Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

K. Kasamatsu
A. Kato
NTT Software Corporation
M. Scott
CertiVox
T. Kobayashi
Y. Kawahara
NTT

July 6, 2015

Barreto-Naehrig Curves
draft-kasamatsu-bncurves-01

Abstract

Elliptic curves with pairings are useful tools for constructing cryptographic primitives. In this memo, we specify domain parameters of Barreto-Naehrig curves (BN-curves) [8]. The BN-curve is an elliptic curve suitable for pairings and allows us to achieve high security and efficiency of cryptographic schemes. This memo specifies domain parameters of four 254-bit BN-curves [1] [2] [5] which allow us to obtain efficient implementations.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Elliptic curves with a special map called a pairing or bilinear map allow cryptographic primitives to achieve functions or efficiency which cannot be realized by conventional mathematical tools. There are identity-based encryption (IBE), attribute-based encryption (ABE), ZSS signature, broadcast encryption (BE) as examples of such primitives. IBE realizes powerful management of public keys by allowing us to use a trusted identifier as a public key. ABE provides a rich decryption condition based on boolean functions and attributes corresponding to a secret key or a ciphertext. The ZSS signature gives a shorter size of signature than that of ECDSA. BE provides an efficient encryption procedure in a broadcast setting.

Some of these cryptographic schemes based on elliptic curves with pairings were proposed in the IETF (e.g. [9], [10], and [11]) and used in some protocols (e.g. [12], [13], [14], [15], and [16]). These cryptographic primitives will be used actively more in the IETF due to their functions or efficiency.

We need to choose an appropriate type of elliptic curve and parameters for the pairing-based cryptographic schemes, because the choice has great impact on security and efficiency of these schemes. However, an RFC on elliptic curves with pairings has not yet been provided in the IETF.

In this memo, we specify domain parameters of Barreto-Naehrig curve (BN-curve) [8]. The BN-curve allows us to achieve high security and efficiency with pairings due to an optimum embedding degree for 128-bit security. This memo specifies domain parameters of four 254-bit BN-curves ([1] and [2]) because of these efficiencies ([5]). These BN-curves are known as efficient curves in academia and particularly provide efficient pairing computation which is generally slowest operation in pairing-based cryptography. There are optimized source codes of ([1] and [2]) as open source software ([20] , [21], and [23]), respectively. This memo describes domain parameters of 224, 256, 384, and 512-bit curves which are compliant with ISO document [3] and organizes differences between types of elliptic curves which are compliant with ISO document [3] in Appendix A.

2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this memo are to be interpreted as described in [4].

3. Preliminaries

In this section, we introduce the definition of elliptic curve and bilinear map, notation used in this memo.

3.1. Elliptic Curve

Throughout this memo, let $p > 3$ be a prime, $q = p^n$, and n be a natural number. Also, let F_q be a finite field. The curve defined by the following equation E is called an elliptic curve.

```
E : y^2 = x^3 + A * x + B such that A, B are in F_q,
          4 * A^3 + 27 * B^2 != 0 mod F_q
```

Solutions (x, y) for an elliptic curve E, as well as the point at infinity, are called F_q-rational points. The additive group is constructed by a well-defined operation in the set of F_q-rational points. Typically, the cyclic additive group with prime order r and the base point G in its group is used for the cryptographic applications. Furthermore, we define terminology used in this memo as follows.

O_E: the point at infinity over elliptic curve E.

#E(F_q): number of points on an elliptic curve E over F_q.

cofactor h: h = #E(F_p)/r.

embedding degree k: minimum integer k such that r is a divisor of q^k - 1

3.2. Bilinear Map

Let G_1 be an additive group of prime order r and let G_2 and G_T be additive and multiplicative groups, respectively, of the same order. Let P, Q be generators of G_1, G_2 respectively. We say that (G_1, G_2, G_T) are asymmetric bilinear map groups if there exists a bilinear map e: (G_1, G_2) -> G_T satisfying the following properties:

1. Bilinearity: for any S in G_1, for any T in G_2, for any a, b in Z_r, we have the relation e([a]S, [b]T) = e(S, T)^{a * b}.

2. Non-degeneracy: for any T in G_2, e(S, T) = 1 if and only if S = O_E. Similarly, for any S in G_1, e(S, T) = 1 if and only if T = O_E.

3. Computability: for any S in G_1, for any T in G_2, the bilinear map is efficiently computable.

For BN-curves, G_1 is a r-order cyclic subgroup of E(F_p) and G_2 is a subgroup of E(F_{p^k}), where k is the embedding degree of the curve. The group G_T is the set of r-th roots of unity in the finite field F_{p^k}.

4. Domain Parameter Specification

In this section, this memo specifies the domain parameters for four 254-bit elliptic curves which allow us to efficiently compute the operation of a pairing at high levels of security.

4.1. Notation for Domain Parameters and Types of Sextic Twists

Here, we define notations for specifying domain parameters and explain types of pairing friendly curves.

The BN-curves E over F_p satisfy following equation.

y^2 = x^3 + B for B in F_p

The values p and r are computed from a suitable integer t.

p is a characteristic of a prime field F_p: p = 36 * t^4 + 36 * t^3 + 24 * t^2 + 6 * t + 1.

r is order of group E over F_p: r = 36 * t^4 + 36 * t^3 + 18 * t^2 + 6 * t + 1.

Also, the value b in the constant of the irreducible field polynomial u^2 + b in F_{p^2}.

Domain parameters of the elliptic curve E(F_p) and E(F_{p^12}) are needed for computation of the pairing. In the pairing over BN-curves, we usually use a sextic twist curve group E’(F_{p^2}) and a map I from the sextic twist E’(F_{p^2}) to E(F_{p^12}) instead of E(F_{p^12}). Hence, this memo follows the group and the map. For the details of the group and the map, refer to [8].

The sextic twist curves are classified in two types, which are called D-type and M-type respectively [22]. The D-type sextic twist curve is defined by equation E’: y’^2 = x’^3 + B/s when elliptic curve E(F_p) is set to be y^2 = x^3 + B and represent of F_{p^12} is set to be F_{p^2}[u]/(u^6 - s), where s is in F_{p^2}^*. Let z be a root of u^6 - s, where z is in F_{p^12}. The corresponding map I: E’(F_{p^2}) -> E(F_{p^12}) is (x’, y’) -> (z^2 * x’, z^3 * y’). The M-type sextic twist curve is defined by equation E’: y^2 = x^3 + B * s when elliptic curve E(F_p) is set to be y^2 = x^3 + B and represent of F_{p^12} is set to be F_{p^2}[u]/(u^6 - s), where s is in F_{p^2}^*. The corresponding map I: E’(F_{p^2}) -> E(F_{p^12}) is (x’, y’) -> (x’ * s^{-1} * z^4, y’ * s^{-1} * z^3), with z^6 = s.

For the pairing, the group G_1 is defined as the subgroup of order r in E(F_p). Then, the group G_2 is defined as the subgroup of order r in E’(F_{p^2}). The group G_T is subgroup of order r in the multiplicative group F_{p^12}^*. The output of pairing is an element on G_T. The order of F_{p^12}^* can be decomposed into (p^12 - 1) = (p^6 - 1) * (p^2 + 1) * (p^4 - p^2 + 1)/r. Let the cofactor h’’ of r on F_{p^12} be h’’_1 * h’’_2, where h’’_1 = (p^4 - p^2 + 1)/r and h’’_2 = (p^6 - 1) * (p^2 + 1).

These domain parameters are described in the following way.

For elliptic curve E(F_p)

G1-Curve-ID is an identifier of the G_1 curve with which the curve can be referenced.

p_b is a prime specifying a base field F_p.

B is the coefficient of the equation y^2 = x^3 + B mod p defining E.

G = (x, y) is the base point, i.e., a point with x and y being its x- and y-coordinates in E, respectively.

r is the prime order of the group generated by G.

h is the cofactor of G in E(F_p)

For twisted curve E’(F_{p^2})

G2-Curve-ID is an identifier of the G_2 curve with which the curve can be referenced.

p_b is a prime specifying a base field.

e2 is the constant of an irreducible polynomial specifying extension field F_{p^2} = F_p[u]/(u^2 - e2).

B’ is the coefficient of the equation y’^2 = x’^3 + B’ mod F_p^2 defining E’.

G’ = (x’, y’) is the base point, i.e., a point with x’ and y’ being its x’- and y’-coordinates in E’, respectively.

r’ is the prime order of the group generated by G’.

h’ is the cofactor of r’ in #E’(F_{p^2})

For F_{p^12}^*

GT-Field-ID is an identifier of the F_{p^12}^*.

p_b is a prime specifying base field.

r’’ is the prime order of the group.

e2 is the constant of the irreducible polynomial of F_{p^2} = F_p [u]/(u^2 - e2).

e6 is the constant of the irreducible polynomial of F_{p^6} = F_{p^2}[v]/(v^3 - e6).

e12 is the constant of the irreducible polynomial of F_{p^12} = F_{p^6}[w]/(w^2 - e12).

h’’ is the cofactor of r in F_{p^12}^* s.t. h’’ = h’’_1 * h’’_2

h’’_1 is the part of cofactor of r in F_{p^12}^* s.t. h’’_1 = (p^4 - p^2 + 1)/r

h’’_2 is the part of cofactor of r in F_{p^12}^* s.t. h’’_2 = (p^6 - 1) * (p^2 + 1)

For the definition of the pairing parameter

Pairing-Param-ID is the set of the identifiers G1-Curve-ID, G2-Curve-ID and GT-Field-ID.

4.2. Efficient Domain Parameters for 254-Bit-Curves

This section specifies the domain parameters for four 254-bit elliptic curves. All twisted domain parameters specified in this section are D-type.

4.2.1. Domain Parameters by Beuchat et al.

The domain parameters by Beuchat et al. [1] generated by t = 3fc0100000000000.

The domain parameters described in this subsection are defined by elliptic curve E(F_p) : y^2 = x^3 + 5 and sextic twist E’(F_{p^2}) : x’^3 + 5/s = x’^3 - u, where F_{p^2} = F_p[u]/(u^2 + 5), F_{p^6} = F_{p^2}[v]/(v^3 - u), F_{p^12} = F_{p^6}[w]/(w^2 - v), s = - 5/u. We describe domain parameters of elliptic curves E and E’. The parameter p_b is 1 mod 8. For the details of these parameters, refer to [1].

