Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2019

S. Yonezawa
Lepidum
S. Chikara
NTT TechnoCross
T. Kobayashi
T. Saito
NTT

March 11, 2019

Pairing-Friendly Curves
draft-yonezawa-pairing-friendly-curves-01

Abstract

This memo introduces pairing-friendly curves used for constructing pairing-based cryptography. It describes recommended parameters for each security level and recent implementations of pairing-friendly curves.

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 https://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 September 12, 2019.

Table of Contents

1. Introduction

1.1. Pairing-Based Cryptography

Elliptic curve cryptography is one of the important areas in recent cryptography. The cryptographic algorithms based on elliptic curve cryptography, such as ECDSA, is widely used in many applications.

Pairing-based cryptography, a variant of elliptic curve cryptography, is attracted the attention for its flexible and applicable functionality. Pairing is a special map defined over elliptic

curves. As the importance of pairing grows, elliptic curves where pairing is efficiently computable are studied and the special curves called pairing-friendly curves are proposed.

Thanks to the characteristics of pairing, it can be applied to construct several cryptographic algorithms and protocols such as identity-based encryption (IBE), attribute-based encryption (ABE), authenticated key exchange (AKE), short signatures and so on. Several applications of pairing-based cryptography is now in practical use.

1.2. Applications of Pairing-Based Cryptography

Several applications using pairing-based cryptography are standardized and implemented. We show example applications available in the real world.

IETF issues RFCs for pairing-based cryptography such as identity-based cryptography [9], Sakai-Kasahara Key Encryption (SAKKE) [10], and Identity-Based Authenticated Key Exchange (IBAKE) [11]. SAKKE is applied to Multimedia Internet KEYing (MIKEY) [12] and used in 3GPP [13].

Pairing-based key agreement protocols are standardized in ISO/IEC [14]. In [14], a key agreement scheme by Joux [15], identity-based key agreement schemes by Smart-Chen-Cheng [16] and by Fujioka-Suzuki-Ustaoglu [17] are specified.

MIRACL implements M-Pin, a multi-factor authentication protocol [18]. M-Pin protocol includes a kind of zero-knowledge proof, where pairing is used for its construction.

Trusted Computing Group (TCG) specifies ECDAA (Elliptic Curve Direct Anonymous Attestation) in the specification of Trusted Platform Module (TPM) [19]. ECDAA is a protocol for proving the attestation held by a TPM to a verifier without revealing the attestation held by that TPM. Pairing is used for constructing ECDAA. FIDO Alliance [20] and W3C [21] also published ECDAA algorithm similar to TCG.

Zcash implements their own zero-knowledge proof algorithm named zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) [22]. zk-SNARKs is used for protecting privacy of transactions of Zcash. They use pairing for constructing zk-SNARKS.

Cloudflare introduced Geo Key Manager [23] to restrict distribution of customers' private keys to the subset of their data centers. To achieve this functionality, attribute-based encryption is used and pairing takes a role as a building block.

Recently, Boneh-Lynn-Shacham (BLS) signature schemes are being standardized [24] and utilized in several blockchain projects such as Ethereum [25], Algorand [26], Chia Network [27] and DFINITY [28]. The threshold functionality and aggregation functionality of BLS signatures are effective for their applications of decentralization and scalability.

1.3. Goal

The goal of this memo is to consider the security of pairing-friendly curves used in pairing-based cryptography and introduce secure parameters of pairing-frindly curves. Specifically, we explain the recent attack against pairing-friendly curves and how much the security of the curves is reduced. We show how to evaluate the security of pairing-friendly curves and give the parameters for 100 bits of security, which is no longer secure, 128 and 256 bits of security.

1.4. Requirements Terminology

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 [1].

2. Preliminaries

2.1. Elliptic Curve

Let p > 3 be a prime and F_p be a finite field. The curve defined by the following equation E is called an elliptic curve.

$$E : y^2 = x^3 + A * x + B,$$

where A, B are in F_p and satisfies $4 * A^3 + 27 * B^2 != 0 \bmod p$.

Solutions (x, y) for an elliptic curve E, as well as the point at infinity, O_E, are called F_p-rational points. If P and Q are two points on the curve E, we can define R = P + Q as the opposite point of the intersection between the curve E and the line that intersects P and Q. We can define P + O_E = P = O_E + P as well. The additive group is constructed by the well-defined operation in the set of F_p-rational points. Similarly, a scalar multiplication S = [a]P for a positive integer a can be defined as an a-time addition of P.

Typically, the cyclic additive group with a prime order r and the base point G in its group is used for the elliptic curve

cryptography. Furthermore, we define terminology used in this memo as follows.

O_E: the point at infinity over an elliptic curve E.

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

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

k: an embedding degree, a minimum integer such that r is a divisor of p^k - 1.

2.2. Pairing

Pairing is a kind of the bilinear map defined over an elliptic curve. Examples include Weil pairing, Tate pairing, optimal Ate pairing [2] and so on. Especially, optimal Ate pairing is considered to be efficient to compute and mainly used for practical implementation.

Let E be an elliptic curve defined over the prime field F_p. Let G_1 be a cyclic subgroup generated by a rational point on E with order r, and G_2 be a cyclic subgroup generated by a twisted curve E' of E with order r. Let G_T be an order r subgroup of a field F_p^k, where k is an embedded degree. Pairing is defined as a bilinear map e: (G_1, G_2) -> G_T satisfying the following properties:

1. Bilinearity: for any S in G_1, T in G_2, 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 and T in G_2, the bilinear map is efficiently computable.

2.3. Barreto-Naehrig Curve

A BN curve [3] is one of the instantiations of pairing-friendly curves proposed in 2005. A pairing over BN curves constructs optimal Ate pairings.

A BN curve is an elliptic curve E defined over a finite field F_p, where p is more than or equal to 5, such that p and its order r are prime numbers parameterized by

$$p = 36 * t^4 + 36 * t^3 + 24 * t^2 + 6 * t + 1$$
$$r = 36 * t^4 + 36 * t^3 + 18 * t^2 + 6 * t + 1$$

for some well chosen integer t. The elliptic curve has an equation of the form E: y^2 = x^3 + b, where b is an element of multiplicative group of order p.

BN curves always have order 6 twists. If m is an element which is neither a square nor a cube in a finite field F_p2, the twisted curve E' of E is defined over a finite field F_p2 by the equation E': y^2 = x^3 + b' with b' = b / m or b' = b * m. The embedded degree k is 12.

A pairing e is defined by taking G_1 as a cyclic group composed by rational points on the elliptic curve E, G_2 as a cyclic group composed by rational points on the elliptic curve E', and G_T as a multiplicative group of order p^12.

2.4. Barreto-Lynn-Scott Curve

A BLS curve [4] is another instantiations of pairings proposed in 2002. Similar to BN curves, a pairing over BLS curves constructs optimal Ate pairings.

A BLS curve is an elliptic curve E defined over a finite field F_p by an equation of the form E: y^2 = x^3 + b and has a twist of order 6 defined in the same way as BN curves. In contrast to BN curves, a BLS curve does not have a prime order but its order is divisible by a large parameterized prime r and the pairing will be defined on the r-torsions points.

BLS curves vary according to different embedding degrees. In this memo, we deal with BLS12 and BLS48 families with embedding degrees 12 and 48 with respect to r, respectively.

In BLS curves, parameterized p and r are given by the following equations:

```
BLS12:
    p = (t - 1)^2 * (t^4 - t^2 + 1) / 3 + t
    r = t^4 - t^2 + 1
BLS48:
    p = (t - 1)^2 * (t^16 - t^8 + 1) / 3 + t
    r = t^16 - t^8 + 1
```

for some well chosen integer t.

2.5. Representation Convention for an Extension Field

Pairing-friendly curves uses some extension fields. In order to encode an element of an extension field, we adopt the convention shown in [29].

For an element s of an extension field of degree d such that s = s_0 + s_1 * i + s_2 * i^2 + ... + s_{d-1} * i^{d-1} for an indeterminant i, s is represented by

s = s_0 + s_1 * p + s_2 * p^2 + ... + s_{d-1} * p^{d-1}.

The parameters and test vectors of extension fields described in this memo are encoded by this convention and represented in octet stream.

3. Security of Pairing-Friendly Curves

3.1. Evaluating the Security of Pairing-Friendly Curves

The security of pairing-friendly curves is evaluated by the hardness of the following discrete logarithm problems.

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

- The finite field discrete logarithm problem (FFDLP) in G_T

There are other hard problems over pairing-friendly curves, which are used for proving the security of pairing-based cryptography. Such problems include computational bilinear Diffie-Hellman (CBDH) problem or bilinear Diffie-Hellman (BDH) Problem, decision bilinear Diffie-Hellman (DBDH) problem, gap DBDH problem, etc [30]. Almost all of these variants are reduced to the hardness of discrete logarithm problems described above and believed to be easier than the discrete logarithm problems.

