Abstract

Pairing-based cryptography, a subfield of elliptic curve cryptography, has received attention due to its flexible and practical functionality. Pairings are special maps defined using elliptic curves and it can be applied to construct several cryptographic protocols such as identity-based encryption, attribute-based encryption, and so on. At CRYPTO 2016, Kim and Barbulescu proposed an efficient number field sieve algorithm named exTNFS for the discrete logarithm problem in a finite field. Several types of pairing-friendly curves such as Barreto-Naehrig curves are affected by the attack. In particular, a Barreto-Naehrig curve with a 254-bit characteristic was adopted by a lot of cryptographic libraries as a parameter of 128-bit security, however, it ensures no more than the 100-bit security level due to the effect of the attack. In this memo, we list the security levels of certain pairing-friendly curves, and motivate our choices of curves. First, we summarize the adoption status of pairing-friendly curves in standards, libraries and applications, and classify them in the 128-bit, 192-bit, and 256-bit security levels. Then, from the viewpoints of "security" and "widely used", we select the recommended pairing-friendly curves considering exTNFS.

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 20 May 2021.

Copyright Notice

Copyright (c) 2020 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 (https://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
   1.1. Pairing-based Cryptography
   1.2. Applications of Pairing-based Cryptography
   1.3. Motivation and Contribution
   1.4. Requirements Terminology
2. Preliminaries
   2.1. Elliptic Curves
   2.2. Pairings
   2.3. Barreto-Naehrig Curves
   2.4. Barreto-Lynn-Scott Curves
   2.5. Representation Convention for an Extension Field
3. Security of Pairing-Friendly Curves
   3.1. Evaluating the Security of Pairing-Friendly Curves
   3.2. Impact of Recent Attacks
4. Selection of Pairing-Friendly Curves
   4.1. Adoption Status of Pairing-friendly Curves
      4.1.1. International Standards
      4.1.2. Cryptographic Libraries
      4.1.3. Applications
   4.2. For 128-bit Security
      4.2.1. BLS Curves for the 128-bit security level
      4.2.2. BN Curves for the 128-bit security level
   4.3. For 256-bit Security
5. Security Considerations
6. IANA Considerations
7. Acknowledgements
1. Introduction

1.1. Pairing-based Cryptography

Elliptic curve cryptography is an important area in currently deployed cryptography. The cryptographic algorithms based on elliptic curve cryptography, such as the Elliptic Curve Digital Signature Algorithm (ECDSA), are widely used in many applications.

Pairing-based cryptography, a subfield of elliptic curve cryptography, has attracted much attention due to its flexible and practical functionality. Pairings are special maps defined using elliptic curves. Pairings are fundamental in the construction of 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 are currently in practical use.

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

1.2. Applications of Pairing-based Cryptography

Several applications using pairing-based cryptography have already been standardized and deployed. We list here some examples of applications available in the real world.

IETF published RFCs for pairing-based cryptography such as Identity-Based Cryptography [RFC5091], Sakai-Kasahara Key Encryption (SAKKE) [RFC6508], and Identity-Based Authenticated Key Exchange (IBAKE) [RFC6539]. SAKKE is applied to Multimedia Internet KEYing (MIKEY) [RFC6509] and used in 3GPP [SAKKE].
Pairing-based key agreement protocols are standardized in ISO/IEC [ISOIEC11770-3]. In [ISOIEC11770-3], a key agreement scheme by Joux [Joux00], identity-based key agreement schemes by Smart-Chen-Cheng [CCS07] and Fujioka-Suzuki-Ustaoglu [FSU10] are specified.

MIRACL implements M-Pin, a multi-factor authentication protocol [M-Pin]. The M-Pin protocol includes a type of zero-knowledge proof, where pairings are used for its construction.

The Trusted Computing Group (TCG) specified the Elliptic Curve Direct Anonymous Attestation (ECDAA) in the specification of a Trusted Platform Module (TPM) [TPM]. ECDAA is a protocol for proving the attestation held by a TPM to a verifier without revealing the attestation held by that TPM. Pairings are used in the construction of ECDAA. FIDO Alliance [FIDO] and W3C [W3C] also published an ECDAA algorithm similar to TCG.

Intel introduced Intel Enhanced Privacy ID (EPID) that enables remote attestation of a hardware device while preserving the privacy of the device as part of the functionality of Intel Software Guard Extensions (SGX) [EPID]. They extended TPM ECDAA to realize such functionality. A pairing-based EPID was proposed [BL10] and distributed along with Intel SGX applications.

Zcash implemented their own zero-knowledge proof algorithm named Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARKs) [Zcash]. zk-SNARKs are used for protecting the privacy of transactions of Zcash. They use pairings to construct zk-SNARKs.

Cloudflare introduced Geo Key Manager [Cloudflare] to restrict distribution of customers' private keys to a subset of their data centers. To achieve this functionality, ABE is used, and pairings take a role as a building block. In addition, Cloudflare published a new cryptographic library, the Cloudflare Interoperable, Reusable Cryptographic Library (CIRCL) [CIRCL] in 2019. They plan to include securely implemented subroutines for pairing computations on certain secure pairing-friendly curves in CIRCL.

Currently, Boneh-Lynn-Shacham (BLS) signature schemes are being standardized [I-D.boneh-bls-signature] and utilized in several blockchain projects such as Ethereum [Ethereum], Algorand [Algorand], Chia Network [Chia], and DFINITY [DFINITY]. The aggregation functionality of BLS signatures is effective for their applications of decentralization and scalability.

### 1.3. Motivation and Contribution

At CRYPTO 2016, Kim and Barbulescu proposed an efficient number field sieve (NFS) algorithm for the discrete logarithm problem in a finite field $GF(p^k)$ [KB16]. The attack improves the polynomial
selection that is the first step in the number field sieve algorithm for discrete logarithms in GF(p^k). The idea is applicable when the embedding degree k is a composite that satisfies k = i*j (gcd (i, j) = 1, i, j > 1). The basic idea is based on the equality GF(p^k) = (GF(p^i)^j) and one of the improvement for reducing the amount of cost for solving the discrete logarithm problem is using sub-field calculation. Several types of pairing-friendly curves such as Barreto-Naehrig curves (BN curves)[BN05] and Barreto-Lynn-Scott curves (BLS curves)[BLS02] are affected by the attack, since a pairing-friendly curve suitable for cryptographic applications requires that the discrete logarithm problem is sufficiently difficult. Please refer to [KB16] for detailed ideas and calculation algorithms of the attack by Kim. In particular, BN254, which is a BN curve with a 254-bit characteristic effective for pairing calculations, was adopted by a lot of cryptographic libraries as a parameter of the 128-bit security level, however, BN254 ensures no more than the 100-bit security level due to the effect of the attack, where the security levels described in this memo correspond to the security strength of NIST recommendation [NIST].

To resolve this effect immediately, several research groups and implementers re-evaluated the security of pairing-friendly curves and they respectively proposed various curves that are secure against the attack [BD18] [BLS12-381].

In this memo, we list the security levels of certain pairing-friendly curves, and motivate our choices of curves. First, we summarize the adoption status of pairing-friendly curves in international standards, libraries and applications, and classify them in the 128-bit, 192-bit, and 256-bit security levels. Then, from the viewpoints of "security" and "widely used", pairing-friendly curves corresponding to each security level are selected in accordance with the security evaluation by Barbulescu and Duquesne [BD18].

As a result, we recommend the BLS curve with 381-bit characteristic of embedding degree 12 and the BN curve with the 462-bit characteristic for the 128-bit security level, and the BLS curves of embedding degree 48 with the 581-bit characteristic for the 256-bit security level. This memo shows their specific test vectors.

1.4. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.
2. Preliminaries

2.1. Elliptic Curves

Let \( p \) be a prime number and \( q = p^n \) for a natural number \( n > 0 \), where \( p \) at least 5. Let \( GF(q) \) be a finite field. The curve defined by the following equation \( E \) is called an elliptic curve:

\[
E : y^2 = x^3 + a * x + b,
\]

and \( a \) and \( b \) in \( GF(q) \) satisfy the discriminant inequality \( 4 * a^3 + 27 * b^2 \neq 0 \mod q \). This is called the Weierstrass normal form of an elliptic curve.

A solution \((x,y)\) to the equation \( E \) can be thought of as a point on the corresponding curve. For a natural number \( k \), we define the set of \((GF(q^k))\)-rational points of \( E \), denoted by \( E(GF(q^k)) \), to be the set of all solutions \((x,y)\) in \( GF(q^k) \), together with a 'point at infinity' \( O_E \), which is defined to lie on every vertical line passing through the curve \( E \).