G1-Curve-ID: Fp254BNa

p_b = 0x2370fb049d410fbe4e761a9886e502417d023f40180000017e80600000
000001

x = 1

y = 0xd45589b158faaf6ab0e4ad38d998e9982e7ff63964ee1460342a592677cc
cb0

r = 0x2370fb049d410fbe4e761a9886e502411dc1af70120000017e8060000000
0001

h = 1

G2-Curve-ID: Fp254n2BNa

p_b = 0x2370fb049d410fbe4e761a9886e502417d023f40180000017e80600000
000001

e2 = -5 in F_p

B’ = -u

x’ = 0x19b0bea4afe4c330da93cc3533da38a9f430b471c6f8a536e81962ed967
909b5 + (0xa1cf585585a61c6e9880b1f2a5c539f7d906fff238fa6341e1de1a2
e45c3f72) u

y’ = 0x17abd366ebbd65333e49c711a80a0cf6d24adf1b9b3990eedcc91731384
d2627 + (0x0ee97d6de9902a27d00e952232a78700863bc9aa9be960C32f5bf9f
d0a32d345) u

r’ = r

h’ = 0x2370fb049d410fbe4e761a9886e50241dc42cf101e0000017e806000000
00001

GT-Field-ID: Fp254n12a

p_b = 0x2370fb049d410fbe4e761a9886e502417d023f40180000017e80600000
000001

r’’ = r

e2 = -5 in F_p

e6 = u in F_{p^2}

e12 = v in F_{p^6}

```
h’’ = 0x189b459262d16204423a54bb8427aba5530e63254675b78cca28b1f810
476f6b3c53ed0eec245d3ffa0db96f3d713f434a4870545018ff4ea2c361c594bb
b978ce81c80fd1d1cc16cdde274c80f3345359b79069f453e128c1502c0939bbc7
c5cd822ab539b98c5bd283a3377cf7638d91a123a167c510e55bbf53609af49c01
b9c0678c1c10f11cc862018f8fca977741390b5093031edcef806a7301b263b23c
97ea03430da6512a4d5f6df97e761baaf604e724be4f5aafd48fe75994131f2c78
5e364e09256e04dbd1c5eb89733e8ad5a1dacfbb082f399a0d0ea0ab73d6478a96
4221656337a971792a7a42902fcce7c32eb12ab7225b55bf4c7c56d697e0481cb6
23808f99ac23c352660bfd238ab5347121765223970ad69ad7343393718708bd0f
613e4596afede064f7eea9f73082070596e8c495b49fab1bed21ac7b33b5d084c7
ed91d1ae8c38a69d0fa48b8000011ee04800000000000

h’’_1 = 0xade56cf7e1002629c65ca37294ca9149f129ccbb50212575b3d18098
dac4072302eae88c14b40564d9b21719304c9efd7c907850461e1ce3a37da6d40b
e2032e03c8c76238b30af10d6da963854a4aca504a90ae0000017e806000000000
01

h’’_2 = 0x24396d2e7daaf102f72fc17484da5601e50a8e4fe4101271d84f0639
930313fae7dbbc4b6f64a48a9bbc8b65632eea8295222ece92adb1fdad8a57b84b
13025fd1c64ebe9b3daa6b9be21c2330e997025161babcc1d0eb55d93939c5fd02
e02f1c269f16c3785aef71f0ef1c256be2bf9de36925b42004c3d390638c802e46
f220bf63cc039d8ab7e73ad426b32f383084672ea9f0fe34d053a6184768d21c52
cfd50313acaeed74538e4cd07c1827e7e9a8f14eac8401482fefa2e06ec810f407
882b548ea549c760b3e2013b5a299a6cd7395bbd58ebd04400e5e193fcae081e0b
e4dae5650bb8707a73b116f9fa887c708000011ee04800000000000

Pairing-Param-ID: Beuchat = {
    G1-Curve-ID: Fp254BNa
    G2-Curve-ID: Fp254n2BNa
    GT-Field-ID: Fp254n12a
}
```

4.2.2. Domain Parameters by Nogami et al. / Aranha et al.

The domain parameters by Nogami et al. [2] generated by t = -0x4080000000000001. Aranha et al. presented an open source library of the pairing using this parameter [2].

The domain parameters described in this subsection are defined by elliptic curve E(F_p) : y^2 = x^3 + 2 and sextic twist E’(F_{p^2}) : x’^3 + 2/s = x’^3 + 1 - u, where F_{p^2} = F_p[u]/(u^2 + 1), F_{p^6} = F_{p^2}[v]/(v^3 - (1 + u)), F_{p^12} = F_{p^6}[w]/(w^2 - v), 1/s = 1/(1 + u). We describes domain parameters of elliptic curves E and E’. The parameter p_b is 3 mod 4. For the details of these parameters, refer to [2].

```
G1-Curve-ID: Fp254BNb
```

p_b = 0x2523648240000001ba344d80000000086121000000000013a700000000
000013

B = 2

x = 0x2523648240000001ba344d80000000086121000000000013a70000000000
0012

y = 1

r = 0x2523648240000001ba344d8000000007ff9f800000000010a10000000000
000d

h = 1

G2-Curve-ID: Fp254BNb

p_b = 0x2523648240000001ba344d80000000086121000000000013a700000000
000013

e2 = -1 in F_p

B’ = 1 + (-1) u

x’ = 0x061a10bb519eb62feb8d8c7e8c61edb6a4648bbb4898bf0d91ee4224c80
3fb2b + (0x0516aaf9ba737833310aa78c5982aa5b1f4d746bae3784b70d8c34c
1e7d54cf3) u

y’ = 0x021897a06baf93439a90e096698c822329bd0ae6bdbe09bd19f0e07891c
d2b9a + (0x0ebb2b0e7c8b15268f6d4456f5f38d37b09006ffd739c9578a2d1ae
c6b3ace9b) u

r’ = r

h’ = 0x2523648240000001ba344d8000000008c2a2800000000016ad000000000
00019

GT-Field-ID: Fp254n12b

p_b = 0x2523648240000001ba344d80000000086121000000000013a700000000
000013

r’’ = r

e2 = -1 in F_p

e6 = 1 + u in F_{p^2}

```
e12 = v in F_{p^6}

h’’ = 0x2928fbb36b391596ee3fe4cbe857330da83e46fedf04d235a4a8daf5ff
9f6eabcb4e3f20aa06f0a0d96b24f9af0cbbce750d61627dcbf5fec9139b8f1c46
c86b49b4f8a202af26e4504f2c0f56570e9bd5b94c403f385d1908556486e24b39
6ddc2cdf13d06542f84fe8e82ccbad7b7423fc1ef4e8cc73d605e3e867c0a75f45
ea7f6356d9846ce35d5a34f30396938818ad41914b97b99c289a7259b5d2e09477
a77bd3c409b19f19e893f8ade90b0aed1b5fc8a07a3cebb41d4e9eee96b21a832d
db1e93e113edfb704fa532848c18593cd0ee90444a1b3499a800177ea38bdec62e
c5191f2b6bbee449722f98d2173ad33077545c2ad10347e125a56fb40f086e9a4e
62ad336a72c8b202ac3c1473d73b93d93dc0795ca0ca39226e7b4c1bb92f99248e
c0806e0ad70744e9f2238736790f5185ea4c70808442a7d530c6ccd56b55a69738
67ec6c73599bbd020bbe105da9c6b5c009ad8946cd6f0

h’’_1 = 0xc816ed457c4f0cbba598fbf85278d6a283736855af2828a32ad1c29a
144223e6281b946847fdfeb69c50d19a04e83b02b9108347fe83011a78b30ec3c0
4f5235bd893d800083e82c022780000099261da2800000006fd671000000000027
0d

h’’_2 = 0x34a94d3d1f0dc12947911459f9c97e1cafcb74609938a7cd37a11adf
6b9bd9bba488c257f6684b18eaf5e67df52cac7666c59efee0438bd28494fdda8d
885b39a9fcdc9ec6fccae4176a422f3f96db68ff3d696b0842dfed0d2ba7e853d9
cb6ea2194a2457251fa44e714cea395c60ea4852c28305971c9405144476d3cad8
a7fdcb78a53125d893e87ac3969ecf74ddd99f9e6ba4fc7d0d8c6b607840f2b9a2
5cf964bff87e6160db1954275f370301029b0b18e809ac493883635763bd991d19
19680457071767d197dfed87a2112b74feaec3e7e276b2c884552cc2543491bfb5
420df1026219e849c1f94a4d35e0020c9d8849b5c000003f71a76b0



Pairing-Param-ID: Nogami-Aranha = {
    G1-Curve-ID: Fp254BNb
    G2-Curve-ID: Fp254n2BNb
    GT-Field-ID: Fp254n12b
}
```

4.2.3. Domain Parameters Scott

The domain parameters by Scott generated by t = -0x4000806000004081 [6].

The domain parameters described in this subsection are defined by elliptic curve $E(F_p) : y^2 = x^3 + 2$ and sextic twist $E'(F_{p^2}) : x'^3 + 2/s = x'^3 + 1 - u$, where $F_{p^2} = F_p[u]/(u^2 + 1)$, $F_{p^6} = F_{p^2}[v]/(v^3 - (1 + u))$, $F_{p^12} = F_{p^6}[w]/(w^2 - v)$, $1/s = 1/(1 + u)$. We describes domain parameters of elliptic curves E and E’. The parameter p_b is 3 mod 4. For the details of these parameters, refer to [2].

G1-Curve-ID: Fp254BNc

p_b = 0x240120db6517014efa0bab3696f8d5f06e8a555614f464babe9dbbfeeeb4a713

B = 2

x = 0x240120db6517014efa0bab3696f8d5f06e8a555614f464babe9dbbfeeeb4a712

y = 1

r = 0x240120db6517014efa0bab3696f8d5f00e88d43492b2cb363a75777e8d30210d

h = 1

G2-Curve-ID: Fp254n2BNc

p_b = 0x240120db6517014efa0bab3696f8d5f06e8a555614f464babe9dbbfeeeb4a713

e2 = -1 in F_p

B’ = 1 + (-1) u

r’ = r

x’ = 0x0571af2ea9666eb2a53f3fb837172bdd809c03a95c5870f34a8cb340220bf9c0 + (0x0f71abb712a9e6e12c07b58bc01f2f994c3b5a1531cf96609b838e5ccf05bc71) u

y’ = 0x0b88822fe134c1695b21419bb1ab9732f707701046a2e6ff3ad10f3c70284b93 + (0x1659b723676b5af5231fb045b3d822c0de6fcaab171bad9c8951afc800a26775) u

h’ = 0x240120db6517014efa0bab3696f8d5f0ce8bd6779735fe3f42c6007f50392d19

GT-Field-ID: Fp254n12c

p_b = 0x240120db6517014efa0bab3696f8d5f06e8a555614f464babe9dbbfeeeb4a713

r’’ = r

e2 = -1 in F_p

```
e6 = 1 + u in F_{p^2}

e12 = v in F_{p^6}

h’’ = 0x1d43e8fcd92a8e7d54f5820d5a3701e694bad5ec9021a8a58128e0908b
cb1747bc941f92c7713cf91dc9a015614324e892b37c0bbcc7873897da12bded8e
e32461e008c9b2e43e5a5d6498bb1b44874b164fc2f8cb2e02847eb2550ef4fb67
ebba59d2dc7b7fa6b348d432b00916f8fafd5ec31daed9dc0c9790d7640fd2085e
d6bf6796b5634709896c13aabbc8ad817ce596a31e581258e2d88985978f27e6b4
b5daadbe327cb2dfc0220f0dfb61a1fe9dc7f88e061d67a0c1f6dac9b1d839e046
ecbd957bb030322f4ab982f624f1aa8c1d8f97661f7d6fe0f01660b845948d1ca4
db92203ccb50779ccb981ba37248a67f2f5f7201dd03efbadd98232ffec54f723b
583c0df642183ad006819a33e938fd763efee80a64a5aa7092ce5e4bf7f40c9442
5a83e47b6f0e685bf5a801c864f76637225082c61c7fda904ac0d5fc90ee608f9c
b5f79b6e69c217097de370e7a0f22ae9afbb992f232f0

h’’_1 = 0xb651238d914d6ec916c6f4c59202389fb75a267e7c7feabf4a5ee9ef
5aa0b588f60d6f5d737b92988f3253f3d3c8aa439f0743d28102d47dc7e0b0ff07
f71e282739c9d5a3236579d81733eaf9269bb184134d7ac2c082e05ea6e634f918
0d

h’’_2 = 0x2917c05fa90fae306d470d8d5d3f04e9265a173b6c281349dab6abff
e85c4b6129d208e97f9d6240137b86473a62a611475435473877667777a255874c9
16f826d23df531380749423add88352eb9838833969e3fcc2b61bbfa62ab642308
509c7ef4dddc267f1f9ab38047837b4618a6d477a9c3067cd2d5711c450915e9a6
fd49ee049860c56da205aaf066dfab99472a91a225abcaa4051b77ee0f8c811889
384be038871765c7e4ade3fe391232d04f4397c94f1273cf057a6552123e1c30d6
e0dd4536a32d372a3d426d1d9046f5da0ffdfe53ab2d4a4fa6604b6c224c04e916
90d605d0bd8be366a4bd78b4bfeafb9c7face675844fd40ed13d2b0

Pairing-Param-ID: Scott = {
    G1-Curve-ID: Fp254BNc
    G2-Curve-ID: Fp254n2BNc
    GT-Field-ID: Fp254n12c
}
```

4.2.4. Domain Parameters by BCMNPZ

The domain parameters by BCMNPZ generated by t = -0x4000020100608205 [7].