There would be the case where the attacker solves these reduced problems to break the pairing-based cryptography. Since such attacks have not been discovered yet, we discuss the hardness of the discrete logarithm problems in this memo.

The security level of pairing-friendly curves is estimated by the computational cost of the most efficient algorithm to solve the above discrete logarithm problems. The well-known algorithms for solving the discrete logarithm problems includes Pollard's rho algorithm [31], Index Calculus [32] and so on. In order to make index calculus algorithms more efficient, number field sieve (NFS) algorithms are utilized.

In addition, the special case where the cofactors of G_1, G_2 and G_T are small should be taken care [33]. In such case, the discrete logarithm problem can be efficiently solved. One has to choose parameters so that the cofactors of G_1, G_2 and G_T contain no prime factors smaller than |G_1|, |G_2| and |G_T|.

3.2. Impact of the Recent Attack

In 2016, Kim and Barbulescu proposed a new variant of the NFS algorithms, the extended number field sieve (exTNFS), which drastically reduces the complexity of solving FFDLP [5]. Due to exTNFS, the security level of pairing-friendly curves asymptotically dropped down. For instance, Barbulescu and Duquesne estimates that the security of the BN curves which was believed to provide 128 bits of security (BN256, for example) dropped down to approximately 100 bits [6].

Some papers show the minimum bitlength of the parameters of pairing-friendly curves for each security level when applying exTNFS as an attacking method for FFDLP. For 128 bits of security, Menezes, Sarkar and Singh estimated the minimum bitlength of p of BN curves after exTNFS as 383 bits, and that of BLS12 curves as 384 bits [7]. For 256 bits of security, Kiyomura et al. estimated the minimum bitlength of p^k of BLS48 curves as 27,410 bits, which implied 572 bits of p [8].

4. Security Evaluation of Pairing-Friendly Curves

We give security evaluation for pairing-friendly curves based on the evaluating method presented in Section 3. We also introduce secure parameters of pairing-friendly curves for each security level. The parameters introduced here are chosen with the consideration of security, efficiency and global acceptance.

For security, we introduce 100 bits, 128 bits and 256 bits of security. We note that 100 bits of security is no longer secure and recommend 128 bits and 256 bits of security for secure applications. We follow TLS 1.3 [34] which specifies the cipher suites with 128 bits and 256 bits of security as mandatory-to-implement for the choice of the security level.

implementers of the applications have to choose the parameters with appropriate security level according to the security requirements of the applications. For efficiency, we refer to the benchmark by mcl [35] for 128 bits of security, and by Kiyomura et al. [8] for 256 bits of security and choose sufficiently efficient parameters. For global acceptance, we give the implementations of pairing-friendly curves in Section 5.

4.1. For 100 Bits of Security

Before exTNFS, BN curves with 256-bit size of underlying finite field (so-called BN256) were considered to have 128 bits of security.

After exTNFS, however, the security level of BN curves with 256-bit size of underlying finite field fell into 100 bits.

implementers who will newly develop the applications of pairing-based cryptography SHOULD NOT use BN256 as a pairing-friendly curve when their applications require 128 bits of security. In case an application does not require higher security level and is sufficient to have 100 bits of security (i.e. Internet of Things), implementers MAY use BN256.

4.2. For 128 Bits of Security

A BN curve with 128 bits of security is shown in [6], which we call BN462. BN462 is defined by a parameter t = 2^114 + 2^101 - 2^14 - 1 for the definition in Section 2.3. Defined by t, the elliptic curve E and its twisted curve E' are represented by E: y^2 = x^3 + 5 and E': y^2 = x^3 - u + 2, where u is an element of an extension field F_p^2, respectively. The size of p becomes 462-bit length.

For the finite field F_p, the towers of extension field F_p2, F_p6 and F_p12 are defined by indeterminants u, v, w as follows:

```
F_p2 = F_p[u] / (u^2 + 1)
F_p6 = F_p2[v] / (v^3 - u - 2)
F_p12 = F_p6[w] / (w^2 - v).
```

As the parameters for BN462, we give a characteristic p, an order r, a base point G = (x, y), a cofactor h of an elliptic curve E: y^2 = x^3 + b, and an order r', a base point G' = (x', y'), a cofactor h' of an elliptic curve E': y^2 = x^3 + b'.

p: 0x2404 80360120 023fffff fffff6ff 0cf6b7d9 bfca0000 000000d8
12908f41 c8020fff fffffff6 ff66fc6f f687f640 00000000 2401b008
40138013

r: 0x2404 80360120 023fffff fffff6ff 0cf6b7d9 bfca0000 000000d8
12908ee1 c201f7ff fffffff6 ff66fc7b f717f7c0 00000000 2401b007
e010800d

x: 0x21a6 d67ef250 191fadba 34a0a301 60b9ac92 64b6f95f 63b3edbe
c3cf4b2e 689db1bb b4e69a41 6a0b1e79 239c0372 e5cd7011 3c98d91f
36b6980d

y: 0x0118 ea0460f7 f7abb82b 33676a74 32a490ee da842ccc fa7d788c
65965042 6e6af77d f11b8ae4 0eb80f47 5432c666 00622eca a8a5734d
36fb03de

h: 1

b: 5

r': 0x2404 80360120 023fffff fffff6ff 0cf6b7d9 bfca0000 000000d8
12908ee1 c201f7ff fffffff6 ff66fc7b f717f7c0 00000000 2401b007
e010800d

x': 0x041b04cb e3413297 c49d8129 7eed0759 47d86135 c4abf0be 9d5b64be
02d6ae78 34047ea4 079cd30f e28a68ba 0cb8f7b7 2836437d c75b2567
ff2b98db b93f68fa c828d822 1e4e1d89 475e2d85 f2063cbc 4a74f6f6
6268b6e6 da1162ee 055365bb 30283bde 614a17f6 1a255d68 82417164
bc500498

y': 0x0104fa79 6cbc2989 0f9a3798 2c353da1 3b299391 be45ddb1 c15ca42a
bdf8bf50 2a5dd7ac 0a3d351a 859980e8 9be676d0 0e92c128 714d6f3c
6aba56ca 6e0fc6a5 468c12d4 2762b29d 840f13ce 5c3323ff 016233ec
7d76d4a8 12e25bbe b2c25024 3f2cbd27 80527ec5 ad208d72 24334db3
c1b4a49c

h': 0x2404 80360120 023fffff fffff6ff 0cf6b7d9 bfca0000 000000d8
12908fa1 ce0227ff fffffff6 ff66fc63 f5f7f4c0 00000000 2401b008
a0168019

b': -u + 2

A BLS12 curve with 128 bits of security shown in [36], BLS12-381, is defined by a parameter t = -2^63 - 2^62 - 2^60 - 2^57 - 2^48 - 2^16 and the size of p becomes 381-bit length. Defined by t, the elliptic curve E and its twisted curve E' are represented by E: y^2 = x^3 + 4 and E': y^2 = x^3 + 4(u + 1), where u is an element of an extension field F_p^2, respectively.

For the finite field F_p, the towers of extension field F_p2, F_p6 and F_p12 are defined by indeterminants u, v, w as follows:

F_p2 = F_p[u] / (u^2 + 1)
F_p6 = F_p2[v] / (v^3 - u - 1)
F_p12 = F_p6[w] / (w^2 - v).

We have to note that, according to [7], the bit length of p for BLS12 to achieve 128 bits of security is calculated as 384 bits and more, which BLS12-381 does not satisfy. Although the computational time is conservatively estimated by 2^110 when exTNFS is applied with index calculus, there is no currently published efficient method for such computational time. They state that BLS12-381 achieves 127-bit security level evaluated by the computational cost of Pollard's rho. Therefore, we regard BN462 as a "conservative" parameter, and BLS12-381 as an "optimistic" parameter.

