A survey of elliptic curves for proof systems[⋆]

Diego F. Aranha[1][0000−0002−2457−0783],
Youssef El Housni[2,3,4][0000−0003−2873−3479], and
Aurore Guillevic[1,5][0000−0002−0824−7273]

[1] Aarhus University, Aarhus, Denmark
dfaranha@cs.au.dk
[2] ConsenSys, gnark, Paris, France
[3] LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris
[4] Inria
youssef.elhousni@consensys.net
[5] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
aurore.guillevic@inria.fr

Abstract. Elliptic curves have become key ingredients for instantiating zero-knowledge proofs and more generally proof systems. Recently, there have been many tailored constructions of these curves that aim at efficiently implementing different kinds of proof systems. In this survey we provide the reader with a comprehensive overview on existing work and revisit the contributions in terms of efficiency and security. We present an overview at three stages of the process: curves to instantiate a SNARK, curves to instantiate a recursive SNARK, and also curves to express an elliptic-curve related statement. We provide new constructions of curves for SNARKs and generalize the state-of-the-art constructions for recursive SNARKs. We also exhaustively document the existing work and open-source implementations.

1 Introduction

A proof system is an interactive protocol where one party (called the prover) tries to convince another party (called the verifier) that a given statement is true. In a zero-knowledge proof, we require further that the proof does not reveal anything beyond the truth of the statement. A classical example is Schnorr's proof of knowledge of a discrete logarithm x in a cyclic group $\mathbb{G}$ of prime order r. The verifier is given $\mathbb{G}$, a generator G, and an element Q. The prover will prove that they know x such that $Q = [x]G$ without revealing it. For that, they take at random a secret nonce $n \in \mathbb{Z}_r$ and send $A = [n]G$ to the verifier. In return, the verifier takes at random a challenge $c \in \mathbb{Z}_r$ and sends it to the verifier, who computes $s = n + c \cdot x$ (to hide x with the secret n) and sends s to the verifier. Finally the verifier checks that $[s]G = A + [c]Q$, indeed, one

[⋆] This article is the authors' full version submitted to Springer on 2022-10-14 for the special issue *Mathematics of zero-knowledge* of Designs, Codes and Cryptography, ePrint 2022/586 and hal-03667798.

should have $[s]G = [n]G + [c][x]G$. A proof is non-interactive if no communication is required between the prover and the verifier except from sending the proof. Schnorr's example is very simple, but when the statement to be proven grows (how to *prove the verification of a signature* for example?), new machinery is sought. In the class of non-interactive proofs, a particularly interesting concept for proving the correctness of a computation is the Succinct Non-interactive ARgument of Knowledge (SNARK). It provides a computationally sound proof cheap to verify and small compared to the size of the statement or the witness (in Schnorr example, the witness is the secret DL x). SNARK systems can be further equipped with a zero-knowledge property that enables the proof to be verified without revealing anything about the intermediate steps (the witness generation).

Proof systems were introduced in [GMR89] and extensively studied both in theoretical and applied settings [Kil92,Mic94,GW11,BCCT12]. Recent constructions focus on a panoply of settings that range from cryptographic assumptions, asymptotic efficiency, concrete performance of implementations to numerous applications. The mathematical security of many schemes relies on variants of the **discrete logarithm problem (DLP)**: given a cyclic group $\mathbb{G}$ of prime order r written additively, a generator G, and an element $P \in \mathbb{G}$, compute $x \in \{0, 1, \ldots, r-1\}$ such that $[x]G = P$. For example, the DLP-based zero-knowledge proofs (e.g. [BCC$^+$16], Bulletproofs [BBB$^+$18], Hyrax [WTs$^+$18]) require a group $\mathbb{G}$ where the discrete logarithm problem is hard. They can be instantiated with any cryptographically secure elliptic curve, where the problem is then referred to as ECDLP. An efficient implementation uses the Ristretto group [Ham15,dV21] over `ed25519` [BDL$^+$12] (e.g. Bulletproofs' Dalek library [dVYA22]). In all these examples, two mathematical structures are combined: a scalar field $\mathbb{Z}_r$ where the *exponents* or *scalars* live, and a cyclic group $\mathbb{G}$ of order r, but defined itself over something else: usually $\mathbb{G}$ is the group of points of an elliptic curve over a prime field $\mathbb{F}_p$, where $\mathbb{F}_p$ is distinct from $\mathbb{Z}_r$. The verifier's task consists in checking an equation hidden in the scalar field ($s = n + c \cdot x$ in Schnorr's example).