The set \( E(GF(q^k)) \) forms a group under a group law that can be defined geometrically as follows. For \( P \) and \( Q \) in \( E(GF(q^k)) \) define \( P + Q \) to be the reflection around the x-axis of the unique third point \( R \) of intersection of the straight line passing through \( P \) and \( Q \) with the curve \( E \). If the straight line is tangent to \( E \), we say that it passes through that point twice. The identity of this group is the point at infinity \( O_E \). We also define scalar multiplication \([K]P\) for a positive integer \( K \) as the point \( P \) added to itself \((K-1)\) times. Here, \([0]P\) becomes the point at infinity \( O_E \) and the relation \([-K]P = -([K]P)\) is satisfied.

2.2. Pairings

A pairing is a bilinear map defined on two subgroups of rational points of an elliptic curve. Examples include the Weil pairing, the Tate pairing, the optimal Ate pairing \([\text{Ver09}]\), and so on. The optimal Ate pairing is considered to be the most efficient to compute and is the one that is most commonly used for practical implementation.

Let \( E \) be an elliptic curve defined over a prime field \( GF(p) \). Let \( k \) be the minimum integer for which \( r \) is a divisor of \( p^k - 1 \); this is called the embedding degree of \( E \) over \( GF(p) \). Let \( G_1 \) be a cyclic subgroup of \( E(GF(p)) \) of order \( r \), there also exists a cyclic subgroup of \( E(GF(p^k)) \) of order \( r \), define this to be \( G_2 \). Let \( d \) be a divisor of \( k \) and \( E' \) be an elliptic curve defined over \( GF(p^k/d) \). If an isomorphism from \( E' \) to \( E(GF(p^k)) \) exists, then \( E' \) is called the twist of \( E \). It can sometimes be convenient for efficiency to do the computations of \( G_2 \) in the twist \( E' \), and so consider \( G_2 \) to instead
be a subgroup of $E'$. Let $G_T$ be an order $r$ subgroup of the multiplicative group $(\mathbb{GF}(p^k))^*$; this exists by definition of $k$.

A pairing is defined as a bilinear map $e: (G_1, G_2) \rightarrow G_T$ satisfying the following properties:


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 $T = O_E$.

In applications, it is also necessary that for any $S$ in $G_1$ and $T$ in $G_2$, this bilinear map is efficiently computable.

We define some of the terminology used in this memo as follows:

- $\mathbb{GF}(p)$: a finite field with characteristic $p$.
- $\mathbb{GF}(p^k)$: an extension field of degree $k$.
- $(\mathbb{GF}(p))^*$: a multiplicative group of $\mathbb{GF}(p)$.
- $(\mathbb{GF}(p^k))^*$: a multiplicative group of $\mathbb{GF}(p^k)$.
- $b$: a primitive element of the multiplicative group $(\mathbb{GF}(p))^*$.
- $O_E$: the point at infinity over an elliptic curve $E$.
- $E(\mathbb{GF}(p^k))$: the group of $\mathbb{GF}(p^k)$-rational points of $E$.
- $#E(\mathbb{GF}(p^k))$: the number of $\mathbb{GF}(p^k)$-rational points of $E$.
- $r$: the order of $G_1$ and $G_2$.
- $BP$: a point in $G_1$. (The 'base point' of a cyclic subgroup of $G_1$)
- $h$: the cofactor $h = \#E(\mathbb{GF}(p)) / r$, where $\gcd(h, r)=1$.

2.3. Barreto-Naehrig Curves

A BN curve [BN05] is a family of pairing-friendly curves proposed in 2005. A pairing over BN curves constructs optimal Ate pairings.

A BN curve is defined by elliptic curves $E$ and $E'$ parameterized by a well-chosen integer $t$. $E$ is defined over $\mathbb{GF}(p)$, where $p$ is a prime number and at least 5, and $E(\mathbb{GF}(p))$ has a subgroup of prime order $r$. The characteristic $p$ and the order $r$ are parameterized by
for an integer $t$.

The elliptic curve $E$ has an equation of the form $E: y^2 = x^3 + b$, where $b$ is a primitive element of the multiplicative group $(\mathbb{GF}(p))^*$ of order $(p - 1)$.

In the case of BN curves, we can use twists of the degree 6. If $m$ is an element that is neither a square nor a cube in an extension field $\mathbb{GF}(p^2)$, the twist $E'$ of $E$ is defined over an extension field $\mathbb{GF}(p^2)$ by the equation $E': y^2 = x^3 + b'$ with $b' = b / m$ or $b' = b * m$. BN curves are called D-type if $b' = b / m$, and M-type if $b' = b * m$. The embedding degree $k$ is 12.

A pairing $e$ is defined by taking $G_1$ as a subgroup of $E(\mathbb{GF}(p))$ of order $r$, $G_2$ as a subgroup of $E'(\mathbb{GF}(p^2))$, and $G_T$ as a subgroup of a multiplicative group $(\mathbb{GF}(p^12))^*$ of order $r$.

### 2.4. Barreto-Lynn-Scott Curves

A BLS curve [BLS02] is another family of pairing-friendly curves proposed in 2002. Similar to BN curves, a pairing over BLS curves constructs optimal Ate pairings.

A BLS curve is defined by elliptic curves $E$ and $E'$ parameterized by a well-chosen integer $t$. $E$ is defined over a finite field $\mathbb{GF}(p)$ by an equation of the form $E: y^2 = x^3 + b$, and its twist $E': y^2 = x^3 + b'$, is defined in the same way as BN curves. In contrast to BN curves, $E(\mathbb{GF}(p))$ does not have a prime order. Instead, its order is divisible by a large parameterized prime $r$ and denoted by $h * r$ with cofactor $h$. The pairing is defined on the $r$-torsion points. In the same way as BN curves, BLS curves can be categorized as D-type and M-type.

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

In BLS curves, parameters $p$ 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 a well chosen integer \( t \) where \( t \) must be \( 1 \pmod{3} \).

A pairing \( e \) is defined by taking \( G_1 \) as a subgroup of \( E(GF(p)) \) of order \( r \), \( G_2 \) as an order \( r \) subgroup of \( E'(GF(p^2)) \) for BLS12 and of \( E'(GF(p^8)) \) for BLS48, and \( G_T \) as an order \( r \) subgroup of a multiplicative group \( (GF(p^{12}))^* \) for BLS12 and of a multiplicative group \( (GF(p^{48}))^* \) for BLS48.

### 2.5. Representation Convention for an Extension Field

Pairing-friendly curves use a tower of some extension fields. In order to encode an element of an extension field, focusing on interoperability, we adopt the representation convention shown in Appendix J.4 of [I-D.ietf-lwig-curve-representations] as a standard and effective method. Note that the big-endian encoding is used for an element in \( GF(p) \) which follows to mcl [mcl], ISO/IEC 15946-5 [ISOIEC15946-5] and etc.

Let \( GF(p) \) be a finite field of characteristic \( p \) and \( GF(p^d) = GF(p)(i) \) be an extension field of \( GF(p) \) of degree \( d \).

For an element \( s \) in \( GF(p^d) \) such that \( s = s_0 + s_1 * i + ... + s_{(d-1)} * i^{(d-1)} \) in the basefield \( GF(p) \), \( s \) is represented as octet string by \( \text{oct}(s) = s_0 || s_1 || ... || s_{(d-1)} \).

Let \( GF(p^{d'}) = GF(p^d)(j) \) be an extension field of \( GF(p^d) \) of degree \( d' / d \).

For an element \( s' \) in \( GF(p^{d'}) \) such that \( s' = s'_0 + s'_1 * j + ... + s'_{(d' / d - 1)} * j^{(d' / d - 1)} \) in the basefield \( GF(p^d) \), \( s' \) is represented as integer by \( \text{oct}(s') = \text{oct}(s'_0) || \text{oct}(s'_1) || ... || \text{oct}(s'_{(d' / d - 1)}) \), where \( \text{oct}(s'_0), ..., \text{oct}(s'_{(d' / d - 1)}) \) are octet strings encoded by above convention.

In general, one can define encoding between integer and an element of any finite field tower by inductively applying the above convention.