The domain parameters described in this subsection are defined by elliptic curve E(F_p) : y^2 = x^3 + 2 and sextic twist E’(F_{p^2}) : x’^3 + 2/s = x’^3 + 1 - u, where F_{p^2} = F_p[u]/(u^2 + 1), F_{p^6} = F_{p^2}[v]/(v^3 - (1 + u)), F_{p^12} = F_{p^6}[w]/(w^2 - v), 1/s = 1/(1 + u). We describes domain parameters of elliptic curves E and E’. The parameter p_b is 3 mod 4. For the details of these parameters, refer to [2].

G1-Curve-ID: Fp254BNd

p_b = 0x24000482410f5aadb74e200f3b89d00081cf93e428f0d651e8b2dc2bb4
60a48b

x = 0x24000482410f5aadb74e200f3b89d00081cf93e428f0d651e8b2dc2bb460
a48a

y = 1

B = 2

r = 0x24000482410f5aadb74e200f3b89d00021cf8de127b73833d7fb71a511aa
2bf5

h = 1

G2-Curve-ID: Fp254BNd

p_b = 0x24000482410f5aadb74e200f3b89d00081cf93e428f0d651e8b2dc2bb4
60a48b

e2 = -1 in F_p

B’ = 1 + (-1) u

r’ = r

x’ = 0x20cfe8b965fc444008a21b12cd2a55f843c1dd68ba12a8bb1f1dde3533b
91a32 + (0x0176f822a5ee7ada449f8f876ee001508dd43b5413e03c8f4ad3e3b
38dadaf51) u

y’ = 0x02b27f22c2920fee3b4af218b6d92421780a9bdc66155142fecef3af7f5
8e872 + (0x14e9c62a36ebce710810576b5401fdf0b28126ad2d563bf5043be33
47646dfb4) u

h’ = 0x24000482410f5aadb74e200f3b89d000e1cf99e72a2a746ff96a46b2571
71d21

GT-Field-ID: Fp254n12d

p_b = 0x24000482410f5aadb74e200f3b89d00081cf93e428f0d651e8b2dc2bb4
60a48b

r’’ = r

e2 = -1 in F_p

```
e6 = 1 + u in F_{p^2}

e12 = v in F_{p^6}

h’’ = 0x1d39fc2421c459d1f0de7cde7c1285648918cd045a503063f111e3aaba
83df215962969c6fceb6f999c374d7c0fb36eb380701566be2e2b206368ba4f04e
ebcdf9c008c23935547b5a46e37a5f1f6e26745bf3219c8b4456c4fbc261596000
4d5f42547d6b9a867244929fd958b2f962fb35d58f0225a524e4199f3e961c67e9
b1618141cbe93892841e90040854c324d828bcabba01c45b1c8d62829192d22d2f
a7281370c28fe7449df33a45af6bf04c8fc54e271bd28c671b5ef06591044fce06
13d7a0fb7a9f4467428dcdf071e85f86bf6097ec6dd14b974aa94a1d189b2227ae
75851160753faac94c2bcb2c15fd5be5e68fc316683ac92cf07b7030c91b25e4dd
40f8a6fc9c128f52b060f4be0c33dd22007c9df38874bf6ce8f21736b6ce5b2d0a
69d802b0efe5d3a05fe0fa939f27bdb66812f89bfef4c3852044c18aa3059d5b63
505ec878753497904916ce2ede9dd267ccd69fcf26c50

h’’_1 = 0xb640447a44acc2b50912a1528832c5f4358315c85cd27dc4629b83ad
23ca6447537784d1adc703cf92a32bf736604c22f7fc113e08bd1a0f4061cc8a1c
c42f380317a331d6cb9e0fbbb55404de8fbd905999f354e0c0a9d80c9dbebc66ca
35

h’’_2 = 0x290d9d32167d7406812204488b22639b77897f44694c058dd022c218
16fc3e82f03b87223ac3b8fba7a347184422c7278b0d501d0de0374429d873e7ef
5c86ca749bc6bc55607d2f6dc47fc8fa1abf770d4341041836d6de95ffa72e2cee
6b0ace366bdd8d94be2d4c7c4a4f2312b12932ca02c795a69a53467ce26ae7afb2
f5d99e43aec676bc1564aad101c07a096650986516e4680683384113fcb842d1d4
b6dc261a852b3e85e2b39d159189a82de7794fe53d10feec08ec3521b110b1cfc4
d9d49204f248f9d162489f3bb2c5c0725a1e6da1e0b7df86f8464cc6df13439cd2
5d90d220d3514c1824b5917c5713a224dcd44c8e2c08fbe2e9fc510

Pairing-Param-ID: BCMNPZ = {
    G1-Curve-ID: Fp254BNd
    G2-Curve-ID: Fp254n2BNd
    GT-Field-ID: Fp254n12d
}
```

5. Object Identifiers

We need to define the following object identifiers. Which organization is suitable for the allotment of these object identifiers?

```
Beuchat OBJECT IDENTIFIER ::= {TBD}

Nogami-Aranha OBJECT IDENTIFIER ::= {TBD}

Scott OBJECT IDENTIFIER ::= {TBD}
```

```
BCMNPZ OBJECT IDENTIFIER ::= {TBD}
```

6. Security Considerations

For above sections, G_1 is a r-order cyclic subgroup of E(F_p) and G_2 is a subgroup of E'(F_{p^2}), where k is the embedding degree of the curve and the group G_T is the set of r-th roots of unity in the finite field F_{p^12}^*. In this section, G_1, G_2 and G_T imply E(F_p), E'(F_{p^2}) and F_{p^12}^* respectively.

Pairing-based cryptographic primitives are often based on the hardness of the following problems, so when the elliptic curves from this document are used in such schemes, these problems would apply.

The elliptic curve discrete logarithm problem in G_1 and G_2 (ECDLP)

The finite field discrete logarithm problem in G_T (FFDLP)

The elliptic curve computational Diffie-Hellman (CDH) problem in G_1 and G_2

The elliptic curve computational co-Diffie-Hellman problem in G_1 and G_2

The elliptic curve decisional Diffie-Hellman (DDH) problem in G_1

The bilinear Diffie-Hellman (BDH) problem

Algorithms to efficiently solve the problems above, aside from special cases, are unknown. Mainly, there are Pollard-rho algorithm [18] against point of an elliptic curve G_1 and G_2, and Number Field Sieve method [17] against G_T which is output of pairing as generic attacks against elliptic curve with pairing .

G_T to be larger than G_1 and G_2, because FFDLP can be computed more efficiently than ECDLP in most cases. Security level of schemes based on pairing depends most weak level for each problems. Thus implementors should necessary to ensure adequate security level for both of problems.

Table 1 shows the security level of elliptic curves described in this memo Schemes based on the elliptic curves (i.e. G_1 and G_2) and the finite fields (i.e. G_T) must be combined with cryptographic primitives which have similar or greater security level than the scheme.

Pairing-Param-ID	Security Level for ECDLP in G_1, G_2 (bits)	Security Level for FFDLP in G_T (bits)
Beuchat	128	128
Nogami-Aranha	128	128
Scott	128	128
BCMNPZ	128	128

Table 1: security level of elliptic curves and finite field specified in this memo

6.1. Subgroup Security (OPTIONAL requirement)

For BN-curves, G_1 is cryptographic group of large prime order and cofactor h is always 1. On the other hand, G_2, G_T are consisted of subgroup of order h’ and h’’ that are not equal to 1 in addition to subgroup of order r , resp. Thus implementors who provided groups in G_2 and G_T, MUST check element of those groups included in subgroup of order r (see [7]) .

The order check of G_T can be performed by exponentiation of h’’_1 and h’’_2. The exponentiation of h’’_2 can be easily computed by using Frobenius map. Whereas the exponentiation of h’’_1 is complicated.

For simplification of the order check which is the smallest prime factor of h’ and h’’_1 will be greater than r, of element, we define OPTIONAL security G_2-strong and G_T-strong security. G_2-strong and G_T-strong means those order of cryptographic group MUST have the smallest prime factor greater than r. Therefore implementors could not check of order, G_2-strong and G_T-strong cryptographic group will not be insecure

Table 2 shows the G_2, G_T-strong security of parameters described in this memo.

Pairing-Param-ID	Have G_2-Strong?	Have G_T-Strong?
Beuchat	no	no
Nogami-Aranha	no	no
Scott	no	yes
BCMNPZ	yes	yes

Table 2: G2, G3-strong security

7. Acknowledgements

This memo was inspired by the content and structure of [19].

8. Change log

NOTE TO RFC EDITOR: Please remove this section in before final RFC publication.

9. References

9.1. Normative References

[1] Beuchat, J., Gonzalez-Diaz, J., Mitsunari, S., Okamoto, E., Rodriguez-Henriquez, F., and T. Teruya, "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves", Proceedings Lecture notes in computer sciences; Pairing-Based Cryptography --Pairing2010, 2010.

[2] Aranha, D., Karabina, K., Longa, P., Gebotys, C., Rodriguez-Henriquez, F., and J. Lopez, "Faster Explicit Formulas for Computing Pairings over Ordinary Curves", Proceedings Lecture notes in computer sciences; EUROCRYPT --EUROCRYPT2011, 2011.

[3] International Organization for Standardization, "Information Technology - Security Techniques -- Cryptographic techniques based on elliptic curves . Part 5: Elliptic curve generation", ISO/IEC 15946-5, 2009.

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

[5] Nogami, Y., Akane, M., Sakemi, Y., Kato, H., and Y. Morikawa, "Integer Variable \chi Based Ate Pairing", Proceedings Pairing 2008, LNCS 5209, pp. 178.191, Springer-Verlag , 2008.

9.2. Informative References

[6] Scott, M., "Unbalancing Pairing-Based Key Exchange Protocols", ePrint http://eprint.iacr.org/2013/688.pdf, 2013.

[7] Barreto, P., Costello, C., Misoczki, R., Naehrig, M., Pereira, G., and G. Zanon, "Subgroup security in pairing-based cryptography", ePrint http://eprint.iacr.org/2015/247.pdf, 2015.

[8] Barreto, P. and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order", Proceedings Lecture notes in computer sciences; 3897 in Selected Areas in Cryptgraphy -- SAC2005, 2006.

[9] Boyen, X. and L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, December 2007.

[10] Groves, M., "Sakai-Kasahara Key Encryption (SAKKE)", RFC 6508, February 2012.

[11] Hitt, L., "ZSS Short Signature Scheme for Supersingular and BN Curves", draft-irtf-cfrg-zss-02 (work in progress), 2013.

[12] Martin, L. and M. Schertler, "Using the Boneh-Franklin and Boneh-Boyen Identity-Based Encryption Algorithms with the Cryptographic Message Syntax (CMS)", RFC 5409, January 2009.

[13] Cakulev, V. and G. Sundaram, "MIKEY-IBAKE: Identity-Based Authenticated Key Exchange (IBAKE) Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)", RFC 6267, June 2011.

[14] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", RFC 6507, February 2012.

[15] Groves, M., "MIKEY-SAKKE: Sakai-Kasahara Key Encryption in Multimedia Internet KEYing (MIKEY)", RFC 6509, February 2012.

[16] Cakulev, V., Sundaram, G., and I. Broustis, "IBAKE: Identity-Based Authenticated Key Exchange", RFC 6539, March 2012.

[17] Joux, A., Lercier, R., Smart, P., and F. Vercauteren, "The number field sieve in the medium prime case", Proceedings Lecture notes in computer sciences; 4117 in Comput. Sci. -- CRYPTO2006, 2006.

[18] Pollard, J., "Monte Carlo Methods for Index Computation (mod p)", Proceedings Mathematics of Computation, Vol.32, 1978.

[19] Lochter, M. and J. Merkle, "Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation", RFC 5639, March 2010.

[20] "University of Tsukuba Elliptic Curve and Pairing Library", 2013, <http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html>.

[21] Aranha, D. and C. Gouv, "RELIC is an Efficient LIbrary for Cryptography", 2013, <https://code.google.com/p/relic-toolkit/>.

[22] Aranha, D., Barreto, P., Longa, P., and J. Rocardini, "The Realm of the Pairings", SAC 2013, to appear, 2013.

[23] Scott, M., "The MIRACL IoT Multi-Lingual Crypto Library", 2015, <https://github.com/CertiVox/MiotCL.git>.

