Workgroup: CFRG
Internet-Draft:
draft-irtf-cfrg-pairing-friendly-curves-07
Published: 18 June 2020
Intended Status: Experimental
Expires: 20 December 2020
Authors: Y. Sakemi, Ed. T. Kobayashi T. Saito
Lepidum NTT NTT
R. Wahby
Stanford University

Pairing-Friendly Curves

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 will expire on 20 December 2020.

Copyright Notice

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 192-bit Security
  4.4. For 256-bit Security
5. Security Considerations
6. IANA Considerations
7. Acknowledgements

8. References
   8.1. Normative References
   8.2. Informative References
Appendix A. Computing the Optimal Ate Pairing
   A.1. Optimal Ate Pairings over Barreto-Naehrig Curves
   A.2. Optimal Ate Pairings over Barreto-Lynn-Scott Curves
Appendix B. Test Vectors of Optimal Ate Pairing
Appendix C. ZCash serialization format for BLS12-381
   C.1. Point Serialization Procedure
   C.2. Point deserialization procedure
Authors' Addresses

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 [KB16]. 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. 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 level described in this memo corresponds 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 > 3 be a prime and q = p^n for a natural number n. Let F_q be a finite field. The curve defined by the following equation E is called an elliptic curve:

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

and A and B in F_q satisfy the discriminant inequality 4 * A^3 + 27 * B^2 != 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 (F_q^k)-rational points of E, denoted by E(F_q^k), to be the set of all solutions (x,y) in F_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(F_q^k) forms a group under a group law which can be defined geometrically as follows. For P and Q in E(F_q^k) define P + Q to be the reflection about the x-axis of the unique third point 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 [a]P for a positive integer a as the point P added to itself (a-1) times.

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

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

E(F_q^k): the group of F_q-rational points of E.

#E(F_q^k): the number of F_q-rational points of E.

r: the largest prime divisor of #E(F_q).

BP: a point in E(F_q) of order r. (The 'base point' of a cyclic subgroup of E(F_q))

h: the cofactor h = #E(F_q) / r.

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 [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 F_p. Let k be the minimum integer for which r is a divisor of p^k - 1; this is called the embedding degree. Let G_1 be a cyclic subgroup of E(F_p) of order r, there also exists a cyclic subgroup of E(F_p^k) of order r, define this to be G_2. 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 (F_p^k)^*; this exists by definition of k.

A 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, and integers a and b, 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.

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

2.3. Barreto-Naehrig Curves

A BN curve [BN05] 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 defined by elliptic curves E and E' parameterized by a well-chosen integer t. E is defined over F_p, where p is a prime more than or equal to 5, and E(F_p) has a subgroup of prime order r. The characteristic p and the order r are 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 an integer t.

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

BN curves always have order 6 twists. If m is an element that is neither a square nor a cube in an extension field F_p^2, the twist E' of E is defined over an extension field F_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 embeddiing degree k is 12.

A pairing e is defined by taking G_1 as a subgroup of E(F_p) of order r, G_2 as a subgroup of E'(F_p^2), and G_T as a subgroup of a multiplicative group (F_p^12)^* of order r.

2.4. Barreto-Lynn-Scott Curves

A BLS curve [BLS02] is another instantiation of pairings 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 F_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(F_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 into 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, 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 a well chosen integer t.

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

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

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

Let F_p^d' = F_p^d(j) be an extension field of F_p^d of degree d' / d.

For an element s' in F_p^d' such that s' = s'_0 + s'_1 * j + ... + s'_{d' / d - 1} * j^{d' / d - 1} where s'_0, s'_1, ..., s'_{d' / d - 1} in the basefield F_p^d, s' is represented as integer by oct(s') = oct(s'_0) || oct(s'_1) || ... || oct(s'_{d' / d - 1}), where oct(s'_0), ... , 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 paiting-based crytography, the only known attacks thus far attack the discrete logarithm problem directly, so we focus on the discrete logarithm 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 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]. Due to exTNFS, the security level 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 level described in this memo corresponds 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.

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 into '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. Due to space limitations, Table 1 omits libraries that have not been maintained since the 2016 exTNFS attacks and curves that have had the security levels below 128 bits since before 2016 (ex. BN160). The full version of Table 1 is available at https://lepidum.co.jp/blog/2020-03-27/ietf-draft-pfc/. In this table, 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 level of these curves.

Category	**Name**	**Curve Type**	**Security Levels (bit)**					
			~	Arnd 128	~	Arnd 192	~	Arnd 256
Standard	ISO/IEC	BN256I	X					
		BN384		X				
		BN512I			X			
	TCG	BN256I	X					
		BN638			X			
	FIDO/W3C	BN256I	X					
		BN256D	X					
		BN512I			X			
		BN638			X			
Library	mcl	BLS12_381		X				
		BN254N	X					
		BN_SNARK1	X					
		BN382M		X				
		BN462		X				
	TEPLA	BN254B	X					

Category	Name	Curve Type	Security Levels (bit)					
			~	Arnd 128	~	Arnd 192	~	Arnd 256
		BN254N	X					
	RELIC	BLS12_381		X				
		BLS12_446		X				
		BLS12_455		X				
		BLS12_638			X			
		BLS24_477				X		
		BLS48_575						X
		BN254N	X					
		BN256D	X					
		BN382R		X				
		BN446		X				
		BN638			X			
		CP8_544		X				
		K54_569						X
		KSS18_508			X			
		OT8_511		X				
	AMCL	BLS12_381		X				
		BLS12_383		X				
		BLS12_461		X				
		BLS24_479				X		
		BLS48_556						X
		BN254N	X					
		BN254CX	X					
		BN256I	X					
		BN512I			X			
	Intel IPP	BN256I	X					
	Kyushu Univ.	BLS48_581						X
	MIRACL	BLS12_381		X				
		BLS12_383		X				
		BLS12_461		X				
		BLS24_479				X		
		BLS48_556						X
		BLS48_581						X
		BN254N	X					
		BN254CX	X					
		BN256I	X					
		BN462		X				
		BN512I			X			
	Adjoint	BLS12_381		X				
		BN_SNARK1	X					
		BN254B	X					
		BN254N	X					

Category	Name	Curve Type	Security Levels (bit)					
			~	Arnd 128	~	Arnd 192	~	Arnd 256
		BN254S1	X					
		BN254S2	X					
		BN462		X				
Application	Zcash	BLS12_381		X				
		BN_SNARK1	X					
	Ethereum	BLS12_381		X				
	Chia Network	BLS12_381		X				
	DFINITY	BLS12_381		X				
		BN254N	X					
		BN_SNARK1	X					
		BN382M		X				
		BN462		X				
	Algorand	BLS12_381		X				

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 [ISOIEC15946-5] 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 K54_569)[MAF19], a KSS curve [KSS08] 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 developper's name because he proposed these parameters.