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

When applications communicate elements in an extension field, using the compression method [MP04] may be more effective. In that case, care for interoperability must be taken.
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 used for proving the security of pairing-based cryptography. Such problems include the computational bilinear Diffie-Hellman (CBDH) problem, the bilinear Diffie-Hellman (BDH) problem, the decision bilinear Diffie-Hellman (DBDH) problem, the gap DBDH problem, etc. [ECRYPT]. Almost all of these variants are reduced to the hardness of discrete logarithm problems described above and are believed to be easier than the discrete logarithm problems.

Although it would be sufficient to attack any of these problems to attack pairing-based cryptography, the only known attacks thus far attack the discrete logarithm problem directly, so we focus on the discrete logarithm in this memo.

The security levels of pairing-friendly curves are estimated by the computational cost of the most efficient algorithm for solving the above discrete logarithm problems. The best-known algorithms for solving the discrete logarithm problems are based on Pollard's rho algorithm [Pollard78] and Index Calculus [HR83]. To make index calculus algorithms more efficient, number field sieve (NFS) algorithms are utilized.

3.2. Impact of Recent Attacks

In 2016, Kim and Barbulescu proposed a new variant of the NFS algorithms, the extended tower number field sieve (exTNFS), which drastically reduces the complexity of solving FFDLP [KB16]. The exTNFS improves the polynomial selection that is the first step in the number field sieve algorithm for discrete logarithms in $GF(p^k)$. The idea is applicable when the embedding degree $k$ is a composite that satisfies $k = i \times j$ ($gcd(i, j) = 1, i, j > 1$). Since the above condition is satisfied especially when $k = 2^n \times 3^m$ ($n, m > 1$), BN curves and BLS curves whose embedding degree is divisible by 6 are affected by the exTNFS. The basic idea of the exTNFS is based on the equality $GF(p^k) = (GF(p^i)^j)$ and one of the improvement for reducing the amount of cost for solving FFDLP is using sub-field calculation. Please refer to [KB16] for detailed ideas and calculation algorithms of exTNFS. Due to exTNFS, the security levels
of certain pairing-friendly curves asymptotically dropped down. For instance, Barbulescu and Duquesne estimated that the security of the BN curves, which had been believed to provide 128-bit security (BN256, for example) was reduced to approximately 100 bits [BD18]. Here, the security levels described in this memo correspond to the security strength of NIST recommendation [NIST].

There has since been research into the minimum bit length of the parameters of pairing-friendly curves for each security level when applying exTNFS as an attacking method for FFDLP. For 128-bit security, Barbulescu and Duquesne estimated the minimum bit length of $p$ of BN curves and BLS12 curves after exTNFS as 461 bits [BD18]. For 256-bit security, Kiyomura et al. estimated the minimum bit length of $p^k$ of BLS48 curves as 27,410 bits, which indicated 572 bits of $p$ [KIK17].

4. Selection of Pairing-Friendly Curves

In this section, we introduce some of the known secure pairing-friendly curves that consider the impact of exTNFS.

First, we show the adoption status of pairing-friendly curves in standards, libraries and applications, and classify them in accordance with the 128-bit, 192-bit, and 256-bit security levels. Then, from the viewpoints of "security" and "widely used", pairing-friendly curves corresponding to each security level are selected and their parameters are indicated.

In our selection policy, it is important that selected curves are shown in peer-reviewed papers for security and that they are widely used in cryptographic libraries. In addition, "efficiency" is one of the important aspects but greatly dependant on implementations, so we choose to prioritize "security" and "widely used" over "efficiency" in consideration of future interconnections and interoperability over the internet.

As a result, we recommend the BLS curve with 381-bit characteristic of embedding degree 12 and the BN curve with the 462-bit characteristic for the 128-bit security level, and the BLS curves of embedding degree 48 with the 581-bit characteristic for the 256-bit security level. On the other hand, we do not show the parameters for 192-bit security here because there are no curves that match our selection policy.

4.1. Adoption Status of Pairing-friendly Curves

We show the pairing-friendly curves that have been selected by existing standards, cryptographic libraries, and applications.
Table 1 summarizes the adoption status of pairing-friendly curves. In this table, "Arnd" is an abbreviation for "Around". The curves categorized as 'Arnd 128-bit', 'Arnd 192-bit' and 'Arnd 256-bit' for each label show that their security levels are within the range of plus/minus 5 bits for each security level. Other labels shown with '~-' mean that the security level of the categorized curve is outside the range of each security level. Specifically, the security level of the categorized curves is more than the previous column and is less than the next column. The details are described as the following subsections. A BN curve with a XXX-bit characteristic p is denoted as BNXXX and a BLS curve of embedding degree k with a XXX-bit p is denoted as BLSk_XXX.

Table 1 omits parameters with security levels below the "Arnd 128-bit" range due to space limitations and viewpoints of secure usage of parameters. On the other hand, indicating which standards, libraries, and applications use these lower security level parameters would be useful information for implementers, therefore Appendix D shows these parameters. In addition, the full version of Table 1 is available at https://lepidum.co.jp/blog/2020-03-27/ietf-draft-pfc/.

In Table 1, the security level for each curve is evaluated in accordance with [BD18],[GMT19],[MAF19] and [FK18]. Note that the Freeman curves and MNT curves are not included in this table because [BD18] does not show the security levels of these curves.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Curve Type</th>
<th>Security Levels (bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Arnd 128</td>
</tr>
<tr>
<td>Standard</td>
<td>ISO/IEC</td>
<td>BN384</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN512I</td>
<td>X</td>
</tr>
<tr>
<td>TCG</td>
<td></td>
<td>BN638</td>
<td>X</td>
</tr>
<tr>
<td>FIDO/W3C</td>
<td></td>
<td>BN512I</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN638</td>
<td>X</td>
</tr>
<tr>
<td>Library</td>
<td>mcl</td>
<td>BLS12_381</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN382M</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN462</td>
<td>X</td>
</tr>
<tr>
<td>RELIC</td>
<td></td>
<td>BLS12_381</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLS12_446</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLS12_455</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLS12_638</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLS24_477</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLS48_575</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN382R</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN446</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BN638</td>
<td>X</td>
</tr>
<tr>
<td>Category</td>
<td>Name</td>
<td>Curve Type</td>
<td>Security Levels (bit)</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Arnd 128</td>
</tr>
<tr>
<td>AMCL</td>
<td>CP8_544</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>K54_569</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KSS18_508</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OT8_511</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS12_381</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS12_383</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS12_461</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS24_479</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS48_556</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BN512I</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Kyushu Univ.</td>
<td>BLS48_581</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIRACL</td>
<td>BLS12_381</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS12_383</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS12_461</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS24_479</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS48_556</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BLS48_581</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BN462</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BN512I</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ADjoint</td>
<td>BLS12_381</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BN462</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>bls12377js</td>
<td>BLS12_377</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>Zcash</td>
<td>BLS12_381</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Ethereum</td>
<td>BLS12_381</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Chia Network</td>
<td>BLS12_381</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>DFINITY</td>
<td>BLS12_381</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Adoption Status of Pairing-Friendly Curves

4.1.1. International Standards

ISO/IEC 15946 series specifies public-key cryptographic techniques based on elliptic curves. ISO/IEC 15946-5 [ISO/IEC15946-5] shows numerical examples of MNT curves [MNT01] with 160-bit p and 256-bit p, Freeman curves [Freeman06] with 224-bit p and 256-bit p, and BN curves with 160-bit p, 192-bit p, 224-bit p, 256-bit p, 384-bit p, and 512-bit p. These parameters do not take into account the effects of the exTNFS. On the other hand, the parameters may be revised in future versions since ISO/IEC 15946-5 is currently under development. As described below, BN curves with 256-bit p and 512-
bit p specified in ISO/IEC 15946-5 used by other standards and libraries, these curves are especially denoted as BN256I and BN512I. The suffix 'I' of BN256I and BN512I are given from the initials of the standard name ISO. TCG adopts the BN256I and a BN curve with 638-bit p specified by their own[TPM]. FIDO Alliance [FIDO] and W3C [W3C] adopt BN256I, BN512I, the BN638 by TCG, and the BN curve with 256-bit p proposed by Devegili et al.[DSD07] (named BN256D). The suffix 'D' of BN256D is given from the initials of the first author's name of the paper which proposed the parameter.

4.1.2. Cryptographic Libraries

There are a lot of cryptographic libraries that support pairing calculations.