We give the parameters for BLS12-381 as follows.

p: 0x1a0111ea 397fe69a 4b1ba7b6 434bacd7 64774b84 f38512bf 6730d2a0
f6b0f624 1eabfffe b153ffff b9feffff ffffaaab

r: 0x73eda753 299d7d48 3339d808 09a1d805 53bda402 fffe5bfe ffffffff
00000001

x: 0x17f1d3a7 3197d794 2695638c 4fa9ac0f c3688c4f 9774b905 a14e3a3f
171bac58 6c55e83f f97a1aef fb3af00a db22c6bb

y: 0x08b3f481 e3aaa0f1 a09e30ed 741d8ae4 fcf5e095 d5d00af6 00db18cb
2c04b3ed d03cc744 a2888ae4 0caa2329 46c5e7e1

h: 0x396c8c00 5555e156 8c00aaab 0000aaab

b: 4

r': 0x1a0111ea 397fe69a 4b1ba7b6 434bacd7 64774b84 f38512bf 6730d2a0
f6b0f624 1eabfffe b153ffff b9feffff ffffaaab

x': 0x204d9ac 05ffbfeb ac60c8f3 e4143831 567c7063 d38b0595 9c12ec06
3fd7b99a b4541ece faa3f0ec 1a0a33da 0ff56d7b 45b2ca9f f8adbac4
78790d52 dc45216b 3e272dce a7571e71 81b20335 695608a3 0ea1f83e
53a80d95 ad3a0c1e 7c4e76e2

y': 0x09cb66a fff60c18 9da2c655 d4eccad1 5dba53e8 a3c89101 aba0838c
17ad69cd 096844ba 7ec246ea 99be5c24 9aea2f05 c14385e9 c53df5fb
63ddecfe f1067e73 5cc17763 97138d4c b2ccdfbe 45b5343e eadf6637
08ae1288 aa4306db 8598a5eb

h': 0x5d543a9 5414e7f1 091d5079 2876a202 cd91de45 47085aba a68a205b
2e5a7ddf a628f1cb 4d9e82ef 21537e29 3a6691ae 1616ec6e 786f0c70
cf1c38e3 1c7238e5

b': 4 * (u + 1)

4.3. For 256 Bits of Security

As shown in Section 3.2, it is unrealistic to achieve 256 bits of security by BN curves since the minimum size of p becomes too large to implement. Hence, we consider BLS48 for 256 bits of security.

A BLS48 curve with 256 bits of security is shown in [8], which we call BLS48-581. It is defined by a parameter t = -1 + 2^7 - 2^10 - 2^30 - 2^32 and the elliptic curve E and its twisted curve E' are represented by E: y^2 = x^3 + 1 and E': y^2 = x^3 - 1 / w, where w is

an element of an extension field F_p^8. The size of p becomes 581-bit length.

For the finite field F_p, the towers of extension field F_p2, F_p4, F_p8, F_p24 and F_p48 are defined by indeterminants u, v, w, z, s as follows:

```
F_p2 = F_p[u] / (u^2 + 1)
F_p4 = F_p2[v] / (v^2 + u + 1)
F_p8 = F_p4[w] / (w^2 + v)
F_p24 = F_p8[z] / (z^3 + w)
F_p48 = Fp24[s] / (s^2 + z)
```

We then give the parameters for BLS48-581 as follows.

p: 0x12 80f73ff3 476f3138 24e31d47 012a0056 e84f8d12 2131bb3b
e6c0f1f3 975444a4 8ae43af6 e082acd9 cd30394f 4736daf6 8367a551
3170ee0a 578fdf72 1a4a48ac 3edc154e 6565912b

r: 0x23 86f8a925 e2885e23 3a9ccc16 15c0d6c6 35387a3f 0b3cbe00
3fad6bc9 72c2e6e7 41969d34 c4c92016 a85c7cd0 562303c4 ccbe5994
67c24da1 18a5fe6f cd671c01

x: 0x02 af59b7ac 340f2baf 2b73df1e 93f860de 3f257e0e 86868cf6
1abdbaed ffb9f754 4550546a 9df6f964 5847665d 859236eb dbc57db3
68b11786 cb74da5d 3a1e6d8c 3bce8732 315af640

y: 0x0c efda44f6 531f91f8 6b3a2d1f b398a488 a553c9ef eb8a52e9
91279dd4 1b720ef7 bb7beffb 98aee53e 80f67858 4c3ef22f 487f77c2
876d1b2e 35f37aef 7b926b57 6dbb5de3 e2587a70

h: 0x85555841 aaaec4ac

b: 1

r': 0x23 86f8a925 e2885e23 3a9ccc16 15c0d6c6 35387a3f 0b3cbe00
3fad6bc9 72c2e6e7 41969d34 c4c92016 a85c7cd0 562303c4 ccbe5994
67c24da1 18a5fe6f cd671c01

x': 0x01 690ae060 61530e31 64040ce6 e7466974 a0865edb 6d5b825d
f11e5db6 b724681c 2b5a805a f2c7c45f 60300c3c 4238a1f5 f6d3b644
29f5b655 a4709a8b ddf790ec 477b5fb1 ed4a0156 dec43f7f 6c401164
da6b6f9a f79b9fc2 c0e09d2c d4b65900 d2394b61 aa3bb48c 7c731a14
68de0a17 346e34e1 7d58d870 7f845fac e35202bb 9d64b5ef f29cbfc8
5f5c6d60 1d794c87 96c20e67 81dffed3 36fc1ff6 d3ae3193 dec00603
91acb681 1f1fbde3 8027a0ef 591e6b21 c6e31c5f 1fda66eb 05582b6b
0399c6a2 459cb2ab fd0d5d95 3447a927 86e194b2 89588e63 ef1b8b61
ad354bed 299b5a49 7c549d7a 56a74879 b7665a70 42fbcaf1 190d915f

```
      945fef6c 0fcec14b 4afc403f 50774720 4d810c57 00de1692 6309352f
      660f26a5 529a2f74 cb9d1044 0595dc25 d6d12fcc e84fc565 57217bd4
      bc2d645a b4ca167f b812de7c acc3b942 7fc78212 985680b8 83bf7fee
      7eae0199 1eb7a52a 0f4cbb01 f5a8e3c1 6c41350d c62be2c1 9cbd2b98
      d9c9d268 7cd811db 7863779c 97e9a15b d6967d5e b21f972d 28ad9d43
      7de41234 25249319 98f280a9 a9c799c3 3ff8f838 ca35bdde bbb79cdc
      2967946c c0f77995 411692e1 8519243d 5598bdb4 623a11dc 97ca3889
      49f32c65 db3fc6a4 7124bd5d 063549e5 0b0f8b03 0d3a9830 e1e3bef5
      cd428393 9d33a28c fdc3df89 640df257 c0fc2544 77a9c8ef f69b57cf
      f042e6fd 1ef3e293 c57beca2 cd61dc44 838014c2 08eda095 e10d5e89
      e705ff69 07047895 96e41969 96508797 71f58935 d768cdc3 b55150cc
      a3693e28 33b62df3 4f1e2491 ef8c5824 f8a80cd8 6e65193a

   y':  0x00 951682f0 10b08932 b28b4a85 1ec79469 f9437fc4 f9cfa8cc
      dec25c3c c847890c 65e1bcd2 df994b83 5b71e49c 0fc69e6d 9ea5da9d
      bb020a9d fb2942dd 022fa962 fb0233de 016c8c80 e9387b0b 28786785
      523e68eb 7c008f81 b99ee3b5 d10a72e5 321a09b7 4b39c58b 75d09d73
      e4155b76 dc11d8dd 416b7fa6 3557fcdd b0a955f6 f5e0028d 4af2150b
      fd757a89 8b548912 e2c0c6e5 70449113 fcee54cd a9cb8bfd 7f182825
      b371f069 61b62ca4 41bfcb3d 13ce6840 432bf8bc 4736003c 64d695e9
      84ddc2ef 4aee1747 044157fd 2f9b81c4 3eed97d3 45289899 6d24c66a
      ad191dba 634f3e04 c89485e0 6f8308b8 afaedf1c 98b1a466 deab2c15
      81f96b6f 3c64d440 f2a16a62 75000cf3 8c09453b 5b9dc827 8eabe442
      92a154dc 69faa74a d76ca847 b786eb2f d686b9be 509fe24b 8293861c
      c35d76be 88c27117 04bfe118 e4db1fad 86c2a642 4da6b3e5 807258a2
      d166d3e0 e43d15e3 b6464fb9 9f382f57 fd10499f 8c8e11df 718c98a0
      49bd0e5d 1301bc9e 6ccd0f06 3b06eb06 422afa46 9b5b529b 8bba3d4c
      6f219aff e4c57d73 10a92119 c98884c3 b6c0bbcc 113f6826 b3ae70e3
      bbbaadab 3ff8abf3 b905c231 38dfe385 134807fc c1f9c19e 68c0ec46
      8213bc9f 0387ca1f 4ffe406f da92d655 3cd4cfd5 0a2c895e 85fe2540
      9ffe8bb4 3b458f9b efab4d59 bee20e2f 01de48c2 affb03a9 7ceede87
      214e3bb9 0183303b 672e50b8 7b36a615 34034578 db0195fd 81a46beb
      55f75d20 049d044c 3fa5c367 8c783db3 120c2580 359a7b33 cac5ce21
      e4cecda9 e2e2d6d2 ff202ff4 3c1bb2d4 b5e53dae 010423ce

   h':  0x170e915c b0a6b740 6b8d9404 2317f811 d6bc3fc6 e211ada4 2e58ccfc
      b3ac076a 7e4499d7 00a0c23d c4b0c078 f92def8c 87b7fe63 e1eea270
      db353a4e f4d38b59 98ad8f0d 042ea24c 8f02be1c 0c83992f e5d77252
      27bb2712 3a949e08 76c0a8ce 0a67326d b0e955dc b791b867 f31d6bfa
      62fbdd5f 44a00504 df04e186 fae033f1 eb43c1b1 a08b6e08 6eff03c8
      fee9ebdd 1e191a8a 4b0466c9 0b389987 de5637d5 dd13dab3 3196bd2e
      5afa6cd1 9cf0fc3f c7db7ece 1f3fac74 2626b1b0 2fcee040 43b2ea96
      492f6afa 51739597 c54bb78a a6b0b993 19fef9d0 9f768831 018ee656
      4c68d054 c62f2e0b 4549426f ec24ab26 957a669d ba2a2b69 45ce40c9
      aec6afde da16c79e 15546cd7 771fa544 d5364236 690ea068 32679562
      a6873142 0ae52d0d 35a90b8d 10b688e3 1b6aee45 f45b7a50 83c71732
      105852de cc888f64 839a4de3 3b99521f 0984a418 d20fc7b0 609530e4
      54f0696f a2a8075a c01cc8ae 3869e8d0 fe1f3788 ffac4c01 aa2720e4
```

```
   31da333c 83d9663b fb1fb7a1 a7b90528 482c6be7 89229903 0bb51a51
   dc7e91e9 15687441 6bf4c26f 1ea7ec57 80585639 60ef92bb bb8632d3
   a1b695f9 54af10e9 a78e40ac ffc13b06 540aae9d a5287fc4 429485d4
   4e6289d8 c0d6a3eb 2ece3501 24527518 39fb48bc 14b51547 8e2ff412
   d930ac20 307561f3 a5c998e6 bcbfebd9 7effc643 3033a236 1bfcdc4f
   c74ad379 a16c6dea 49c209b1
```