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 F_p, the towers of extension field F_p^2, F_p^6 and F_p^12 are defined by indeterminates u, v, and w as follows:

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

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 into 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(F_p)

-r : the order of G_1

-BP = (x,y) : a 'base point', i.e., a generator of G_1

-h : the cofactor #E(F_p)/r

*G_2 is an r-order subgroup of E'(F_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 in F_p)

oy' = y'_0 + y'_1 * u (y'_0, y'_1 in F_p)

-h' : the cofactor #Et(F_p^8)/r

p:
```
0x1a0111ea397fe69a4b1ba7b6434bacd764774b84f38512bf6730d2a0f6b0f6241eabfffeb153ffffb9fef
```

r:
```
0x73eda753299d7d483339d80809a1d80553bda402fffe5bfeffffffff00000001
```

x:
```
0x17f1d3a73197d7942695638c4fa9ac0fc3688c4f9774b905a14e3a3f171bac586c55e83ff97a1aeffb3af
```

y:
```
0x08b3f481e3aaa0f1a09e30ed741d8ae4fcf5e095d5d00af600db18cb2c04b3edd03cc744a2888ae40caa2
```

h: 0x396c8c005555e1568c00aaab0000aaab

b: 4

r':
```
0x1a0111ea397fe69a4b1ba7b6434bacd764774b84f38512bf6730d2a0f6b0f6241eabfffeb153ffffb9fef
```

x'_0:
```
0x024aa2b2f08f0a91260805272dc51051c6e47ad4fa403b02b4510b647ae3d1770bac0326a805bbefd4805
```

x'_1:
```
0x13e02b6052719f607dacd3a088274f65596bd0d09920b61ab5da61bbdc7f5049334cf11213945d57e5ac7
```

y'_0:
```
0x0ce5d527727d6e118cc9cdc6da2e351aadfd9baa8cbdd3a76d429a695160d12c923ac9cc3baca289e1935
```

y'_1:
```
0x0606c4a02ea734cc32acd2b02bc28b99cb3e287e85a763af267492ab572e99ab3f370d275cec1da1aaa9c
```

h':
```
0x5d543a95414e7f1091d50792876a202cd91de4547085abaa68a205b2e5a7ddfa628f1cb4d9e82ef21537e
```

b': 4 * (u + 1)

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].

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 F_p, the towers of extension field F_p^2, F_p^6 and F_p^12 are defined by indeterminates u, v, and w as follows:

```
F_p^2 = F_p[u] / (u^2 + 1)
F_p^6 = F_p^2[v] / (v^3 - u - 2)
F_p^12 = F_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 + 5 and E': y^2 = x^3 - u + 2, respectively. The size of p becomes 462-bit length. BN462 is categorized into 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 more efficient than BN462, 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(F_p)

-r : the order of G_1

-BP = (x,y) : a 'base point', i.e., a generator of G_1

-h : the cofactor #E(F_p)/r

*G_2 is an r-order subgroup of E'(F_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 in F_p) oy' = y'_0 + y'_1 * u (y'_0, y'_1 in F_p)

-h' : the cofactor #Et(F_p^8)/r

p:
0x240480360120023ffffffffff6ff0cf6b7d9bfca0000000000d812908f41c8020ffffffffff6ff66fc6ff

r:
0x240480360120023ffffffffff6ff0cf6b7d9bfca0000000000d812908ee1c201f7ffffffffff6ff66fc7bf

x:
0x21a6d67ef250191fadba34a0a30160b9ac9264b6f95f63b3edbec3cf4b2e689db1bbb4e69a416a0b1e792

y:
0x0118ea0460f7f7abb82b33676a7432a490eeda842cccfa7d788c659650426e6af77df11b8ae40eb80f475

h: 1

b: 5

r':
0x240480360120023ffffffffff6ff0cf6b7d9bfca0000000000d812908ee1c201f7ffffffffff6ff66fc7bf

x'_0:
0x0257ccc85b58dda0dfb38e3a8cbdc5482e0337e7c1cd96ed61c913820408208f9ad2699bad92e0032ae1f

x'_1:
0x1d2e4343e8599102af8edca849566ba3c98e2a354730cbed9176884058b18134dd86bae555b783718f50a

y'_0:
0x0a0650439da22c1979517427a20809eca035634706e23c3fa7a6bb42fe810f1399a1f41c9ddae32e03695

y'_1:
0x073ef0cbd438cbe0172c8ae37306324d44d5e6b0c69ac57b393f1ab370fd725cc647692444a04ef87387a

h':
0x240480360120023ffffffffff6ff0cf6b7d9bfca0000000000d812908fa1ce0227ffffffffff6ff66fc63f

b': -u + 2

4.3. For 192-bit Security

As shown in Table 1, there are only two candidates of pairing-friendly curves for the 192-bit security level, BLS24_477 and BLS24_479. BLS24_477 has only one implementation and BLS24_479 is an experimental parameter that is not shown in any peer-reviewed paper. Therefore, because neither match our selection policy, we do not show the parameters for 192-bit security here.

4.4. 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 F_p, the towers of extension field F_p^2, F_p^4, F_p^8, F_p^24 and F_p^48 are defined by indeterminates u, v, w, z, and s as follows:

```
 F_p^2 = F_p[u] / (u^2 + 1)
 F_p^4 = F_p^2[v] / (v^2 + u + 1)
 F_p^8 = F_p^4[w] / (w^2 + v)
 F_p^24 = F_p^8[z] / (z^3 + w)
 F_p^48 = F_p^24[s] / (s^2 + z).
```

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 into D-type.

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

- *G_1 is the largest prime-order subgroup of E(F_p)
  - -r : the order of G_1
  - -BP = (x,y) : a 'base point', i.e., a generator of G_1
  - -h : the cofactor #E(F_p)/r
- *G_2 is an r-order subgroup of E'(F_p^8)
  - -r': an order

-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'_2 * v + x'_3 * u * v + x'_4 * w +
 x'_5 * u * w + x'_6 * v * w + x'_7 * u * v * w (x'_0, ...,
 x'_7 in F_p)

oy' = 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 (y'_0, ...,
 y'_7 in F_p)
```