PBC is a library for pairing-based cryptography published by Stanford University that supports BN curves, MNT curves, Freeman curves, and supersingular curves [PBC]. Users can generate pairing parameters by using PBC and use pairing operations with the generated parameters.

mcl[mcl] is a library for pairing-based cryptography that supports four BN curves and BLS12_381 [GMT19]. These BN curves include BN254 proposed by Nogami et al. [NASKM08] (named BN254N), BN_SNARK1 suitable for SNARK applications[libsnark], BN382M, and BN462. The suffix 'N' of BN256N and the suffix 'M' of BN382M are respectively given from the initials of the first author's name of the proposed paper and the library's name mcl. Kyushu University published a library that supports the BLS48_581 [BLS48]. The University of Tsukuba Elliptic Curve and Pairing Library (TEPLA) [TEPLA] supports two BN curves, BN254N and BN254 proposed by Beuchat et al. [BGMORT10] (named BN254B). The suffix 'B' of BN254B is given from the initials of the first author's name of the proposed paper. Intel published a cryptographic library named Intel Integrated Performance Primitives (Intel-IPP) [Intel-IPP] and the library supports BN256I.

RELIC [RELIC] uses various types of pairing-friendly curves including six BN curves (BN158, BN254R, BN256R, BN382R, BN446, and BN638), where BN254R, BN256R, and BN382R are RELIC specific parameters that are different from BN254N, BN254B, BN256I, BN256D, and BN382M. The suffix 'R' of BN382R is given from the initials of the library's name RELIC. In addition, RELIC supports six BLS curves (BLS12_381, BLS12_446, BLS12_445, BLS12_638, BLS24_477, and BLS48_575 [MAF19]), Cocks-Pinch curves of embedding degree 8 with 544-bit p[GMT19], pairing-friendly curves constructed by Scott et al. [SG19] based on Kachisa-Scott-Schaefer curves with embedding degree 54 with 569-bit p (named KS4_569)[MAF19], a KSS curve.
of embedding degree 18 with 508-bit p (named KSS18_508) [AFKMR12], Optimal TNFS-secure curve [FM19] of embedding degree 8 with 511-bit p(OT8_511), and a supersingular curve [S86] with 1536-bit p (SS_1536).

Apache Milagro Crypto Library (AMCL) [AMCL] supports four BLS curves (BLS12_381, BLS12_461, BLS24_479 and BLS48_556) and four BN curves (BN254N, BN254CX proposed by CertiVox, BN256I, and BN512I). In addition to AMCL’s supported curves, MIRACL [MIRACL] supports BN462 and BLS48_581.

Adjoint published a library that supports the BLS12_381 and six BN curves (BN_SNARK1, BN254B, BN254N, BN254S1, BN254S2, and BN462) [AdjointLib], where BN254S1 and BN254S2 are BN curves adopted by an old version of AMCL [AMCLv2]. The suffix 'S' of BN254S1 and BN254S2 are given from the initials of developer's name because he proposed these parameters.

The Celo foundation published the bls12377js library [bls12377js]. The supported curve is the BLS12_377 curve which is shown in [BCGMMW20].

4.1.3. Applications

Zcash uses a BN curve (named BN128) in their library libsnark [libsnark]. In response to the exTNFS attacks, they proposed new parameters using BLS12_381 [BLS12-381] [GMT19] and published its experimental implementation [zkcrypto].

Ethereum 2.0 adopted BLS12_381 and uses the implementation by Meyer [pureGo-bls]. Chia Network published their implementation [Chia] by integrating the RELIC toolkit [RELIC]. DFINITY uses mcl, and Algorand published an implementation which supports BLS12_381.

4.2. For 128-bit Security

Table 1 shows a lot of cases of adopting BN and BLS curves. Among them, BLS12_381 and BN462 match our selection policy. Especially, the one that best matches the policy is BLS12_381 from the viewpoint of "widely used" and "efficiency", so we introduce the parameters of BLS12_381 in this memo.

On the other hand, from the viewpoint of the future use, the parameter of BN462 is also introduced. As shown in recent security evaluations for BLS12_381[BD18] [GMT19], its security level close to 128-bit but it is less than 128-bit. If the attack is improved even a little, BLS12_381 will not be suitable for the curve of the 128-bit security level. As curves of 128-bit security level are currently the most widely used, we recommend both BLS12-381 and
BN462 in this memo in order to have a more efficient and a more prudent option respectively.

4.2.1. BLS Curves for the 128-bit security level

In this part, we introduce the parameters of the Barreto-Lynn-Scott curve of embedding degree 12 with 381-bit $p$ that is adopted by a lot of applications such as Zcash [Zcash], Ethereum [Ethereum], and so on.

The BLS12_381 curve is shown in [BLS12-381] and it is defined by the parameter

$$t = -2^{63} - 2^{62} - 2^{60} - 2^{57} - 2^{48} - 2^{16}$$

where the size of $p$ becomes 381-bit length.

For the finite field $GF(p)$, the towers of extension field $GF(p^2)$, $GF(p^6)$ and $GF(p^{12})$ are defined by indeterminates $u$, $v$, and $w$ as follows:

$$GF(p^2) = GF(p)[u] / (u^2 + 1)$$
$$GF(p^6) = GF(p^2)[v] / (v^3 - u - 1)$$
$$GF(p^{12}) = GF(p^6)[w] / (w^2 - v).$$

Defined by $t$, the elliptic curve $E$ and its twist $E'$ are represented by $E$: $y^2 = x^3 + 4$ and $E'$: $y^2 = x^3 + 4(u + 1)$. BLS12_381 is categorized as M-type.

We have to note that the security level of this pairing is expected to be 126 rather than 128 bits [GMT19].

Parameters of BLS12_381 are given as follows.

* $G_1$ is the largest prime-order subgroup of $E(GF(p))$
  - $BP = (x,y)$: a 'base point', i.e., a generator of $G_1$

* $G_2$ is an $r$-order subgroup of $E'(GF(p^2))$
  - $BP' = (x',y')$: a 'base point', i.e., a generator of $G_2$
    (encoded with [I-D.ietf-lwig-curve-representations])
    $$ox' = x'_0 + x'_1 * u \ (x'_0, x'_1 \text{ in } GF(p))$$
    $$oy' = y'_0 + y'_1 * u \ (y'_0, y'_1 \text{ in } GF(p))$$
    $$-h' : \text{the cofactor } #E'(GF(p^2))/r$$

$p$:
As mentioned above, BLS12_381 is adopted in a lot of applications. Since it is expected that BLS12_381 will continue to be widely used more and more in the future, Appendix C shows the serialization format of points on an elliptic curve as useful information. This serialization format is also adopted in [I-D.boneh-bls-signature] [zkcrypto].

In addition, many pairing-based cryptographic applications use a hashing to an elliptic curve procedure that outputs a rational point
on an elliptic curve from an arbitrary input. A standard specification of ciphersuites for a hashing to an elliptic curve, including BLS12-381, is under discussion in the IETF [I-D.irtf-cfrg-hash-to-curve] and it will be valuable information for implementers.

4.2.2. BN Curves for the 128-bit security level

A BN curve with the 128-bit security level is shown in [BD18], which we call BN462. BN462 is defined by the parameter

\[ t = 2^{114} + 2^{101} - 2^{14} - 1 \]

for the definition in Section 2.3.

For the finite field \( \text{GF}(p) \), the towers of extension field \( \text{GF}(p^2) \), \( \text{GF}(p^6) \), and \( \text{GF}(p^{12}) \) are defined by indeterminates \( u \), \( v \), and \( w \) as follows:

\[
\begin{align*}
\text{GF}(p^2) & = \text{GF}(p)[u] / (u^2 + 1) \\
\text{GF}(p^6) & = \text{GF}(p^2)[v] / (v^3 - u - 2) \\
\text{GF}(p^{12}) & = \text{GF}(p^6)[w] / (w^2 - v).
\end{align*}
\]

Defined by \( t \), the elliptic curve \( E \) and its twist \( E' \) are represented by \( E: y^2 = x^3 + 5 \) and \( E': y^2 = x^3 - u + 2 \), respectively. The size of \( p \) becomes 462-bit length. BN462 is categorized as D-type.

We have to note that BN462 is significantly slower than BLS12_381, but has 134-bit security level [GMT19], so may be more resistant to future small improvements to the exTNFS attack.

We note also that CP8_544 is about 20% faster that BN462 [GMT19], has 131-bit security level, and that due to its construction will not be affected by future small improvements to the exTNFS attack. However, as this curve is not widely used (it is only implemented in one library), we instead chose BN462 for our 'safe' option.