Alternatively, a **bilinear pairing** is required in certain schemes. A cryptographic pairing is a bilinear non-degenerate map from two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ to a target group $\mathbb{G}_T$, where the three groups have the same prime order r. The pairing is denoted $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It allows the verifier to multiply *in the exponents*, that is in the scalar field $\mathbb{Z}_r$, two unknown exponents: knowing $A = [a]G_1$ and $B = [b]G_2$, the verifier can get $e(A, B) = e([a]G_1, [b]G_2) = G_T^{ab}$, where G_1, G_2, G_T are generators of the thee pairing groups. (Note that in Schnorr's example above, the verifier can get $[c][x]G = [c]Q$ only because they know the scalar c). The only known instantiation of such a cryptographic pairing is over dedicated algebraic curves named pairing-friendly, and the most efficient ones are specific elliptic curves named after Barreto, Lynn and Scott (BLS), and Barreto–Naehrig (BN). In this context, $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct subgroups of the same order of an elliptic curve, and $\mathbb{G}_T$ is a multiplicative subgroup of some finite field. Pairing-based SNARKs cannot be instantiated with generic-purpose elliptic curves, but instead require tailored constructions of elliptic curves. More precisely, they need pairing-friendly elliptic curves with additional properties, designed on purpose to provide an efficient implementation. In this paper, we give an overview on the elliptic curves designed for different proof systems, revisit some constructions and propose some new ones.

Contributions. We present an overview of elliptic curves tailored for SNARKs, at three stages of the process: curves to instantiate a SNARK, curves to instantiate a recursive SNARK, and also curves to express an elliptic-curve related statement. We investigate the literature and collect the milestones in the constructions of pairing-friendly elliptic curves and 2-chains for SNARKs. We also cover different kinds of cycles of curves and summarize the impossibility results and open questions.

In the case of recursive proofs, the promising (pairing-friendly) curves with room for improvements are the 2-chains of curves. A 2-chain has first an **inner curve** and second an **outer curve**. We generalize the constructions from [EHG22a] and include the BN curves in the general framework of 2-chains, in addition to BLS12 and BLS24 curves. The pairing computation is made of a Miller loop (optimized in [EHG22a]) followed by a final exponentiation. We improve that final exponentiation for the outer curves in the three cases (BLS12, BLS24 and BN).

We propose a systematic way to obtain seeds for inner curves such that the scalar field $\mathbb{Z}_r$ is suited to FFT-efficient implementation of arithmetic circuits, that is, $r-1$ the predecessor of the curve order r has a much higher 2-valuation L (also called 2-adicity, i.e. $2^L \mid r-1$). For BLS24 curves, this more than doubles the 2-valuation of $r-1$. Although the results were already satisfying for BLS12, it was a drawback for using BLS24. As an application, we propose a new inner BLS24 curve more suitable for KZG-based SNARKs, that can be used to prove deeper arithmetic circuits ($2^{60} \mid r-1$ instead of $2^{22} \mid r-1$ at the 128-bit security level).

There are various libraries for SNARKs. We list in Table 12 all known open-source library developments (to the best of our knowledge in 2022), together with the main projects using them, excluding forks and stand-alone implementations.

Organization. Section 2 provides a historical overview on the state of the art and Section 3 lays out briefly mathematical preliminaries on bilinear pairings and zk-SNARKs. The core of the paper are Sections 4, 5 and 6. First, in Section 4, we look at pairing-friendly elliptic curves suitable to some popular SNARKs, provide existing and new constructions and compare them in terms of efficiency and security. Next, in Section 5, we investigate tailored constructions for recursive SNARKs: different combinations of pairing-friendly and plain cycles, and 2-chains of pairing-friendly curves. Finally, in Section 6, we consider how to efficiently prove elliptic-curve cryptographic statements in a SNARK. After concluding, we catalog in Appendix A a list of different independent implementations of discussed curves, cycles and 2-chains. We give in Appendix B tables of parameters of outer curves obtained from inner BN, BLS12 and BLS24. Appendix C optimizes the final exponentiation of the pairing computation for these outer curves.

2 State of the art

In this section we relate the building steps of SNARKs that we date back to 2006.

Building on ideas from the pairing-based doubly-homomorphic encryption scheme [BGN05], Groth, Ostrovsky and Sahai [GOS06,Gro06,GS08] introduced the pairing-based non-interactive zero-knowledge proofs, yielding the first linear-size proofs based on standard assumptions. Groth [Gro10] combined these techniques with ideas from interactive zero-knowledge proofs to give the first constant-size proofs which are based on constructing a set of polynomial equations and using pairings to efficiently verify these equations. This work relies on two new introduced pairing-based cryptographic assumptions, namely the q-computational power Diffie-Hellman (q-CPDH) and the q-power knowledge of Exponent (q-PKE).