b': -1 / w

5. Implementations of Pairing-Friendly Curves

We show the pairing-friendly curves selected by existing standards, applications and cryptographic libraries.

ISO/IEC 15946-5 [37] shows examples of BN curves with the size of 160, 192, 224, 256, 384 and 512 bits of p. There is no action so far after the proposal of exTNFS.

TCG adopts an BN curve of 256 bits specified in ISO/IEC 15946-5 (TPM_ECC_BN_P256) and of 638 bits specified by their own (TPM_ECC_BN_P638). FIDO Alliance [20] and W3C [21] adopt the BN curves specified in TCG, a 512-bit BN curve shown in ISO/IEC 15946-5 and another 256-bit BN curve.

MIRACL [38] implements BN curves and BLS12 curves.

Zcash implemented a BN curve (named BN128) in their library libsnark [39]. After exTNFS, they propose a new parameter of BLS12 as BLS12-381 [36] and publish its experimental implementation [40].

Cloudflare implements a 256-bit BN curve (bn256) [41]. There is no action so far after exTNFS.

Ethereum 2.0 adopts BLS12-381 (BLS12_381), BN curves with 254 bits of p (CurveFp254BNb) and 382 bits of p (CurveFp382_1 and CurveFp382_2) [42]. Their implementation calls mcl [35] for pairing computation. Chia Network publishs their implementation [27] by integrating the RELIC toolkit [44].

Cryptographic libraries which implement pairings include PBC [43], mcl [35], RELIC [44], TEPLA [45], AMCL [46], Intel IPP [47] and a library by Kyushu University [48].

Table 1 shows the adoption of pairing-friendly curves in existing standards, applications and libraries. In this table, the curves marked as (*) indicate that there is no research result on the security evaluation, but that the implementers states that they hold 128 bits of security.

Category	Name	100 bit	128 bit	256 bit
standards	ISO/IEC 15946-5	BN256	BN384	
	TCG	BN256		
	FIDO/W3C	BN256		
applications	MIRACL	BN254	BLS12	
	Zcash	BN128 (CurveSNARK)	BLS12-381	
	Cloudflare	BN256		
	Ethereum	BN254	BN382 (*) / BLS12-381 (*)	
	Chia Network		BLS12-381 (*)	
libraries	PBC	BN		
	mcl	BN254 / BN_SNARK1	BN381_1 (*) / BN462 / BLS12-381	
	RELIC	BN254 / BN256	BLS12-381 / BLS12-455	
	TEPLA	BN254		
	AMCL	BN254 / BN256	BLS12-381 (*) / BLS12-383 (*) / BLS12-461	BLS48
	Intel IPP	BN256		
	Kyushu Univ.			BLS48

Table 1: Adoption of Pairing-Friendly Curves

6. Security Considerations

This memo entirely describes the security of pairing-friendly curves, and introduces secure parameters of pairing-friendly curves. We give these parameters in terms of security, efficiency and global acceptance. The parameters for 100, 128 and 256 bits of security are introduced since the security level will different in the requirements of the pairing-based applications.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The authors would like to thank Akihiro Kato for his significant contribution to the early version of this memo.

9. References

9.1. Normative References

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

[2] Vercauteren, F., "Optimal Pairings", IEEE Transactions on Information Theory Vol. 56, pp. 455-461, DOI 10.1109/tit.2009.2034881, January 2010.

[3] Barreto, P. and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order", Selected Areas in Cryptography pp. 319-331, DOI 10.1007/11693383_22, 2006.

[4] Barreto, P., Lynn, B., and M. Scott, "Constructing Elliptic Curves with Prescribed Embedding Degrees", Security in Communication Networks pp. 257-267, DOI 10.1007/3-540-36413-7_19, 2003.

[5] Kim, T. and R. Barbulescu, "Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case", Advances in Cryptology - CRYPTO 2016 pp. 543-571, DOI 10.1007/978-3-662-53018-4_20, 2016.

[6] Barbulescu, R. and S. Duquesne, "Updating Key Size Estimations for Pairings", Journal of Cryptology, DOI 10.1007/s00145-018-9280-5, January 2018.

[7] Menezes, A., Sarkar, P., and S. Singh, "Challenges with Assessing the Impact of NFS Advances on the Security of Pairing-Based Cryptography", Lecture Notes in Computer Science pp. 83-108, DOI 10.1007/978-3-319-61273-7_5, 2017.

[8] Kiyomura, Y., Inoue, A., Kawahara, Y., Yasuda, M., Takagi, T., and T. Kobayashi, "Secure and Efficient Pairing at 256-Bit Security Level", Applied Cryptography and Network Security pp. 59-79, DOI 10.1007/978-3-319-61204-1_4, 2017.

9.2. Informative References

[9] Boyen, X. and L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, DOI 10.17487/RFC5091, December 2007, <https://www.rfc-editor.org/info/rfc5091>.

[10] Groves, M., "Sakai-Kasahara Key Encryption (SAKKE)", RFC 6508, DOI 10.17487/RFC6508, February 2012, <https://www.rfc-editor.org/info/rfc6508>.

[11] Cakulev, V., Sundaram, G., and I. Broustis, "IBAKE: Identity-Based Authenticated Key Exchange", RFC 6539, DOI 10.17487/RFC6539, March 2012, <https://www.rfc-editor.org/info/rfc6539>.

[12] Groves, M., "MIKEY-SAKKE: Sakai-Kasahara Key Encryption in Multimedia Internet KEYing (MIKEY)", RFC 6509, DOI 10.17487/RFC6509, February 2012, <https://www.rfc-editor.org/info/rfc6509>.

[13] 3GPP, "Security of the mission critical service (Release 15)", 3GPP TS 33.180 15.3.0, 2018.

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

[15] Joux, A., "A One Round Protocol for Tripartite Diffie-Hellman", Lecture Notes in Computer Science pp. 385-393, DOI 10.1007/10722028_23, 2000.

[16] Chen, L., Cheng, Z., and N. Smart, "Identity-based key agreement protocols from pairings", International Journal of Information Security Vol. 6, pp. 213-241, DOI 10.1007/s10207-006-0011-9, January 2007.

[17] Fujioka, A., Suzuki, K., and B. Ustaoğlu, "Ephemeral Key Leakage Resilient and Efficient ID-AKEs That Can Share Identities, Private and Master Keys", Lecture Notes in Computer Science pp. 187-205, DOI 10.1007/978-3-642-17455-1_12, 2010.

[18] Scott, M., "M-Pin: A Multi-Factor Zero Knowledge Authentication Protocol", March 2019, <https://www.miracl.com/miracl-labs/m-pin-a-multi-factor-zero-knowledge-authentication-protocol>.

[19] Trusted Computing Group (TCG), "Trusted Platform Module Library Specification, Family \"2.0\", Level 00, Revision 01.38", <https://trustedcomputinggroup.org/resource/tpm-library-specification/>.