We give the following parameters for BN462.

* \( G_1 \) is the largest prime-order subgroup of \( E(\text{GF}(p)) \)
  - \( BP = (x, y) \): a 'base point', i.e., a generator of \( G_1 \)

* \( G_2 \) is an \( r \)-order subgroup of \( E'(\text{GF}(p^2)) \)
  - \( BP' = (x', y') \): a 'base point', i.e., a generator of \( G_2 \)
    (encoded with [I-D.ietf-lwig-curve-representations])

\[
\begin{align*}
x'_0 & = x'_0 + x'_1 * u \quad (x'_0, x'_1 \text{ in } \text{GF}(p)) \\
y'_0 & = y'_0 + y'_1 * u \quad (y'_0, y'_1 \text{ in } \text{GF}(p))
\end{align*}
\]
- $h'$: the cofactor $E'(\text{GF}(p^2))/r$

$p$:  
0x240480360120023fffffffef0cf6b7d9bfca0000000000d812908f41c8020ffffffff6ff66fc6ff

$r$:  
0x240480360120023fffffffef0cf6b7d9bfca0000000000d812908ee1201f7ffffffff6ff66fc6ff

$x$:  
0x21a667ef250191fadba34a0a30160b9ac92646f9f63b3edbec3cf4b2e689db1b4e69a416a0b1e792

$y$:  
0x0118ea0460f7f7abb2b33676a7432a490eeda842cccfa7d788c659650426e6af77df11b8ae40eb80f475

$h$: 1  
$b$: 5

$x'_0$:  
0x0257ccc85b58db0d0fb38e3a8cbdc5482e0337e7c1cd96ed61c91382040828f9ad2699bad92e032ae1f

$x'_1$:  
0x1d2e4343e8599102af8edca84566ba3c98e2a354730cbed9176684058b18134dd86bae555b783718f50a

$y'_0$:  
0x0a0650439da22c1979517427a20809eca035634706e23c3fa7a6b42fe810f1399a1f41c9ddae32e03695

$y'_1$:  
0x073ef0cbd438cbe0172c8ae37306324d44d5e6b0c69ac57b393f1ab370fd725cc647692444a04ef87387a

$h'$:  
0x240480360120023fffffffef0cf6b7d9bfca0000000000d812908fa1ce0227ffffffff6ff66fc6ff

$b'$: -u + 2

4.3. For 256-bit Security

As shown in Table 1, there are three candidates of pairing-friendly curves for 256-bit security. According to our selection policy, we select BLS48_581, as it is the most widely adopted by cryptographic libraries.
The selected BLS48 curve is shown in [KIK17] and it is defined by the parameter
\[ t = -1 + 2^7 - 2^{10} - 2^{30} - 2^{32}. \]

In this case, the size of \( p \) becomes 581-bit.

For the finite field \( \text{GF}(p) \), the towers of extension field \( \text{GF}(p^2) \), \( \text{GF}(p^4) \), \( \text{GF}(p^8) \), \( \text{GF}(p^24) \) and \( \text{GF}(p^48) \) are defined by indeterminates \( u, v, w, z, \) and \( s \) as follows:

\[
\begin{align*}
\text{GF}(p^2) &= \text{GF}(p)[u] / (u^2 + 1) \\
\text{GF}(p^4) &= \text{GF}(p^2)[v] / (v^2 + u + 1) \\
\text{GF}(p^8) &= \text{GF}(p^4)[w] / (w^2 + v) \\
\text{GF}(p^{24}) &= \text{GF}(p^8)[z] / (z^3 + w) \\
\text{GF}(p^{48}) &= \text{GF}(p^{24})[s] / (s^2 + z).
\end{align*}
\]

The elliptic curve \( E \) and its twist \( E' \) are represented by \( E: y^2 = x^3 + 1 \) and \( E': y^2 = x^3 - 1 / w \). BLS48-581 is categorized as D-type.

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

- \( G_1 \) is the largest prime-order subgroup of \( E(\text{GF}(p)) \)
  - \( \text{BP} = (x,y) \) : a 'base point', i.e., a generator of \( G_1 \)

- \( G_2 \) is an \( r \)-order subgroup of \( E'(\text{GF}(p^8)) \)
  - \( \text{BP}' = (x',y') \) : a 'base point', i.e., a generator of \( G_2 \)
    (encoded with [I-D.ietf-lwig-curve-representations])
  
    \[ \begin{align*}
    x' &= x'_0 + x'_1 * u + x'_2 * v + x'_3 * u * v + x'_4 * w + x'_5 * u * w + x'_6 * v * w + x'_7 * u * v * w \\
    & \quad \text{in } \text{GF}(p) \\
    y' &= y'_0 + y'_1 * u + y'_2 * v + y'_3 * u * v + y'_4 * w + y'_5 * u * w + y'_6 * v * w + y'_7 * u * v * w \\
    & \quad \text{in } \text{GF}(p)
    \end{align*} \]

- \( h' \) : the cofactor \( \#E'(\text{GF}(p^8))/r \)

\[ p: \]
0x1280f73ff3476f313824e31d47012a0056e84f8d122131bb3be6c0f1f3975444a48ae43af6e082ac9d9cd3

\[ r: \]
0x2386f8a925e2885e233a9ccc1615c0d6c635387a3f0b3cbe003fad6bc972c2e6e741969d34c4c92016a85
5. Security Considerations

The recommended pairing-friendly curves are selected by considering the exTNFS proposed by Kim et al. in 2016 [KB16] and they are categorized in each security level in accordance with [BD18]. Implementers who will newly develop pairing-based cryptography applications SHOULD use the recommended parameters. As of 2020, as far as we’ve investigated the top cryptographic conferences in the past, there are no fatal attacks that significantly reduce the security of pairing-friendly curves after exTNFS.

BLS curves of embedding degree 12 typically require a characteristic p of 461 bits or larger to achieve the 128-bit security level [BD18]. Note that the security level of BLS12-381, which is adopted by a lot of libraries and applications, is slightly below 128 bits because a 381-bit characteristic is used [BD18] [GMT19].
BN254 is used in most of the existing implementations as shown in Section 4.1 (and Appendix D), however, BN curves that were estimated as the 128-bit security level before exTNFS including BN254 ensure no more than the 100-bit security level by the effect of exTNFS.

In addition, implementors should be aware of the following points when they implement pairing-based cryptographic applications using recommended curves. Regarding the use case and applications of pairing-based cryptographic applications, please refer Section 1.2.

In applications such as key agreement protocols, users exchange the elements in G_1 and G_2 as public keys. To check these elements are so-called sub-group secure [BCM15], implementors should validate if the elements have the correct order r. Specifically, for public keys P in G_1 and Q in G_2, a receiver should calculate scalar multiplications \([r]P\) and \([r]Q\), and check the results become points at infinity.

The pairing-based protocols, such as the BLS signatures, use a scalar multiplication in G_1, G_2 and an exponentiation in G_3 with the secret key. In order to prevent the leakage of secret key due to side channel attacks, implementors should apply countermeasure techniques such as montgomery ladder [Montgomery] [CF06] when they implement modules of a scalar multiplication and an exponentiation. Please refer [Montgomery] and [CF06] for the detailed algorithms of montgomery ladder.

When converting between an element in extension field and an octet string, implementors should check that the coefficient is within an appropriate range [IEEE1363]. If the coefficient is out of range, there is a possible that security vulnerabilities such as the signature forgery may occur.

Recommended parameters are affected by the Cheon's attack which is a solving algorithm for the strong DH problem [Cheon06]. The mathematical problem that provides the security of the strong DH problem is called ECDLP with Auxiliary Inputs (ECDLPwAI). In ECDLPwAI, given rational points P, [K]P, [K^i]P, for i=1,...,n, then we find a secret K. Since the complexity of ECDLPwAI is given as \(O(\sqrt{(r-1)/n} + \sqrt{n})\) where \(n|r-1\) by using Cheon's algorithm whereas the complexity of ECDLP is given as \(O(\sqrt{r})\), the complexity of ECDLPwAI with the ideal value n becomes dramatically smaller than that of ECDLP. Please refer [Cheon06] for the details of Cheon's algorithm. Therefore, implementers should be careful when they design cryptographic protocols based on the strong DH problem. For example, in the case of Short Signatures, they can prevent the Cheon's attack by carefully setting the maximum number of queries which corresponds to the parameter n.
6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The authors would like to appreciate a lot of authors including Akihiro Kato for their significant contribution to early versions of this memo. The authors would also like to acknowledge Kim Taechan, Hoeteck Wee, Sergey Gorbunov, Michael Scott, Chloe Martindale as an Expert Reviewer, Watson Ladd, Armando Faz, Rene Struik, and Satoru Kanno for their valuable comments.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Computing the Optimal Ate Pairing

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

```plaintext
if (A = B) then
else if (A = -B) then
  return P[1] - A[1];
else
end if;
```

When implementing the line function, implementers should consider the isomorphism of \( E \) and its twist curve \( E' \) so that one can reduce the computational cost of operations in \( G_2 \) [CLN09][KIK17]. We note that `Line_function` does not consider such an isomorphism.

