Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition

Youssef El Housni[1,2,3] and Aurore Guillevic[4]

[1] EY Blockchain, Paris, France
[2] École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[3] Inria, équipe-projet GRACE, Université Paris–Saclay, France
youssef.el.housni@fr.ey.com
[4] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
aurore.guillevic@inria.fr

Abstract. A zero-knowledge proof is a method by which one can prove knowledge of general non-deterministic polynomial (NP) statements. SNARKs are in addition non-interactive, short and cheap to verify. This property makes them suitable for recursive proof composition, that is proofs attesting to the validity of other proofs. Recursive proof composition has been empirically demonstrated for pairing-based SNARKs via tailored constructions of expensive elliptic curves. We thus construct on top of the curve BLS12-377 a new pairing-friendly elliptic curve which is STNFS-secure and optimized for one layer composition. We show that it is at least five times faster to verify a composed SNARK proof on this curve compared to the previous state-of-the-art. We propose to name the new curve BW6-761.

1 Introduction

Proofs of knowledge are a powerful tool that was introduced in [21] and is studied both in theoretical and applied cryptography. Since then, a lot of work has been conducted to design short non-interactive proofs that are cheap to verify, resulting in succinct non-interactive arguments of knowledge (SNARKs). Zero-knowledge (zk) SNARKs allow a prover to convince a verifier that they know a witness to an instance being member of a language in NP, whilst revealing no information about this witness. As of today, the most efficient scheme due to Groth [22] is a pre-processing zk-SNARK that requires pairings of elliptic curves.

Besides efficiency, SNARKs' succinctness makes them good candidates for recursive proof composition. That is, proofs that could themselves verify the correctness of other proofs, allowing a single proof to inductively attest to the correctness of many previous proofs as suggested by Valiant in [34]. Unfortunately, making recursive composition of Groth's proofs practical requires expensive elliptic curve constructions.

1.1 Previous work

Ben-Sasson et al. [5] presented the first practical setting of recursive proof composition. This setting uses two MNT pairing-friendly elliptic curves [18, Sec. 5]. These curves are constructed in a way such that proofs generated using one of the curves can feasibly reason about proofs generated using the other curve, but both are quite expensive at 128-bit security level. The two curves have low embedding degrees making it necessary to build them over large fields to achieve the standard security as it is implemented by Coda blockchain [30]. Moreover, Chiesa et al. [14] established some limitations on finding other suitable curves.

On the other hand, Bowe et al. proposed the Zexe system [8] which uses a suitable pair of elliptic curves for one layer proof composition. The authors constructed a BLS12 curve that is suitable for both levels of recursion and on top of it another curve via the Cocks–Pinch method [18, Sec. 4.1]. It is to note that while the inner curve is efficient at 128-bit security level, the outer curve is quite expensive.

1.2 Our contributions

We present a new secure and optimized pairing-friendly elliptic curve that is suitable for one layer proof composition. Our curve can substitute Zexe's outer curve while enjoying more properties for very efficient implementation. The curve is defined over a 761-bit prime field instead of 782 bits, we save one machine-word of 64 bits. The curve has CM discriminant $-D = -3$, allowing fast GLV scalar multiplication on $\mathbb{G}_1$ and $\mathbb{G}_2$. The curve has embedding degree 6 and a twist of degree 6, and $\mathbb{G}_2$ has coordinates in the same prime field as $\mathbb{G}_1$ (factor 6 compression). The curve also has fast subgroup check and fast cofactor multiplication. Finally we obtain a very efficient optimal ate pairing on this curve. In particular, it is at least five times faster to verify a Groth proof, compared to Zexe.

1.3 Applications

We mention briefly some applications from the blockchain community projects that can benefit from this work:

Zexe The authors introduced the notion of Decentralized Private Computation (DPC) that uses one layer proof composition [8]. As an application, they described in [8, § V] user-defined assets, decentralized exchanges and policy-enforcing stablecoins.

Celo The project aims at developing a mobile-first oriented blockchain platform. Celo is verifying BLS signatures by generating a single SNARK proof that verifies a bunch of signatures [11].

EY Blockchain The firm released into the public domain its Nightfall tool [16], a smart-contract based solution leveraging zkSNARKs for private transactions of fungible and non-fungible tokens on the Ethereum blockchain. Recently, EY unveiled its latest Nightfall upgrade allowing for transaction batching. This work can be used to aggregate many Nightfall proofs into a single one and thus reducing the overall gas cost.

Filecoin The protocol [31] is a decentralized storage blockchain. Protocol Labs introduced Proof-of-Replication that can be used to prove that some data has been replicated to its own uniquely dedicated physical storage. This proof is then compressed using a SNARK proof but this results in a massive arithmetic circuit. Filecoin is considering to split the circuit into 20 smaller ones and generate small proofs that can be aggregated into a single one using one layer proof composition.

Organization of the paper. The rest of the paper is organized as follows. First, in section 2, we provide preliminaries on pairing-friendly elliptic curves and recursive proof composition. Then, in section 3, we introduce our curve, discuss the optimizations it allows and compare it to Zexe's outer curve. Finally, before concluding, we estimate in section 4 the security of Zexe's inner curve and our curve, taking into account the Special Tower NFS algorithm.

2 Preliminaries

2.1 Pairing-friendly elliptic curves

Background on pairings. We briefly recall here elementary definitions on pairings and present the computation of two pairings used in practice, the Tate and ate pairings. All elliptic curves are *ordinary* (i.e. non-supersingular).

Let E be an elliptic curve defined over a field $\mathbb{F}_q$, where q is a prime power. Let π_q be the Frobenius endomorphism $(x, y) \mapsto (x^q, y^q)$. Its minimal polynomial is $X^2 - tX + q$ where t is called the *trace*. Let r be a prime divisor of the curve order $\#E(\mathbb{F}_q) = q + 1 - t$. The r-torsion subgroup of E is denoted $E[r] \coloneqq \{P \in E(\overline{\mathbb{F}_q}), [r]P = \mathcal{O}\}$ and has two subgroups of order r (eigenspaces of ϕ_q in $E[r]$) that are useful for pairing applications. Following [35], we define the two groups $\mathbb{G}_1 = E[r] \cap \ker(\pi_q - [1])$ with a generator denoted by G_1, and $\mathbb{G}_2 = E[r] \cap \ker(\pi_q - [q])$ with a generator G_2. The group $\mathbb{G}_2$ is defined over $\mathbb{F}_{q^k}$, where the embedding degree k is the smallest integer $k \in \mathbb{N}^*$ such that $r \mid q^k - 1$.

We recall the Tate and ate pairing definitions, based on the same two steps: evaluating a function $f_{s,Q}$ at a point P (Miller loop), and then raising it to the power $(q^k - 1)/r$ (final exponentiation). The function $f_{s,Q}$ has divisor $div(f_{s,Q}) = s(Q) - ([s]Q) - (s - 1)(\mathcal{O})$ and satisfies for integers i and j

$$f_{i+j,Q} = f_{i,Q} f_{j,Q} \frac{\ell_{[i]Q,[j]Q}}{v_{[i+j]Q}}$$

where $\ell_{[i]Q,[j]Q}$ and $v_{[i+j]Q}$ are the two lines needed to compute $[i + j]Q$ from $[i]Q$ and $[j]Q$ (ℓ through the two points, v the vertical). We compute $f_{s,Q}(P)$ with the Miller loop presented in Algorithm 1.

The Tate and ate pairings are defined by

$$\text{Tate}(P, Q) \coloneqq f_{r,P}(Q)^{(q^k-1)/r}$$
$$\text{ate}(P, Q) \coloneqq f_{t-1,Q}(P)^{(q^k-1)/r}$$

Algorithm 1: MillerLoop(s, P, Q)

Output: $m = f_{s,Q}(P)$

$m \leftarrow 1$; $S \leftarrow Q$;
for *b from the second most significant bit of s to the least* **do**
    $\ell \leftarrow \ell_{S,S}(P)$; $S \leftarrow [2]S$; DoubleLine
    $v \leftarrow v_{[2]S}(P)$; VerticalLine
    $m \leftarrow m^2 \cdot \ell / v$; Update1
    if $b = 1$ **then**
        $\ell \leftarrow \ell_{S,Q}(P)$; $S \leftarrow S + Q$; AddLine
        $v \leftarrow v_{S+Q}(P)$; VerticalLine
        $m \leftarrow m \cdot \ell / v$; Update2
return m;

where $P \in \mathbb{G}_1 \subset E[r](\mathbb{F}_q)$ and $Q \in \mathbb{G}_2 \subset E[r](\mathbb{F}_{q^k})$. The values Tate$(P, Q)$ and ate(P, Q) are in the "target" group $\mathbb{G}_T$ of r-th roots of unity in $\mathbb{F}_{q^k}$. In the sequel, when abstraction is needed, we denote a pairing as follows $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

It is also important to recall some results with respect to the complex multiplication (CM) discriminant $-D$. When $D = 3$ (resp. $D = 4$), the curve has CM by $\mathbb{Q}(\sqrt{-3})$ (resp. $\mathbb{Q}(\sqrt{-1})$) so that twists of degrees 3 and 6 exist (resp. 4). When E has d-th order twists for some $d \mid k$, then $\mathbb{G}_2$ is isomorphic to $E'[r](\mathbb{F}_{q^{k/d}})$ for some twist E'. Otherwise, in the general case, E admits a single twist (up to isomorphism) and it is of degree 2.

Some pairing-friendly constructions. We recall here some methods from the literature for constructing pairing-friendly ordinary elliptic curves that will be of interest in the sequel. We focus on Cocks–Pinch [18, Sec. 4.1], Barreto–Lynn–Scott [18, Sec. 6.1] and Brezing–Weng [18, Sec. 6.1] methods.

Cocks–Pinch is the most flexible method and can be used to construct curves with arbitrary embedding degrees but with ratio $\rho = \log_2 q / \log_2 r \approx 2$. It works by fixing the subgroup order r and the CM discriminant D and then computing the trace t and the prime q s.t. the CM equation $4q = t^2 + Dy^2$ (for some $y \in \mathbb{Z}$) is satisfied (cf. Alg. 2).

Brezing and Weng[18, Sec. 6.1], and independently, Barreto, Lynn and Scott [18, Sec. 6.1] generalized the Cocks–Pinch method by parametrizing t, r, q as polynomials. This led to curves with $\rho < 2$. We sketch below the idea of the algorithm in its generality for both BLS and BW constructions (cf. Alg. 3). A particular choice of polynomials for $k = 12$ yields a family of curves with a good security/performance tradeoff, denoted BLS12 [3]. The parameters are given in Table 1.

Pairing-friendly chains and cycles. A *chain* of elliptic curves is a list of curves defined over finite fields in which the number of points on one curve equals the characteristic of the field of definition of the next curve. If this property is cyclic, then it is called a *cycle*.