Appendix A. Domain Parameters Based on ISO Document

We describe the domain parameters for 224, 256, 384, and 512-bit elliptic curves which are compliant with the ISO document and are based on M-type twisted curve. The domain parameters described in below subsections are defined by Elliptic curve E(F_p): y^2 = x^3 + 3 and sextic twist E’(F_{p^2}): y’^2 = x’^3 + 3 * s, where F_{p^2} = F_p[u]/(u^2 + 1), F_{p^12} = F_{p^2}[w]/(w^6 - s), s = 1 + u. We describe domain parameters of elliptic curves E. Detailed information on these domain parameters is given in [3].

A.1. Specific ISO domain parameters

A.1.1. Domain Parameters for 224-Bit Curves

G1-Curve-ID: Fp224BN

p_b = 0xfffffffffff107288ec29e602c4520db42180823bb907d1287127833

B = 3

x = 1

y = 2

r = 0xfffffffffff107288ec29e602c4420db4218082b36c2accff76c58ed

h = 1

A.1.2. Domain Parameters for 256-Bit Curves

G1-Curve-ID: Fp256BN

p_b = 0xfffffffffffcf0cd46e5f25eee71a49f0cdc65fb12980a82d3292ddbae
d33013

B = 3

x = 1

y = 2

r = 0xfffffffffffcf0cd46e5f25eee71a49e0cdc65fb1299921af62d536cd10b
500d

h = 1

A.1.3. Domain Parameters for 384-Bit Curves

G1-Curve-ID: Fp384BN

p_b = 0xfffffffffffffffffff2a96823d5920d2a127e3f6fbca024c8fbe29531
892c79534f9d306328261550a7cabd7cccd10b

B = 3

x = 1

y = 2

r = 0xfffffffffffffffffff2a96823d5920d2a127e3f6fbca023c8fbe2953189
2c795356487d8ac63e4f4db17384341a5775

h = 1

A.1.4. Domain Parameters for 512-Bit Curves

G1-Curve-ID: Fp512BN

p_b = 0xfffffffffffffffffffffffffff9ec7f01c60ba1d8cb5307c0bbe3c111
b0ef455146cf1eacbe98b8e48c65deab236fel916a55ce5f4c6467b4eb280922ad
ef33

B = 3

x = 1

y = 2

r = 0xfffffffffffffffffffffffffff9ec7f01c60ba1d8cb5307c0bbe3c111b0
ef445146cf1eacbe98b8e48c65deab2679a34a10313e04f9a2b406a64a5f519a09
ed

h = 1

A.1.5. Security of ISO curves

In this section, this memo describes ECDLP on G_1 and G_2, FFDLP on G_T and subgroup security over G_2 and G_T, for ISO curves.

Table 3 shows the security level of ISO curves.

Pairing-Param-ID	Security Level for ECDLP in G_1, G_2 (bits)	Security Level for FFDLP in G_T (bits)
ISO-Fp224	112	112
ISO-Fp256	128	128
ISO-Fp384	192	128
ISO-Fp512	256	128

Table 3: security level of ISO elliptic curves and finite field specified in this memo

Table 4 shows the G_2, G_T-strong security of ISO curves.

Pairing-Param-ID	Have G_2-Strong?	Have G_T-Strong?
ISO-Fp224	no	no
ISO-Fp256	no	no
ISO-Fp384	no	no
ISO-Fp512	no	no

Table 4: G2, G3-strong security of ISO curves

Authors' Addresses

Kohei Kasamatsu
NTT Software Corporation

EMail: kasamatsu.kohei-at-po.ntts.co.jp

Akihiro Kato
NTT Software Corporation

EMail: kato.akihiro-at-po.ntts.co.jp

Michael Scott
CertiVox

EMail: mike.scott-at-certivox.com

Tetsutaro Kobayashi
NTT

EMail: kobayashi.tetsutaro-at-lab.ntt.co.jp

Yuto Kawahara
NTT

EMail: kawahara.yuto-at-lab.ntt.co.jp

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

A. Kato
NTT Software Corporation
T. Hardjono
MIT
T. Kobayashi
T. Saito
K. Suzuki
NTT

July 6, 2015

FSU Key Exchange
draft-kato-fsu-key-exchange-00

Abstract

This draft proposes an identity-based authenticated key exchange protocol following the extended Canetti-Krawczyk (id-eCK) model. The protocol is currently the most efficient among the id-eCK protocols.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. ‘Introduction

Authenticated key exchange (AKE) is a core security function within many deployed systems today. It is a foundational function that allows end-users and systems alike to be authenticated prior to access to resource and services. Over the past two decades key exchange schemes have been proposed, based on symmetric and asymmetric key cryptography.

A more recent approach to AKE protocol has been the introduction of identity binding to the exchange [7] [8], obviating the need to rely on a public key infrastructure in which digital certificates need to be exchanged by users or end-points that wish to communicate signed and/or encrypted messages.

Identity-based AKE (ID-AKE) schemes rely on the use of the trusted intermediary referred to as the Key Generation Center (KGC). The role of the KGC, among others, is to generate a pair of master public and secret keys based on the user’s identity and to extract a user’s secret key corresponding to his or her identity.

In a 2-pass ID-AKE scheme, an "initiator" entity wishing to share a key with a second entity (referred to as the "responder") sends ephemeral public information to the responder. In its turn, the responder sends another ephemeral public information to the initiator entity. Following this, each entity would then generate a session from a number of parameters, notably their respective secret keys (given by the KGC), their own secret values of the ephemeral information, the identity of the peer they’re communicating with, and the ephemeral information they received from that peer.

We propose a provably secure ID-AKE scheme called "FSU" [4] [5] [6] based on the previous model of [9] and which builds on the previous efforts in [10] [11] . The model underlying the FSU was chosen due to the merit of provable security based on an adversarial model in which the adversary has the freedom to choose keys reveal.

2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this memo are to be interpreted as described in [1].

3. Notation

This section shows notation used in this memo.

Let F_q be a finite field with q = p^n elements for a prime p and an integer n and let E(F_q) be an elliptic curve with an order r and an embedding degree k defined over F_q. An embedding degree k is defined as a minimum integer k such that r is a divisor of q^k - 1.

Let G_1 (resp. G_2) be an additive group with an order r generated by E(F_q) (resp. E'(F_q)). Let G_T be multiplicative groups with the same order r. Let P_1, P_2 be generators of G_1, G_2 respectively. We say that (G_1, G_2, G_T) are bilinear map groups if there exists a pairing e: (G_1, G_2) -> G_T satisfying the following properties:

1. Bilinearity: for any Q_1 in G_1, for any Q_2 in G_2, for any a, b in Z_r, we have the relation e(aQ_1, bQ_2) = e(Q_1,Q_2)^{ab}.

2. Non-degeneracy: for any Q_1 in G_1, e(Q_1,Q_2) = 1 only if Q_2 = O_{G_2} and for any Q_2 in G_2, e(Q_1,Q_2) = 1 only if Q_1 = O_{G_1}.

3. Computability: for any Q_1 in G_1, for any Q_2 in G_2, the bilinear map is efficiently computable.

This pairing is described in specification of optimal ate pairing specification[3]. It is defined by Pairing-Parm-ID following way.

```
Pairing-Param-ID = {
    G1-Curve-ID,
    G2-Curve-ID
    GT-Field-ID
}
```

G1-Curve-ID and G2-Curve-ID is an identifiers of elliptic curve. And GT-Field-ID is an identifier of the G_T range of finite field. G1-Curve-ID , G2-Curve-ID and GT-Field-ID are described in [2] the following way.

```
 G1-Curve-ID = {
     p_b    : A prime specifying base field F_p.
     A, B   : The coefficients of the equation y^2 = x^3 A * x + B
             defining E.
     G = (x, y) : The base point, i.e., a point with x and y
              being its x- and y-coordinates in E, respectively.
     r      : The prime order of the group generated by G.
     h      : The cofactor of G in E.
}
 G2-Curve-ID = {
     p_b    : A prime specifying base field F_p.
     e2     : The constant of an irreducible polynomial specifying
              extension field F_{p^2} = Fp[u] / (u^2 - e2).
     A’, B’ : The coefficients of the equation y^2 = x^3 A’ * x +
                B’ defining E’.
     G’ = (x’, y’) : The base point, i.e., a point with x’ and y’
              being its x- and y-coordinates in E’, respectively.
     r’      : The prime order of the group generated by G’.
     h’      : The cofactor of G’ in E’.
}

GT-Filed-ID = {
     p_b    : A prime specifying base field.
     r      : The prime order of the subgroup of F_{p^12}.
     e2     : The constant of the irreducible polynomial of F_{p^2} =
              F_p [u] / (u^2 - e2).
     e6     : The constant of the irreducible polynomial of F_{p^6} =
              F_{p^2}[v] / (v^3 - e6).
     e12    : The constant of the irreducible polynomial of F_{p^12}
              = F_{p^6}[w] / (w^2 - e12).
     h’’    : The cofactor of G_T
 }
```

In addition, this memo uses the following functions.

floor(x) : The function returning an integer such that max{x’ in Z | x’ <= x}.

ceil(x) : The function returning an integer such that min{x’ in Z | x’ >= x}.

O_E : The point at infinity over elliptic curve E.

4. Data Type and Its Conversions

This section describes data type and its conversion used in this memo.

4.1. BitString-to-OctetString Conversion (BS2OSP)

This memo uses conversion from bit strings to octet strings. Informally, the idea is to pad the bit string with 0's on the left to make its length a multiple of 8, then chop the result up into octets. Formally, the conversion routine, BS2OSP(B), is specified in Appendix A.1

4.2. OctetString-to-BitString Conversion (OS2BSP)

This memo uses conversion from octet strings to bit strings. Informally, the idea is simply to view the octet string as a bit string. Formally, the conversion routine, OS2BSP(M), is specified in Appendix A.2

4.3. FieldElement-to-Integer Conversion (FE2IP)

This memo uses conversion from field elements to integers. An finite field element should be represented as a polynomial with subfield coefficients, which can be represented as a sequence of the coefficients. Informally, the idea is simply to view the sequence of the coefficients as the radix-p^m representation of the base field elements, where p^m is the number of the subfield elements. Formally, the conversion routine, FE2IP(a), is specified in Appendix A.3

4.4. Integer-to-FieldElement Conversion (I2FEP)

This memo uses conversion from integers to field elements. A field element should be represented as a polynomial with subfield coefficients, and it can be represented as a sequence of the coefficients. Informally, the idea is to represent the integer with radix-p^m positional number system where p^m is the number of the subfield element, and then convert the each digit to the each coefficient of the polynomial. Formally, the conversion routine, I2FEP(x), is specified in Appendix A.4:

4.5. FieldElement-to-OctetString Conversion (FE2OSP)

This memo uses conversion from field elements to octet strings. This conversion is constructed by using FE2IP and I2SOP conversions. Formally, the conversion routine, FE2OSP(a), is specified in Appendix A.5.

4.6. OctetString-to-FieldElement Conversion (OS2FEP)

This memo uses conversion from octet strings to field elements. This conversion is constructed by using OS2IP and I2FEP conversions. Formally, the conversion routine, OS2FEP(M), is specified in Appendix A.6.

4.7. EllipticCurvePoint-to-OctetString Conversion (ECP2OSP)

This memo uses conversion from elliptic curve points to octet strings. Informally the idea is that, if point compression is being used, the compressed y-coordinate is placed in the leftmost octet of the octet string along with an indication that point compression is on, and the x-coordinate is placed in the remainder of the octet string; otherwise if point compression is off, the leftmost octet indicates that point compression is off, and remainder of the octet string contains the x-coordinate followed by the y-coordinate. Formally, the conversion routine, ECP2OSP(P,R), is specified in Appendix A.7.

4.8. OctetString-to-EllipticCurvePoint Conversion (OS2ECPP)

This memo uses conversion from octet strings to elliptic curve points. Informally, the idea is that, if the octet string represents a compressed point, the compressed y-coordinate is recovered from the leftmost octet, the x-coordinate is recovered from the remainder of the octet string, and then the point compression process is reversed; otherwise the leftmost octet of the octet string is removed, the x-coordinate is recovered from the left half of the remaining octet string, and the y-coordinate is recovered from the right half of the remaining octet string. Formally, the conversion routine, OS2ECPP(M), is specified in Appendix A.8.