The computation of the optimal Ate pairing for BN curves uses the Frobenius map. The p-power Frobenius map \( \pi \) for a point \( Q = (x, y) \) over \( E' \) is \( \pi(p, Q) = (x^p, y^p) \).

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

Let \( c = 6 \times t + 2 \) for a parameter \( t \) and \( c_0, c_1, \ldots, c_L \) in \( \{-1,0,1\} \) such that the sum of \( c_i \times 2^i \) (\( i = 0, 1, \ldots, L \)) equals \( c \).

The following algorithm shows the computation of the optimal Ate pairing on BN curves. It takes \( P \) in \( G_1 \), \( Q \) in \( G_2 \), an integer \( c \), \( c_0, \ldots, c_L \) in \( \{-1,0,1\} \) such that the sum of \( c_i \times 2^i \) (\( i = 0, 1, \ldots, L \)) equals \( c \), and the order \( r \) of \( G_1 \) as input, and outputs \( e(P, Q) \).
A.2. Optimal Ate Pairings over Barreto-Lynn-Scott Curves

Let \( c = t \) for a parameter \( t \) and \( c_0, c_1, \ldots, c_L \) in \{-1,0,1\} such that the sum of \( c_i \cdot 2^i \) (\( i = 0, 1, \ldots, L \)) equals \( c \). 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 \( c \), \( c_0, c_1, \ldots, c_L \) in \{-1,0,1\} such that the sum of \( c_i \cdot 2^i \) (\( i = 0, 1, \ldots, L \)), and an order \( r \) as input, and outputs \( e(P, Q) \).

\[
f := 1; \quad T := Q;
\]
\[
\text{if } (c_L = -1) \quad T := -T;
\]
\[
\text{end if}
\]
\[
\text{for } i = L-1 \text{ to } 0
\]
\[
f := f^2 \cdot \text{Line}_{\text{function}}(T, T, P); \quad T := 2 \cdot T;
\]
\[
\text{if } (c_i = 1 \lor c_i = -1) \quad f := f \cdot \text{Line}_{\text{function}}(T, c_i \cdot Q); \quad T := T + c_i \cdot Q;
\]
\[
\text{end if}
\]
\[
\text{end for}
\]
\[
Q_1 := \pi(p, Q); \quad Q_2 := \pi(p, Q_1);
\]
\[
f := f \cdot \text{Line}_{\text{function}}(T, Q_1, P); \quad T := T + Q_1;
\]
\[
f := f \cdot \text{Line}_{\text{function}}(T, -Q_2, P);
\]
\[
f := f^{(p^k - 1) / r};
\]
\[
\text{return } f;
\]

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 BLS12-381, BN462 and BLS48-581 given in Section 4. Here, the inputs \( P = (x, y) \) and \( Q = (x', y') \) are the corresponding base points \( BP \) and \( BP' \) given in Section 4.

For BLS12-381 and BN462, \( Q = (x', y') \) is given by

\[
x' = x'_0 + x'_1 \cdot u \quad \text{and} \quad y' = y'_0 + y'_1 \cdot u,
\]
where $u$ is a indeterminate and $x'_0$, $x'_1$, $y'_0$, $y'_1$ are elements of $GF(p)$.

For BLS48-581, $Q = (x', y')$ is given by

$$x' = x'_0 + x'_1 * u + x'_2 * v + x'_3 * u * v + x'_4 * w + x'_5 * u * w + x'_6 * v * w + x'_7 * u * v * w$$

$$y' = y'_0 + y'_1 * u + y'_2 * v + y'_3 * u * v + y'_4 * w + y'_5 * u * w + y'_6 * v * w + y'_7 * u * v * w,$$

where $u$, $v$ and $w$ are indeterminates and $x'_0$, ..., $x'_7$ and $y'_0$, ..., $y'_7$ are elements of $GF(p)$. The representation of $Q = (x', y')$ given below is followed by [I-D.ietf-lwig-curve-representations].

BLS12-381:
Input x value:
0x17f1d3a73197d7942695638c4fa9ac0fc3688c4f9774b905a14e3a3f171bac586c55e83ff97a1eфф3aф

Input y value:
0x08b3f481e3aa0f1a09e30ed741d8ae4fcf5e095d5d00af600db18cb2c04b3edd03cc7442888ae40ca2

Input x'_0 value:
0x024aa2b2f08f0a91260805272dc51051c6e47ad4fa403b02b4510b647ae3d1770bac0326a805b8efd4805

Input x'_1 value:
0x13e02b6052719f607dacad3a088274f65596bd0d09920b61ab5da61bbd7f5049334cf11213945d75e5ac7

Input y'_0 value:
0xce5d527727d6e118cc9c64da2e351adfd9baa8cbdd3a76429a69516d0d12c923ac9cc3baca289e1935

Input y'_1 value:
0x0606c4a2ea734cc32ac2b02bc2899cb3e287e85a763af267492ab572e99ab3f370d275c1da1aa96

\( e_0 \):
0x11619b45f61edfe3b47a15fac19442526ff489dca25e59121d9931438907dfd448299a87dde3a649bbdb

\( e_1 \):
0x153ce1a76a53e205ba8f275ef1137c56a566f638b52d34ba3bf3bf22f277d70f76316218c0df5d583a94

\( e_2 \):
0x095668fb4a02fe930ed44767834c915b283bc6ca98c047bd4c272e9ac3f3ba6ff0b05a93e59c71fba77b

\( e_3 \):
0x16deedaa683124fe7260085184d88f7d036b86f53bb5b7f1fc5e248814782065413e7d958d17960109e6

\( e_4 \):
0x9c92cf02fd3d3d2f9d34bc44ee0dd50314ed44ca5d30ce6a9ec0539be7a86b121edc61839ccc908c4b
e

\( e_5 \):
0x111061f398efc2a97ff825b04d21089e24fd8b93a47e41e60eae7e9b2a38d54fa4dedced0811c34ce528e7

\( e_6 \):
0x01ecfcf31c86257ab00b4709c33f1c9c4e007659dd5ffcc4a735192167ce197058cfd4c94225e7f1b6c26ad9ba68f63bc

e_7:
0x08890726743a1f94a8193a166800b7787744a8ad8e2f9365db76863e894b7a11d83f90d873567e9d645dcd7d9d0ac018

e_8:
0x0e61c752414ca5dfd258e9606bac08daec29b3e2c57062669566954fb227d3f1260eef825446a086b084e938211

e_9:
0x0fe63f185f56dd29150fc498bee87e783043620db33f75a05a0a2ce5c442beaff9da195ff15164b24558e5fb44b54b

e_10:
0x10900338a92ed0b47af21163f7cfdec717b7ee43900ee9b5fc24f0000c5874d4801372db478987691c59e833f3a2f

BN462:
Input x value:
0x21a6d67ef250191fadba34a0a30160b9ac9264b6f95f63b3edbec3cf4b2e689db1bb4e69a416a0b1e7929

Input y value:
0x0118ea0460f7f7abb82b33676a7432a490eea842ccfafa7d788c659650426e6af77df11b8ae40eb80f4757

Input x'_0 value:
0x0257ccc85b58d0d0db38e3a8cbdc5482e033e7c1cd96ed61c913820408020f9ad2699bad92e0032ae1f

Input x'_1 value:
0x1d2e4343e8599102af8edca849566ba3c98e2a354730cbedd976884058b18134dd86bae555b783718f50a

Input y'_0 value:
0x0a0650439da22c1979517427a20809eca035634706e2c35a7a6bb42fe810f1399a1f41c9daae32e03695