Algorithm 2: Cocks–Pinch method

Input: A positive integer k and a positive square-free integer D

Output: $E/\mathbb{F}_q$ with an order-r subgroup and embedding degree k

1 Fix k and D and choose a prime r s.t. k divides $r-1$ and $-D$ is a square modulo r;
2 Compute $t = 1 + x^{(r-1)/k}$ for x a generator of $(\mathbb{Z}/r\mathbb{Z})^\times$;
3 Compute $y = (t-2)/\sqrt{-D} \bmod r$;
4 Lift t and y in $\mathbb{Z}$;
5 Compute $q = (t^2 + Dy^2)/4$ in $\mathbb{Q}$;
6 **if** *q is a prime integer* **then**
7 | Use CM method ($D < 10^{12}$) to construct $E/\mathbb{F}_q$ with order-r subgroup;
8 **else**
9 | Go back to 1;
10 **return** *$E/\mathbb{F}_q$ with an order-r subgroup and embedding degree k*

Algorithm 3: Idea of BLS and BW methods

Input: A positive integer k and a positive square-free integer D

Output: $E/\mathbb{F}_{q(x)}$ with an order-$r(x)$ subgroup and embedding degree k

1 Fix k and D and choose an irreducible polynomial $r(x) \in \mathbb{Z}[x]$ with positive leading coefficient[1] s.t. $\sqrt{-D}$ and the primitive k-th root of unity ζ_k are in $K = \mathbb{Q}[x]/(r(x))$;
2 Choose $t(x) \in \mathbb{Q}[x]$ be a polynomial representing $\zeta_k + 1$ in K;
3 Set $y(x) \in \mathbb{Q}[x]$ be a polynomial mapping to $(\zeta_k - 1)/\sqrt{-D}$ in K;
4 Compute $q(x) = (t^2(x) + Dy^2(x))/4$ in $\mathbb{Q}[x]$;
5 **return** *$E/\mathbb{F}_{q(x)}$ with an order-$r(x)$ subgroup and embedding degree k*

[1]conditions to satisfy Bunyakovsky conjecture which states that such a polynomial produces infinitely many primes for infinitely many integers.

Definition 1. *An m-chain of elliptic curves is a list of distinct curves $E_1/\mathbb{F}_{q_1}, \ldots, E_m/\mathbb{F}_{q_m}$ where $q_1, \ldots, q_m$ are large primes and*

$$\#E_1(\mathbb{F}_{q_1}) = q_2, \ldots, \#E_i(\mathbb{F}_{q_i}) = q_{i+1}, \ldots, \#E_{m-1}(\mathbb{F}_{q_{m-1}}) = q_m \tag{1}$$

Definition 2. *An m-cycle of elliptic curves is a list of distinct curves $E_1/\mathbb{F}_{q_1}, \ldots, E_m/\mathbb{F}_{q_m}$ where $q_1, \ldots, q_m$ are large primes and*

$$\#E_1(\mathbb{F}_{q_1}) = q_2, \ldots, \#E_i(\mathbb{F}_{q_i}) = q_{i+1}, \ldots, \#E_{m-1}(\mathbb{F}_{q_{m-1}}) = q_m, \#E_m(\mathbb{F}_{q_m}) = q_1 \tag{2}$$

In the literature, a 2-cycle of ordinary curves is called an *amicable pair*. Following the same logic, we call *pairing-friendly amicable chain* a 2-chain of pairing-

BLS12, $k = 12$, $D = 3$, $x = 1 \bmod 3$
$q_{\mathrm{BLS12}}(x) = (x^6 - 2x^5 + 2x^3 + x + 1)/3$, $x = 1 \bmod 3$
$r_{\mathrm{BLS12}}(x) = x^4 - x^2 + 1$
$t_{\mathrm{BLS12}}(x) = x + 1$

Table 1. Polynomial parameters of BLS12 curve family.

friendly ordinary elliptic curves. In this paper, we are interested in constructing a pairing-friendly amicable chain of curves with efficient arithmetic.

2.2 Recursive proof composition

To date the most efficient zkSNARK is due to Groth [22]. Here, we briefly sketch the construction and refer the reader to the original paper. The construction consists of a trapdoored setup, a proof of 3 group elements and the verification is one equation of pairings product (Eq. 3). Given an instance $\Phi = (a_0, \ldots, a_l) \in \mathbb{F}_q^l$, a proof $\pi = (A, C, B) \in \mathbb{G}_1^2 \times \mathbb{G}_2$ and a verification key $vk = (vk_{\alpha,\beta}, \{vk_{\pi_i}\}_{i=0}^{l}, vk_\gamma, vk_\delta) \in \mathbb{G}_T \times \mathbb{G}_1^{l+1} \times \mathbb{G}_2^2$, the verifier must check that

$$e(A, B) = vk_{\alpha,\beta} \cdot e(vk_x, vk_\gamma) \cdot e(C, vk_\delta) \quad (3)$$

where $vk_x = \sum_{i=0}^{l} [a_i] vk_{\pi_i}$ depends only on the instance Φ and $vk_{\alpha,\beta} = e(vk_\alpha, vk_\beta)$ can be computed in the trusted setup for $(vk_\alpha, vk_\beta) \in \mathbb{G}_1 \times \mathbb{G}_2$.

It is also to note that for efficient implementation the subgroup order r is chosen to allow efficient FFT-based polynomial multiplications, as proposed in [4]. To achieve this, we require *high 2-adicity*: $r - 1$ should be divisible by a "large enough" power of 2.

To allow recursive proof composition, one needs to write the verification equation (3) as an instance in the prover language. In pairing-based SNARKs such as [22], the verification arithmetic is in an extension of $\mathbb{F}_q$ up to a degree k while the proving arithmetic is in $\mathbb{F}_r$. Since a pairing-friendly curve with $q = r$ doesn't exist[2], one needs to simulate $\mathbb{F}_q$ operations via $\mathbb{F}_r$ operations which results in a blowup of the order $\log q$ compared to native arithmetic. A practical alternative was suggested in [5] using a pairing-friendly amicable pair. The authors proposed two MNT curves [18, Sec.5] with embedding degrees 4 and 6 and primes q, r of 298 bits. While this solves the problem, the security level of the curves is "low". To remediate this, Coda protocol [30] uses a larger MNT-based amicable pair proposed by [5] that targets 128-bit security with primes q, r of 753 bits at the cost of expensive computations (cf. Fig. 1).

While an amicable pair allows an infinite recursion loop, an amicable chain allows a bounded recursion. In some applications such as those we mentioned a one layer composition is sufficient. To this end, Zexe's authors proposed an amicable chain consisting of an inner BLS12 curve called BLS12-377 and an outer Cocks–Pinch curve called SW6 (for Short Weierstrass form, and embedding degree 6). BLS12-377 was constructed in a way to have both $r - 1$ and $q - 1$ highly 2-adic while enjoying all the efficient implementation properties of the BLS12 family. Once the inner curve is constructed, the authors looked for a pairing-friendly curve with pre-determined subgroup order r equal to the field size q of BLS12-377. The only construction from the literature to allow such flexibility is Cocks–Pinch but it unfortunately results in a curve on which operations are at

[2] r needs to divide $q^k - 1$ for $k \in_{\mathbb{N}^*} [\![1, 20]\!]$, thus $r = 1$ is the only solution.

least two times more costly (in time and space) than BLS12-377. Furthermore, SW6 doesn't allow efficient pairing computation and efficient scalar multiplication via endomorphisms.

In the sequel, we refer to BLS12-377 as $E_{\mathrm{BLS12}}(\mathbb{F}_{q_{\mathrm{BLS12}}})$ with a subgroup of order r_{BLS12}, SW6 as $E_{\mathrm{SW6}}(\mathbb{F}_{q_{\mathrm{SW6}}})$ with a subgroup of order r_{SW6} and our curve as $\tilde{E}(\mathbb{F}_{\tilde{q}})$ with a subgroup of order $\tilde{r}$ (cf. Fig. 1).

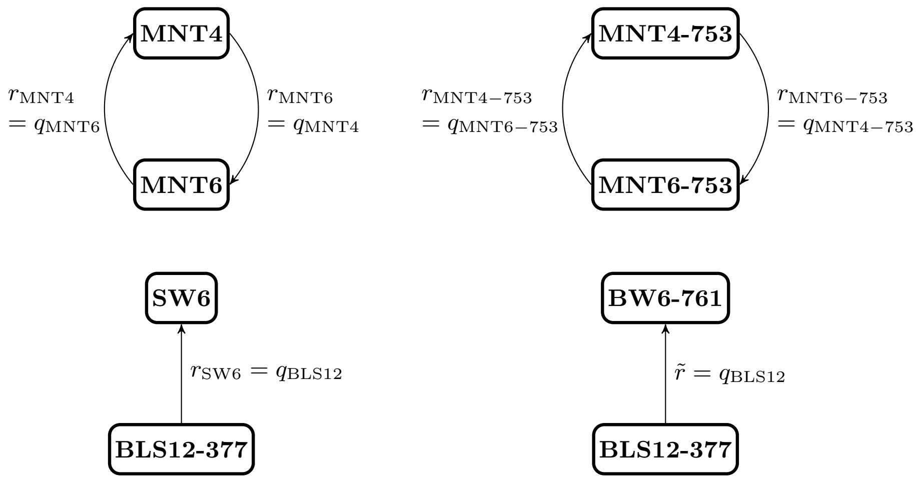

Fig. 1. Examples of pairing-friendly amicable cycles and chains

3 The proposed elliptic curve: BW6-761

The authors in [8] proposed two curves for the one-layer proof-composition: BLS12-377 and SW6 whose parameters are given in Table 2. Note that because the Cocks–Pinch method has a ratio $\rho \approx 2$, the SW6 curve characteristic q_{SW6} is 782-bit long (832 bits in Montgomery domain). Since q_{SW6} is already very large, an embedding degree $k = 6$ is sufficient for the security of $\mathbb{F}_{q_{\mathrm{SW6}}}^{k}$. Moreover, because $D = 339$, E_{SW6} has only a quadratic twist E' and thus $\mathbb{G}_2 \subset E(\mathbb{F}_{q_{\mathrm{SW6}}^6})[r_{\mathrm{SW6}}]$ is isomorphic to $E'(\mathbb{F}_{q_{\mathrm{SW6}}^3})[r_{\mathrm{SW6}}]$, and $\mathbb{G}_2$ elements can be compressed to only $3 \times 832 = 2496$ bits.