[20] Lindemann, R., "FIDO ECDAA Algorithm - FIDO Alliance Review Draft 02", <https://fidoalliance.org/specs/fido-v2.0-rd-20180702/fido-ecdaa-algorithm-v2.0-rd-20180702.html>.

[21] Lundberg, E., "Web Authentication: An API for accessing Public Key Credentials Level 1 - W3C Recommendation", <https://www.w3.org/TR/webauthn/>.

[22] Lindemann, R., "What are zk-SNARKs?", <https://z.cash/technology/zksnarks.html>.

[23] Sullivan, N., "Geo Key Manager: How It Works", <https://blog.cloudflare.com/geo-key-manager-how-it-works/>.

[24] Boneh, D., Gorbunov, S., Wee, H., and Z. Zhang, "BLS Signature Scheme", draft-boneh-bls-signature-00 (work in progress), February 2019.

[25] Jordan, R., "Ethereum 2.0 Development Update #17 - Prysmatic Labs", <https://medium.com/prysmatic-labs/ethereum-2-0-development-update-17-prysmatic-labs-ed5bcf82ec00>.

[26] Gorbunov, S., "Efficient and Secure Digital Signatures for Proof-of-Stake Blockchains", <https://medium.com/algorand/digital-signatures-for-blockchains-5820e15fbe95>.

[27] Chia Network, "BLS signatures in C++, using the relic toolkit", <https://github.com/Chia-Network/bls-signatures>.

[28] Williams, D., "DFINITY Technology Overview Series Consensus System Rev. 1", n.d., <https://dfinity.org/pdf-viewer/library/dfinity-consensus.pdf>.

[29] "IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques", IEEE standard, DOI 10.1109/ieeestd.2004.94612, n.d..

[30] ECRYPT, "Final Report on Main Computational Assumptions in Cryptography".

[31] Pollard, J., "Monte Carlo methods for index computation $({\rm mod}\ p)$", Mathematics of Computation Vol. 32, pp. 918-918, DOI 10.1090/s0025-5718-1978-0491431-9, September 1978.

[32] Hellman, M. and J. Reyneri, "Fast Computation of Discrete Logarithms in GF (q)", Advances in Cryptology pp. 3-13, DOI 10.1007/978-1-4757-0602-4_1, 1983.

[33] Barreto, P., Costello, C., Misoczki, R., Naehrig, M., Pereira, G., and G. Zanon, "Subgroup Security in Pairing-Based Cryptography", Progress in Cryptology -- LATINCRYPT 2015 pp. 245-265, DOI 10.1007/978-3-319-22174-8_14, 2015.

[34] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[35] Mitsunari, S., "mcl - A portable and fast pairing-based cryptography library", 2016, <https://github.com/herumi/mcl>.

[36] Bowe, S., "BLS12-381: New zk-SNARK Elliptic Curve Construction", <https://blog.z.cash/new-snark-curve/>.

[37] ISO/IEC, "ISO/IEC 15946-5:2017", ISO/IEC Information technology -- Security techniques -- Cryptographic techniques based on elliptic curves -- Part 5: Elliptic curve generation, 2017.

[38] MIRACL Ltd., "MIRACL Cryptographic SDK", 2018, <https://github.com/miracl/MIRACL>.

[39] SCIPR Lab, "libsnark: a C++ library for zkSNARK proofs", 2012, <https://github.com/zcash/libsnark>.

[40] zkcrypto, "zkcrypto - Pairing-friendly elliptic curve library", 2017, <https://github.com/zkcrypto/pairing>.

[41] Cloudflare, "package bn256", March 2019, <https://godoc.org/github.com/cloudflare/bn256>.

[42] Prysmatic Labs, "go-bls - Go wrapper for a BLS12-381 Signature Aggregation implementation in C++", 2018, <https://godoc.org/github.com/prysmaticlabs/go-bls>.

[43] Lynn, B., "PBC Library - The Pairing-Based Cryptography Library", 2006, <https://crypto.stanford.edu/pbc/>.

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

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

[46] The Apache Software Foundation, "The Apache Milagro Cryptographic Library (AMCL)", 2016, <https://github.com/apache/incubator-milagro-crypto>.

[47] Intel Corporation, "Developer Reference for Intel Integrated Performance Primitives Cryptography 2019", 2018, <https://software.intel.com/en-us/ipp-crypto-reference-arithmetic-of-the-group-of-elliptic-curve-points>.

[48] Kyushu University, "bls48 - C++ library for Optimal Ate Pairing on BLS48", 2017, <https://github.com/mk-math-kyushu/bls48>.

Appendix A. Computing Optimal Ate Pairing

Before presenting the computation of optimal Ate pairing e(P, Q) satisfying the properties shown in Section 2.2, we give subfunctions used for pairing computation.

The following algorithm Line_Function shows the computation of the line function. It takes A = (A[1], A[2]), B = (B[1], B[2]) in G_2 and P = ((P[1], P[2])) in G_1 as input and outputs an element of G_T.

```
    if (A = B) then
        l := (3 * A[1]^2) / (2 * A[2]);
    else if (A = -B) then
        return P[1] - A[1];
    else
        l := (B[2] - A[2]) / (B[1] - A[1]);
    end if;
    return (l * (P[1] -A[1]) + A[2] -P[2]);
```

When implementing the line function, implementer should consider the isomorphism of E and its twisted curve E' so that one can reduce the computational cost of operations in G_2. We note that the function Line_function does not consider such isomorphism.

Computation of optimal Ate pairing for BN curves uses Frobenius map. Let a Frobenius map pi for a point Q = (x, y) over E' be pi(p, Q) = (x^p, y^p).

A.1. Optimal Ate Pairings over Barreto-Naehrig Curves

Let s = 6 * t + 2 for a parameter t and s_0, s_1, ... , s_L in {-1,0,1} such that the sum of s_i * 2^i (i = 0, 1, ..., L) equals to s.

The following algorithm shows the computation of optimal Ate pairing over Barreto-Naehrig curves. It takes P in G_1, Q in G_2, an integer s, s_0, ...,s_L in {-1,0,1} such that the sum of s_i * 2^i (i = 0, 1, ..., L) equals to s, and an order r as input, and outputs e(P, Q).

```
    f := 1; T := Q;
    if (s_L = -1)
        T := -T;
    end if
    for i = L-1 to 0
        f := f^2 * Line_function(T, T, P); T := 2 * T;
        if (s_i = 1 | s_i = -1)
            f := f * Line_function(T, s_i * Q); T := T + s_i * Q;
        end if
    end for
    Q_1 := pi(p, Q); Q_2 := pi(p, Q_1);
    f := f * Line_function(T, Q_1, P); T := T + Q_1;
    f := f * Line_function(T, -Q_2, P);
    f := f^{(p^k - 1) / r}
    return f;
```

A.2. Optimal Ate Pairings over Barreto-Lynn-Scott Curves

Let s = t for a parameter u and s_0, s_1, ... , s_L in {-1,0,1} be a sign-digit representation of s such that the sum of s_i * 2^i (i = 0, 1, ..., L) equals to s. The following algorithm shows the computation of optimal Ate pairing over Barreto-Lynn-Scott curves. It takes P in G_1, Q in G_2, a parameter s, s_0, s_1, ..., s_L in {-1,0,1} such that the sum of s_i * 2^i (i = 0, 1, ..., L), and an order r as input, and outputs e(P, Q).

```
f := 1; T := Q;
if (s_L = -1)
    T := -T;
end if
for i = L-1 to 0
    f := f^2 * Line_function(T, T, P); T := 2 * T;
    if (s_i = 1 | s_i = -1)
        f := f * Line_function(T, s_i * Q, P); T := T + s_i * Q;
    end if
end for
f := f^{(p^k - 1) / r};
return f;
```

Appendix B. Test Vectors of Optimal Ate Pairing

We provide test vectors for Optimal Ate Pairing e(P, Q) given in Appendix A for the curves BN462, BLS12-381 and BLS48-581 given in Section 4. Here, the inputs P = (x, y) and Q = (x', y') are the corresponding base points G and G' given in {{secure_params}.