5. Building Block of FSU Key Exchange

This section describes building block for constructing FSU Key Exchange.

5.1. Key Derivation Function

MGF1 is a mask generation function, parameterized by a hash function. MGF1(M,n) is defined as follows:

System parameters:

- Hash : a hash function

- hashLen : the length in octets of the hash function output

Input:

- M : a seed from which a mask is generated, an octet string
- n : the octet length of the output, a positive integer

Output:

- mask : a mask, an octet string of length n

Method:

1. Let n_0 be the octet length of M. If n_0 + 4 is greater than the input limitation for the hash function, output INVALID and stop.
2. Set cThreshold = ceil(n / hashLen)
3. If cThreshold > 2^32, output INVALID and stop
4. Let M’ be the empty octet string
5. Set counter = 0
6. B = B_{0}, ..., B{31} such that counter = B_{31} + B_{30}*2 + ... + B_{0}*2^{31}
7. Compute C = BS2OSP(B)
8. Compute H = Hash(M||C)
9. Set M’ = M’||H
10. Set counter = counter + 1
11. If counter < cThreshold, go back to step 6.
12. Set mask = M’_0M’_1...M’_{n-1} where M’ = M’_0M’_1M’_2...
13. Output mask

5.2. Hashing to Point

Hashed value should be converted to elliptic curve point as described in this section. Formally, the conversion routine, HASHINGTOPOINT(Curve-ID, Hash, M), is specified as follows:

Input:

- Curve-ID : an elliptic curve parameter
- Hash : a hash function
- M : an octet string

Output:

- P : an elliptic curve point

Method:

```
1.   Set i = 0

2.   B = B_{0}, ..., B{15} such that counter = B_{15} + B_{14}*2 +
     ... + B_{0}*2^{15}

3.   Compute C = BS2OSP(B)

4.   x_0 = OS2FQE(C||M, Hash, F_{p^m}) in F_{p^m}

5.   t = x_0^3 + A * x_0 + B

6.   If t=0, set P =(x_0, 0) and output h'*P

7.   If t is not square in F_{p^m}, set i = i + 1 and go back to step
     2

8.   Set alpha be one of square roots of t.  Then, -alpha is another
     square root of t.

9.   Set y_1 = FE2IP(alpha)

10.  Set y_2 = FE2IP(-alpha)

11.  If y_1 > y_2, set y_0 = -alpha

12.  Else (i.e. y_1 <= y_2), set y_0 = alpha

13.  Set P = (x_0, y_0)

14.  Output h * P
```

5.2.1. IHF1

Bit string should be converted to hashed non-negative integer less than an assigned integer as described in this section. Formally, the conversion routine, IHF1(s,n,Hash) is defined as follows:

Input:

- s: an octet string
- n : an integer
- Hash : a hash function

Output:

- v in Z_n

Method:

1. Set hashLen be the length of the output of the hash function Hash
2. Set h_0 be the zero string of length hashLen
3. $h_1 = Hash(h_0 || s)$
4. $B = B_0,...,B_{l-1} = OS2BSP(h_1)$
5. $a_1 = sum_{i = 0}^{l-1} 2^{l-1-i} * B_{i}$
6. $h_2 = Hash(h_1 || s)$
7. $B = B_0,...,B_{l-1} = OS2BSP(h_2)$
8. $a_2 = sum_{i=0}^{l-1} 2^{l-1-i} * B_{i}$
9. $v = 2^{hashLen} * a_1 + a_2 \mod n$
10. Output v

5.2.2. OS2FQE

Octet string should be converted to hashed finite field element as described in this section. Formally, the conversion routine, OS2FQE(s,Hash,F_{p^m}) is defined as follows:

Input:

- s: an octet string
- Hash : a hash function

- o F_{p^m} : a finite field with p^m elements where p is a prime, and m > 0 is an integer

Output:

- o a: an element in F_{p^m}

Method:

1. Set i = 0
2. B = B_{0}, ..., B{31} such that counter = B_{31} + B_{30}*2 + ... + B_{0}*2^{31}
3. Compute C = BS2OSP(B)
4. Compute t_i = IHF1(C||s,p,Hash)
5. If i < m, set i = i + 1 and go back to step2
6. Compute a = sum_{i=0}^{m-1} t_i * beta^i where beta is the variable of the polynomial
7. Output a

5.3. Group Membership Test Function

GROUPMEMBERSHIPTEST(Curve-ID, P) is a test function that an elliptic curve point is on the correct curve and group. GROUPMEMBERSHIPTEST is defined as follows:

Input:

- o Curve-ID : an elliptic curve identifier
- o P = (x,y) : an elliptic curve point

Output:

- o boolean : an integer in {0,1}

Method:

1. If P = O_E, then output 1
2. If y^2 != x^3 + A * x + B, then output 0
3. If h != 1 && r * P != O_E, then output 0

4. Output 1

6. FSU Key Exchange

This section provides the specification of ID-based authenticated key exchange protocol FSU [4] that is an extension of FSU (Fujioka-Suzuki-Ustaoglu) protocol standardized in ISO/IEC11770-3 [5] [6].

6.1. System Parameter Setup

Key Generation Center (KGC) defines the following system parameters in FSU:

- Pairing-Param-ID : An identifier for showing asymmetric pairing. i.e., G1-Curve-ID, G2-Curve-ID and GT-Filed-ID.

- G1-Curve-ID is an identifier for showing an elliptic curve which defines cyclic groups G_1 with prime p_b_1, coefficients A_1 and B_1, generator P_1, order r, and cofactor h_1.

- G2-Curve-ID is an identifier for showing an elliptic curve which defines cyclic groups G_2 with prime p_b_2, irreducible polynomial e2_2, coefficients A_2 and B_2, generator P_2, order r, and cofactor h_2.

- GT-Field-ID is an identifier for showing a pairing co-domain group which is subgroup of order r in G_{phi_12(p)}. G_{phi_12(p)} is the 12-th cyclotomic subgroup of order p^4-p^2+1 in F_{p^12}^*.

- HASH-ID : An identifier for showing a hash function i,e., Hash : {0,1}^* -> {0,1}^hashLen.

- hashLen : Length of output by Hash.

- KDF-ID : An identifier for showing key derivation function, i.e., MGF1: {0,1}^* -> {0,1}^n.

- n : Length of output by key derivation function.

- R : A point compression type of conversion between elliptic curve point and octet string specifically "Compressed", "Uncompressed", or "Hybrid".

KGC generates the master secret key MSK and master public key MPK from system parameters as following.

1. KGC selects a random integer z in Z_r.

2. KGC computes Z_v = z * P_v for v is in {1, 2}.

3. KGC sets MSK = z and MPK = (Z_1, Z_2).

Hash function H_v are defined as H_v(M) = HASHINGTOPOINT(Gv-Curve-ID, Hash, "FSU"||ECP2OSP(Z_1, R)||ECP2OSP(Z_2, R)||M) for v in {1, 2}. Hash function H is defined as H(M) = MGF1("FSU"||ECP2OSP(Z_1, R)||ECP2OSP(Z_2, R)||M, n).

6.2. Key Distribution by KGC

This subsection explains operations of key distribution by KGC. There are two types of static secret key in FSU Key Exchange, respectively static secret key based on cyclic groups in G_1 and in G_2. FSU Key Exchange requires that an initiator and a responder use static secret key with different types, respectively. Hence, KCG needs to define a rule for key distribution for users. For example, clients use static secret keys in G_1 and servers use them in G_2.

KGC generates static secret key D_{i, v} for an identifier ID_i for i in {A, B} of user in G_v as following.

1. Let MPK be (Z_1, Z_2) and MSK be z.

2. KGC Compute D_{i ,v} = z*H_v(ID_i).

3. Distribute D_{i ,v} to a user with ID_i.

6.3. FSU Key Exchange Protocol

This subsection describes FSU Key Exchange Protocol in an initiator U_A with an identifier ID_A and static secret key D_{A,1} and a responder U_B with an identifier ID_B and static secret key D_{B,2}.

Computation of ephemeral public key by U_A

1. U_A selects a random integer x_A in Z_r.

2. U_A computes the ephemeral public key X_{A,v} = x_A * P_v for v in {1,2}.

3. U_A computes XOS_{A,v} = ECP2OSP(X_{A,v}, R) for v in {1,2}.

4. U_A sends (ID_A, ID_B, XOS_{A,1}, XOS_{A,2}) to U_B.

Computation of ephemeral public key by U_B

1. U_B receives (ID_A, ID_B, XOS_{A,1}, XOS_{A,2}).

2. U_B computes X_{A,v} = OS2ECPP(XOS_{A,v}) for v in {1,2}.

3. If (GROUPMEMBERSHIPTEST(G1-Curve-ID, X_{A,1}) = 0 || GROUPMEMBERSHIPTEST(G2-Curve-ID, X_{A,2}) = 0 || e(X_{A,1}, P_2) != e(P_1, X_{A,2})), then abort.

4. U_B selects a random ephemeral secret key x_B in Z_r.

5. U_B computes the ephemeral public key X_{B,v} = x_B * P_v for v in {1,2}.

6. U_B computes XOS_{B,v} = ECP2OSP(X_{B,v}, R) for v in {1,2}.

7. U_B sends (ID_B, ID_A, XOS_{B,1}, XOS_{B,2}) to U_A.

Computation of session key by U_B

1. U_B computes sigma_1 = e(H_1(ID_A), D_{B,2}).

2. U_B computes sigma_2 = e(H_1(ID_A) + X_{A,1}, D_{B,2} + x_B * Z_2).

3. U_B computes sigma_3 = x_B * X_{A,1}.

4. U_B computes sigma_4 = x_B * X_{A,2}.

5. U_B computes sigmaOS_j = FE2OSP(sigma_j) for j in {1,2}.

6. U_B computes sigmaOS_j’ = ECP2OSP(simga_j’,R) for j’ in {3,4}.

7. Set sid = (ID_A||ID_B||XOS_{A,1}||XOS_{A,2}||XOS_{B,1}||XOS_{B,2}).

8. U_B computes session key K = H(sigmaOS_1||sigmaOS_2||sigmaOS_3||sigmaOS_4||sid).

Computation of session key by U_A

1. U_A computes X_{B,v} = OS2ECPP(XOS_{B,v}) for v in {1,2}.

2. If (GROUPMEMBERSHIPTEST(G1-Curve-ID, X_{B,1}) = 0 || GROUPMEMBERSHIPTEST(G2-Curve-ID, X_{B, 2}) = 0 || e(X_{B,1}, P_2) != e(P_1, X_{B,2})), then abort.

3. U_A computes sigma_1 = e(D_{A,1}, H_2(ID_B)).

4. U_A computes sigma_2 = e(D_{A,1} + x_A * Z_1, H_2(ID_B) + X_{B,2}).

5. U_A computes sigma_3 = x_A * X_{B,1}.

6. U_A computes sigma_4 = x_A * X_{B,2}.

7. U_A computes sigmaOS_j = FE2OSP(sigma_j) for j in {1,2}.

8. U_A computes sigmaOS_j’ = ECP2OSP(simga_j’,R) for j’ in {3,4}.

9. Set sid = (ID_A||ID_B||XOS_{A,1}||XOS_{A,2}||XOS_{B,1}||XOS_{B,2}).

10. U_A compute session key K = H(sigmaOS_1||sigmaOS_2||sigmaOS_3||sigmaOS_4||sid).

7. Security Considerations

This memo specifies identity-based authenticated key exchange protocol FSU [4] [6] [5] which is secure in the id-eCK(id-based extended Canetti-Krawczyk) security model under the GBDH(gap bilinear DH) assumption [4].

id-eCK security model is the most strong security model in the meaning of that it ensures the safety of session key if any non-trivial combinations of master key, static key, and ephemeral key are leaked.

And id-eCK security model guarantees following 4 security notions:

MitM(resistance to man in the middle attacks),

wPFS(weak perfect forward security),

KCI(resistance to key compromise impersonation attacks),

RLE(resilience to leakage of ephemeral private keys).