Since we are stuck with $\rho \approx 2$, we searched for a Cocks–Pinch curve $\tilde{E}$ with $k = 6$ and smallest $\tilde{q}$ less or equal to 768 bits. We restricted our search to curves with CM discriminant $D = 3$ to allow optimal $\mathbb{G}_2$ compression (a sextic twist $\tilde{E}'/\mathbb{F}_{\tilde{q}}$ of $\tilde{E}$ s.t. $\mathbb{G}_2$ is isomorphic to $\tilde{E}'(\mathbb{F}_{\tilde{q}})[r]$ of 768 bits) and GLV fast scalar multiplication [20] on $\mathbb{G}_1$ and $\mathbb{G}_2$. Then, following the work [23], we computed the polynomial form of $\tilde{q}$ which allowed us to compute the coefficients of an

name	curve type	k	D	r	q	compressed $\mathbb{G}_1$ in bits	compressed $\mathbb{G}_2$ in bits
E_{BLS12}	BLS12	12	3	r_{BLS12}	q_{BLS12}	384	768
E_{SW6}	short Weierstrass	6	339	$r_{\text{SW6}} = q_{\text{BLS12}}$	q_{SW6}	832	2496

prime	value	size in bits	2-adicity
r_{BLS12}	`0x12ab655e9a2ca55660b44d1e5c37b00159aa76fed00000010a11800000000001`	253	47
$q_{\text{BLS12}} = r_{\text{SW6}}$	`0x1ae3a4617c510eac63b05c06ca1493b1a22d9f300f5138f1ef3622fba094800170b5d44300000008508c00000000001`	377	46
q_{SW6}	`0x3848c4d2263babf8941fe959283d8f526663bc5d176b746af0266a7223ee72023d07830c728d80f9d78bab3596c8617c579252a3fb77c79c13201ad533049cfe6a399c2f764a12c4024bee135c065f4d26b7545d85c16dfd424adace79b57b942ae9`	782	3

Table 2. Parameters of BLS12-377 and SW6 curves

optimal lattice-based final exponentiation as in [19] and also faster subgroup checks. Finally, the constructed curve has a 2-torsion point allowing fast and secure Elligator 2 hashing-to-point [6]. We investigate also Wahby and Boneh work [36] as an alternative hashing method.

The short Weierstrass forms of the curve $\tilde{E}$ and its sextic twist $\tilde{E}'$ are

$$\tilde{E}/\mathbb{F}_{\tilde{q}} \colon y^2 = x^3 - 1 \quad (4)$$

$$\tilde{E}'/\mathbb{F}_{\tilde{q}} \colon y^2 = x^3 + 4 \quad (5)$$

and the parameters are given in Table 3.

name	curve type	k	D	r	q	compressed $\mathbb{G}_1$ in bits	compressed $\mathbb{G}_2$ in bits
$\tilde{E}$	short Weierstrass	6	3	$\tilde{r} = q_{\text{BLS12}}$	$\tilde{q}$	768	768

prime	value	size in bits	2-adicity
$\tilde{r} = q_{\text{BLS12}}$	`0x1ae3a4617c510eac63b05c06ca1493b1a22d9f300f5138f1ef3622fba094800170b5d44300000008508c00000000001`	377	46
$\tilde{q}$	`0x122e824fb83ce0ad187c94004faff3eb926186a81d14688528275ef8087be41707ba638e584e91903cebaff25b423048689c8ed12f9fd9071dcd3dc73ebff2e98a116c25667a8f8160cf8aeeaf0a437e6913e6870000082f49d00000000008b`	761	1

Table 3. Parameters of our curve

Given that q_{BLS12} is parameterized by a polynomial $q(u)$ with $u =$ `0x8508c00000000`, we can apply the Brezing–Weng method (cf. Alg. 3) with $k = 6$, $D = 3$ and

$\tilde{r}(x) = (x^6 - 2x^5 + 2x^3 + x + 1)/3$. There are two primitive 6-th roots of unity in $\mathbb{Q}[x]/\tilde{r}(x)$, and two sets of solutions

$$\begin{cases} \tilde{t}_0(x) = x^5 - 3x^4 + 3x^3 - x + 3 \\ \tilde{y}_0(x) = (x^5 - 3x^4 + 3x^3 - x + 3)/3 \end{cases} \text{or} \begin{cases} \tilde{t}_1(x) = -x^5 + 3x^4 - 3x^3 + x \\ \tilde{y}_1(x) = (x^5 - 3x^4 + 3x^3 - x)/3 \end{cases}$$

Unfortunately neither $(\tilde{t}_0(x) + 3\tilde{y}_0(x))/4$ nor $(\tilde{t}_1(x) + 3\tilde{y}_1(x))/4$ are irreducible polynomials, and we cannot construct a polynomial family of amicable 2-chain elliptic curves. But following [23], with well-chosen lifting cofactors h_t and h_y, we can obtain valid parameters $\tilde{t} = \tilde{r} \times h_t + \tilde{t}_i(u)$ and respectively $\tilde{y} = \tilde{r} \times h_y + \tilde{y}_i(u)$ for $i \in \{0, 1\}$. We found $i = 0$, $h_t = 13$, $h_y = 9$. We summarize the polynomial form of the parameters in Table 4. We propose to name our curve BW6-761 as it is a Brezing–Weng curve of embedding degree 6 over a 761-bit prime field.

Our curve, $k = 6$, $D = 3$, $\tilde{r} = q_{\text{BLS12}}$
$\tilde{r}(x) = (x^6 - 2x^5 + 2x^3 + x + 1)/3$
$\tilde{t}(x) = x^5 - 3x^4 + 3x^3 - x + 3 + h_t\tilde{r}(x)$
$\tilde{y}(x) = (x^5 - 3x^4 + 3x^3 - x + 3)/3 + h_y\tilde{r}(x)$
$\tilde{q}(x) = (\tilde{t}^2 + 3\tilde{y}^2)/4$
$\tilde{q}_{h_t=13,h_y=9}(x) = (103x^{12} - 379x^{11} + 250x^{10} + 691x^9 - 911x^8$
$-79x^7 + 623x^6 - 640x^5 + 274x^4 + 763x^3 + 73x^2 + 254x + 229)/9$

Table 4. Polynomial parameters of our curve.

3.1 Optimizations in $\mathbb{G}_1$

GLV scalar multiplication. We have $\tilde{q} \equiv 1 \pmod 3$ and $\tilde{E}(\mathbb{F}_{\tilde{q}})$ has j-invariant 0. Let ω be an element of order 3 in $\mathbb{F}_{\tilde{q}}$. Then the endomorphism $\phi : \tilde{E} \to \tilde{E}$ defined by $(x, y) \mapsto (\omega x, y)$ (and $\mathcal{O} \mapsto \mathcal{O}$) acts on a point $P \in \tilde{E}(\mathbb{F}_{\tilde{q}})[\tilde{r}]$ as $\phi(P) = [\lambda]P$ where λ is an integer satisfying $\lambda^2 + \lambda + 1 \equiv 0 \mod \tilde{r}$. Since we expressed $\tilde{q}$ and $\tilde{r}$ as polynomials, we find $\omega(x)$ and $\lambda(x)$ such that

$$\begin{aligned} \omega(x)^2 + \omega(x) + 1 &\equiv 0 \pmod{\tilde{q}(x)} \text{ (cube root of unity)} \\ \lambda(x)^2 + \lambda(x) + 1 &\equiv 0 \pmod{\tilde{r}(x)} \end{aligned}$$

where $\lambda_1(x) = x^5 - 3x^4 + 3x^3 - x + 1$, $\lambda_2(x) = -\lambda_1 - 1$, $\omega_1(x) = (103x^{11} - 482x^{10} + 732x^9 + 62x^8 - 1249x^7 + 1041x^6 + 214x^5 - 761x^4 + 576x^3 + 11x^2 - 265x + 66)/21$, $\omega_2(x) = -\omega_1(x) - 1$. Evaluating the polynomials at $u =$ `0x8508C00000000001`, we find that for $P(x, y) \in \tilde{E}(\mathbb{F}_{\tilde{q}})$ of order $\tilde{r}$,

$$\begin{aligned} [\lambda_1]P &= (\omega_1 x, y) \\ [\lambda_2]P &= (\omega_2 x, y) \end{aligned} \quad (6)$$

where

$\lambda_1 =$ `0x9b3af05dd14f6ec619aaf7d34594aabc5ed1347970dec00452217cc9000`
`00008508c00000000001`

$\lambda_2 =$ `-0x9b3af05dd14f6ec619aaf7d34594aabc5ed1347970dec00452217cc900`
`000008508c00000000002`

$\omega_1 =$ `0x531dc16c6ecd27aa846c61024e4cca6c1f31e53bd9603c2d17be416c5e44`
`26ee4a737f73b6f952ab5e57926fa701848e0a235a0a398300c65759fc4518315`
`1f2f082d4dcb5e37cb6290012d96f8819c547ba8a4000002f962140000000002a`

$\omega_2 =$ `0xcfca638f1500e327035cdf02acb2744d06e68545f7e64c256ab7ae14297a`
`1a823132b971cdefc65870636cb60d217ff87fa59308c07a8fab8579e02ed3cdd`
`ca5b093ed79b1c57b5fe3f89c11811c1e214983de300000535e7bc00000000060`

Hashing-to-point. Elligator 2 [6] is an injective map to any elliptic curve of the form $y^2 = x^3 + Ax^2 + Bx$ with $AB(A^2 - 4B) \neq 0$ over any finite field of odd characteristic. Since the point $(1, 0) \in \tilde{E}(\mathbb{F}_{\tilde{q}})$ is of order 2, we can map $\tilde{E}$ to a curve of Montgomery form, precisely $\tilde{M}(\mathbb{F}_{\tilde{q}}) : y^2 = x^3 + 3x^2 + 3x$ (cf. Alg. 4). We denote `hash2base(.)` in Algorithm 4 a cryptographic hash function to $\mathbb{F}_{\tilde{q}}$,

Algorithm 4: Elligator 2

Input: Θ an octet string to be hashed, $A = 3, B = 3$ coefficients of the curve $\tilde{M}$ and N a constant non-square in $\mathbb{F}_{\tilde{q}}$

Output: A point (x, y) in $\tilde{M}(\mathbb{F}_{\tilde{q}})$

Define $g(x) = x(x^2 + Ax + B)$;
$u =$ `hash2base`(Θ);
$v = -A/(1 + Nu^2)$;
$e =$ `Legendre`$(g(v), \tilde{q})$;
if $u \neq 0$ **then**
    $x = ev - (1 - e)A/2$;
    $y = -e\sqrt{g(x)}$;
else
    $x = 0$ and $y = 0$;
return (x, y);

and `Legendre`$(a, \tilde{q})$ is the Legendre symbol of an integer a modulo $\tilde{q}$ which takes values $1, -1, 0$ for when the input is a quadratic residue, non-quadratic-residue, or zero respectively. Elligator 2 is parametrized by a non-square N. Finding a non-square is an easy computation in general since about half of the elements of $\mathbb{F}_{\tilde{q}}$ are non-squares. For efficiency it is desirable to choose N to be small, or otherwise in a way to speed up multiplications by N. In our case, we can choose $N = -1$ because $\tilde{q} \equiv 3 \pmod 4$.

Wahby and Boneh introduced in [36] an "indirect" map for the BLS12-381 curve [7] based on the simplified SWU map [10], which works by mapping to an isogenous curve with nonzero j-invariant, then evaluating the isogeny map. We hence check if $\tilde{E}$ has a low-degree rational isogeny. Since 2×11 divides $\tilde{y}$ (CM equation $4\tilde{q} = \tilde{t}^2 + D\tilde{y}^2$), there should be isogenies of degree 2 and 11. Noting that the point $(1, 0)$ is a 2-torsion point, we obtain the rational 2-isogeny $(x, y) \mapsto ((x^2 - x + 3)/(x - 1), y(x^2 - 2x - 2)/(x - 1)^2)$ to the curve $y^2 = x^3 - 15x - 22$ of j-invariant 54000. A 2-torsion point on this curve is $(-2, 0)$ and the dual isogeny to work with Wahby–Boneh hash function is $(x', y') \mapsto ((x'^2 + 2x' - 3)/(x' + 2), y(x'^2 + 4x' + 7)/(x'^2 + 2)^2)$.