Input y'_1 value:
0x073ef0cbd438cbe0172c8ae37306324d44d5e6b0c69ac57b393f1ab370fd725cc647692444a04ef87387a

e_0:
0xc2f70f2e01610804272f4a7a24014ac085543d787c8f8bf07059f93f87ba7e2a4ac77835d4ff10e78669

e_1:
0x0def2c737515694ee5b85051e39970f2e27ca278847c7cfa709bd4f48b830b3763b1b001f1194445b62

e_2:
0x4d685b29fd2b8faedacd36873f24a06158742bb2328740f93827934592d6f1723e0772bb9ccd3025f88c

e_3:
0x090d7e8f2892dec0c4be49cbe4ff1f835286c70000c8d191574cb424019de1112b3c722cc5083017191241

e_4:
0x1437603b60dce235a090c43f5147d9c03bd63081c8bb1fft7a78a2c31d673230860bb3df4e4ca85581f7459

e_5:
0x13191b1111d13650b80e76b356fe776eb9d7a03fe33f82e3fe5732071f305d201843238cc96f0e92a6c

e_6:
0x07b1ce375c0191c786bb184cc9c08a6ae5a569dd7586f75d6d2de2b2f075787ee5082d44ca4b8009b32857

e_7:
0x05b64add5e49574b124a02d85f508c8d2d37993ae4c370a9cdac89a100c4b5e1d441b57768dbc68429ffafed

e_8:
0x0fd9a3271854a2b4542b42c55916e1fa7a8b87a7d10907179ac7073f6a1de044906ffaf4760d11c8f92d

e_9:
0x17fa0c7fa60c9a6d4d8bb9897991ef0d087899edc776f33743db921a689720c82257ee3c788e8160c112f

e_10:
0x0c901397a62bb185a8f9cf336e28c5b0f354e2313f99c538cdceef8b8aa22c23b896201170fc915690f7

e_11:
0x20f27fde93cee94ca4bf9ded1b1378c1b0d80439eeb1d0c8daef30db0037104a5e32a2ccc94fa1860a95e

BLS48-581:
Input x value:
0x02af59b7ac340f2baf2b73df1e93f860de3f257e0e86868cf61abdbaedffbf9f7544550546a9df6f964584

Input y value:
0xcefda44f6531f91f86b3a2d1fb398a488a553c9efeb8a52e991279dd41b720ef7bb7beff98ae53e0f

x'_0:
0x05d615d9a7871e4a38237fa45a2775debabbefc70344dbcc7de64db3a2ef156c46ff79baad1a8c42281a

x'_1:
0x07c4973ece2258512069b0e86abc07e8b22bb6d980e1623e9526f6da12307f4e1c3943a00abfedf16214a

x'_2:
0x01fccc70198f1334e1b2ea1853ad83bc73a8a6ca9ae237ca7a6d6957ccbab5ab6860161c1dbd19242ffae

x'_3:
0x0be2218c25ceb6185c78d8012954d4be8f5985ac62f3e5821b7b92a393f8be0cc218a95f63e1c776e6e42

x'_4:
0x038b91c600b35913a3c598e4caa9dd63007c675d0b1642b5675ff0e7c580538669981f9e48199d5ac0b

x'_5:
0x0c96c7797eb0738603f1311e4edca088f7b8f35dcef977a3d1a58677bb037418181df63835d28997eb57

x'_6:
0x0b9b7951c6061ee3f0197a498908aee660dea41b39d13852b6db908ba2c0b7a449cef11f293b13ced0f0d

x'_7:
0x0827d5c22fb2bdec5282624c4f4aaa2b1e5d7a9defaf47b5211cf741719728a7f98cfca93f29ccf364a7

y'_0:
0x0eb53356c375b5dfa497216452f3024b970b4238059a577e6f3b39ebfc435faa0906235afa27748d90f

y'_1:
0x0284dc75979e0ff144da6531815fcadc2b75a422ba325e6fba01d72964732fcbf3af0b96b243b1f192c5d

y'_2:
y'_3:
0x0a0c25a4621edc0688223fbd478762b1c2cded3360dce23dd8b0e710e122d2742c89b224333fa40dced6e9

y'_4:
0x0d209d5a223a9c46916503fa5a88325a2554dc541b43dd93b5a959805f1129857ed85c77fa238cdce8a1e6

y'_5:
0x07d0d03745736b7a513d339d5ad537b90421ad66eb16722b589d82e2055ab7504fa83420e8c270841f682f

y'_6:
0x0896767811be65ea25c2d05dfdd17af8a006f364fc0841b064155f14e4c819a6df98f425ae3a2864f22c17

y'_7:
0x035e2524ff89029d393a5c07e84f981b5e068f1406be8e50c87549b6ef8eca9a9533a3f8e69c31e97e1ad0

e_0:
0x0e26c3fcb8ef67417814098de511ffccc1d003d15b367bad07cef2291a93d31db03e3f03376f3beae2b0

e_1:
0x069061b8047279aa5c2d25cdf676ddf34eddbc8ec2ec0f03614886fa828e1fc066b26d35744c0c3827184

e_2:
0x02b9bce645fbf9d8f97025a1545359f6fe3ffab3cd57094f862f7fb9ca01c88705c26675bccc723878e942

e_3:
0x080d0267bf036c1e61d7cf3905e8c630b97aa05e3f2d66c82e7a111072c0d2056ba8137fba111c9650df0

e_4:
0x03c6b4c12f338f94016a493a405b33e64389338db8c5e592a8dd7eac7720d83dd6b0c189eeeda208093d

e_5:
0x016e46224f28bdf8833f76ac29ee6e406a9da1bde55f5e82b3bd977897a9104f18b9ee41e9af7d4183d88

e_6:
0x08ddce7a4a1b94be5df3ceea56bef0077dcdde86d579938a50933a47296337b7629934128e2457e2414
Appendix C. ZCash serialization format for BLS12-381

This section describes the serialization format defined by [ZCashRep]. This format applies to points on the BLS12-381 elliptic curves $E$ and $E'$, whose parameters are given in Section 4.2.1. Note that this serialization method is based on the representation shown in [SEC1] and it is a tiny tweak so as to apply to $\text{GF}(p^m)$. It is not officially standardized by the standards organization, however we show it in this appendix as a useful reference for implementers.

At a high level, the serialization format is defined as follows:

*Serialized points include three metadata bits that indicate whether a point is compressed or not, whether a point is the point at infinity or not, and (for compressed points) the sign of the point's $y$-coordinate.*

*Points on $E$ are serialized into 48 bytes (compressed) or 96 bytes (uncompressed). Points on $E'$ are serialized into 96 bytes (compressed) or 192 bytes (uncompressed).*

*The serialization of a point at infinity comprises a string of zero bytes, except that the metadata bits may be nonzero.*

*The serialization of a compressed point other than the point at infinity comprises a serialized $x$-coordinate.*

*The serialization of an uncompressed point other than the point at infinity comprises a serialized $x$-coordinate followed by a serialized $y$-coordinate.*
Below, we give detailed serialization and de-serialization procedures. The following notation is used in the rest of this section:

*Elements of GF(p^2) are represented as polynomial with GF(p) coefficients like Section 2.5.

*For a byte string str, str[0] is defined as the first byte of str.

*The function sign_GF_p(y) returns one bit representing the sign of an element of GF(p). This function is defined as follows:

\[
\text{sign}_{\text{GF}_p}(y) := \begin{cases} 
1 & \text{if } y > \frac{p - 1}{2}, \\
0 & \text{otherwise}.
\end{cases}
\]

*The function sign_GF_p^2(y') returns one bit representing the sign of an element in GF(p^2). This function is defined as follows:

\[
\text{sign}_{\text{GF}_p^2}(y') := \begin{cases} 
\text{sign}_{\text{GF}_p}(y'_0) & \text{if } y'_1 \text{ equals } 0, \\
1 & \text{if } y'_1 > \frac{p - 1}{2}, \\
0 & \text{otherwise}.
\end{cases}
\]

C.1. Point Serialization Procedure

The serialization procedure is defined as follows for a point P = (x, y). This procedure uses the I2OSP function defined in [RFC8017].

1. Compute the metadata bits C_bit, I_bit, and S_bit, as follows:

   *C_bit is 1 if point compression should be used, otherwise it is 0.

   *I_bit is 1 if P is the point at infinity, otherwise it is 0.

   *S_bit is 0 if P is the point at infinity or if point compression is not used. Otherwise (i.e., when point compression is used and P is not the point at infinity), if P is a point on E, S_bit = sign_GF_p(y), else if P is a point on E', S_bit = sign_GF_p^2(y).