BN462:

Input x value: 0x17f1d3a7 3197d794 2695638c 4fa9ac0f c3688c4f
9774b905 a14e3a3f 171bac58 6c55e83f f97a1aef fb3af00a db22c6bb

Input y value: 0x08b3f481 e3aaa0f1 a09e30ed 741d8ae4 fcf5e095
d5d00af6 00db18cb 2c04b3ed d03cc744 a2888ae4 0caa2329 46c5e7e1

Input x' value: 0x041b04cb e3413297 c49d8129 7eed0759 47d86135
c4abf0be 9d5b64be 02d6ae78 34047ea4 079cd30f e28a68ba 0cb8f7b7
2836437d c75b2567 ff2b98db b93f68fa c828d822 1e4e1d89 475e2d85
f2063cbc 4a74f6f6 6268b6e6 da1162ee 055365bb 30283bde 614a17f6
1a255d68 82417164 bc500498

Input y' value: 0x0104fa79 6cbc2989 0f9a3798 2c353da1 3b299391
be45ddb1 c15ca42a bdf8bf50 2a5dd7ac 0a3d351a 859980e8 9be676d0
0e92c128 714d6f3c 6aba56ca 6e0fc6a5 468c12d4 2762b29d 840f13ce

5c3323ff 016233ec 7d76d4a8 12e25bbe b2c25024 3f2cbd27 80527ec5
ad208d72 24334db3 c1b4a49c

Output e(P, Q): 0x3c8193be d979f4ea 5012851f 2eb824ba 7d21f584
5c041641 0f26a097 72ab76e9 153ad0df 012c6aad b90b2db7 91b4865d
c0dce2a1 deec6844 03bc919a 350d0077 25f4aaf2 eb1c6a6a 84c3d68b
3eb71c8f 6d21a669 6d3b70b8 2ab34ca3 d8e362f8 570aeec9 78d92761
54940920 d459438e 2141831f e6813bda 88f0a239 d693499f 07806c58
8498a881 870a313a 26c03629 44f48955 25034b30 becb184d 52e0d806
62215750 4c392059 308a3ab6 0f567fce 39169b96 36932988 abb8689f
709a4ad2 6ebc8dc2 65f79386 19357015 cf8aebec 069412b8 82dd79bf
112713ec 241d5e20 d6003b61 4943d666 403cb445 8e4ac7b8 800873eb
55d6e5aa bb6398c1 c0f349f0 9e4ba501 f239e3f5 4d09ad61 40452545
f07a20d5 3d075b26 30fefb61 f46eae73 4a0098c1 b7b7baf5 33a6072a
5e288590 cfb0c85b 4edf7be9 06e01b4f 023387d8 5e9ca9aa 70ec3b5c
c1acb450 685f4134 5391a237 182295b9 fe23ebbb f8485065 da74c4a5
91a01f3e 5e3c5710 337c050b 01c17724 d7cc02b4 f0f512da 2af14526
0028235d e26fe897 7bc0a2e5 7c183729 661ed63f aef45700 78e30f16
d09e1c58 888e561d a1cb4adb 1390360d d8d7b7b5 f097b0e0 7c988743
001eb5d1 50da6218 428c960f 10441667 e4ce905c e9cb9176 987e1731
403181b2 d197c267 27f8fef0 0a612956 a43b172f ea11ba6f 660be51f
12c1f80a 3697b28b 4d685ab6 7816e4b9 a8265a21 825a059d b092cfc1
cc28d142 8988b01a 3ec9a14b 8b95bf1d 3d111be3 82848f2c 1e9ae9fe
dbefb52e 8eba88ea 17ad2e37 30ba6d0e fcd916de 43c1666f 3b25cd7f
e72147f4 5e6c55cc 701a6469 1426d6cd 9fbbe831 6a00537a 53650496
d27c2819 4b5c1d2c 4909bcf0 06ab5e0f 90fd82a1 4e45c5d0 08448080
154b4723 b44bbc0d 48911427 dbc54e0c 0d41a043 6a1c2d36 252b921a
2560ddcc ad362cb9 02f79d7f 1210ddac 950bf406 d0f0c79f 299bcebd

BLS12-381:

Input x value: 0x17f1d3a7 3197d794 2695638c 4fa9ac0f c3688c4f
9774b905 a14e3a3f 171bac58 6c55e83f f97a1aef fb3af00a db22c6bb

Input y value: 0x08b3f481 e3aaa0f1 a09e30ed 741d8ae4 fcf5e095
d5d00af6 00db18cb 2c04b3ed d03cc744 a2888ae4 0caa2329 46c5e7e1

Input x' value: 0x204d9ac 05ffbfeb ac60c8f3 e4143831 567c7063
d38b0595 9c12ec06 3fd7b99a b4541ece faa3f0ec 1a0a33da 0ff56d7b
45b2ca9f f8adbac4 78790d52 dc45216b 3e272dce a7571e71 81b20335
695608a3 0ea1f83e 53a80d95 ad3a0c1e 7c4e76e2

Input y' value: 0x09cb66a fff60c18 9da2c655 d4eccad1 5dba53e8
a3c89101 aba0838c 17ad69cd 096844ba 7ec246ea 99be5c24 9aea2f05
c14385e9 c53df5fb 63ddecfe f1067e73 5cc17763 97138d4c b2ccdfbe
45b5343e eadf6637 08ae1288 aa4306db 8598a5eb

Output e(P, Q): 0x1099133 07699946 ffb01bb6 a8708efd ae7a380d
3d0eeed9 b73440e4 6ba128c4 db75a7b2 1df4cabb a9722393 01955454
f8e43f34 37f83953 557f251d 93c11e38 91134da9 e9d0a017 db6bbef8
f9f00689 b05a4e7b 66ca3b5e bd345258 f776e9e1 117e41ec 69120956
0e0ac469 921183f4 76c8dc14 0d30c301 a7a673e4 fc51655e 0c4130e5
47e1f648 386a1555 7ab73dd7 b113ee92 60869568 7dd9cb79 5060647e
feac9894 c0049ab3 4f1cbdea c9527013 ef5810ec f5672692 74a0425a
ff778592 49fdd23a af67e366 5c40a2b9 4a0aae91 112fb6aa d05ac5ca
8e3fa0bc 6b185c94 447d2368 136ba383 bbec5528 af53f298 1628ba4b
906e54f0 60383b92 fc46f84e 2e7c50d7 9cf7ff6d 21e81a67 15b31a66
0aa418ea de81887e c995285a 656d0a43 208ef518 27fd935a 1d617142
ad008f36 15201e00 017154ac 5aee4c2d aa96433f 97cc4705 59f94d64
dcf4c69c fc254475 d1365bc4 a3d18524 be1f6a7a cd1ad2e6 4a901e5c
f97c6291 efa4951c dba232e2 172c7e94 4c89f6fc f6074d23 f41ea7ed
783be9ba ace67ae2 7e9c682f d8fc347e 533d5e2e 550b72ee ee9f250f
427ac1f1 a0bb315e e5582635 065ec196 f68776ab 97cdd86e 9f1117b3
873800a1 9c2221c9 3a810a71 7fb6ae09 a8eac52d 707158c7 83d45b9e
ec3adfa8 3b626448 60679b2e e242888d c0ae8c3f 7a4e2c5f 8d9060b8
4b7c53c2 3992b502 15170d86 a8a1a1e6 2737e951 647ce587 2379cc01
ab977532 cefddf85 39a8223f 3df88acc dd1c9fa2 c66227b9 5549471b
462370aa 61b58c57 e9035ef9 0d630357 d852eaa3

BLS48-581:

Input x value: 0x02 af59b7ac 340f2baf 2b73df1e 93f860de 3f257e0e
86868cf6 1abdbaed ffb9f754 4550546a 9df6f964 5847665d 859236eb
dbc57db3 68b11786 cb74da5d 3a1e6d8c 3bce8732 315af640

Input y value: 0x0c efda44f6 531f91f8 6b3a2d1f b398a488 a553c9ef
eb8a52e9 91279dd4 1b720ef7 bb7beffb 98aee53e 80f67858 4c3ef22f
487f77c2 876d1b2e 35f37aef 7b926b57 6dbb5de3 e2587a70