Clearing cofactor. Another important step is clearing the cofactor. Our curve has a large 384-bit long cofactor due to the Cocks–Pinch method. The cofactor has a polynomial form in the seed u like the other parameters, $\tilde{c}(x) = (103x^6 - 173x^5 - 96x^4 + 293x^3 + 21x^2 + 52x + 172)/3$ and $\tilde{c}(x)\tilde{r}(x) = \tilde{q}(x) + 1 - \tilde{t}(x)$. Because the curve has j-invariant 0, it has a fast endomorphism $\phi : (x, y) \mapsto (\omega_1 x, y)$ and such that $\phi^2 + \phi + 1$ is the identity map. We apply the same technique as in [19]. First we compute the eigenvalue $\lambda(x)$ of the endomorphism modulo the cofactor. We obtain $\lambda(x) = (-385941x^5 + 1285183x^4 - 1641034x^3 - 121163x^2 + 1392389x - 1692082)/1250420 \bmod \tilde{c}(x)$. Then we reduce with LLL the lattice spanned by the rows of the matrix $M = \begin{bmatrix} \tilde{c}(x) & 0 \\ \lambda(x) \bmod \tilde{c}(x) & 1 \end{bmatrix}$. We obtain a reduced matrix

$$\begin{bmatrix} m_{00} & m_{01} \\ m_{10} & m_{11} \end{bmatrix} = \frac{1}{103} \begin{bmatrix} 103x^3 - 83x^2 - 40x + 136 & 7x^2 + 89x + 130 \\ -7x^2 - 89x - 130 & 103x^3 - 90x^2 - 129x + 6 \end{bmatrix}$$

We check that $m_{i0}(x) + \lambda(x)m_{i1}(x) = 0 \bmod \tilde{c}(x)$ and $\gcd(m_{i0}(x) + \lambda(x)m_{i1}(x), \tilde{r}(x)) = 1$. Now $x = u$ and for clearing the cofactor, we first precompute uP, u^2P, u^3P which costs three scalar multiplications by $u =$ `0x8508c00000000`, then we compute $R = 103(u^3P) - 83(u^2P) - 40(uP) + 136P + \phi(7(u^2P) + 89(uP) + 130P)$, and R has order $\tilde{r}$. This formula is compatible with ω_1 to compute ϕ.

Subgroup check. Subgroup check can benefit of the same technique. We need to check if a point $P \in \tilde{E}(\mathbb{F}_{\tilde{q}})$ has order $\tilde{r}$, that is, $\tilde{r}P = \mathcal{O}$. Instead of multiplying by $\tilde{r}$ of 377 bits, we can use the endomorphism ϕ as above. This time we need $\lambda_1(x) = x^5 - 3x^4 + 3x^3 - x + 1$ modulo $\tilde{r}(x)$. We reduce the matrix $M = \begin{bmatrix} \tilde{r}(x) & 0 \\ -\lambda_1(x) \bmod \tilde{r}(x) & 1 \end{bmatrix}$ and obtain $\begin{bmatrix} x + 1 & x^3 - x^2 + 1 \\ x^3 - x^2 - x & -x - 1 \end{bmatrix}$ and check that $(x + 1) + \lambda_1(x)(x^3 - x^2 + 1) = 0 \bmod \tilde{r}(x)$. For a faster subgroup check of a point P, one can precompute uP, u^2P, u^3P with three scalar multiplications by u, and check if $uP + P + \phi(u^3P - u^2P + P)$ is the point at infinity.

3.2 Optimizations in $\mathbb{G}_2$

Compression of $\mathbb{G}_2$ elements. The curve $\tilde{E}/\mathbb{F}_{\tilde{q}}$ has CM discriminant $D = 3$ so it has a twist of degree $d = 6$ and because the embedding degree is $k = 6$ the $\tilde{r}$-torsion of $\mathbb{G}_2 \subset \tilde{E}[\tilde{r}](\mathbb{F}_{\tilde{q}^6})$ is isomorphic to $\tilde{E}'[\tilde{r}](\mathbb{F}_{\tilde{q}^{k/d}}) = \tilde{E}'[\tilde{r}](\mathbb{F}_{\tilde{q}})$. Thus, elements in $\mathbb{G}_2$ can be compressed from 4608 bits to 768 bits. We choose the irreducible polynomial $x^6 + 4$ in $\mathbb{F}_{\tilde{q}}[x]$ and construct the M-twist curve $\tilde{E}'/\mathbb{F}_{\tilde{q}} \colon y^2 = x^3 + 4$. To map a point $Q(x, y) \in \tilde{E}(\mathbb{F}_{\tilde{q}^6})$ to a point on the M-twist curve we use $\xi \colon (x, y) \mapsto (x/\nu^2, y/\nu^3)$ where $\nu^6 = -4$.

GLV scalar multiplication. Since the group $\mathbb{G}_2$ is isomorphic to the $\tilde{r}$-torsion in $\tilde{E}'(\mathbb{F}_{\tilde{q}})$ (Eq. (5)), we can apply the same GLV decomposition as in equation (6).

Hashing-to-point. The curve $\tilde{E}'(\mathbb{F}_{\tilde{q}}) : y^2 = x^3 + 4$ doesn't have a point of order 2 so Elligator 2 doesn't apply. Furthermore, we didn't find a low-degree rational isogeny and thus Wahby–Boneh method is not efficiently applicable. However, we can apply the more generic Shallue–Woestijn algorithm [32] that works for any elliptic curve over a finite field $\mathbb{F}_q$ of odd characteristic with $\#\mathbb{F}_q > 5$. It is based on a generalization of Skalba identity [33] and runs deterministically in time $O(\log^4 q)$ or in time $O(\log^3 q)$ if $q \equiv 3 \mod 4$ (a square root costs an exponentiation). Fouque and Tibouchi [17] adapted this algorithm to BN curves ($j = 0$) assuming that $q \equiv 7 \mod 12$ and that the point $(1, y) \neq (1, 0)$ lies on the curve. Noting that $\tilde{E}' : y^2 = f(x) = x^3 + 4$ satisfies these assumptions, we propose to use Fouque–Tibouchi map for our curve:

$$\begin{aligned} g : \mathbb{F}_{\tilde{q}}^* &\to \tilde{E}'(\mathbb{F}_{\tilde{q}}) \\ z &\mapsto \Big(x_i(z), \texttt{Legendre}(z, \tilde{q}) \times \sqrt{f(x_i(z))}\Big)_{i \in \{1,2,3\}} \end{aligned}$$

where

$$\begin{aligned} x_1(z) &= \frac{-1 + \sqrt{-3}}{2} - \frac{\sqrt{-3} \cdot z^2}{5 + z^2} \\ x_2(z) &= \frac{-1 - \sqrt{-3}}{2} + \frac{\sqrt{-3} \cdot z^2}{5 + z^2} \\ x_3(z) &= 1 - \frac{(5 + z^2)^2}{3z^2} \end{aligned}$$

Because of Skalba identity $y^2 = f(x_1) \cdot f(x_2) \cdot f(x_3)$, at least one of the x_i is an abscissa of a point on $\tilde{E}'$ — we choose the smallest index such that $f(x_i)$ is a square in $\mathbb{F}_{\tilde{q}}$.

Clearing cofactor. We apply the same technique as for $\mathbb{G}_1$. The co-factor is $\tilde{c}'(x) = (103x^6 - 173x^5 - 96x^4 + 293x^3 + 21x^2 + 52x + 151)/3$. The choice of $\lambda'(x) = (-1413572x^5 + 3625805x^4 - 1877140x^3 - 2124857x^2 + 1372633x - 2268778)/296619$ is compatible with ω_1. We obtain after LLL reduction the matrix

$$\begin{bmatrix} -7x^2 + 117x + 109 & 103x^3 - 90x^2 - 26x + 136 \\ 103x^3 - 83x^2 - 143x + 27 & 7x^2 - 117x - 109 \end{bmatrix}$$

Given a point $P' \in \tilde{E}'(\mathbb{F}_{\tilde{q}})$, we precompute uP', u^2P', u^3P' with three scalar multiplications, then the point R' has order $\tilde{r}$, with

$$R' = (103u^3P' - 83u^2P' - 143uP' + 27P') + \phi(7u^2P' - 117uP' - 109P') .$$

Subgroup check. We obtain similar formulas as for $\mathbb{G}_1$. Note that since the twist has order 6, the untwist-Frobenius-twist endomorphism corresponds to ψ where $\psi^2 - \psi + 1 = 0$ and whose eigenvalue is a 6-th root of unity. This is almost the same as considering ϕ, where $\phi^2 + \phi + 1 = 0$ (we have $\phi = -\psi$). The formula compatible with ω_1 for ϕ is

$$-uP' - P' + \phi(u^3P' - u^2P' - uP') = 0 \iff \tilde{r}P' = \mathcal{O} .$$

3.3 Pairing computation

The verification equation of proof composition (3) requires three pairing computations. When an even-degree twist is available, the denominators $v_{[2]S}(P), v_{S+Q}(P)$ in Algorithm 1 are in a proper subgroup of $\mathbb{F}_{q^k}$ and can be removed as they become one after the final exponentiation. This is the case for BLS12, SW6 and our curve. We estimate in terms of multiplications in the base fields $\mathbb{F}_{q_{\mathrm{BLS12}}}, \mathbb{F}_{q_{\mathrm{SW6}}}, \mathbb{F}_{\tilde{q}}$ a pairing on the curves BLS12-377 and SW6 (see Appendix A), and an optimal ate pairing on our curve. We follow the estimate in [23]. We model the cost of arithmetic in a degree 6, resp. degree 12 extension in the usual way, where multiplications and squarings in quadratic and cubic extensions are obtained recursively with Karatsuba and Chung–Hasan formulas, summarized in Table 5. We denote by $\mathbf{m}_k$, $\mathbf{s}_k$, $\mathbf{i}_k$ and $\mathbf{f}_k$ the costs of multiplication, squaring, inversion, and q-th power Frobenius in an extension $\mathbb{F}_{q^k}$, and by $\mathbf{m} = \mathbf{m}_1$ the multiplication in a base field $\mathbb{F}_q$. We neglect additions and multiplications by small constants.