-h' : the cofactor #E'(F_p^8)/r

p:

0x1280f73ff3476f313824e31d47012a0056e84f8d122131bb3be6c0f1f3975444a48ae43af6e082acd9cd

r:

0x2386f8a925e2885e233a9ccc1615c0d6c635387a3f0b3cbe003fad6bc972c2e6e741969d34c4c92016a8

x:

0x02af59b7ac340f2baf2b73df1e93f860de3f257e0e86868cf61abdbaedffb9f7544550546a9df6f96458

y:

0x0cefda44f6531f91f86b3a2d1fb398a488a553c9efeb8a52e991279dd41b720ef7bb7beffb98aee53e80

x'_0:

0x05d615d9a7871e4a38237fa45a2775debabbefc70344dbccb7de64db3a2ef156c46ff79baad1a8c42281

x'_1:

0x07c4973ece2258512069b0e86abc07e8b22bb6d980e1623e9526f6da12307f4e1c3943a00abfedf16214

x'_2:

0x01fccc70198f1334e1b2ea1853ad83bc73a8a6ca9ae237ca7a6d6957ccbab5ab6860161c1dbd19242ffa

x'_3:

0x0be2218c25ceb6185c78d8012954d4bfe8f5985ac62f3e5821b7b92a393f8be0cc218a95f63e1c776e6e

x'_4:

0x038b91c600b35913a3c598e4caa9dd63007c675d0b1642b5675ff0e7c5805386699981f9e48199d5ac10

x'_5:

0x0c96c7797eb0738603f1311e4ecda088f7b8f35dcef0977a3d1a58677bb037418181df63835d28997eb5

x'_6:

0x0b9b7951c6061ee3f0197a498908aee660dea41b39d13852b6db908ba2c0b7a449cef11f293b13ced0fd

x'_7:

0x0827d5c22fb2bdec5282624c4f4aaa2b1e5d7a9defaf47b5211cf741719728a7f9f8cfca93f29cff364a

y'_0:

0x00eb53356c375b5dfa497216452f3024b918b4238059a577e6f3b39ebfc435faab0906235afa27748d90f

y'_1:
0x0284dc75979e0ff144da6531815fcadc2b75a422ba325e6fba01d72964732fcbf3afb096b243b1f192c5c

y'_2:
0x0b36a201dd008523e421efb70367669ef2c2fc5030216d5b119d3a480d370514475f7d5c99d0e90411515

y'_3:
0x0aec25a4621edc0688223fbbd478762b1c2cded3360dcee23dd8b0e710e122d2742c89b224333fa40dced

y'_4:
0x0d209d5a223a9c46916503fa5a88325a2554dc541b43dd93b5a959805f1129857ed85c77fa238cdce8a1e

y'_5:
0x07d0d03745736b7a513d339d5ad537b90421ad66eb16722b589d82e2055ab7504fa83420e8c270841f682

y'_6:
0x0896767811be65ea25c2d05dfdd17af8a006f364fc0841b064155f14e4c819a6df98f425ae3a2864f22c1

y'_7:
0x035e2524ff89029d393a5c07e84f981b5e068f1406be8e50c87549b6ef8eca9a9533a3f8e69c31e97e1ad

h: 0x85555841aaaec4ac

b: 1

r':
0x2386f8a925e2885e233a9ccc1615c0d6c635387a3f0b3cbe003fad6bc972c2e6e741969d34c4c92016a85

h':
0x170e915cb0a6b7406b8d94042317f811d6bc3fc6e211ada42e58ccfcb3ac076a7e4499d700a0c23dc4b0c

b': -1 / w

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 know, 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 Table 1, 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. Implementers MAY use pairing-friendly curves with 100-bit security only if they need to keep interoperability with the existing applications.

In addition, implementors should be aware of the following points when they implement pairing-based cryptographic applications using recommended curves.

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, calculate a scalar multiplication 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 when they implement a module of scalar multiplication[Montgomery] [RFC7748].

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]. 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.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The authors would like to thank Akihiro Kato and Shoko Yonezawa for their significant contribution to an early version of this memo. The authors would also like to acknowledge Sakae Chikara, Kim Taechan, Hoeteck Wee, Sergey Gorbunov, Michael Scott, Chloe Martindale as an Expert Reviewer, Watson Ladd, Armand Faz, and Satoru Kanno for their valuable comments.

8. References

8.1. Normative References

[BD18] Barbulescu, R. and S. Duquesne, "Updating Key Size Estimations for Pairings", DOI 10.1007/s00145-018-9280-5, Journal of Cryptology, January 2018, <https://doi.org/10.1007/s00145-018-9280-5>.

[BLS02] Barreto, P., Lynn, B., and M. Scott, "Constructing Elliptic Curves with Prescribed Embedding Degrees", DOI 10.1007/3-540-36413-7_19, Security in Communication Networks pp. 257-267, 2003, <https://doi.org/10.1007/3-540-36413-7_19>.

[BN05] Barreto, P. and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order", DOI 10.1007/11693383_22, Selected Areas in Cryptography pp. 319-331, 2006, <https://doi.org/10.1007/11693383_22>.

[GMT19] Guillevic, A., Masson, S., and E. Thome, "Cocks-Pinch curves of embedding degrees five to eight and optimal ate pairing computation", DOI 10.1007/s10623-020-00727-w, International Journal of Designs, Codes and Cryptography vol. 88, pp. 1047-1081, 2019, <https://doi.org/10.1007/s10623-020-00727-w>.

[KB16] Kim, T. and R. Barbulescu, "Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case", DOI 10.1007/978-3-662-53018-4_20, Advances in Cryptology - CRYPTO 2016 pp. 543-571, 2016, <https://doi.org/10.1007/978-3-662-53018-4_20>.