8. Acknowledgements

TBD

9. Algorithm Identifiers

TBD

10. Change log

NOTE TO RFC EDITOR: Please remove this section in before final RFC publication.

11. Test Vectors

TBD

12. References

12.1. Normative References

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

[2] Kasamatsu, K., Kanno, S., Kato, A., Scott, M., Kobayashi, T., and Y. Kawahara, "Barreto-Naehrig Curves", draft-kasamatsu-bncurves-02 (work in progress), 2015.

[3] Kato, A., Scott, M., Kobayashi, T., and Y. Kawahara, "Barreto-Naehrig Curves", draft-kato-optimal-ate-pairings-00 (work in progress), 2015.

12.2. Informative References

[4] Fujioka, A., Hoshino, F., Kobayashi, T., Suzuki, K., Ustaglu, B., and K. Yoneyama, "id-eCK Secure ID-Based Authenticated Key Exchange on Symmetric and Asymmetric Pairing", Proceedings IEICE Transactions 96-A(6): 1139-1155, 2013.

[5] Fujioka, A., Suzuki, K., and B. Ustaglu, "Ephemeral Key Leakage Resilient and Efficient ID-AKEs That Can Share Identities, Private and Master Keys", Proceedings Pairing 2010 Lecture Notes in Computer Science Volume 6487, pp 187-205, 2010.

[6] "Information technology -- Security techniques -- Key management -- Part 3: Mechanisms using asymmetric techniques.", ISO/IEC 11770-3: 2015, 2015.

[7] Shamir, A., "Identity-based Cryptosystems and Signature Schemes", Proceedings CRYPTO ’84, LNCS 196, pages 47-53, Springer-Verlag, 1984.

[8] Boneh, D. and M. Franklin, "Identity-Based Encryption from the Weil Pairing", Proceedings CRYPTO 2001, LNCS 2139, pages 213-229, Springer-Verlag, 2001.

[9] Huang, H. and Z. Cao, "An ID-based Authenticated Key Exchange Protocol Based on Bilinear Diffie-Hellman Problem", Proceedings the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS ’09) pp. 333-342, ACM, 2009.

[10] Canetti, R. and H. Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", Proceedings Eurocrypt 2001 (LNCS2015), pp. 453-474, Springer-Verlag, 2001.

[11] LaMacchia, B., Lauter, K., and A. Mityagin, "Stronger Security of Authenticated Key Exchange", Proceedings in Provable Security (LNCS 4784), pp. 1-16, Springer, 2007.

Appendix A. Construction of Data Conversion

A.1. Construction of BS2OSP

Concrete construction of BS2OSP(B) is specified as follows:

Input:

- B = B_0 B_1 ... B_{l-1} : a bit string of length l

Output:

- M = M_0 M_1 ... M_{n-1}: an octet string of length n = ceil(l/8).

Method:

1. If l = 0, then output empty octet string and stop.
2. For j in {0,...,8n-1}, if j >= 8n - l, set B’_j = B_{j-(8n-l)}, otherwise set B’_j = 0.
3. For i in {0,...,n-1}, set M_i = B’_{8i} B’_{8i+1}...B’_{8i+7}.
4. Output M = M_0 M_1 ... M_{n-1}.

A.2. Construction of OS2BSP

Concrete construction of OS2BSP(M) is specified as follows:

Input:

- M = M_0M_1...M{n-1}: an octet string of length n.

Output:

- B = B_0B_1...B_{l-1} : a bit string of lenth l = 8*n

Method:

1. If l = 0, then output empty octet string and stop.
2. For i in {0, ..., n-1} , j in {0,...,7}, set B_{8i + j} in {0,1} as M_i = B_{8i} B_{8i+1}...B_{8i+7}.
3. Output B = B_0 B_1 ... B_{l-1}.

A.3. Construction of FE2IP

Concrete construction of FE2IP(a) is specified as follows:

System parameters:

- F_{p^{m_2}}/F_{p^{m_1}}: a field extension with an irreducible polynomial Irr(F_{p^m_2} / F_{p^m_1};beta)

Input:

- a : a field element in F_{p^m_2}

Output:

- x : an integer in {0,...,p^{m_2} - 1}

Method:

1. If m_2 = 1 (i.e. F_{p^m_2} is prime field)

   A field element of F_{p^{m_2}} must be represented an an integer in {0, ..., p-1}

   (A) Set x = a

   (B) Output x

2. Else (i.e. m_2 > 1)

   (A) Let the coefficients a_i in F_{p^m_1} for i in {0,...,m_2 / m_1 - 1} such that a = sum_{i=0}^{m_2 / m_1 - 1} a_i * beta^i

   (B) Compute x = sum_{i=0}^{m_2 / m_1 - 1} FE2IP(a_i) * (p^{m_1})^i

   (C) Output x

A.4. Construction of I2FEP

Concrete construction of I2FEP(x) is specified as follows:

System parameters:

- F_{p^{m_2}}/F_{p^{m_1}}: a field extension with an irreducible polynomial Irr(F_{p^m_2} / F_{p^m_1};beta)

Input:

- o x : an integer in {0,...,p^{m_2} - 1}

Output:

- o a : a field element in F_{p^m_2}

Method:

```
1.  If m_2 = 1 (i.e.  F_{p^m_2} is prime field)

        A field element of F_{p^{m_2}} must be represented an an
        integer in {0, ..., p-1}

        (A) Set a = x

        (B) Output a

2.  Else (i.e. m_2 > 1)

        (A) Let x_i be an element in {0,...,p^{m_1}-1} for i in
        {0,...,m_2 / m_1 - 1} such that x = sum_{i=0}^{m_2 / m_1 -1}
        x_i * p^{m_1}^i

        (B) Compute a = sum_{i=0}^{m_2 / m_1 - 1} I2FEP(x_i) * beta^i

        (C) Output a
```

A.5. Construction of FE2OSP

System parameter:

- o F_{p^m} : a finite field with p^m elements where p is a prime, and m > 0 is an integer
- o n : an integer equivalent to ceil(m * log_2 p / 8)

Input:

- o a : a field element in F_{p^m}

Output:

- o M : an octet string

Method:

1. Compute I = FE2IP(a)

2. Compute X = x_{0}, ..., X_{n-1} such that I = x_{n-1} + x_{n-2}*2 + ... + x_{1}*2^{n-2} + x_{0}*2^{n-1}

3. Compute M = BS2OSP(X)

4. Output M

A.6. Construction of OS2FEP

System parameter:

- F_{p^m} : a finite field with p^m elements where p is a prime, and m > 0 is an integer

- n : an integer equivalent to ceil(m * log_2 p / 8)

Input:

- M : an octet string

Output:

- a : a field element in F_{p^m}

Method:

1. Compute X = OS2BSP(M)

2. Let X be x_0,...,x_{l-1}

3. Compute I = sum_{i=0}^{l-1} 2^{l-1-i} * x_{i}

4. Compute a = I2FEP(I)

5. Output a

A.7. Construction of ECP2OSP

Concrete construction of ECP2OSP(P,R), is specified as follows:

System parameters:

- Curve-ID : an elliptic curve parameter

Input:

- o P : a point on an elliptic curve over F_{p^m}
- o R : compression type specifically "Compressed", "Uncompressed", or "Hybrid"

Output:

- o M : an octet string of length n

Method:

```
1.  If P = O_E

        (A) Compute M = BS2OSP(00000000)

        (B) Output M

2.  If P = (x,y) != O_E && R = Compressed

        (A) Set X = FE2OSP(x)

        (B) If p is odd && y = 0 , set y’ = 0

        (C) Else if p is odd && y != 0, set y’ = y_i mod 2 such that y
        = y_{m-1} * beta^{m-1} + ... + y_1 * beta + y_0 and i is the
        smallest integer such that y_i != 0

        (D) If y’ = 0, compute L = BS2OSP(00000100)

        (E) If y’ = 1, compute L = BS2OSP(00000101)

        (F) Output M = L || X

3.  If P = (x,y) != O_E && R = Uncompressed

        (A) Set X = FE2OSP(x)

        (B) Set Y = FE2OSP(y)

        (C) Compute L = BS2OSP(00000100)

        (D) Output M = L || X || Y

4.  If P = (x,y) != O_E && R = Hybrid

        (A) Set X = FE2OSP(x)

        (B) Set Y = FE2OSP(y)
```

    (C) If y = 0, set y’ = 0

    (D) Else (i.e. y != 0) y’ = y_i mod 2 such that y = y_{m-1} * beta^{m-1} + ... + y_1 * beta + y_0 and i is the smallest integer such that y_i != 0

    (E) If y’ = 0, compute L = BS2OSP(00000110)

    (F) If y’ = 1, compute L = BS2OSP(00000111)

    (G) Output M = L || X || Y

A.8. Construction of OS2ECPP

Concrete construction of OS2ECPP(M), is specified as follows:

System parameters

- Curve-ID : an elliptic curve parameter

Input:

- M : an octet string

Output:

- P : an elliptic curve point

Method:

1. If M = BS2OSP(00000000), output P = O_E

2. If M has length ceil(m * log_2 p / 8) + 1

    (A) Let M be L||X where L is a single octet

    (B) Compute x = OS2FEP(X)

    (C) If L = BS2OSP(00000010), then set y’ = 0

    (D) Else if L = BS2OSP(00000011), then set y’ = 1

    (E) Else output INVALID and stop

    (F) Compute w = x^3 + A * x + B

    (G) Compute gamma = square(w)

(H) If there is no gamma in F_{p^m}, then output INVALID and stop

(I) Else if gamma = 0, then set y = 0

(J) Else if gamma_i = y’ mod 2 where gamma = gamma_{m-1} * beta^{m-1} + ... + gamma_{1} * beta + gamma_{0} and i is the smallest integer such that gamma_i != 0

(K) Else if gamma_i != y’ mod 2, set y = -gamma where gamma = gamma_{m-1} * beta^{m-1} + ... + gamma_{1} * beta + gamma_{0} and i is the smallest integer such that gamma_i != 0

(L) Output P = (x,y)

3. If M has length 2 * floor(m * log_2 p / 8) + 1

(A) Let M be L || X || Y where L is a single octet, X is floor(m * log_2 p / 8) octets, and Y is floor(m * log_2 p / 8) octets

(B) Unless L is BS2OSP(00000100), BS2OSP(00000110) or BS2OSP(00000111), output INVALID and stop.

(a) Compute x = OS2FEP(X)

(b) Compute y = OS2FEP(Y)

(c) If (x,y) does not satisfy the equation of elliptic curve, then output INVALID and stop

(d) Output P = (x,y)

Authors’ Addresses

Akihiro Kato
NTT Software Corporation

EMail: kato.akihiro-at-po.ntts.co.jp

Thomas Hardjono
MIT

EMail: hardjono-at-mit.edu

Tetsutaro Kobayashi
NTT

EMail: kobayashi.tetsutaro-at-lab.ntt.co.jp

Tsunekazu Saito
NTT

EMail: saito.tsunekazu-at-lab.ntt.co.jp

Koutarou Suzuki
NTT

EMail: suzuki.koutarou-at-lab.ntt.co.jp

Network Working Group A. Kato
Internet-Draft NTT Software Corporation
Intended status: Informational M. Scott
Expires: January 7, 2016 CertiVox
T. Kobayashi
Y. Kawahara
NTT
July 6, 2015

Optimal Ate Pairing
draft-kato-optimal-ate-pairings-00

Abstract

Pairing is a special map from two elliptic curve that called Pairing-friend curves to a finite field and is useful mathematical tools for constructing cryptographic primitives. It allows us to construct powerful primitives. (e.g. [3] and [4])

There are some types of pairing and its choice has an impact on the performance of the primitive. For example, Tate Pairing [3] and Ate Pairing [4] are specified in IETF. This memo focuses on Optimal Ate Pairing [2] which is an improvement of Ate Pairing.