Input x' value: 0x01 690ae060 61530e31 64040ce6 e7466974 a0865edb
6d5b825d f11e5db6 b724681c 2b5a805a f2c7c45f 60300c3c 4238a1f5
f6d3b644 29f5b655 a4709a8b ddf790ec 477b5fb1 ed4a0156 dec43f7f
6c401164 da6b6f9a f79b9fc2 c0e09d2c d4b65900 d2394b61 aa3bb48c
7c731a14 68de0a17 346e34e1 7d58d870 7f845fac e35202bb 9d64b5ef
f29cbfc8 5f5c6d60 1d794c87 96c20e67 81dffed3 36fc1ff6 d3ae3193
dec00603 91acb681 1f1fbde3 8027a0ef 591e6b21 c6e31c5f 1fda66eb
05582b6b 0399c6a2 459cb2ab fd0d5d95 3447a927 86e194b2 89588e63
ef1b8b61 ad354bed 299b5a49 7c549d7a 56a74879 b7665a70 42fbcaf1
190d915f 945fef6c 0fcec14b 4afc403f 50774720 4d810c57 00de1692
6309352f 660f26a5 529a2f74 cb9d1044 0595dc25 d6d12fcc e84fc565
57217bd4 bc2d645a b4ca167f b812de7c acc3b942 7fc78212 985680b8
83bf7fee 7eae0199 1eb7a52a 0f4cbb01 f5a8e3c1 6c41350d c62be2c1
9cbd2b98 d9c9d268 7cd811db 7863779c 97e9a15b d6967d5e b21f972d
28ad9d43 7de41234 25249319 98f280a9 a9c799c3 3ff8f838 ca35bdde
bbb79cdc 2967946c c0f77995 411692e1 8519243d 5598bdb4 623a11dc

```
      97ca3889 49f32c65 db3fc6a4 7124bd5d 063549e5 0b0f8b03 0d3a9830
      e1e3bef5 cd428393 9d33a28c fdc3df89 640df257 c0fc2544 77a9c8ef
      f69b57cf f042e6fd 1ef3e293 c57beca2 cd61dc44 838014c2 08eda095
      e10d5e89 e705ff69 07047895 96e41969 96508797 71f58935 d768cdc3
      b55150cc a3693e28 33b62df3 4f1e2491 ef8c5824 f8a80cd8 6e65193a

   Input y' value:  0x00 951682f0 10b08932 b28b4a85 1ec79469 f9437fc4
      f9cfa8cc dec25c3c c847890c 65e1bcd2 df994b83 5b71e49c 0fc69e6d
      9ea5da9d bb020a9d fb2942dd 022fa962 fb0233de 016c8c80 e9387b0b
      28786785 523e68eb 7c008f81 b99ee3b5 d10a72e5 321a09b7 4b39c58b
      75d09d73 e4155b76 dc11d8dd 416b7fa6 3557fcdd b0a955f6 f5e0028d
      4af2150b fd757a89 8b548912 e2c0c6e5 70449113 fcee54cd a9cb8bfd
      7f182825 b371f069 61b62ca4 41bfcb3d 13ce6840 432bf8bc 4736003c
      64d695e9 84ddc2ef 4aee1747 044157fd 2f9b81c4 3eed97d3 45289899
      6d24c66a ad191dba 634f3e04 c89485e0 6f8308b8 afaedf1c 98b1a466
      deab2c15 81f96b6f 3c64d440 f2a16a62 75000cf3 8c09453b 5b9dc827
      8eabe442 92a154dc 69faa74a d76ca847 b786eb2f d686b9be 509fe24b
      8293861c c35d76be 88c27117 04bfe118 e4db1fad 86c2a642 4da6b3e5
      807258a2 d166d3e0 e43d15e3 b6464fb9 9f382f57 fd10499f 8c8e11df
      718c98a0 49bd0e5d 1301bc9e 6ccd0f06 3b06eb06 422afa46 9b5b529b
      8bba3d4c 6f219aff e4c57d73 10a92119 c98884c3 b6c0bbcc 113f6826
      b3ae70e3 bbbaadab 3ff8abf3 b905c231 38dfe385 134807fc c1f9c19e
      68c0ec46 8213bc9f 0387ca1f 4ffe406f da92d655 3cd4cfd5 0a2c895e
      85fe2540 9ffe8bb4 3b458f9b efab4d59 bee20e2f 01de48c2 affb03a9
      7ceede87 214e3bb9 0183303b 672e50b8 7b36a615 34034578 db0195fd
      81a46beb 55f75d20 049d044c 3fa5c367 8c783db3 120c2580 359a7b33
      cac5ce21 e4cecda9 e2e2d6d2 ff202ff4 3c1bb2d4 b5e53dae 010423ce

   Output e(P, Q):  0x276 43363d8d 5bcd15f0 b28c5097 eef37de3 71c67b59
      9c1edb17 6719754f f47a6ea7 5ad80480 617a2769 333ed4d1 89c7e90b
      36a7f3c9 873a13af f7cd1604 7a3c7fdc bf0c8471 bb510164 94c2e075
      1da368ed 41b9c2ae 12694b36 3abc6e6e 4f179278 71d4d68d b391f9be
      d8df5283 1e6efe0a fa3816fd 8387ca11 5d2cba0c 16a34759 0cea2fa9
      d358712d cf6c023a d4f5dfb9 ffd8cb2e 2778e6f6 95185e58 e2d81c2f
      0cebcc27 0e0fae67 8f758ef6 e319d634 f860a03f 10953e20 3c7e419f
      9eb8a0b1 af2890b2 b4182d8a f9fcbd41 0527e637 8c1a5aca 0ea8c085
      e2784e78 7c1e4712 fd891dc6 c20e5c72 a2b899cd 21aede46 358bc4c0
      17270f72 3cc29866 17fbabd4 b9c24ff7 44b9a032 1a2ebb39 2034d2ae
      4d415eb8 fd45330e 8fae1c93 f5d8849f 558e14dc f20e60b5 19122407
      cdb876b2 eb7897a8 9a602b0c 61093e87 d7dfc7c1 5cacf77a 9c71e9ea
      e43a6dfb fa4f68b8 9019f394 5caf71c1 9682779c 533e0ce0 002c16cf
      123d3aff b99103d0 4d3fdb2a 6a1682ab 3a34cc41 7f369850 a22db3a6
      c5b2d34b e0e97fa1 eb9e7d6e 45431e64 e6f0d20b f725cb43 23807531
      fca7994a 1447bb9c 1fc39fa8 87750f53 76c69ecb 8c58c70f f11a9f6d
      5f5f780a 831be9a6 1729ee17 78f050d6 c2afcdda 6cab13bd 952de323
      222fbaea 4638e029 5fbd5b8f 5c60ee91 429b078f 36f3dc2b e436236d
      5fd2282c ff620325 2456f683 4b9aea62 12d9d478 ddcdb1fc 0537db30
      5cc854f3 9aca8619 38d0aab5 53f0d449 f4ca5bc7 653101b8 f90b70c9
```