[KIK17] Kiyomura, Y., Inoue, A., Kawahara, Y., Yasuda, M., Takagi, T., and T. Kobayashi, "Secure and Efficient Pairing at 256-Bit Security Level", DOI 10.1007/978-3-319-61204-1_4, Applied Cryptography and

Network Security pp. 59-79, 2017, <https://doi.org/10.1007/978-3-319-61204-1_4>.

[NIST] Barker, E., "NIST special publication 800-57 part 1 (revised) : Recommendation for key management, part 1: General (revised)", National Institute of Standards and Technology (NIST), 2020.

[RFC2119] 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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[Ver09] Vercauteren, F., "Optimal Pairings", DOI 10.1109/tit.2009.2034881, IEEE Transactions on Information Theory Vol. 56, pp. 455-461, January 2010, <https://doi.org/10.1109/tit.2009.2034881>.

8.2. Informative References

[AdjointLib] Adjoint Inc., "Optimised bilinear pairings over elliptic curves", 2018, <https://github.com/adjoint-io/pairing>.

[AFKMR12] Aranha, D.F., Fuentes-Castaneda, L., Knapp, E., Menezes, A., and F. Rodríguez-Henríquez, "Implementing Pairings at the 192-Bit Security Level", DOI / 10.1007/978-3-642-36334-4_11, Pairing 2012 pp. 177-195, 2012, <https://doi.org//10.1007/978-3-642-36334-4_11>.

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

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

[AMCLv2] The Apache Software Foundation, "Old version of the Apache Milagro Cryptographic Library", 2016, <https://github.com/miracl/amcl/tree/master/version22>.

[BCM15] Barreto, P. S. L. M., Costello, C., Misoczki, R., Naehrig, M., Pereira, G. C. C. F., and G. Zanon, "Subgroup security in pairing-based cryptography", Cryptology ePrint Archive Report 2015/247, 2015, <https://eprint.iacr.org/2015/247.pdf>.

[BGMORT10] Beuchat, J., González-Díaz, J., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., and T. Teruya, "High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves", DOI 10.1007/978-3-642-17455-1_2, Pairing 2010 pp. 21-39, 2010, <https://doi.org/10.1007/978-3-642-17455-1_2>.

[BL10] Brickell, E. and J. Li, "Enhanced Privacy ID from Bilinear Pairing for Hardware Authentication and Attestation", DOI 10.1109/socialcom.2010.118, 2010 IEEE Second International Conference on Social Computing, August 2010, <https://doi.org/10.1109/socialcom.2010.118>.

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

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

[CCS07] Chen, L., Cheng, Z., and N. Smart, "Identity-based key agreement protocols from pairings", DOI 10.1007/s10207-006-0011-9, International Journal of Information Security Vol. 6, pp. 213-241, January 2007, <https://doi.org/10.1007/s10207-006-0011-9>.

[Cheon06] Cheon, J. H., "Security Analysis of the Strong Diffie-Hellman Problem", DOI 10.1007/11761679_1, EUROCRYPT 2006 pp. 1-11, 2006, <https://doi.org/10.1007/11761679_1>.

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

[CIRCL] Cloudflare, "CIRCL: Cloudflare Interoperable, Reusable Cryptographic Library", 2019, <https://github.com/cloudflare/circl>.

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

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

[DSD07]
Devegili, A. J., Scott, M., and R. Dahab, "Implementing Cryptographic Pairings over Barreto-Naehrig Curves", DOI 10.1007/978-3-540-73489-5_10, Pairing 2007 pp. 197-207, 2007, <https://doi.org/10.1007/978-3-540-73489-5_10>.

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

[EPID] Intel Corporation, "Intel (R) SGX: Intel (R) EPID Provisioning and Attestation Services", , <https://software.intel.com/en-us/download/intel-sgx-intel-epid-provisioning-and-attestation-services>.

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

[FIDO] 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>.

[FK18] Fotiadis, G. and E. Konstantinou, "TNFS Resistant Families of Pairing-Friendly Elliptic Curves", Cryptology ePrint Archive Report 2018/1017, 2018, <https://eprint.iacr.org/2018/1017.pdf>.

[FM19] Fotiadis, G. and C. Martindale, "Optimal TNFS-secure pairings on elliptic curves with composite embedding degree", Cryptology ePrint Archive Report 2019/555, 2019, <https://eprint.iacr.org/2019/555.pdf>.

[Freeman06] Freeman, D., "Constructing pairing-friendly elliptic curves with embedding degree 10", DOI 10.1007/11792086_32, ANTS 2006 pp. 452-465, 2006, <https://doi.org/10.1007/11792086_32>.

[FSU10] Fujioka, A., Suzuki, K., and B. Ustaoglu, "Ephemeral Key Leakage Resilient and Efficient ID-AKEs That Can Share Identities, Private and Master Keys", DOI 10.1007/978-3-642-17455-1_12, Lecture Notes in Computer Science pp. 187-205, 2010, <https://doi.org/10.1007/978-3-642-17455-1_12>.

[HR83] Hellman, M. and J. Reyneri, "Fast Computation of Discrete Logarithms in GF (q)", DOI 10.1007/978-1-4757-0602-4_1, Advances in Cryptology pp. 3-13, 1983, <https://doi.org/10.1007/978-1-4757-0602-4_1>.

[I-D.boneh-bls-signature]
Boneh, D., Gorbunov, S., Wee, H., and Z. Zhang, "BLS Signature Scheme", Work in Progress, Internet-Draft, draft-boneh-bls-signature-00, 8 February 2019, <https://tools.ietf.org/html/draft-boneh-bls-signature-00>.

[I-D.ietf-lwig-curve-representations]
Struik, R., "Alternative Elliptic Curve Representations", Work in Progress, Internet-Draft, draft-ietf-lwig-curve-representations-08, 24 July 2019, <https://tools.ietf.org/html/draft-ietf-lwig-curve-representations-08>.

[IEEE1363] "IEEE Standard Specifications for Public-Key Cryptography", IEEE standard, DOI 10.1109/IEEESTD.2000.92292, 2000, <https://doi.org/10.1109/IEEESTD.2000.92292>.

[Intel-IPP] 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>.

[ISOIEC11770-3] ISO/IEC, "ISO/IEC 11770-3:2015", ISO/IEC Information technology -- Security techniques -- Key management -- Part 3: Mechanisms using asymmetric techniques, 2015.

[ISOIEC15946-5] 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.