This memo defines Optimal Ate Pairing for any pairing-friendly curve. We can obtain concrete algorithm by deciding parameters and building blocks based on the form of a curve and the description in this memo. It enables us to reduce the cost for specifying Optimal Ate Pairing over additional curves. Furthermore, this memo provides concrete algorithm for Optimal Ate Pairing over BN-curves [7] and its test vectors.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Pairing is a special map from two elliptic curve that called Pairing-friend curves (PFCs) to a finite field and is useful mathematical tools for constructing cryptographic primitives. It allows us to construct powerful primitives like Identity-Based Encryption (IBE) [5] and Functional Encryption (FE) [6]. The IBE and FE provide a rich decryption condition. Some Pairing-Based Cryptography is specified in IETF. (e.g. [3] and [4])

There are some types of pairing and its choice has an impact on the performance of the primitive. For example, primitives by using Tate Pairing [3] and Ate Pairing [4] are specified in IETF. This memo focuses on Optimal Ate Pairing which is an improvement of Ate Pairing. Optimal Ate Pairing allows us to construct Pairing-Based Cryptography with high performance and is implemented in some open source softwares. ([8], [9], and [10])

This memo defines Optimal Ate Pairing [2] for any PFC. We can obtain concrete algorithm by deciding parameters and two building blocks based on the form of a curve. It enables us to reduce the cost for describing the body of Optimal Ate Pairing when Optimal Ate Pairing is specified over additional curves in IETF. Furthermore, this memo provides concrete algorithm for Optimal Ate Pairing over BN-curves [7] and its test vectors. This memo is expected to use by combining Optimal Ate Pairing with a suitable PFC for a primitive in order to realize same functional structure of ECDSA and ECDH. (i.e. DSA over elliptic curve and DH over elliptic curve)

2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this memo are to be interpreted as described in [1].

3. Preliminaries

In this section, we introduce the definition of elliptic curve and bilinear map, notation used in this memo.

3.1. Elliptic Curve

Throughout this memo, let p > 3 be a prime, q = p^n, and n be a natural number. Also, let F_q be a finite field. The curve defined by the following equation E is called an elliptic curve.

```
E : y^2 = x^3 + A * x + B such that A, B are in F_q,
          4 * A^3 + 27 * B^2 != 0 mod F_q
```

Solutions (x, y) for an elliptic curve E, as well as the point at infinity, are called F_q-rational points. The additive group is constructed by a well-defined operation in the set of F_q-rational points. Typically, the cyclic additive group with prime order r and the base point G in its group is used for the cryptographic applications. Furthermore, we define terminology used in this memo as follows.

O_E: the point at infinity over elliptic curve E.

#E(F_q): number of points on an elliptic curve E over F_q.

cofactor h: h = #E(F_p)/r.

embedding degree k: minimum integer k such that r is a divisor of q^k - 1

3.2. Bilinear Map

Let G_1 be an additive group of prime order r and let G_2 and G_T be additive and multiplicative groups, respectively, of the same order. Let P, Q be generators of G_1, G_2 respectively. We say that (G_1, G_2, G_T) are asymmetric bilinear map groups if there exists a bilinear map e: (G_1, G_2) -> G_T satisfying the following properties:

1. Bilinearity: for any S in G_1, for any T in G_2, for any a, b in Z_r, we have the relation e([a]S, [b]T) = e(S, T)^{a * b}.

2. Non-degeneracy: for any T in G_2, e(S, T) = 1 if and only if S = O_E. Similarly, for any S in G_1, e(S, T) = 1 if and only if T = O_E.

3. Computability: for any S in G_1, for any T in G_2, the bilinear map is efficiently computable.

4. Optimal Ate Pairing

This section specifies Optimal Ate Pairing e for c_0, ..., c_l and s_i = sum_{j=i}^l c_j * q^j with following conditions

1. c_l is not 0

2. r is a divisor of s_0

3. r^2 is not a divisor of s_0

4. r does not divide s_0 * k * q^{k-1} - (q^k - 1)/r * sum_{i=0}^l i * c_i * q^{i - 1}

Section 4.1 shows a guide to decide these parameters c_0, ..., c_l. Optimal Ate Pairing is specified below and Miller Loop f which are its building blocks are introduced in Section 4.2. Straight Line Function l which is building blocks of Optimal Ate Pairing and Miller Loop are defined in Section 4.3. Section 4.3 only show the definitions because its descriptions are based on the form (of the PFC?). Practically, concrete algorithms need to be specified for a form of PFC.

Input:

- A point P in G_1
- A point Q in G_2

Output:

- The value e(P, Q) in G_T

Method:

```
1.  f = 1

2.  ln = 1

3.  for i = 0 to l

       (a) f = f * f_{c_i, Q}^{q^i}(P)

    end for

4.  for i = 0 to l - 1

       (a) ln = ln * l_{[s_i + 1]Q, [c_i * q^i]Q}(P)

    end for

5.  return (f * ln)^{(q^k - 1)/r}
```

4.1. Guide for Decision on Parameters for Optimal Ate Pairing

This subsection shows a guide for decision on parameters c_0, ..., c_l for Optimal Ate Pairing. According to [2], a way is to choice coefficients of short vector of the following lattice L with a minimal number of coefficients as parameters c_0, ..., c_l.

L = (v_1, ..., v_phi(k)) where

- v_1 is column vector t(r, -q, -q^2, ..., -q^{phi(k) - 1})
- v_i is column vector whose i component is 1 and other components is 0 for i = 2, ..., phi(k)

4.2. Miller Loop

In this subsection, we specify Miller Loop f which is building block of Optimal Ate Pairing.

Input:

- A point P in G_1
- A point Q in G_2
- An integer s

Output:

- f_{s, Q}(P)

Method:

```
1.  compute s_0, ..., s_L such that |s| = sum_{j=0}^L s_j * 2^j with
    s_j is in {0, 1} and s_L = 1

2.  T = Q

3.  f = 1

4.  for j = L - 1 down to 0

       (A) Doubling Step

          (a) ln = l_{T, T}(P)

          (b) T = 2 * T

       (B) f = f^2 * ln

       (C) if s_j = 1

          (a) Addition Step

             (i) ln = l_{T, Q}(P)
```

```
                (ii) T = T + Q

             (b) f = f' * ln

          end if

       end for

   5.  if s < 0, then f = f^{-1}

   6.  return f
```

4.3. Straight Line Function

Straight Line Function l_{Q, Q'}(P) is calculated by a point P for linear equation defined as a line l though points Q, Q'. Note that Straight Line Function l_{Q, Q'}(P) is calculated by a point P for linear equation defined as a tangent line to an elliptic curve E at a point Q of E on condition that Q = Q'. The function is used for Optimal Ate Pairing in Section 4 and Miller Loop in Section 4.2

5. Optimal Ate Pairing over BN-curves

In this section, we specify Optimal Ate Pairing over BN-curves [7]. BN-curves define over a finite field F_p, and have embedding degree k = 12, r(t) = 36 * t^4 + 36 * t^3 + 18 * t^2 + 6 * t + 1, and p(t) = 36 * t^4 + 36 * t^3 + 24 * t^2 + 6 * t + 1, where t is the specific integer in [7].

The extension fields are defined by following:

F_{p^2} is set to F_p[u]/(u^2 - e2)

F_{p^6} is set to F_{p^2}[v]/(u^3 - e6)

F_{p^12} is set to F_{p^6}[w]/(w^2 - e12)

The constants e3, e6 and e6 which are varied by G_T are defined in [7].

Hence parameters for Optimal Ate Pairing over D-Type twisted curve are following by the method in Section 4.1:

1. l = 3

2. c_0 = 6 * t + 2

3. c_1 = 1

4. c_2 = -1

5. c_3 = 1

These short vectors are specified in section 4. A of [2].

Algorithm of Optimal Ate Pairing by Miller Loop in Section 4.2 based on building blocks specified in Section 5.2 and Section 5.3 and Straight Line Function f in Section 5.1 over BN-curves is as following:

Input:

- A point P in G_1

- A point Q in G_2

Output:

- The value e(P, Q) in G_T

Method:

1. f_1 = f_{c_0, Q}(P)

2. l_1 = l_{[p^3]Q}, - [p^2]Q}(P)

3. l_2 = l_{[p^3]Q - [p^2]Q, [p]Q}(P)

4. l_3 = l_{[p]Q - [p^2]Q + [p^3]Q, [6 * t + 2]Q}

5. return (f_1 * l_1 * l_2 * l_3)^{(p^k - 1)/r}

5.1. Straight Line Function over BN-curves

This subsection shows an operation of Straight Line Function over BN-curves for Optimal Ate Pairing.

Input:

- A point Q = (x_1, y_1) in G_2

- A point Q’ = (x_2, y_2) in G_2

- A point P = (x, y) in G_1

Output:

- l_{Q, Q’}(P)

Method:

```
1.  If Q != +- Q’

        (A) lambda = (y_2 - y_1)/(x_2 - x_1)

        (B) t0 = -lambda * x

        (C) t1 = lambda * x_1 - y_1

        (D) ln = y + t0 * w + t1 w^3

2.  If Q = Q’

        (A) lambda = (3 * x_1^2)/(2 * y_1)

        (B) t0 = -lambda * x

        (C) t1 = lambda * x_1 - y_1

        (D) ln = y + t0 w + t1 w^3

        (E) return ln

3.  If Q = -Q’

        (A) ln = x - x_1 w^3

4.  return ln
```

5.2. Doubling Step of Miller Loop over BN-Curves

This subsection shows an operation of Doubling Step of Miller Loop over BN-curves. (i.e. operation of method 4-(A) in Section 4.2 over BN-curves)

Input:

- A point P = (x, y) in G_1

- A point Q = (x_1, y_1) in G_2

Output:

- ln such that l_{Q, Q}(P)

- o A point T = (x_3, y_3) such that [2]Q

Method:

1. lambda = (3 * x_1^2)/(2 * y_1)
2. x_3 = lambda^2 - 2 * x_1
3. y_3 = lambda * (x_1 - x_3) - y_1
4. t0 = -lambda * x
5. t1 = lambda * x_1 - y_1
6. ln = y + t0 w + t1 w^3
7. return ln and T

5.3. Addition Step of Miller Loop over BN-Curves

This subsection shows an operation of Addition Step of Miller Loop over BN-curves. (i.e. operation of method 4-(C)-(a) in Section 4.2 over BN-curves)

Input:

- o A point Q = (x_1, y_1) in G_2
- o A point Q’ = (x_2, y_2) in G_2
- o A point P = (x, y) in G_1

Output:

- o ln such that l_{Q, Q’}(P)
- o A point T = (x_3, y_3) such that Q + Q’

Method:

1. lambda = (y_2 - y_1)/(x_2 - x_1)
2. x_3 = lambda^2 - x_1 - x_2
3. y_3 = lambda * (x_1 - x_3) - y_1
4. t0 = -lambda * x

```
5.  t1 = lambda * x_1 - y_1

6.  ln = y + t0 w + t1 w^3

7.  return ln and T
```

6. Algorithm Identifiers

TBD

7. Security Considerations

The security of cryptographic primitive which is constructed by pairing depends on pairing-friendly curves (PFC). PFC must satisfy computational assumption which the primitive requires at the level of security strength in system when the primitive is constructed by using Optimal Ate Pairing.

8. Acknowledgements

TBD

9. Change log

NOTE TO RFC EDITOR: Please remove this section in before final RFC publication.

10. References

10.1. Normative References

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

[2] Vercauteren, F., "Optimal pairings", Proceedings IEEE Transactions on Information Theory 56(1): 455-461 (2010), 2010.

10.2. Informative References

[3] Boyen, X. and l. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, December 2007.

[4] Hitt, L., "ZSS Short Signature Scheme for Supersingular and BN Curves", draft-irtf-cfrg-zss-02 (work in progress), 2013.

[5] Boneh, D. and M. Franklin, "Identity-based encryption from the Weil pairing", Proceedings Lecture notes in computer sciences; CRYPTO --CRYPTO2001, 2001.

[6] Okamoto, T. and K. Takashima, "Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption", Proceedings Lecture notes in computer sciences; CRYPTO --CRYPTO2011, 2010.

[7] Kasamatsu, K., Kanno, S., Kobayashi, T., and Y. Kawahara, "Barreto-Naehrig Curves", draft-kasamatsu-bncurves-01 (work in progress), 2015.