k	1	2	3	6	12
$\mathbf{m}_k$	$\mathbf{m}$	$3\mathbf{m}$	$6\mathbf{m}$	$18\mathbf{m}$	$54\mathbf{m}$
$\mathbf{s}_k$	$\mathbf{m}$	$2\mathbf{m}$	$5\mathbf{m}$	$12\mathbf{m}$	$36\mathbf{m}$
$\mathbf{f}_k$	0	0	$2\mathbf{m}$	$4\mathbf{m}$	$10\mathbf{m}$
$\mathbf{s}_k^{\mathrm{cyclo}}$				$6\mathbf{m}$	$18\mathbf{m}$
$\mathbf{i}_k - \mathbf{i}_1$	0	$4\mathbf{m}$	$12\mathbf{m}$	$34\mathbf{m}$	$94\mathbf{m}$
$\mathbf{i}_k$, with $\mathbf{i}_1 = 25\mathbf{m}$	$25\mathbf{m}$	$29\mathbf{m}$	$37\mathbf{m}$	$59\mathbf{m}$	$119\mathbf{m}$

Table 5. Cost from [23, Table 6] of $\mathbf{m}_k$, $\mathbf{s}_k$ and $\mathbf{i}_k$ for finite field extensions involved.

k	D	curve	DoubleLine and AddLine	Update1 and Update2	ref
$6 \mid k$	-3	$y^2 = x^3 + b$	$2\mathbf{m}_{k/6} + 7\mathbf{s}_{k/6} + (k/3)\mathbf{m}$	$\mathbf{s}_k + 13\mathbf{m}_{k/6}$	[15, §5]
		sextic twist	$10\mathbf{m}_{k/6} + 2\mathbf{s}_{k/6} + (k/3)\mathbf{m}$	$13\mathbf{m}_{k/6}$	
$6 \mid k$	-3	$y^2 = x^3 + b$	$3\mathbf{m}_{k/6} + 6\mathbf{s}_{k/6} + (k/3)\mathbf{m}$	$\mathbf{s}_k + 13\mathbf{m}_{k/6}$	[1, §4,6]
		sextic twist	$11\mathbf{m}_{k/6} + 2\mathbf{s}_{k/6} + (k/3)\mathbf{m}$	$13\mathbf{m}_{k/6}$	

Table 6. Miller loop cost in Weierstrass model from [15,1], homogeneous coordinates.

Miller loop for our curve. For our curve, the Tate pairing has Miller loop length $\tilde{r}(x) = (x^6 - 2x^5 + 2x^3 + x + 1)/3$, and the ate pairing has Miller loop length $\tilde{t}(x) - 1 = (13x^6 - 23x^5 - 9x^4 + 35x^3 + 10x + 19)/3$, hence the ate pairing will be slightly slower. Recall that thanks to a degree 6 twist, the two points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$ have coordinates in $\mathbb{F}_{\tilde{q}}$, hence swapping the two points for the ate pairing does not slow down the Miller loop in itself. We can apply Vercauteren's method [35] to obtain a minimal Miller loop. We define the lattice spanned by the rows of the matrix $M = \begin{bmatrix} \tilde{r}(x) & 0 \\ -\tilde{q}(x) \bmod \tilde{r}(x) & 1 \end{bmatrix}$ and reduce it with LLL. We obtain the short basis $\begin{bmatrix} x+1 & x^3 - x^2 - x \\ x^3 - x^2 + 1 & -x - 1 \end{bmatrix}$. We check that $(x+1) + (x^3 - x^2 - x)\tilde{q}(x) \equiv 0 \bmod \tilde{r}(x)$. The optimal ate pairing on our curve can be computed as

$$\mathrm{ate}_{\mathrm{opt}}(P,Q) = (f_{u+1,Q}(P) f^{\tilde{q}}_{u^3-u^2-u,Q}(P) \ell_{[u^3-u^2-u]\pi(Q),[u+1]Q}(P))^{(\tilde{q}^6-1)/\tilde{r}}$$

and since $(u+1) + \tilde{q}(u^3 - u^2 - u) = 0 \bmod \tilde{r}$, we have $[u+1]Q + \pi_{\tilde{q}}([u^3 - u^2 - u]Q) = \mathcal{O}$. The line $\ell_{[u^3-u^2-u]\pi(Q),[u+1]Q}$ is vertical and can be removed. Finally,

$$\mathrm{ate}_{\mathrm{opt}}(P,Q) = (f_{u+1,Q}(P) f^{\tilde{q}}_{u^3-u^2-u,Q}(P))^{(\tilde{q}^6-1)/\tilde{r}} \ . \qquad (7)$$

More precisely, $f_{u+1,Q}(P) = f_{u,Q}(P) \cdot \ell_{[u]Q,Q}(P)$. We can re-use $f_{u,Q}(P)$ to compute the second part $f_{u(u^2-u-1),Q}$ since $f_{uv,Q} = f^v_{u,Q} f_{v,[u]Q}$. We obtain Algorithm 5. We can write $u^2 - u - 1$ in 2-non-adjacent-form (2-NAF) to minimize the addition steps in the Miller loop, and replace Q by $-Q$ in the algorithm when the bit b_i is -1. The scalar $u^2 - u - 1$ is 127-bit long and has $\mathrm{HW}_{\text{2-NAF}} = 19$.

$$\begin{aligned}\mathrm{Cost}_{\textsc{MillerLoop}} = {} & (\mathrm{nbits}(u) - 1)\mathrm{Cost}_{\textsc{DoubleLine}} + (\mathrm{nbits}(u) - 2)\mathrm{Cost}_{\textsc{Update1}} \\ & + (\mathrm{HW}(u) - 1)(\mathrm{Cost}_{\textsc{AddLine}} + \mathrm{Cost}_{\textsc{Update2}}) \\ & + (\mathrm{nbits}(u^2 - u - 1) - 1)(\mathrm{Cost}_{\textsc{DoubleLine}} + \mathrm{Cost}_{\textsc{Update1}}) \\ & + (\mathrm{HW}_{\text{2-NAF}}(u^2 - u - 1) - 1)(\mathrm{Cost}_{\textsc{AddLine}} + \mathrm{Cost}_{\textsc{Update2}} + \mathbf{m}_6) \\ & + \mathbf{i} + 2\mathbf{m} + \mathbf{i}_6 + \mathbf{f}_6 + \mathbf{m}_6\end{aligned}$$

We compute $(64 - 1)(3\mathbf{m} + 6\mathbf{s} + 2\mathbf{m}) + (64 - 2)(\mathbf{s}_6 + 13\mathbf{m}) + (7 - 1)(11\mathbf{m} + 2\mathbf{s} + 2\mathbf{m} + 13\mathbf{m}) + (127 - 1)(3\mathbf{m} + 6\mathbf{s} + 2\mathbf{m} + 13\mathbf{m} + \mathbf{s}_6) + (19 - 1)(11\mathbf{m} + 2\mathbf{s} + 2\mathbf{m} + 13\mathbf{m} + \mathbf{m}_6) + \mathbf{i} + 2\mathbf{m} + \mathbf{i}_6 + 13\mathbf{m} + \mathbf{f}_6 + \mathbf{m}_6 = 7861\mathbf{m} + 2\mathbf{i} \approx 7911\mathbf{m}$ with $\mathbf{m} = \mathbf{s}$ and $\mathbf{i} \approx 25\mathbf{m}$. We note that since $\tilde{q}(x) \equiv \tilde{t}(x) - 1 \equiv \lambda(x) + 1 \bmod \tilde{r}(x)$, the formulas are similar as for subgroup check in $\mathbb{G}_1$ in Section 10, where we found that $(x + 1) + \lambda(x)(x^3 - x^2 + 1) = 0 \bmod \tilde{r}(x)$.

Final exponentiation for our curve. Given that $\tilde{q}$ has a polynomial form, we can compute the coefficients of a fast final exponentiation as in [19]. As for SW6 curve, the easy part is raising to $(\tilde{q}^3 - 1)(\tilde{q} + 1)$ and costs $99\mathbf{m}$ (see Appendix A). For the hard part $\sigma = (\tilde{q}^2 - \tilde{q} + 1)/\tilde{r}$, we raise to a multiple $\sigma'(u)$ of σ with $\tilde{r} \nmid \sigma$.

Algorithm 5: Miller Loop for our curve

$m \leftarrow 1$; $S \leftarrow Q$;
for *b from the second most significant bit of u to the least* **do**
  $\ell \leftarrow \ell_{S,S}(P)$; $S \leftarrow [2]S$; DoubleLine
  $m \leftarrow m^2 \cdot \ell$; Update1
  if $b = 1$ **then**
    $\ell \leftarrow \ell_{S,Q}(P)$; $S \leftarrow S + Q$; AddLine
    $m \leftarrow m \cdot \ell$; Update2
$Q_u \leftarrow \text{AffineCoordinates}(S)$; Homogeneous ($\mathcal{H}$): $\mathbf{i} + 2\mathbf{m}$; Jacobian ($\mathcal{J}$): $\mathbf{i} + \mathbf{s} + 3\mathbf{m}$
$m_{-u} \leftarrow 1/m_u$; $S \leftarrow Q_u$; $m_u \leftarrow m$; $\mathbf{i}_6$
$\ell \leftarrow \ell_{Q_u,Q}(P)$; $Q_{u+1} \leftarrow Q_u + Q$; AddLine
$m_{u+1} \leftarrow m_u \cdot \ell$; Update2
for *b from the second most significant bit of* $(u^2 - u - 1)$ *to the least* **do**
  $\ell \leftarrow \ell_{S,S}(P)$; $S \leftarrow [2]S$; DoubleLine
  $m \leftarrow m^2 \cdot \ell$; Update1
  if $b = 1$ **then**
    $\ell \leftarrow \ell_{S,Q_u}(P)$; $S \leftarrow S + Q_u$; AddLine
    $m \leftarrow m \cdot m_u \cdot \ell$; $\mathbf{m}_6$+Update2
  else if $b = -1$ **then**
    $\ell \leftarrow \ell_{S,-Q_u}(P)$; $S \leftarrow S - Q_u$; AddLine
    $m \leftarrow m \cdot m_{-u} \cdot \ell$; $\mathbf{m}_6$+Update2
return $m_{u+1} \cdot m^{\tilde{q}}$; $\mathbf{f}_6 + \mathbf{m}_6$

Following [19], we find that $\sigma'(x) = R_0(x) + \tilde{q} \times R_1(x)$ with

$$R_0(x) = -103x^7 + 70x^6 + 269x^5 - 197x^4 - 314x^3 - 73x^2 - 263x - 220$$
$$R_1(x) = 103x^9 - 276x^8 + 77x^7 + 492x^6 - 445x^5 - 65x^4 + 452x^3 - 181x^2 + 34x + 229$$

and a polynomial cofactor $3(x^3 - x^2 + 1)$ to $\sigma(x)$. The exponentiation is dominated by exponentiations to $u, u^2, \ldots, u^9$. With the same analysis as for BLS12-377 in Appendix A (this is the same u), raising to u costs $\exp_u = 4(\text{nbits}(u) - 1)\mathbf{m} + (6\,\text{HW}(u) - 3)\mathbf{m} + (\text{HW}(u) - 1)\mathbf{m}_6 + 3(\text{HW}(u) - 1)\mathbf{s} + \mathbf{i} = 4 \cdot 63\mathbf{m} + 39\mathbf{m} + 6\mathbf{m}_6 + 18\mathbf{s} + \mathbf{i} = 417 + \mathbf{i} = 442\mathbf{m}$; and nine such u-powers cost $3978\mathbf{m}$. Eight Frobenius powers $\mathbf{f}_6 = 4\mathbf{m}$ occur, and exponentiations to the small coefficients of R_0, R_1. They do not seem suited for short addition chain so we designed a multi-exponentiation in 2-NAF in Algorithm 6 (Appendix B) which costs $9\mathbf{s}_6^{\text{cyclo}} + 51\mathbf{m}_6 = 972\mathbf{m}$. The total count is $(99+3978+32+972)\mathbf{m} = 5081\mathbf{m}$.