2. Let \( m_{\text{byte}} = (C_{\text{bit}} \times 2^7) + (I_{\text{bit}} \times 2^6) + (S_{\text{bit}} \times 2^5) \).

3. Let x_string be the serialization of x, which is defined as follows:

   *If P is the point at infinity on E, let x_string = I2OSP(0, 48).
If P is a point on E other than the point at infinity, then x is an element of GF(p), i.e., an integer in the inclusive range \([0, p - 1]\). In this case, let \(x\text{\_string} = \text{I2OSP}(x, 48)\).

If P is the point at infinity on E', let \(x\text{\_string} = \text{I2OSP}(0, 96)\).

If P is a point on E' other than the point at infinity, then x can be represented as \((x_0, x_1)\) where \(x_0\) and \(x_1\) are elements of GF(p), i.e., integers in the inclusive range \([0, p - 1]\) (see discussion of vector representations above). In this case, let \(x\text{\_string} = \text{I2OSP}(x_1, 48) \parallel \text{I2OSP}(x_0, 48)\).

Notice that in all of the above cases, the 3 most significant bits of \(x\text{\_string}[0]\) are guaranteed to be 0.

4. If point compression is used, let \(y\text{\_string}\) be the empty string. Otherwise (i.e., when point compression is not used), let \(y\text{\_string}\) be the serialization of y, which is defined in Step 3.

5. Let \(s\text{\_string} = x\text{\_string} \parallel y\text{\_string}\).

6. Set \(s\text{\_string}[0] = x\text{\_string}[0] \text{ OR } m\text{\_byte}\), where OR is computed bitwise. After this operation, the most significant bit of \(s\text{\_string}[0]\) equals \(C\text\_bit\), the next bit equals \(I\text\_bit\), and the next equals \(S\text\_bit\). (This is true because the three most significant bits of \(x\text{\_string}[0]\) are guaranteed to be zero, as discussed above.)

7. Output \(s\text{\_string}\).

**C.2. Point deserialization procedure**

The deserialization procedure is defined as follows for a string \(s\text{\_string}\). This procedure uses the OS2IP function defined in [RFC8017].

1. Let \(m\text{\_byte} = s\text{\_string}[0] \text{ AND } 0xE0\), where AND is computed bitwise. In other words, the three most significant bits of \(m\text{\_byte}\) equal the three most significant bits of \(s\text{\_string}[0]\), and the remaining bits are 0.

   If \(m\text{\_byte}\) equals any of \(0x20\), \(0x60\), or \(0xE0\), output INVALID and stop decoding.

   Otherwise:

   *Let \(C\text\_bit\) equal the most significant bit of \(m\text{\_byte}\),
Let \( I_{\text{bit}} \) equal the second most significant bit of \( m_{\text{byte}} \), and

Let \( S_{\text{bit}} \) equal the third most significant bit of \( m_{\text{byte}} \).

2. If \( C_{\text{bit}} \) is 1:

   * If \( s_{\text{string}} \) has length 48 bytes, the output point is on the curve \( E \).
   
   * If \( s_{\text{string}} \) has length 96 bytes, the output point is on the curve \( E' \).
   
   * If \( s_{\text{string}} \) has any other length, output INVALID and stop decoding.

   If \( C_{\text{bit}} \) is 0:

   * If \( s_{\text{string}} \) has length 96 bytes, the output point is on \( E \).
   
   * If \( s_{\text{string}} \) has length 192 bytes, the output point is on \( E' \).
   
   * If \( s_{\text{string}} \) has any other length, output INVALID and stop decoding.

3. Let \( s_{\text{string}}[0] = s_{\text{string}}[0] \) AND 0x1F, where AND is computed bitwise. In other words, set the three most significant bits of \( s_{\text{string}}[0] \) to 0.

4. If \( I_{\text{bit}} \) is 1:

   * If \( s_{\text{string}} \) is not the all zeros string, output INVALID and stop decoding.
   
   * Otherwise (i.e., if \( s_{\text{string}} \) is the all zeros string), output the point at infinity on the curve that was determined in step 2 and stop decoding.

   Otherwise, \( I_{\text{bit}} \) must be 0. Continue decoding.

5. If \( C_{\text{bit}} \) is 0:

   * Let \( x_{\text{string}} \) be the first half of \( s_{\text{string}} \).
   
   * Let \( y_{\text{string}} \) be the last half of \( s_{\text{string}} \).
   
   * Let \( x = \text{OS2IP}(x_{\text{string}}) \).
   
   * Let \( y = \text{OS2IP}(y_{\text{string}}) \).
If the point \( P = (x, y) \) is not a valid point on the curve that was determined in step 2, output INVALID and stop decoding.

Otherwise, output the point \( P = (x, y) \) and stop decoding.

Otherwise, \( C_{\text{bit}} \) must be 1. Continue decoding.

6. Let \( x = \text{OS2IP}(s_{\text{string}}) \).

7. If the curve that was determined in step 2 is \( E \):

   *Let \( y^2 = x^3 + 4 \) in \( \text{GF}(p) \).

   *If \( y^2 \) is not square in \( \text{GF}(p) \), output INVALID and stop decoding.

     *Otherwise, let \( y = \sqrt{y^2} \) in \( \text{GF}(p) \) and let \( Y_{\text{bit}} = \text{sign}_{\text{GF}\_p}(y) \).

Otherwise, (i.e., when the curve that was determined in step 2 is \( E' \)):

   *Let \( y^2 = x^3 + 4 \cdot (u + 1) \) in \( \text{GF}(p^2) \).

   *If \( y^2 \) is not square in \( \text{GF}(p^2) \), output INVALID and stop decoding.

     *Otherwise, let \( y = \sqrt{y^2} \) in \( \text{GF}(p^2) \) and let \( Y_{\text{bit}} = \text{sign}_{\text{GF}\_p^2}(y) \).

8. If \( S_{\text{bit}} \) equals \( Y_{\text{bit}} \), output \( P = (x, y) \) and stop decoding. Otherwise, output \( P = (x, -y) \) and stop decoding.

Appendix D. Adoption Status of Pairing-Friendly Curves with the 100-bit Security Level

BN curves including BN254 that were estimated as the 128-bit security level before exTNFS ensure no more than the 100-bit security level by the effect of exTNFS. Table 2 summarizes the adoption status of the parameters with a security level lower than the "Arnd 128-bit" range. Please refer the Section 4 for the naming conventions for each curve listed in Table 2.

<table>
<thead>
<tr>
<th>Category</th>
<th>Name</th>
<th>Supported 100-bit Curves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>ISO/IEC</td>
<td>BN256I</td>
</tr>
<tr>
<td></td>
<td>TCG</td>
<td>BN256I</td>
</tr>
<tr>
<td></td>
<td>FIDO/W3C</td>
<td>BN256I, BN256D</td>
</tr>
<tr>
<td>Category</td>
<td>Name</td>
<td>Supported 100-bit Curves</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Library</td>
<td>mcl</td>
<td>BN254N, BN_SNARK1</td>
</tr>
<tr>
<td></td>
<td>TEPLA</td>
<td>BN254B, BN254N</td>
</tr>
<tr>
<td></td>
<td>RELIC</td>
<td>BN254N, BN256D</td>
</tr>
<tr>
<td></td>
<td>AMCL</td>
<td>BN254N, BN254CX, BN256I</td>
</tr>
<tr>
<td></td>
<td>Intel IPP</td>
<td>BN256I</td>
</tr>
<tr>
<td></td>
<td>MIRACL</td>
<td>BN254N, BN254CX, BN256I</td>
</tr>
<tr>
<td></td>
<td>Adjoint</td>
<td>BN_SNARK1, BN254B, BN254N, BN254S1, BN254S2</td>
</tr>
<tr>
<td>Application</td>
<td>Zcash</td>
<td>BN_SNARK1</td>
</tr>
<tr>
<td></td>
<td>DFINITY</td>
<td>BN254N, BN_SNARK1</td>
</tr>
</tbody>
</table>

Table 2: Adoption Status of Pairing-Friendly Curves with 100-bit Security Level (Legacy)

Authors' Addresses

Yumi Sakemi (editor)
Lepidum

Email: yumi.sakemi@lepidum.co.jp

Tetsutaro Kobayashi
NTT

Email: tetsutaro.kobayashi.dr@hco.ntt.co.jp

Tsunekazu Saito
NTT

Email: tsunekazu.saito.hg@hco.ntt.co.jp

Riad S. Wahby
Stanford University

Email: rsw@cs.stanford.edu