Following this direction of work, Gennaro et al. [GGPR13] proposed an insightful construction of polynomial equations that resulted in many interesting implementations [PHGR13,BFR$^+$13,BCG$^+$13,BCTV14b,KPP$^+$14] leading to the most succinct and widely implemented pairing-based SNARK [Gro16]. The first implementation, Pinocchio [PHGR13] used a pairing-friendly elliptic curve in the Barreto–Naehrig family [BN06] (BN) targeting a 128-bit security level, but the source code was proprietary. Parno et al. used the BN curve defined over a 256-bit field suggested in [NNS10] (seed $x = 1868033^3$). Next, as part of Pantry [BFR$^+$13], Braun et al. re-implemented Pinocchio under a BSD-style license using a 254-bit BN curve from [BGM$^+$10] (seed $x = -(2^{62} + 2^{55} + 1)$, first introduced in [NAS$^+$08]). This new BN implementation partially builds on techniques from the previous BN paper [NNS10] Pinocchio used.

Later in [BCG$^+$13], Ben-Sasson et al. observed that constructing a pairing-friendly curve with a subgroup order r where the predecessor $r-1$ is divisible by 2^L a large power of 2, results in an efficient proof generation via suitable Fast Fourier Transforms (FFTs) in $\mathbb{F}_r$. To speedup the arithmetic, they proposed to use the elliptic curve in Edwards form, by looking for a group order multiple of 4. To match these two constraints: 2^L divides $r-1$ and the curve has order $4 \cdot r$, they designed a Galbraith-McKee-Valença curve [GMV07] of embedding degree 6 (GMV6) defined over $\mathbb{F}_p$ where p is a prime of 183 bits, and of order $4r$ where r is of 181 bits such that $2^{31} \mid r-1$. This curve was targeting a 80-bit security level in 2013 and was implemented in `libff` [BSCT$^+$a]. We will call such a field $\mathbb{F}_r$ with $2^L \mid r-1$ to be **FFT-friendly**.

The same year, [BCTV14b] improved previous SNARK works and developed their implementations with two different curves: the same GMV6 curve for the estimated 80-bit security level ($2^{31} \mid r-1$) and a new BN curve for the estimated 128-bit security level ($2^{28} \mid r-1$). Note that, as of today (October 2022), this BN curve is the one precompiled in the Ethereum blockchain. Last, Trinocchio [KPP$^+$14] provided an implementation of a privacy-preserving version of

Pinocchio using the BN curve from [BCTV14b]. As the number of applications increases, other implementations were released especially in the blockchain space. In particular, Zcash cryptocurrency first implemented Zerocash protocol [BCG+14] which uses the SNARK of [BCTV14b] and its BN curve, before switching to the SNARK of [Gro16] and a new curve.

The previous paragraphs told an overview over ten years of active and fruitful research on SNARK from the prelude [BGN05] in 2005 to 2016 with [Gro16]. Meanwhile, the research in discrete logarithm computation in finite fields related to pairings also saw a tremendous decade, with a prequel in 2012 with a discrete logarithm record-breaking computation in $\mathrm{GF}(3^{6\cdot 97})$ in [FNK12]. This announcement had a quite important impact over the community at that time [GM16, §9.3.8 p.30], probably because the curve broken in 2012 was introduced in 2001 with the BLS short signatures [BLS01] and its target field $\mathrm{GF}(3^{6\cdot 97})$ was still believed to be safe for 80-bit security implementations. In 2014, two algorithms on fast computation of DL in small characteristic finite fields were published [BGJT14,GKZ14] (quasi-polynomial-time algorithm, zig-zag descent). In 2019, Granger et al. announced the largest record computation in a field $\mathrm{GF}(2^{30750})$ [GKL+21], and finally [KW22] published a proved complexity. Nowadays, small-characteristic finite fields should be definitely avoided, as computing DL is not hard anymore especially if the extension degree is composite, which renders supersingular elliptic curves in small characteristic insecure.