Finally, according to Table 7, our curve is much faster than SW6. We obtain a pairing whose cost in terms of base field multiplications is two times cheaper compared to the Tate pairing on SW6 and four times cheaper than the ate pairing available in Zexe Rust implementation [27], and moreover the multiplications take place in a smaller field by one 64-bit machine word. Particularly, it is at least five times cheaper to compute on our curve the product of pairings needed to verify a Groth proof (Eq. (3)). The verification would be even more cheaper if we include the GLV-based multi-scalar multiplication in $\mathbb{G}_1$ needed for vk_x

Curve	Prime	Pairing	Miller loop	Exponentiation	Total
BLS12	377-bit q_{BLS12}	ate	$6705\mathbf{m}_{384}$	$7063\mathbf{m}_{384}$	$13768\mathbf{m}_{384}$
SW6	782-bit q_{SW6}	Tate	$21510\mathbf{m}_{832}$	$10521\mathbf{m}_{832}$	$32031\mathbf{m}_{832}$
		ate	$47298\mathbf{m}_{832}$	$10521\mathbf{m}_{832}$	$57819\mathbf{m}_{832}$
		opt. ate	$21074\mathbf{m}_{832}$	$10521\mathbf{m}_{832}$	$31595\mathbf{m}_{832}$
Our curve	761-bit $\tilde{q}$	opt. ate	$7911\mathbf{m}_{768}$	$5081\mathbf{m}_{768}$	$12992\mathbf{m}_{768}$

Table 7. Pairing cost estimation in terms of base field multiplications $\mathbf{m}_b$, where b is the bitsize in Montgomery domain on a 64-bit platform.

and subgroup checks in $\mathbb{G}_1$ and $\mathbb{G}_2$ needed for the proof and the verification keys. Finally, We note that proof generation is also faster on our curve but we chose to base the comparison on the verification equation for two reasons: The cost of proof generation depends on the NP-statement being proved while the verification cost is constant for a given curve, and in blockchain applications, only the verification is performed on the blockchain, costing execution fees.

Implementation. We provide a Sagemath script and a Magma script to check the formulas and algorithms of this section at `https://gitlab.inria.fr/zk-curves/bw6-761/`. We also provide, based on `libff` library [26], a C++ implementation of the curve arithmetic for $\tilde{E}$ and $\tilde{E}'$, and of the optimal ate pairing described in Alg. 5 and Alg. 6. The open source code is available under MIT public licence at: `https://github.com/EYBlockchain/zk-swap-libff/tree/ey/libff/algebra/curves/bw6_761`.

For benchmarking, we choose to compare the timings of a pairing computation, and an evaluation of Eq. (3) which costs mainly 3 Miller loops and 1 final exponentiation (cf. Tab. 8). Since SW6 original implementation is in Rust [27], we implemented both SW6 and our curve in C++ for a like-for-like comparison. It is to note that the Rust code implements an ate pairing with affine coordinates as those should lead, for SW6, to a slightly faster Miller loop than the projective coordinates, as suggested in [28,29]. For completeness, we report also timings for SW6 ate pairing in projective coordinates.
This was tested on a 2.2 GHz Intel Core i7 x86_64 processor with 16 Go 2400 MHz DDR4 memory running macOS Mojave 10.14.6. C++ compiler is clang 10.0.1. Profiling routines use `clock_gettime` and `readproc` calls.

Curve	Pairing	Miller loop	Exponentiation	Total	Eq. 3
SW6	ate (projective)	0.0388s	0.0110s	0.0499s	0.1274s
SW6	ate (affine)	0.0249s	0.0110s	0.0361s	0.0857s
Our curve	opt. ate	0.0053s	0.0044s	0.0097s	0.0203s

Table 8. C++ implementation timings of ate pairing computation on SW6 and optimal ate pairing on our curve.

We conclude that using this work, a pairing computation is 3.7 times faster (or 5 times faster with projective SW6 coordinates) and the verification of a Groth16 proof is 6.27 times faster (or respectively 4.22 times), compared to the curve SW6. Finally, one should also include the cost to check that the proof and the verification key elements are in the right subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$, which should also be faster on our curve as described in §10, §3.2.

4 Security estimate of the curves

We estimate the security in $\mathbb{G}_T$ against a discrete logarithm computation for BLS12-377, SW6, and our curve. BLS12-377 has a security of about 2^{125} in $\mathbb{F}_{q_{\mathrm{BLS}}^{12}}$, SW6 has security about 2^{138} in $\mathbb{F}_{q_{\mathrm{SW6}}^{6}}$ and our curve has a security of about 2^{126} in $\mathbb{F}_{\tilde{q}^6}$, taking into account the Special Tower NFS algorithm and the model of estimation in [24]. For SW6, the characteristic q_{SW6} has not a special form and we consider the TNFS algorithm with the same set of parameters as for a MNT curve of embedding degree 6. The security on the curve BLS12-377 ($\mathbb{G}_1$ and $\mathbb{G}_2$) is 126 bits because r_{BLS12} is 253-bit long. The security in $\mathbb{G}_1$ and $\mathbb{G}_2$ for SW6 and our curve is 188 bits as $r_{\mathrm{SW6}} = \tilde{r}$ is 377-bit long.

The curve parameters are given in Tables 1 and 2 for BLS12-377 and SW6, and 4, 3 for our curve. The parameters for the estimation are given in Tables 9 and 10. In [24, Table 5], the BLS12-381 curve has a security in $\mathbb{G}_T$ estimated about 2^{126}. We obtain a very similar result for BLS12-377: 2^{125}, indeed the curves are almost the same size. Very luckily, we also obtained a security of 2^{126} in $\mathbb{F}_{\tilde{q}^6}$.

4.1 A note on Cheon's attack

Cheon [13] showed that given $G, [\tau]G$ and $[\tau^{\mathrm{T}}]G$, with G a point in a subgroup $\mathbb{G}$ of order r with $\mathrm{T}|r-1$, it is possible to recover τ in $2\left(\left\lceil\sqrt{\frac{r-1}{\mathrm{T}}}\right\rceil + \left\lceil\sqrt{\mathrm{T}}\right\rceil\right) \times (\mathtt{Exp}_{\mathbb{G}}(r) + \log r \times \mathtt{Comp}_{\mathbb{G}})$ where $\mathtt{Exp}_{\mathbb{G}}(r)$ stands for the cost of one exponentiation in $\mathbb{G}$ by a positive integer less than r and $\mathtt{Comp}_{\mathbb{G}}$ for the cost to determine if two elements are equal in $\mathbb{G}$. According to Theorem 2 in [13], if $\mathrm{T} \leq r^{1/3}$, then the complexity of the attack is about $O(\sqrt{r/\mathrm{T}})$ exponentiation by using $O(\sqrt{r/\mathrm{T}})$ storage.

In zkSNARK settings such as in [22], the preprocessing phase includes elements $[\tau^{\ i}]_{i=0}^{\tau=\mathrm{T}}G_1 \in \ \mathbb{G}_1$ and $[\tau^{\ i}]_{i=0}^{\tau=\mathrm{T}}G_2 \in \ \mathbb{G}_2$ for $\mathrm{T} \in \mathbb{N}^*$ the size of the arithmetic circuit related to the NP-statement being proved where τ is a secret trapdoor (usually called a *toxic waste* in SNARK language and is generated by a trustworthy party or in a multi-party computation ceremony [9]). The $\mathrm{T} \mid r-1$ also holds since we need r to be highly *2-adic*. So, given these auxiliary inputs, an attacker would consider to recover the *toxic waste* using Cheon's algorithm in $O(\sqrt{r/\mathrm{T}})$ allowing him thus the possibility of breaking the zkSNARK soundness. This attack is directly related to the specific T-sized circuit being proved, so we estimate the security of the curves for the circuits of the applications we mentioned, precisely Nightfall fungible tokens transfer circuit ($\mathrm{T}_{\mathbb{G}_1} = 2^{22}-1, \mathrm{T}_{\mathbb{G}_2} = 2^{21}$) and Filecoin

curve	BLS12-377	SW6	Our curve
q (bits)	377	782	761
r (bits)	253	377	377
p^k(bits)	4521	4691	4562
u (bits)	64	64	64
polynomials	STNFS	TNFS-Conj	STNFS
$\deg h$	6	2	6
$\deg f_y$	12	6	12
$\deg g_y$	2	3	1
$\|f_y\|_\infty$	2	7	911
$\|g_y\|_\infty$ ($\approx u$ for STNFS)	$2^{63.06}$	$\approx 2^{391}$	$2^{63.06}$
$1/\zeta_{K_h}(2)$	0.9648	0.8608	0.9399
$\alpha(f_y, h, 10^3)$	2.5374	0.5611	0.9921
$\alpha(g_y, h, 10^3)$	1.7625	0.1522	1.8305
A	648	13530839279	648
B	$2^{64.804}$	$2^{71.149}$	$2^{65.016}$
average N_f (bits)	731	401	774
average N_g (bits)	493	978	436
average $N_f N_g$ (bits)	1224	1380	1210
av. B-smooth proba	$2^{-62.5755}$	$2^{-69.9728}$	$2^{-62.2275}$
relation collection space	$2^{123.092}$	$2^{137.622}$	$2^{123.092}$
factor base size	$2^{60.3480}$	$2^{66.5555}$	$2^{60.5555}$
relations obtained	$2^{60.4645}$	$2^{67.4331}$	$2^{60.7747}$
total cost	2^{125}	2^{138}	2^{126}

Table 9. Summary of parameters and estimated data for the simulation of STNFS ([24, Alg. 6.1], average over 10^5 samples) for BLS12-377 curve, $k = 12$ and $D = 3$, SW6 curve $k = 6, D = 339$ and our Cocks–Pinch curve, $k = 6$ and $D = 3$.

curve	seed $u = $ `0x8508c00000000001`, polynomials
BLS12 377 STNFS	$h = Y^6 - 2Y^4 + Y^3 + 2Y^2 - 1$ $f_y = X^{12} - 2yX^{10} + 2y^3X^6 + y^5X^2 + 2y^4 - y^3 - 2y^2 + 1$ $g_y = X^2 - uy = X^2 - 9586122913090633729y$
SW6 782 TNFS	$h = Y^2 + Y - 1$ $f_y = x^6 + x^5 - 3x^4 - x^3 + 7x^2 + 5x + 1$ $g_y = dX^3 - cX^2 - (c + 3d)X - d$, c/d root of $s^2 + s + 2 \bmod q_{\mathrm{SW6}}$
Our curve 761 STNFS	$h = Y^6 + Y^3 - 2Y^2 + 2Y - 1$ $f_y = 103X^{12} - 379X^{11} + 250X^{10} + 691X^9 - 911X^8 - 79X^7$ $+623X^6 - 640X^5 + 274X^4 + 763X^3 + 73X^2 + 254X + 229$ $g_y = X - u = X - 9586122913090633729$

Table 10. Polynomials h, f_y, g_y chosen to minimize the total estimated cost of STNFS. The simulation of STNFS of [24, Alg. 6.1] with 10^5 samples produced the data of Table 9.

circuit ($T_{\mathbb{G}_1} = 2^{28} - 1, T_{\mathbb{G}_2} = 2^{27}$) which is the biggest circuit of public interest in the blockchain community.