41bbcd9e 7775f810 2009c9c4 f268e0d1 5305c287 41d048b1 78f1f8ce
599c66aa e2401d1f 29e4987d e3d448ab b6462856 fe880511 8fe5d21a
e5ce44bb a7eef2ac 2cd4138a b98152a0 8028275f a08a6d3e fe9344d1
6945016d e83c7287 fff6b6aa 95cf2757 c27ca48e ce8b8397 da3eb8f9
95550853 43aba175 5660283b 5ea74e5e 52652818 4d100002 192e2440
61f917ef f3331617 cc10bd27 6167ec72 579a9421 40e1c538 858e84e5
b7585b17 b0eb8bd4 5b8915d4 48bfe906 1fed09aa 7b91e299 b394ccf8
7c5fbb4c 65ed6ab0 56f33c8b db82aea1 baad3f34 25b457e0 fe6a6193
0d4dd22a e7e1e394 6a655c72 564b4b65 d701a680 c6ccf3fe 88a1327d
3399523a 06e9d2ad 2ecbb9ed 9518634d 2201150f 7133381b d1311540
2f482e17 15b68876 f448c6e0 ff0dfec5 d3174ee8 026f7377 8692c617
82236066 e74b7099 6fa41dae b068c9a7 8f5d8b59 655985e0 bd6e3785
b5846f7e 721339e6 788731c2 4b80e26d d1182223 b669ad45 35e536f3
df711c31 f1aa0227 18fdb6e8 b7ad06c7 64785050 7b3c6a7e 454dc498
b86769bd 1fe57aa7 959f972e 8edb1326 b2a4233a 5d7e34d3 2c545b92
415b1944 3339642d 441fd7ef ab7e16ee aebc422d 92c614c4 85e5e18e
c3b990db 473a2dc2 7277c5cc 408c42d8 e1615de5 151a4eba 594c8d98
27f4b14e 38e3afd1 302aa4f8 6ad4d4ce 621f5de7 7b7f04b8 fa3ca11c
d701eee9 f7094ac3 24259670 5a30ccf2 dbf32491 074ee2ad e6595c11
71cf54d1 2ebad23e 383624b5 63c1eba7 a76e4002 f1cdd4a0 e1f8c24e
9227aa53 b56e077e d371babd 5935c20c d64ea0a0 45037849 98089570
7a54167f 68abcc8e 06d4d4b1 027b6cda 799ceaa2 b254b9ea 7f34a437
8417e12a 1bef63ac 3dd3b156 41f92e83 911ac76d 4ebfaaca bdf0f81e
ba243180 cf9a12d4 0f188a82 2d427533 a06969e9 b15ded57 85971636
f9f980e7 27b22628 97ccb665 a3857ab7 754c2bac 6627a5b1 8fc419da
14b8bb5f 2fe3d29c 62f8be1f bdd078d6 529087e3 002039da 3c193038
3d5582d2 9d061c87 d68acf45 618f7da2 65194f7f 02fae405 b0c08687
15df1c3f af6b71df 7331b79c 0e9fe708 c05ff1cb 11a14b68 684a04a3
38359484 cb994dc2 f4322928 17a1fd00 c12dae52 f8d151c6 fe34b3dc
c9ca7eb5 59d44baa f58cf94f 0646c469 4c49e3f1 81e0b1fb 382056aa
771c4118 33d7cfbf 8212bc96 ee303dfc 15623717 c814fd49 e9355f48
a480af44 bd7d2c94 031d9881 456a9eab 57e43962 668f1797 3cfa0e9a
075d94e3 db2ad9c0 6d4a2ee8 db81db28 62af5f2e ac946630 f61a8ab1
b868469a 9f7c2886 7c45cf5d 801bbbd7 4ecf2fce cd6c9939 865d7e27
dbe56fbf f5321cd8 29acf9c3 16c24778 ecc263e4 91cf8ed0 b23e1f2e
c9bf0036 e9f4442a d14db499 7daf8f4d bd4e7e7d 96fa32f0 474e04aa
dced225e 9b723462 900cb340 9320dfd8 54181422 d8135667 897d62cc
15a7e844 b72f713b 983e33f2 017417fc ef9d384f 58600b02 2fbd93bb
2b405b74 896acdc3 2b5c459a 4584aea9 3facf3d4 1c103d85 38fef57a
8ff66715 f84aab22 d6b2613c ee764038 579a5334 b212a15b 9ddc3914
524f85a8 89ed7cac 72bd23df ee819cb2 073f64a1 bd7a5b30 3119a571
5bae052c 7b71b023 0d941d65 bd186322 7935eb6b 0b648608 d044a254
9edddae2 46f935c6 82bf9200 9fb1d1e4 204a3946 8138e0dd 73b5c451
8c14faf2 69b7a50d 2c187367 748aff27 ee960ef2 92bbbcaa 86db4f6d
ca77dd66 1eca3fbf be11e012 87b40aeb 81fea4a2 0f8870dd fc910b80
e9fa9a7f 3949fbf9 e100c790 26256ffe 35b0b427 e9bf6364 616b3e44
45ba35fc 3c65c442 49544aea 848b2577 8a7d3ced 9a3393cf 6aab49f7
337cda34 dfa94fb7 6283645f 7551a49f 0fc0960d b6538505 374bc05e

```
ba88344a 27710517 00aaed45 6ebe28d2 f987ac36 3dd8a339 5617100c
00355ad3 0a22cd13 34e32157 be954065 4fa2841e b370760f 182d8d8f
11380d0d d358c2be b3c2a9b1 c791a142 36dec80d eaabf2fa d5076703
649c4c67 091e366c 593d5ad7 63598a8a 64bb95c9 1c5d4c1c f3941094
9bc3e0ae 5eaa0e4c afb2e520 3da18549 5f81c36e 309e807a d6b5679b
1978ba41 582c3c75 0d703e95 d77e479d 433867d0 fbe6c074 51984815
0fcec435 1c912ded 64844bf1 e2e966fa f64c520e 89517fa9 2996536a
f096c440 7da5c1b1 bee0f040 cb440d00 95fb43a3 e1ee5846 ccdb0e06
8a8086c0 3c2668ab e161554e fb4f0b19 75df1d81 8cd38614 2e343186
162cce43 c9d05212 ddcad50e 10bdd365 9007f156 dcfd40e2 7b36baef
4cd5184b 71f5c342 bf3b39e3 d0f145f6 515f2eed b926c999 53af0491
68376477 632b7435 5d180b72 109e4049 323d0517 67ec3b58 58c545d8
ed85e074 c839558e efaecfea 112062bb 9e000d82 2fad823c 2f59387a
62687738 fb74c5e1 24b75fc6 384bd4e9 450ad801 db149c4e 89a02a14
45504e84 e9d44b83 04ce52ea 513e923e 40c833ee 663a23fc 64365cc3
ef8abecd e2c35900 9712dec9 511e11e2 82685a62 5580ce56 e517ab30
10cc21d2 b4a656e3 a5babb90 0a1678ee ef9ad5a0 46a8cf8e d82c25b4
29c08d2f f6f3f515 7d589d84 230378a4 5649de30 2ddb5da6 b05790df
8fe450e7 bc583abc 7aaa9b83 1fa93a64 a12c379a f6cab1c9 a6cdf3ea
44a651ce c8d9f607 ab050904 5c6583c3 c3827c0b 990c4d93 63139193
cbdf158a 50b8f989 166865db bb6d6658 7b7f2785 f2fd4810 ceb085af
5cabc661 2e5544f1 d0de7ceb a1803095 03e62544 9749619c 619dd15e
190f46bf 75788250 304f3214 460acd9b a78d1626 b75ebeeb 8f08b9d1
cb6183d1 fcf37b1f ff2cd490 9b05070c cf3d2275 266a42a3 1e5204d1
55ce1587 7cd4e0c6 afa46a05 50be923f 66b6be18 93383dd7 9d231d99
7fa9a89d 039b96f0 a3bb28d9 8080706f dd87ebba ffd5199a a5c0a01e
351c6d9e 4a93c3cd 4bc9eeba afb0bffe 7d9e2d79 6979dedc 16d9964a
449de47c a4b04db2 a192e96e 396ed228 6c8cfc6d 584dbbea b100d82e
97ba3c88 e29a2e56 3e30760d b21ae180 ef884967 22ac5de8 77b85a80
5becaf51 f0cf000e 27fc525d 2f665471 911153a6 d19cbc50 70c8404b
fde12034 254a1ab0 38d6626d c83e9652 b497cfec 9485a9ec 756b7c87
0995de42 0e12b91c dc8664e4 1c760c71 d519381e ebc66a31 5b21429d
11591a5d 6994abb8 d2f511c6 4f500fbe 7eaea507 b7492584 9338be99
84388212 4a2c7dea c03ff2b8 b53e0459 dcf41296 e5a8574d c008f089
f47938c2 789367c5 e27fc634 434a331c 8a227b49 b31ee9ec 5425407e
2c108f12 7ca8324c f96dce1a 3227f6ec 4aea2d9b afa8c2e3 3de6b323
76c2f324 a351e867 8e552ab8 077d1294 ec0954d2 b134cddc f2937952
a2c27f6a 5cb719b6 a14c3908 4095b5d4 2ae8fce2 a7a13a1d b1d9c7b2
e1e07c0e 7dbde19c 62c5eb6b 030faa55 c33b202a 56c15dd9 9fc289e1
4f33ba8b b68c36b0 944c14ef 52cc4078 426ddba5 abb05f32 05882c64
4adb6fbe 0fe6b192 0f505b4a b163def7 8363d152 ca8855ad f93a84bc
d7743570 328ac022 cb287205 8f84e35d 416cb7d6 5a6a5a6c 26c54747
7a86dfa6 c3f59dbb e3d1208c c3157177 7af026b2 4f8aab6c 167f6a60
4708e34a 15c77c9c 5c1bf246 f287a8ab 4f128cf4 8f84cb96 43df8f6e
ee07de97 31c1b614 93a07041 b844e507 f8371476 d544c111 8553b3d6
c5bbfea5 89e4939f 1a0786cb b3432aa8 23dd8b1f 360e5761 48aacd5d
859f8c6d b4d9ead2 5de7e396 2cfd47cd 40b33d31 51e1ab08 f521047a
51046218 78a246b5 a0a675e6 d4c3315d 18a39dc8 456759fb b9a6bda2
```

```
      cc73e391 d050b7dd 4c1a2d73 e07b32fb e7201194 ac67c827 570c17f3
      add5e58e d5951b91 64483445 d8c448ab 486dde3a c677443b 6ba8f7ac
      f2617058 b65500a2 3caf76d8 3e0211d1 966b89a5 5a8dfaec c979a05b
      db64c10b aa802daa fb1cedeb d8eb3bbf 3eda50e5 a1e97398 b1e7112c
      d2d29d8e b71d66ad 1d39f85a f41da1cd 3a3330fc 8c7adfcc 91a13ce5
      ab5b29cf 0eb2cc0d 747fe5f1 588e10d1 be6ed505 2cf5ba84 a9ff273c
      7b14c176 3de0b128 f1a37cf7 46b27933 880550c2 56a0ac85 49a52ef3
      fc0bc8a3 eed01024 ddffe6fc 75d8e8ee 2fc302d4 aa3f556d c16852cb
      53a373a7 555b99a1 e914cbf8 55da764c
```

Authors' Addresses

Shoko Yonezawa
Lepidum

Email: yonezawa@lepidum.co.jp

Sakae Chikara
NTT TechnoCross

Email: chikara.sakae@po.ntt-tx.co.jp

Tetsutaro Kobayashi
NTT

Email: kobayashi.tetsutaro@lab.ntt.co.jp

Tsunekazu Saito
NTT

Email: saito.tsunekazu@lab.ntt.co.jp