[Joux00] Joux, A., "A One Round Protocol for Tripartite Diffie-Hellman", DOI 10.1007/10722028_23, Lecture Notes in Computer Science pp. 385-393, 2000, <https://doi.org/10.1007/10722028_23>.

[KSS08] Kachisa, E., Schaefer, E., and M. Scott, "Constructing Brezing-Weng Pairing-Friendly Elliptic Curves Using Elements in the Cyclotomic Field", DOI 10.1007/978-3-540-85538-5_9, Pairing 2008 pp. 126-135, 2008, <https://doi.org/10.1007/978-3-540-85538-5_9>.

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

[M-Pin] Scott, M., "M-Pin: A Multi-Factor Zero Knowledge Authentication Protocol", July 2019, <https://

www.miracl.com/miracl-labs/m-pin-a-multi-factor-zero-knowledge-authentication-protocol>.

[MAF19] Mbiang, N.B., Aranha, D.F., and E. Fouotsa, "Computing the Optimal Ate Pairing over Elliptic Curves with Embedding Degrees 54 and 48 at the 256-bit security level", International Journal of Applied Cryptography to appear, 2019, <https://www.researchgate.net/publication/337011283_Computing_the_Optimal_Ate_Pairing_over_Elliptic_Curves_with_Embedding_Degrees_54_and_48_at_the_256-bit_security_level>.

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

[MIRACL] MIRACL Ltd., "The MIRACL Core Cryptographic Library", 2019, <https://github.com/miracl/core>.

[MNT01] Miyaji, A., Nakabayashi, M., and S. Takano, "New explicit conditions of Elliptic Curve Traces under FR reduction", IEICE Trans. Fundamentals. E84-A(5) pp. 1234-1243, 2001.

[Montgomery] Montgomery, P., "Speeding the Pollard and Elliptic Curve Methods of Factorization", MATHEMATICS OF COMPUTATION , January, 1987, <https://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866113-7/S0025-5718-1987-0866113-7.pdf>.

[MP04] Guillevic, A., Masson, S., and E. Thome, "Cocks-Pinch curves of embedding degrees five to eight and optimal ate pairing computation", Cryptology ePrint Archive Report 2019/431, 2019, <https://eprint.iacr.org/2004/032.pdf>.

[NASKM08] Nogami, Y., Akane, M., Sakemi, Y., Kato, H., and Y. Morikawa, "Integer Variable X-Based Ate Pairing", DOI 10.1007/978-3-540-85538-5_13, Pairing 2008 pp. 178-191, 2008, <https://doi.org/10.1007/978-3-540-85538-5_13>.

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

[Pollard78] Pollard, J., "Monte Carlo methods for index computation $({\rm mod}\ p)$", DOI 10.1090/s0025-5718-1978-0491431-9, Mathematics of Computation Vol. 32, pp. 918-918, September 1978, <https://doi.org/10.1090/s0025-5718-1978-0491431-9>.

[pureGo-bls] Meyer, J., "Pure GO bls library", 2019, <https://github.com/phoreproject/bls>.

[RELIC]
Gouvea, C.P.L., "RELIC is an Efficient LIbrary for Cryptography", 2013, <https://github.com/relic-toolkit/relic>.

[RFC5091] 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>.

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

[RFC6509] 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>.

[RFC6539] 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>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[S86] Silverman, J. H., "The arithmetic of elliptic curves", Springer GTM 106, 1986.

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

[SG19] Scott, M. and A. Guillevic, "A New Family of Pairing-Friendly elliptic curves", Cryptology ePrint Archive

Report 2019/193, 2019, <https://eprint.iacr.org/2018/193.pdf>.

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

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

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

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

[ZCashRep] Electric Coin Company, "BLS12-381", July 2017, <https://github.com/zkcrypto/pairing/blob/master/src/bls12_381/README.md>.

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

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.

```
 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, 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. 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 * t + 2 for a parameter t and c_0, c_1, ... , c_L in {-1,0,1} such that the sum of c_i * 2^i (i = 0, 1, ..., 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, ...,c_L in {-1,0,1} such that the sum of c_i * 2^i (i = 0, 1, ..., L) equals c, and the order r of G_1 as input, and outputs e(P, Q).

```
f := 1; T := Q;
if (c_L = -1)
    T := -T;
end if
for i = L-1 to 0
    f := f^2 * Line_function(T, T, P); T := 2 * T;
    if (c_i = 1 | c_i = -1)
        f := f * Line_function(T, c_i * Q); T := T + c_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 c = t for a parameter t and c_0, c_1, ... , c_L in {-1,0,1} such that the sum of c_i * 2^i (i = 0, 1, ..., 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, ..., c_L in {-1,0,1} such that the sum of c_i * 2^i (i = 0, 1, ..., L), and an order r as input, and outputs e(P, Q).

```
f := 1; T := Q;
if (c_L = -1)
    T := -T;
end if
for i = L-1 to 0
    f := f^2 * Line_function(T, T, P); T := 2 * T;
    if (c_i = 1 | c_i = -1)
        f := f * Line_function(T, c_i * Q, P); T := T + c_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 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 * u and
y' = y'_0 + y'_1 * u,
```

where u is a indeterminate and x'_0, x'_1, y'_0, y'_1 are elements of F_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 and
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 F_p. The representation of Q = (x', y') given below is followed by [I-D.ietf-lwig-curve-representations].

BLS12-381:

Input x value:

0x17f1d3a73197d7942695638c4fa9ac0fc3688c4f9774b905a14e3a3f171bac586c55e83ff97a1aeffb3af

Input y value:

0x08b3f481e3aaa0f1a09e30ed741d8ae4fcf5e095d5d00af600db18cb2c04b3edd03cc744a2888ae40caa2

Input x'_0 value:

0x024aa2b2f08f0a91260805272dc51051c6e47ad4fa403b02b4510b647ae3d1770bac0326a805bbefd4805

Input x'_1 value:

0x13e02b6052719f607dacd3a088274f65596bd0d09920b61ab5da61bbdc7f5049334cf11213945d57e5ac7

Input y'_0 value:

0x0ce5d527727d6e118cc9cdc6da2e351aadfd9baa8cbdd3a76d429a695160d12c923ac9cc3baca289e1935

Input y'_1 value:

0x0606c4a02ea734cc32acd2b02bc28b99cb3e287e85a763af267492ab572e99ab3f370d275cec1da1aaa90

e_0:

0x11619b45f61edfe3b47a15fac19442526ff489dcda25e59121d9931438907dfd448299a87dde3a649bdba

e_1:

0x153ce14a76a53e205ba8f275ef1137c56a566f638b52d34ba3bf3bf22f277d70f76316218c0dfd583a394

e_2:

0x095668fb4a02fe930ed44767834c915b283b1c6ca98c047bd4c272e9ac3f3ba6ff0b05a93e59c71fba77b

e_3:

0x16deedaa683124fe7260085184d88f7d036b86f53bb5b7f1fc5e248814782065413e7d958d17960109ea0

e_4:

0x09c92cf02f3cd3d2f9d34bc44eee0dd50314ed44ca5d30ce6a9ec0539be7a86b121edc61839ccc908c4bc

e_5:

0x111061f398efc2a97ff825b04d21089e24fd8b93a47e41e60eae7e9b2a38d54fa4dedced0811c34ce5287

e_6:

0x01ecfcf31c86257ab00b4709c33f1c9c4e007659dd5ffc4a735192167ce197058cfb4c94225e7f1b6c26a

e_7:

0x08890726743a1f94a8193a166800b7787744a8ad8e2f9365db76863e894b7a11d83f90d873567e9d645cc

e_8:

0x0e61c752414ca5dfd258e9606bac08daec29b3e2c57062669556954fb227d3f1260eedf25446a086b0844

e_9:

0x0fe63f185f56dd29150fc498bbeea78969e7e783043620db33f75a05a0a2ce5c442beaff9da195ff15164

e_10:

0x10900338a92ed0b47af211636f7cfdec717b7ee43900eee9b5fc24f0000c5874d4801372db478987691c5

e_11:

0x1454814f3085f0e6602247671bc408bbce2007201536818c901dbd4d2095dd86c1ec8b888e59611f60a30

BN462:

Input x value:

0x21a6d67ef250191fadba34a0a30160b9ac9264b6f95f63b3edbec3cf4b2e689db1bbb4e69a416a0b1e792

Input y value:

0x0118ea0460f7f7abb82b33676a7432a490eeda842cccfa7d788c659650426e6af77df11b8ae40eb80f475

Input x'_0 value:

0x0257ccc85b58dda0dfb38e3a8cbdc5482e0337e7c1cd96ed61c913820408208f9ad2699bad92e0032ae1f

Input x'_1 value:

0x1d2e4343e8599102af8edca849566ba3c98e2a354730cbed9176884058b18134dd86bae555b783718f50a

Input y'_0 value:

0x0a0650439da22c1979517427a20809eca035634706e23c3fa7a6bb42fe810f1399a1f41c9ddae32e03695

Input y'_1 value:

0x073ef0cbd438cbe0172c8ae37306324d44d5e6b0c69ac57b393f1ab370fd725cc647692444a04ef87387a

e_0:

0x0cf7f0f2e01610804272f4a7a24014ac085543d787c8f8bf07059f93f87ba7e2a4ac77835d4ff10e7866

e_1:

0x00ef2c737515694ee5b85051e39970f24e27ca278847c7cfa709b0df408b830b3763b1b001f1194445b62

e_2:

0x04d685b29fd2b8faedacd36873f24a06158742bb2328740f93827934592d6f1723e0772bb9ccd3025f88d

e_3:

0x090067ef2892de0c48ee49cbe4ff1f835286c700c8d191574cb424019de11142b3c722cc5083a7191241

e_4:

0x1437603b60dce235a090c43f5147d9c03bd63081c8bb1ffa7d8a2c31d673230860bb3dfe4ca85581f745

e_5:

0x13191b1110d13650bf8e76b356fe776eb9d7a03fe33f82e3fe5732071f305d201843238cc96fd0e892bc

e_6:

0x07b1ce375c0191c786bb184cc9c08a6ae5a569dd7586f75d6d2de2b2f075787ee5082d44ca4b8009b328

e_7:
0x05b64add5e49574b124a02d85f508c8d2d37993ae4c370a9cda89a100cdb5e1d441b57768dbc68429ffa

e_8:
0x0fd9a3271854a2b4542b42c55916e1faf7a8b87a7d10907179ac7073f6a1de044906ffaf4760d11c8f92

e_9:
0x17fa0c7fa60c9a6d4d8bb9897991efd087899edc776f33743db921a689720c82257ee3c788e8160c112f

e_10:
0x0c901397a62bb185a8f9cf336e28cfb0f354e2313f99c538cdceedf8b8aa22c23b896201170fc915690f

e_11:
0x20f27fde93cee94ca4bf9ded1b1378c1b0d80439eeb1d0c8daef30db0037104a5e32a2ccc94fa1860a95

BLS48-581:

Input x value:

0x02af59b7ac340f2baf2b73df1e93f860de3f257e0e86868cf61abdbaedffb9f7544550546a9df6f964584

Input y value:

0x0cefda44f6531f91f86b3a2d1fb398a488a553c9efeb8a52e991279dd41b720ef7bb7beffb98aee53e80f

x'_0:

0x05d615d9a7871e4a38237fa45a2775debabbefc70344dbccb7de64db3a2ef156c46ff79baad1a8c42281a

x'_1:

0x07c4973ece2258512069b0e86abc07e8b22bb6d980e1623e9526f6da12307f4e1c3943a00abfedf16214a

x'_2:

0x01fccc70198f1334e1b2ea1853ad83bc73a8a6ca9ae237ca7a6d6957ccbab5ab6860161c1dbd19242ffae

x'_3:

0x0be2218c25ceb6185c78d8012954d4bfe8f5985ac62f3e5821b7b92a393f8be0cc218a95f63e1c776e6ec

x'_4:

0x038b91c600b35913a3c598e4caa9dd63007c675d0b1642b5675ff0e7c5805386699981f9e48199d5ac10b

x'_5:

0x0c96c7797eb0738603f1311e4ecda088f7b8f35dcef0977a3d1a58677bb037418181df63835d28997eb57

x'_6:

0x0b9b7951c6061ee3f0197a498908aee660dea41b39d13852b6db908ba2c0b7a449cef11f293b13ced0fd0

x'_7:

0x0827d5c22fb2bdec5282624c4f4aaa2b1e5d7a9defaf47b5211cf741719728a7f9f8cfca93f29cff364a7

y'_0:

0x00eb53356c375b5dfa497216452f3024b918b4238059a577e6f3b39ebfc435faab0906235afa27748d90f

y'_1:

0x0284dc75979e0ff144da6531815fcadc2b75a422ba325e6fba01d72964732fcbf3afb096b243b1f192c5c

y'_2:

0x0b36a201dd008523e421efb70367669ef2c2fc5030216d5b119d3a480d370514475f7d5c99d0e90411515

y'_3:
0x0aec25a4621edc0688223fbbd478762b1c2cded3360dcee23dd8b0e710e122d2742c89b224333fa40dce

y'_4:
0x0d209d5a223a9c46916503fa5a88325a2554dc541b43dd93b5a959805f1129857ed85c77fa238cdce8a1

y'_5:
0x07d0d03745736b7a513d339d5ad537b90421ad66eb16722b589d82e2055ab7504fa83420e8c270841f68

y'_6:
0x0896767811be65ea25c2d05dfdd17af8a006f364fc0841b064155f14e4c819a6df98f425ae3a2864f22c

y'_7:
0x035e2524ff89029d393a5c07e84f981b5e068f1406be8e50c87549b6ef8eca9a9533a3f8e69c31e97e1a

e_0:
0x0e26c3fcb8ef67417814098de5111ffcccc1d003d15b367bad07cef2291a93d31db03e3f03376f3beae2

e_1:
0x069061b8047279aa5c2d25cdf676ddf34eddbc8ec2ec0f03614886fa828e1fc066b26d35744c0c382718

e_2:
0x02b9bece645fbf9d8f97025a1545359f6fe3ffab3cd57094f862f7fb9ca01c88705c26675bcc723878e9

e_3:
0x0080d267bf036c1e61d7fc73905e8c630b97aa05ef3266c82e7a111072c0d2056baa8137fba111c9650d

e_4:
0x03c6b4c12f338f9401e6a493a405b33e64389338db8c5e592a8dd79eac7720dd83dd6b0c189eeda20809

e_5:
0x016e46224f28bfd8833f76ac29ee6e406a9da1bde55f5e82b3bd977897a9104f18b9ee41ea9af7d4183d

e_6:
0x008ddce7a4a1b94be5df3ceea56bef0077dcdde86d579938a50933a47296d337b7629934128e2457e241

e_7:

0x060ef6eae55728e40bd4628265218b24b38cdd434968c14bfefb87f0dcbfc76cc473ae2dc0cac6e69dfd

e_8:

0x0c3943636876fd4f9393414099a746f84b2633dfb7c36ba6512a0b48e66dcb2e409f1b9e150e36b0b431

e_9:

0x02d31eb8be0d923cac2a8eb6a07556c8951d849ec53c2848ee78c5eed40262eb21822527a8555b071f1c

e_10:

0x07f19673c5580d6a10d09a032397c5d425c3a99ff1dd0abe5bec40a0d47a6b8daabb22edb6b06dd86919

e_11:

0x0d3fe01f0c114915c3bdf8089377780076c1685302279fd9ab12d07477aac03b69291652e9f179baa0a9

e_12:

0x0662eefd5fab9509aed968866b68cff3bc5d48ecc8ac6867c212a2d82cee5a689a3c9c67f1d611adac72

e_13:

0x0aad8f4a8cfdca8de0985070304fe4f4d32f99b01d4ea50d9f7cd2abdc0aeea99311a36ec6ed18208642

e_14:

0x0ffcf21d641fd9c6a641a749d80cab1bcad4b34ee97567d905ed9d5cfb74e9aef19674e2eb6ce3dfb706

e_15:

0x0cbe92a53151790cece4a86f91e9b31644a86fc4c954e5fa04e707beb69fc60a858fed8ebd53e4cfd515

e_16:

0x0202db83b1ff33016679b6cfc8931deea6df1485c894dcd113bacf564411519a42026b5fda4e16262674

e_17:

0x070a617ed131b857f5b74b625c4ef70cc567f619defb5f2ab67534a1a8aa72975fc4248ac8551ce02b68

e_18:

0x070e1ebce457c141417f88423127b7a7321424f64119d5089d883cb953283ee4e1f2e01ffa7b903fe7a9

e_19:

0x058a06be5a36c6148d8a1287ee7f0e725453fa1bb05cf77239f235b417127e370cfa4f88e61a23ea16df

e_20:
0x0dfdfaaeb9349cf18d21b92ad68f8a7ecc509c35fcd4b8abeb93be7a204ac871f2195180206a2c340fcc

e_21:
0x0d06c8adfdd81275da2a0ce375b8df9199f3d359e8cf50064a3dc10a592417124a3b705b05a7ffe78e20

e_22:
0x0708effd28c4ae21b6969cb9bdd0c27f8a3e341798b6f6d4baf27be259b4a47688b50cb68a69a917a4a1

e_23:
0x09da7c7aa48ce571f8ece74b98431b14ae6fb4a53ae979cd6b2e82320e8d25a0ece1ca1563aa5aa6926e

e_24:
0x0a7150a14471994833d89f41daeaa999dfc24a9968d4e33d88ed9e9f07aa2432c53e486ba6e3b6e4f4b8

e_25:
0x084696f31ff27889d4dccdc4967964a5387a5ae071ad391c5723c9034f16c2557915ada07ec68f18672b

e_26:
0x0398e76e3d2202f999ac0f73e0099fe4e0fe2de9d223e78fc65c56e209cdf48f0d1ad8f6093e924ce5f0

e_27:
0x06d683f556022368e7a633dc6fe319fd1d4fc0e07acff7c4d4177e83a911e73313e0ed980cd9197bd17a

e_28:
0x0d764075344b70818f91b13ee445fd8c1587d1c0664002180bbac9a396ad4a8dc1e695b0c4267df4a090

e_29:
0x0aa6a32fdc4423b1c6d43e5104159bcd8e03a676d055d4496f7b1bc8761164a2908a3ff0e4c4d1f43620

e_30:
0x1147719959ac8eeab3fc913539784f1f947df47066b6c0c1beafecdb5fa784c3be9de5ab282a678a2a0c

e_31:
0x11a377bcebd3c12702bb34044f06f8870ca712fb5caa6d30c48ace96898fcbcddbcf31f331c9e524684c

e_32:

0x0b8b4511f451ba2cc58dc28e56d5e1d0a8f557ecb242f4d994a627e07cf3fa44e6d83cb907deacf303d2f

e_33:

0x090962d632ee2a57ce4208052ce47a9f76ea0fdad724b7256bb07f3944e9639a981d3431087241e30ae9b

e_34:

0x0931c7befc80acd185491c68af886fa8ee39c21ed3ebd743b9168ae3b298df485bfdc75b94f0b21aecd8c

e_35:

0x020ac007bf6c76ec827d53647058aca48896916269c6a2016b8c06f0130901c8975779f1672e581e2dfdb

e_36:

0x0c0aed0d890c3b0b673bf4981398dcbf0d15d36af6347a39599f3a22584184828f78f91bbbbd08124a976

e_37:

0x0ef7799241a1ba6baaa8740d5667a1ace50fb8e63accc3bc30dc07b11d78dc545b68910c027489a0d842c

e_38:

0x016663c940d062f4057257c8f4fb9b35e82541717a34582dd7d55b41ebadf40d486ed74570043b2a3c4de

e_39:

0x1184a79510edf25e3bd2dc793a5082fa0fed0d559fa14a5ce9ffca4c61f17196e1ffbb84326272e0d0793

e_40:

0x120e47a747d942a593d202707c936dafa6fed489967dd94e48f317fd3c881b1041e3b6bbf9e8031d44e39

e_41:

0x026b6e374108ecb2fe8d557087f40ab7bac8c5af0644a655271765d57ad71742aa331326d871610a8c4c3

e_42:

0x041be63a2fa643e5a66faeb099a3440105c18dca58d51f74b3bf281da4e689b13f365273a2ed397e7b1c2

e_43:

0x124018a12f0f0af881e6765e9e81071acc56ebcddadcd107750bd8697440cc16f190a3595633bb8900e68

e_44:

0x0d422de4a83449c535b4b9ece586754c941548f15d50ada6740865be9c0b066788b6078727c7dee299ac

e_45:
0x1119f6c5468bce2ec2b450858dc073fea4fb05b6e83dd20c55c9cf694cbcc57fc0effb1d33b9b5587852

e_46:
0x061eaa8e9b0085364a61ea4f69c3516b6bf9f79f8c79d053e646ea637215cf6590203b275290872e3d7b

e_47:
0x0add8d58e9ec0c9393eb8c4bc0b08174a6b421e15040ef558da58d241e5f906ad6ca2aa5de361421708a

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.

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 F_p^2 are represented as polynomial with F_p coefficients like Section 2.5.

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

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

```
sign_F_p(y) := { 1 if y > (p - 1) / 2, else
               { 0 otherwise.
```

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

```
sign_F_p^2(y') := { sign_F_p(y'_0) if y'_1 equals 0, else
                  { 1 if y'_1 > (p - 1) / 2, else
                  { 0 otherwise.
```

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_F_p(y), else if P is a point on E', S_bit = sign_F_p^2(y).

2. Let m_byte = (C_bit * 2^7) + (I_bit * 2^6) + (S_bit * 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 F_p, i.e., an integer in the inclusive range [0, p - 1]. In this case, let x_string = I2OSP(x, 48).

   * If P is the point at infinity on E', let x_string = 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 F_p, i.e., integers in the inclusive range [0, p - 1] (see discussion of vector representations above). In this case, let x_string = I2OSP(x_1, 48) || I2OSP(x_0, 48).

Notice that in all of the above cases, the 3 most significant bits of x_string[0] are guaranteed to be 0.

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

5. Let s_string = x_string || y_string.

6. Set s_string[0] = x_string[0] OR m_byte, where OR is computed bitwise. After this operation, the most significant bit of s_string[0] equals C_bit, the next bit equals I_bit, and the next equals S_bit. (This is true because the three most significant bits of x_string[0] are guaranteed to be zero, as discussed above.)

7. Output s_string.

C.2. Point deserialization procedure

The deserialization procedure is defined as follows for a string s_string. This procedure uses the OS2IP function defined in [RFC8017].

1. Let m_byte = s_string[0] AND 0xE0, where AND is computed bitwise. In other words, the three most significant bits of m_byte equal the three most significant bits of s_string[0], and the remaining bits are 0.

   If m_byte equals any of 0x20, 0x60, or 0xE0, output INVALID and stop decoding.

   Otherwise:

   * Let C_bit equal the most significant bit of m_byte,

   * Let I_bit equal the second most significant bit of m_byte, and

   * Let S_bit equal the third most significant bit of m_byte.

2. If C_bit is 1:

   * If s_string has length 48 bytes, the output point is on the curve E.

   * If s_string has length 96 bytes, the output point is on the curve E'.

   * If s_string has any other length, output INVALID and stop decoding.

   If C_bit is 0:

   * If s_string has length 96 bytes, the output point is on E.

   * If s_string has length 192 bytes, the output point is on E'.

   * If s_string has any other length, output INVALID and stop decoding.

3. Let s_string[0] = s_string[0] AND 0x1F, where AND is computed bitwise. In other words, set the three most significant bits of s_string[0] to 0.

4. If I_bit is 1:

   * If s_string is not the all zeros string, output INVALID and stop decoding.

   * Otherwise (i.e., if s_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_bit must be 0. Continue decoding.

5. If C_bit is 0:

   * Let x_string be the first half of s_string.

   * Let y_string be the last half of s_string.

   * Let x = OS2IP(x_string).

   * Let y = OS2IP(y_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_bit must be 1. Continue decoding.

6. Let x = OS2IP(s_string).

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

   *Let y2 = x^3 + 4 in F_p.

   *If y2 is not square in F_p, output INVALID and stop decoding.

   *Otherwise, let y = sqrt(y2) in F_p and let Y_bit = sign_F_p(y).

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

   *Let y2 = x^3 + 4 * (u + 1) in F_p^2.

   *If y2 is not square in F_p^2, output INVALID and stop decoding.

   *Otherwise, let y = sqrt(y2) in F_p^2 and let Y_bit = sign_F_p^2(y).

8. If S_bit equals Y_bit, output P = (x, y) and stop decoding. Otherwise, output P = (x, -y) and stop decoding.

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