[8] "University of Tsukuba Elliptic Curve and Pairing Library", 2013, <http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html>.

[9] Aranha, D. and C. Gouv, "RELIC is an Efficient LIbrary for Cryptography", 2013, <https://code.google.com/p/relic-toolkit/>.

[10] Scott, M., "The MIRACL IoT Multi-Lingual Crypto Library", 2015, <https://github.com/CertiVox/MiotCL.git>.

Appendix A. Test Vectors of Optimal Ate Pairing over BN-curves

In this section, we specify test vectors of optimal ate pairing over BN-curves which are specified by [7] in the following way.

Parameter:

Pairing-Param-ID is an identifier with which the pairing parameter set can be referenced.

Input:

P is a point of E in G_1

Q is a point of E’ in G_2

Output:

e(P, Q) is computation of pairing in G_T

A.1. 254-Bit-Curves by Beuchat et al.

This subsection shows test vector of 254-bit curves by Beuchat et al. [7] and reprints its parameters under F_{p^2} = F_p[u]/(u^2 + 5), F_{p^6} = F_{p^2}[v]/(v^3 - u), F_{p^12} = F_{p^6}[w]/(w^2 - v) as a reference.

Parameter:

Pairing-Param-ID: Beuchat

Input:

P = (0x0A971735A70FBDD0F94D7D6EFBBC81BEA78D2D92A8510F3344038A41641
9AD97, 0x09456E41754237447752A448282C0873785F724447E1299826F53AC55
6936D3F)

Q = (0x115231D7B49901BA97CB93B5227F7F7F438A346532893DD5FAFD5189509
24AA9 + 0x0DF12398FB78695A50BB3499B7E23B0D9035989B91A76D13AF7BC643
74BFB8A6 u, 0x051D0E087527BC9F41379FB0272EC91E5F28EE011B183EF7D671
2EF3FC9A1A66 + 0x0107E6654DC6C36E163B7867AECB98E4046084734524DBB56
2E73E5A811F678A u)

Output:

e(P,Q) = (0x06A4E0DD1F7FD2F9E5DACAB02CEC9CE8254925C5DC6697E153F05A
242CBCA8A8 + 0x22A0E22C097AEC1187087B7632C9B963B0E779BC8D09848C44D
3EA95CD1C1F8C u + 0x0751037182B5F93BCAB31B115A2C0A0DCC09C6DB7602E0

```
551DD44925F3D364B3 v + 0x04B6BFFB9EB68AD6A99ACF52B8AAD1D17D328847C
6313201A6B659C9DAA5CDFE uv + 0x13BE65D47487BF6D96C146C18855C1F87BF
994F9F1048524568EA0CB9DC402AD v^2 + 0x1202BE31EB2BDCBEF9F3CC00F1B2
CC35FADBE1A0D66CCBF40B024ADFA84C77D1 uv^2 + 0x15F9E3D10B580FF1AB22
82EF1DC39A88E06F93A18303E9520D99B86D665F5380 w + 0x0A1C6D26A6D6830
31D95C4369DB90F5FEE36D5008AA498D2CB6F2DDE6258CDA6 uw + 0x1611153BF
02F1CF7985B98C3F3CB641D39283DBA55E22D1C614568F84959C6FC vw + 0x10B
EF55B7539743CBEAB13E49116A143302F6F28CCD71A69860CEF5208483809 uvw
+ 0x166BD873D0C65DE66300A168BBDC16F0AB1B57A0809973239F2109A7D25AD3
49 v^2w + 0x14D4B5014F840144D03C0C6B6010BB246EE6A69BF704D7542FBAA8
F2D2A27308 uv^2w)
```

A.2. 254-Bit-Curves by Nogami et al. / Aranha et al.

This subsection shows test vector of 254-bit curves by Nogami et al. / Aranha et al. [7] and reprints its parameters under F_{p^2} = F_p[u]/(u^2 + 1), F_{p^6} = F_{p^2}[v]/(v^3 - (1 + u)), F_{p^12} = F_{p^6}[w]/(w^2 - v) as a reference.

Parameter:

Pairing-Param-ID: Nogami-Aranha

Input:

```
P = (0x2074A81D4402A0B63B947335C14B2FC3C28FEA2973860F686114BEC4670
E4EB7, 0x06A41108087B20038771FC89FB94A82B2006034A6E8D871B3BC284846
631CBEB)
```

```
Q = (0x049EEDB108B71A87BFCFC9B65EB5CF1C2F89554E02DF4F8354E4A00F521
83C77 + 0x1FB93AB676140E87D97226185BA05BF5EC088A9CC76D966697CFB8FA
9AA8845D u, 0x0CD04A1ED14AD3CDF6A1FE4453DA2BB9E686A637FB3FF8E25736
44CC1EDF208A + 0x11FF7795CF59D1A1A7D6EE3C3C2DFC765DEF1CAA9F14EA264
E71BD7630A43C14 u)
```

Output:

```
e(P,Q) = (0x03E1F2693AC6D549898C78897EB158490A4832E296F888D3014050
0DB7BD3D12 + 0x1EBC54A76E844EB5D352945226FB103DE9EC1A4FC689B87FAA6
6EF8ABA79D3ED u + 0x0A5A5405542F67384D683A48C281F3676B67554ED5DA17
00784169A0B47A57E4 v + 0x048B66DAFCAEE86DB4D46AB71A9FE848443EF81F4
88D8366A727B39698CF7201 uv + 0x142715D6482BC6FA77377C9CBC2A51C047C
16DE88483D5A889C7EF4DF5F03BDB v^2 + 0x11EE0C12164133041C3DCF312CE1
11C845B60092818F7B72805D4AFF61427934 uv^2 + 0x22371AF975DAE562F686
988CDBBD02702C959BBF843A1FB3C7532D07BE3D7A3A w + 0x04052CA96090068
4A1B26C434B2776AA70736841474C16208CCD1A7C27927E19 uw + 0x05D259DA3
F3AAAA54A6AE5FE8272A5B79D7F4E5BDF3B5E3C815AD781113F7548 vw + 0x084
3C37BC5BDBF253E3BCE568F5905A63867D8836855B74CBA0C800D5DC41B71 uvw
```

```
+ 0x13CA93E1377EF0F6DD38FC2F96DBD3E8B0922F60D1F274EAC63DC1AF2EE975
4C v^2w + 0x0D467F3DA4FB329A5CB406D0A7B743A3A2FFCD09BF95EE8A856B94
AF191D96AF uv^2w)
```

A.3. 254-Bit-Curves by Scott

This subsection shows test vector of 254-bit curves by Scott [7] and reprints its parameters under F_{p^2} = F_p[u]/(u^2 + 1), F_{p^6} = F_{p^2}[v]/(v^3 - (1 + u)), F_{p^12} = F_{p^6}[w]/(w^2 - v) as a reference.

Parameter:

Pairing-Param-ID: Scott

Input:

```
P = (0x8a9143801f541142f89e498a1c06ba0959b8f9713abda0881e5de80d8af
f11a + 0x17df54e2be5e8afeb9a42f412825f79c32841307471fb2b6a14e3a0f
c6e010f4)
```

```
Q = (0x21794a9da7b34b2c1614315d7d90a282c484c8fd49c0c8ba75b079ae304
7d566 + 0x1a9b474c4519e6faee5b32c7cb65547d8707137bca00c9c182d10b7e
3e305936 u, 0xb00d54bf5a298d0eacdefb0efdb74d1a7e744722f61cc8844884
fcce20ff876 + 0x5ecf8bd02e1f5363c8402163c9a235df56b133cc2c8a926c0e
65e985d746b7b u)
```

Output:

```
e(P,Q) = (0x13d3127ba07feffc8c1a608afc58a33a25148176968ef0ec0a2e09
b62344f984 + 0x1774dfc7361e1d4cd2de4bf62cd9b460f0a78487e75994f9e25
51fed2f9d2b78 u + 0x2c7888f053123b5a815125b2c409e3f986594f6c35585c
fb1ed1a1cbbd2ea65 v + 0xe7e7af51c459f6e0ef489348664bc4277e023a5031
bee98658d5b357c07d7e8 uv + 0x8d0f0dd32f31d3624dd9e179233a1f2f2d13c
c1869f2eb933cd3cded75efe0d v^2 + 0x63e676f8cc5be53e8718cc9e61a8c5a
018ac47e3a66f83f4c403ec8caaa130e v^2u + 0x1643c6ec6cf54a1970bfea19
c55e34a312eb5c825f8d31354200d29339d2ca61 w + 0xaae41d356d24b0234dc
2b714b595aa297f585bbe9a7c4840d58d62cdfaa1764 wu+ 0x1ea5e2efa342adc
bc3ac757254d03bfde32ef6a8445bfa6a7b13aee776430594 wv + 0x3aa5bc92f
95887ce42ef03e666dd1455d640a031b062ed7a65fbf0a59d996b8 wvu + 0xf77
35a9655207b2fe6e8e73d8f8c3f79f8a08aaeb670e6b9059d8f0739891ec wv^2
+ 0x1a501fad47a0406e50b705a544377ee1ad7518adbbb49cbe30ce31770ae9be
2e wv^2u)
```

A.4. 254-Bit-Curves by BCMNPZ

This subsection shows test vector of 254-bit curves by BCMNPZ [7] and reprints its parameters under F_{p^2} = F_p[u]/(u^2 + 1), F_{p^6} = F_{p^2}[v]/(v^3 - (1 + u)), F_{p^12} = F_{p^6}[w]/(w^2 - v) as a reference.

Parameter:

Pairing-Param-ID: BCMNPZ

Input:

```
P = (0x1bec8eae1f1d3959e394588e49d09f2d3070efda1f836640288cf21af54
88765 + 0x2d148d39f9edf5325d9a1f4820774930675669a6fe20284e435f4bfe
3d3273c)

Q = (0xd62cf33cd0e46fdc338cfab52ca5cdebf1a9348e4460545441584ff4f8d
c275 + 0x22701025e0cd2bfed4518febe8e7fa97a3c7f33f2fdd280e24d651be9
d17d7a8, 0x1cc6cbd065535e7f83be0cfc4f39d4687558fc21dcdc6e46aca508c
4f6cc1f90 + 86ee46779f9e9922a870137d033e484ec5c5ba979b75bba179064a
bff0cf2a u)
```

Output:

```
e(P,Q) = (0x20f263ae42e42cfd53cf99dc238ed7b61951c1c767af88a72ad3c1
9ca54cdb2d + 0xa96b263aade3501f7201808028c4ce11793dd84127d80525fa5
7f892d3043f6 u + 0x3a31ca4864d996d64181d9a0b025e7368d60b1f53a8276a
2c39e02a58b6636e v + 0x2301fe7eb607f6dd63b72979753c96d23fdd487f116
77644884f86a83c837174 uv + 0xcbe52ab6e1c210cf80215816f38d8964c4534
7bd3802c66d85e616ca9786dde v^2 + 0x1c039dee75146d8ae6812568e77d11c
fa060d11e0224dc6e28606bfb14090650 v^2u + 0x2344fb2b5dd57710d544583
83cd33bd8f928babfe6f7d641887a565790b88e24 w + 0x8e48a543c2a73cca42
811a2fea2e79eb3e628e27e54a477b5e1652466629608 wu+ 0x96a48564f586e1
d59d8a9393730824b885818e93a3ce4bfae057682efc37aeb wv + 0x17260fa31
ed89d4e90d7a1a2652379e4329927e61f15b11a2ce2a93c84050245 wvu + 0x5b
d893369435b63a10384db8248dab8908f2173e166129d0cccd6d37c89dce6 wv^2
+ 0x2a4dec6bbfe98df2c9169b06410c329d4c699747ca649e611d9960416d615
b5 wv^2u)
```

Authors’ Addresses

Akihiro Kato
NTT Software Corporation

EMail: kato.akihiro-at-po.ntts.co.jp

Michael Scott
CertiVox

EMail: mike.scott-at-certivox.com

Tetsutaro Kobayashi
NTT

EMail: kobayashi.tetsutaro-at-lab.ntt.co.jp

Yuto Kawahara
NTT

EMail: kawahara.yuto-at-lab.ntt.co.jp