We recall that our curve is designed for proof composition and that the verifier circuit of a Groth's proof can be expressed in $\tilde{T} = 40000$ constraints. Hence, the complexity of Cheon's attack on our curve, in the case of composing Groth's proofs, would be $O\left(\sqrt{\tilde{r}/\tilde{T}}\right) \approx 2^{237}$ which is worse than the generic attack. However, for completeness, we state that under this attack, the curve still has at least 126-bit security as previously stated, for circuits of size up to 2^{46} which is, to the authors' knowledge, "large enough" for all the published applications.

For BLS12-377 curve, because the subgroup order is 253-bit, the estimated security for Nightfall setup is ≈ 116-bit in $\mathbb{G}_1$ and for Filecoin is ≈ 113-bit in $\mathbb{G}_1$. While this curve has a standard security under generic attacks (without auxiliary inputs), one should use it with small SNARK circuits or look for alternative inner curves, which we set as a future work.

5 Conclusion

In this work, we construct on top of Zexe's inner curve a new pairing-friendly elliptic curve suitable for one layer proof composition. We discuss its security and the optimizations it allows. We conclude that it is at least five times faster for verifying Groth's proof compared to the previous state-of-the-art, and validate our results with an open-source C++ implementation.

We mentioned several projects in the blockchain community that need one layer proof composition and therefore can benefit from this work. Applications such as Zexe, EY Nightfall, Celo or Filecoin can consequently reduce their operations cost.

References

1. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López-Hernández, J.C.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (May 2011). https://doi.org/10.1007/978-3-642-20465-4_5, https://ia.cr/2010/526
2. Arène, C., Lange, T., Naehrig, M., Ritzenthaler, C.: Faster computation of the tate pairing. Journal of Number Theory **131**(5, Elliptic Curve Cryptography), 842–857 (2011). https://doi.org/10.1016/j.jnt.2010.05.013, http://cryptojedi.org/papers/#edpair
3. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg (Sep 2003). https://doi.org/10.1007/3-540-36413-7_19
4. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_6

5. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 276–294. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44381-1_16
6. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 967–980. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516734
7. Bowe, S.: BLS12-381: New zk-SNARK elliptic curve construction (2017), `https://electriccoin.co/blog/new-snark-curve/`
8. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: Zexe: Enabling decentralized private computation. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 1059–1076. IEEE Computer Society, Los Alamitos, CA, USA (may 2020), `https://www.computer.org/csdl/proceedings-article/sp/2020/349700b059/1i0rIqoBYD6`, `https://ia.cr/2018/962`
9. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-SNARK parameters in the random beacon model. ePrint 2017/1050 (2017), `https://ia.cr/2017/1050`
10. Brier, E., Coron, J.S., Icart, T., Madore, D., Randriam, H., Tibouchi, M.: Efficient indifferentiable hashing into ordinary elliptic curves. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 237–254. Springer, Heidelberg (Aug 2010). https://doi.org/10.1007/978-3-642-14623-7_13
11. Celo: BLS-ZEXE: BLS signatures verification inside a SNARK proof. (2019), `https://github.com/celo-org/bls-zexe`
12. Chatterjee, S., Sarkar, P., Barua, R.: Efficient computation of Tate pairing in projective coordinate over general characteristic fields. In: Park, C., Chee, S. (eds.) ICISC 04. LNCS, vol. 3506, pp. 168–181. Springer, Heidelberg (Dec 2005). https://doi.org/10.1007/11496618_13
13. Cheon, J.H.: Discrete logarithm problems with auxiliary inputs. Journal of Cryptology **23**(3), 457–476 (Jul 2010). https://doi.org/10.1007/s00145-009-9047-0
14. Chiesa, A., Chua, L., Weidner, M.: On cycles of pairing-friendly elliptic curves. SIAM Journal on Applied Algebra and Geometry **3**(2), 175–192 (2019). https://doi.org/10.1137/18M1173708
15. Costello, C., Lange, T., Naehrig, M.: Faster pairing computations on curves with high-degree twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-13013-7_14, `https://ia.cr/2009/615`
16. EY-Blockchain: Nightfall: An open source suite of tools designed to enable private token transactions over the public ethereum blockchain. (2019), `https://github.com/EYBlockchain/nightfall`
17. Fouque, P.A., Tibouchi, M.: Indifferentiable hashing to Barreto-Naehrig curves. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 1–17. Springer, Heidelberg (Oct 2012). https://doi.org/10.1007/978-3-642-33481-8_1
18. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology **23**(2), 224–280 (Apr 2010). https://doi.org/10.1007/s00145-009-9048-z, `https://ia.cr/2006/372`
19. Fuentes-Castañeda, L., Knapp, E., Rodríguez-Henríquez, F.: Faster hashing to $\mathbb{G}_2$. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 412–430. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-28496-0_25

20. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_11
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989). https://doi.org/10.1137/0218012
22. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_11
23. Guillevic, A., Masson, S., Thomé, E.: Cocks–Pinch curves of embedding degrees five to eight and optimal ate pairing computation. Des. Codes Cryptogr. pp. 1–35 (March 2020). https://doi.org/10.1007/s10623-020-00727-w, `https://hal.inria.fr/hal-02305051`
24. Guillevic, A., Singh, S.: On the alpha value of polynomials in the tower number field sieve algorithm. ePrint 2019/885 (2019), `https://ia.cr/2019/885`
25. Karabina, K.: Squaring in cyclotomic subgroups. Math. Comput. **82**(281), 555–579 (2013). https://doi.org/10.1090/S0025-5718-2012-02625-1
26. scipr lab: libff: C++ library for finite fields and elliptic curves. (2018), `https://github.com/scipr-lab/libff`
27. scipr lab: ZEXE rust implementation. (2018), `https://github.com/scipr-lab/zexe`
28. Lauter, K., Montgomery, P.L., Naehrig, M.: An analysis of affine coordinates for pairing computation. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) PAIRING 2010. LNCS, vol. 6487, pp. 1–20. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17455-1_1
29. Le, D.P., Tan, C.H.: Speeding up ate pairing computation in affine coordinates. In: Kwon, T., Lee, M., Kwon, D. (eds.) ICISC 12. LNCS, vol. 7839, pp. 262–277. Springer, Heidelberg (Nov 2013). https://doi.org/10.1007/978-3-642-37682-5_19
30. Meckler, I., Shapiro, E.: Coda: Decentralized cryptocurrency at scale. O(1) Labs whitepaper (2018), `https://cdn.codaprotocol.com/v2/static/coda-whitepaper-05-10-2018-0.pdf`
31. ProtocolLabs: Filecoin: A decentralized storage network. (2017), `https://filecoin.io/filecoin.pdf`
32. Shallue, A., van de Woestijne, C.E.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS VII. LNCS, vol. 4076, pp. 510–524. Springer (2006). https://doi.org/10.1007/11792086_36
33. Skalba, M.: Points on elliptic curves over finite fields. Acta Arithmetica **117**(3), 293–301 (2005), `http://eudml.org/doc/278782`
34. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8_1
35. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (Jan 2010). https://doi.org/10.1109/TIT.2009.2034881, `https://ia.cr/2008/096`
36. Wahby, R.S., Boneh, D.: Fast and simple constant-time hashing to the BLS12-381 elliptic curve. IACR TCHES **2019**(4), 154–179 (2019). https://doi.org/10.13154/tches.v2019.i4.154-179

A Pairing computation for BLS12-377 and SW6 curves

We report here the cost of computing a pairing on BLS12-377 and SW6 curves from Zexe. We estimate the cost of Miller loop and final exponentiation for both ate and Tate pairing in the case of SW6 and ate pairing in the case of BLS12-377.

Miller loop for BLS12-377. For BLS12-377, the ate pairing is $\mathrm{ate}(P,Q) = (f_{u,Q}(P))^{(q_{\mathrm{BLS12}}^k-1)/r_{\mathrm{BLS12}}}$ and it is optimal in the sense of Vercauteren [35], the Miller loop is the shortest thanks to the trace $t_{\mathrm{BLS12}} - 1 = u$. The cost of Miller loop is given by Eq. (9), where nbits is the bitlength and HW is the Hamming weight, and the costs of main steps are given in Table 6.

$$\mathrm{Cost}_{\textsc{MillerLoop}} = (\mathrm{nbits}(u)-1)\mathrm{Cost}_{\textsc{DoubleLine}} + (\mathrm{nbits}(u)-2)\mathrm{Cost}_{\textsc{Update1}} \quad (8)$$
$$+ (\mathrm{HW}(u)-1)(\mathrm{Cost}_{\textsc{AddLine}} + \mathrm{Cost}_{\textsc{Update2}}) \quad (9)$$

The Hamming weight of $u = 2^{63}+2^{58}+2^{56}+2^{51}+2^{47}+2^{46}+1$ is $\mathrm{HW}(u) = 7$ and its length is $\mathrm{nbits}(u) = 64$. We obtain $\mathrm{Cost}_{\textsc{MillerLoop}} = 63(3\mathbf{m}_2+6\mathbf{s}_2+4\mathbf{m})+62(\mathbf{s}_{12}+13\mathbf{m}_2)+6(11\mathbf{m}_2+2\mathbf{s}_2+4\mathbf{m}+13\mathbf{m}_2)$. With $\mathbf{m}_2 = 3\mathbf{m}$, $\mathbf{s}_2 = 2\mathbf{m}$, $\mathbf{s}_{12} = 36\mathbf{m}$ (applying recursively Karatsuba and Chung–Hasan formulas for multiplication and squaring in quadratic and cubic extensions), we obtain $\mathrm{Cost}_{\textsc{MillerLoop}} = 6705\mathbf{m}$. The count is reported in Table 7.

Final exponentiation for BLS12-377. The final exponentiation for BLS12 curves is decomposed in $(q_{\mathrm{BLS}}^{12}-1)/r_{\mathrm{BLS}} = (q_{\mathrm{BLS}}^{12}-1)/\Phi_{12}(q_{\mathrm{BLS}}) \times \Phi_{12}(q_{\mathrm{BLS}})/r_{\mathrm{BLS}}$, where the first ratio simplifies to $(q_{\mathrm{BLS}}^6-1)(q_{\mathrm{BLS}}^2+1)$, and the second ratio is a polynomial of degree 20 that can be decomposed in basis q_{BLS12}, because Frobenius powers are much faster. One can use cyclotomic squarings where $\mathbf{s}_{12}^{\mathrm{cyclo}} = 18\mathbf{m}$, or compressed cyclotomic squarings (method of Karabina [25]) where $\mathbf{s}_{12}^{\mathrm{comp}}$ has a cost dominated by $4\mathbf{s}_2$. Applying [25, Corollary 4.1], the cost of raising to the power u costs $\exp_u = 4(\mathrm{nbits}(u)-1)\mathbf{m}_2 + (6\,\mathrm{HW}(u)-3)\mathbf{m}_2 + (\mathrm{HW}(u)-1)\mathbf{m}_{12} + 3(\mathrm{HW}(u)-1)\mathbf{s}_2 + \mathbf{i}_2$. The final count is $\mathbf{i}_{12} + 2\mathbf{s}_{12}^{\mathrm{cyclo}} + 12\mathbf{m}_{12} + 4\mathbf{f}_{12} + 5\exp_u$. Reusing the script provided with [23], we obtain a final count of $7063\mathbf{m}$, reported in Table 7.

Miller loop for SW6 curve. We consider the ate and the Tate pairing on the curve SW6. The Miller loop of the ate pairing iterates over $T = t_{\mathrm{SW6}} - 1$ which is 388-bit long,

$$\mathrm{Cost}_{\textsc{MillerLoop ate}} = (\mathrm{nbits}(T)-1)\mathrm{Cost}_{\textsc{DoubleLine}} + (\mathrm{nbits}(T)-2)\mathrm{Cost}_{\textsc{Update1}}$$
$$+ (\mathrm{HW}_{\text{2-NAF}}(T)-1)(\mathrm{Cost}_{\textsc{AddLine}} + \mathrm{Cost}_{\textsc{Update2}})$$

where the costs of line and update are given in Table 11, indeed $a_{\mathrm{SW6}} = 5$ and a quadratic twist is available. With $\mathrm{nbits}(T) = 388$, $\mathrm{HW}_{\text{2-NAF}}(T) = 107$, we obtain $(388-1)(5\mathbf{m}_3+6\mathbf{s}_3+6\mathbf{m})+(388-2)(\mathbf{m}_6+\mathbf{s}_6)+(107-1)(10\mathbf{m}_3+3\mathbf{s}_3+6\mathbf{m}+\mathbf{m}_6) = 47298\mathbf{m}$. The Miller loop of the Tate pairing is over r_{SW6} which is 377-bit long and $\mathrm{HW}_{\text{2-NAF}}(r_{\mathrm{SW6}}) = 98$. The costs of line computation are slightly changed:

the point operations are now in the base field, and one can save one multiplication $\mathbf{m}$ when evaluating the line at Q. We obtain $(377-1)(6\mathbf{m}+6\mathbf{s}+6\mathbf{m})+(377-2)(\mathbf{m}_6+\mathbf{s}_6)+(98-1)(9\mathbf{m}+3\mathbf{s}+6\mathbf{m}+\mathbf{m}_6)=21510\mathbf{m}$. There are other formulas in Jacobian coordinates with other trade-off of $\mathbf{m}$ and $\mathbf{s}$, however, if we assume $\mathbf{s}\approx\mathbf{m}$, the costs stay the same. From [2], on Weierstrass curves a doubling an line evaluation costs $\mathbf{m}+11\mathbf{s}+k\mathbf{m}$, if $a=-3$ this is $6\mathbf{m}+5\mathbf{s}+k\mathbf{m}$, and if $a=0$, this is $3\mathbf{m}+8\mathbf{s}+k\mathbf{m}$. Mixed addition and line evaluation costs $6\mathbf{m}+6\mathbf{s}+k\mathbf{m}$.

k	D	curve with quadratic twist	Tate DoubleLine AddLine	ate DoubleLine AddLine	Update1 Update2	ref
$2 \mid k$	any	$y^2=x^3+ax+b$	$6\mathbf{m}+6\mathbf{s}+k\mathbf{m}$ $9\mathbf{m}+3\mathbf{s}+k\mathbf{m}$	$5\mathbf{m}_{k/2}+6\mathbf{s}_{k/2}+k\mathbf{m}$ $10\mathbf{m}_{k/2}+3\mathbf{s}_{k/2}+k/2\mathbf{m}$	$\mathbf{m}_k+\mathbf{s}_k$ $\mathbf{m}_k$	[12]
$2 \mid k$	any	$y^2=x^3-3x+b$	$7\mathbf{m}+4\mathbf{s}+k\mathbf{m}$ $9\mathbf{m}+3\mathbf{s}+k\mathbf{m}$	$6\mathbf{m}_{k/2}+4\mathbf{s}_{k/2}+k\mathbf{m}$ $10\mathbf{m}_{k/2}+3\mathbf{s}_{k/2}+k/2\mathbf{m}$	$\mathbf{m}_k+\mathbf{s}_k$ $\mathbf{m}_k$	[12]

Table 11. Miller loop cost in Weierstrass model from [12] in Jacobian coordinates.

Optimal ate Miller loop for SW6 curve. Like for our curve, an optimal ate pairing is available for SW6. We have $(x^3-x^2-x)+q_{\mathrm{SW6}}(x+1)=0 \bmod r_{\mathrm{SW6}}(x)$ and $\pi_{q_{\mathrm{SW6}}}(Q)=[q]Q$ for all $Q\in\mathbb{G}_2$. An optimal ate pairing is

$$\mathrm{ate}_{\mathrm{opt}}(P,Q)=(f_{u^3-u^2-u,Q}(P)f_{u+1,Q}^{q_{\mathrm{SW6}}}(P))^{(q_{\mathrm{SW6}}^6-1)/r_{\mathrm{SW6}}}\ . \quad (10)$$

The same trick as in Algorithm 5 applies to further optimize the computation of $f_{u^3-u^2-u,Q}(P)=f_{u(u^2-u-1),Q}(P)$. We apply the formula of estimated cost (8) and obtain $(64+127-2)(5\mathbf{m}_3+6\mathbf{s}_3+6\mathbf{m})+(64+127-3)(\mathbf{m}_6+\mathbf{s}_6)+(7+19-1)(10\mathbf{m}_3+3\mathbf{s}_3+3\mathbf{m}+\mathbf{m}_6+\mathbf{m}_6)+\mathbf{i}_6+\mathbf{f}_6+\mathbf{m}_6+\mathbf{i}+\mathbf{s}+3\mathbf{m}=21024\mathbf{m}+2\mathbf{i}\approx21074\mathbf{m}$ with $\mathbf{m}=\mathbf{s}$ and $\mathbf{i}\approx25\mathbf{m}$. The difference between Tate and optimal ate Miller loop for SW6 is $\approx 2\%$; this is within the error margin. Indeed the number of base-field multiplications $\mathbf{m}$ does not capture 100% of the total cost of a pairing computation. Since a Tate pairing is actually easier to implement as it does not involve curve arithmetic in $\mathbb{G}_2$, it could turn out to be faster than the optimal ate pairing.

Final exponentiation for SW6 curve. The final exponentiation raises the Miller loop result to the power $(q_{\mathrm{SW6}}^6-1)/r$. The easy part is raising to $(q_{\mathrm{SW6}}^3-1)(q_{\mathrm{SW6}}+1)$ with one Frobenius q_{SW6}, one Frobenius q_{SW6}^3 which is a conjugation, one inversion and two multiplications, costing $\mathbf{f}_6+\mathbf{i}_6+2\mathbf{m}_6=4\mathbf{m}+34\mathbf{m}+\mathbf{i}+36\mathbf{m}=74\mathbf{m}+\mathbf{i}\approx 99\mathbf{m}$. The hard part $e=(q_{\mathrm{SW6}}^2-q_{\mathrm{SW6}}+1)/r$ is decomposed in basis q_{SW6} so that $-W_0+W_1q_{\mathrm{SW6}}=e$, where W_0 is 721-bit long and W_1 is 406-bit long. At this stage, inversions are free as $f^{-1}=f^{q_{\mathrm{SW6}}^3}$, and cyclotomic squarings $\mathbf{s}_6^{\mathrm{cyclo}}=6\mathbf{m}$ are available. A multi-exponentiation would cost at least $720\mathbf{s}_6^{\mathrm{cyclo}}+478\mathbf{m}_6=12924\mathbf{m}$. Writing W_0,W_1 in $\mathrm{HW}_{\text{2-NAF}}$ form, this is reduced to $720\mathbf{s}_6^{\mathrm{cyclo}}+339\mathbf{m}_6=10422\mathbf{m}$, for a total count of $10521\mathbf{m}$.

B Optimized hard part of final exponentiation for our curve

We include here the algorithm that we designed to optimize the hard step in the final exponentiation for our curve.

Algorithm 6: Optimized hard part of final exponentiation

Input: f in $\mathbb{F}_{\tilde{q}^6}$

Output: $f \leftarrow f^{\sigma'(u)}$ in $\mathbb{F}_{\tilde{q}^6}$

$f_0 \leftarrow f$; $f_{0p} \leftarrow f_0^{\tilde{q}}$;

for $(i = 1,\ i \leq 7,\ i = i+1)$ **do**

  $f_i \leftarrow f_{i-1}^u$; $f_{pi} \leftarrow f_i^{\tilde{q}}$;

$f_{8p} \leftarrow f_{7p}^u$; $f_{9p} \leftarrow f_{8p}^u$;

$f \leftarrow f_{3p} * f_{6p} * (f_{5p})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f_{4,2p} \leftarrow f_4 * f_{2p}$; $f \leftarrow f * f_5 * f_{0p} * (f_0 * f_1 * f_3 * f_{4,2p} * f_{8p})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f \leftarrow f * f_{9p} * (f_7)^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f_{2,4p} \leftarrow f_2 * f_{4p}$; $f_{4,2p,5p} \leftarrow f_{4,2p} * f_{5p}$;
 $f \leftarrow f * f_{4,2p,5p} * f_6 * f_{7p} * (f_{2,4p} * f_3 * f_{3p})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f \leftarrow f * f_0 * f_7 * f_{1p} * (f_{0p} * f_{9p})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f_{6p,8p} \leftarrow f_{6p} * f_{8p}$; $f_{5,7p} \leftarrow f_5 * f_{7p}$; $f \leftarrow f * f_{5,7p} * f_{2p} * (f_{6p,8p})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f_{3,6} \leftarrow f_3 * f_6$; $f_{1,7} \leftarrow f_1 * f_7$; $f \leftarrow f * f_{3,6} * f_{9p} * (f_{1,7} * f_2)^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f \leftarrow f * f_0 * f_{0p} * f_{3p} * f_{5p} * (f_{4,2p} * f_{5,7p} * f_{6p,8p})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f \leftarrow f * f_{1p} * (f_{3,6})^{\tilde{q}^3}$;

$f \leftarrow f^2$; $f \leftarrow f * f_{1,7} * f_{5,7p} * f_{0p} * (f_{2,4p} * f_{4,2p,5p} * f_{9p})^{\tilde{q}^3}$;

return f