Cryptanalysts also considered large and medium characteristic finite fields of the form $\mathrm{GF}(p^k)$ that arise with pairings. In 2016, Kim and Barbulescu [KB16] presented a variant of the Number Field Sieve (NFS) algorithm which reduced the security level of the BN curves previously at 128-bit security to around 110. Menezes, Sarkar and Singh [MSS16] were the first to analyze thoroughly the consequences of the new NFS variants on the security of pairing-friendly curves. Their conclusions recommend the Barreto–Lynn–Scott curves of embedding degree 12 [BLS04] or BN curves over 384-bit prime fields instead of BN curves over 256-bit fields. Based on this work, the Zcash team derived the BLS12-381 curve defined over a 381-bit field with a 255-bit prime subgroup order r such that $2^{32} \mid r - 1$ [Bow17]. This curve is used today (October 2022) in several projects (e.g. Zcash, Ethereum 2.0, Skale, Algorand, Dfinity, Chia), implemented in different libraries and considered for IETF standards.

Besides their efficiency, SNARKs' succinctness makes them good candidates for recursive proof composition. Such proofs could themselves verify the correctness of other proofs. This would allow a single proof to inductively attest to the correctness of a former proof or many former proofs. This is achieved by proving the correct verification of many proofs in a one-layer recursion (using 2-chains) or by proving a new proof and the so far composed proof at each layer in a multi-layer recursion (using 2-cycles). However, once a first proof is generated, it is highly impractical to use the same elliptic curve to generate a second proof verifying the first one. In pairing-based SNARKs the proving algorithm runs in $\mathbb{F}_r$ while the verification algorithm runs in $\mathbb{F}_p$. Ideally, we would like to select a curve E of order r over $\mathbb{F}_p$ with $p = r$, so that the proving arithmetic is over the same field for which the statement (the verification of the previous proof) is defined. Unfortunately, this cannot happen: the condition that E has embedding degree k with respect to r implies that $r \mid p^k - 1$, which implies that $p \neq r$. Furthermore, the curves satisfying the $p = r$ condition have a trivial discrete logarithm problem [Sma99]. So, while appealing, this idea cannot even be instantiated. Since we are stuck with $p \neq r$, we may consider doing "long arithmetic": simulating $\mathbb{F}_p$ operations via $\mathbb{F}_r$ operations, by working with bit chunks to perform the integer arithmetic, and reducing modulo p when needed. Alas, having to work at the "bit level" implies a blowup on the order of $\log p$ compared to the native arithmetic. So, while this approach can at least be instantiated, it is very expensive. A practical approach requires two different curves $E(\mathbb{F}_p)$ and $E'(\mathbb{F}_q)$ that are closely tied together (cf. Fig 1).

$$\begin{tikzcd}
\boxed{E'(\mathbb{F}_q)} \arrow[d, bend left=60, "\#E(\mathbb{F}_p) = q"] \\
\boxed{E(\mathbb{F}_p)} \arrow[u, bend left=60, "\#E'(\mathbb{F}_q) = p"]
\end{tikzcd}$$

Fig. 1. A cycle of elliptic curves.

Therefore, we need to find new pairing-friendly-curve constructions for this problem. Indeed, given a curve equation over a finite field, computing its order can be done efficiently with the SEA algorithm, while given p and r, computing the parameters of the curve equation involves computing a class polynomial (e.g. with Sutherland software [Sut11,ES10]), and it is infeasible if the curve discriminant D is too large (say more than 20 decimal digits).

Ben-Sasson et al. [BCTV14a] presented the first practical setting of a recursive proof composition with a **cycle** of two MNT pairing-friendly elliptic curves [MNT01]. It becomes feasible for proofs generated from one curve to reason about proofs generated from the other curve, resulting in an unbounded number of recursion layers. To achieve this, one curve's order is the other curve's base field order and vice-versa (i.e. $\#E'(\mathbb{F}_q) = p$ and $\#E(\mathbb{F}_p) = q$). The two curves are necessarily of prime order, hence cannot admit an Edwards form. Moreover, the only known such curves have low embedding degrees (4 and 6) resulting in large base fields to achieve a standard security level. The authors proposed a pair of MNT curves with parameters of 298 bits which they estimated to meet the 80-bit security level in 2014.

Around approximately the same time, Costello et al. [CFH+15] built on this idea to obtain a bounded recursive proof composition using a 2-chain of two elliptic curves (cf. Fig 2), i.e. a BN curve E_1 (with seed $x = -(2^{62} + 2^{55} + 1)$ from [NAS+08]) defined over a 254-bit field $\mathbb{F}_p$ and a BW6-509 curve E_2 constructed using the Brezing–Weng method (BW) in a way that it has a prime subgroup order equal to p the field characteristic of BN's $\mathbb{F}_p$. This set of curves only allows a one-layer recursion. Note that both MNT4 and MNT6 have an FFT-friendly prime order ($2^{17} \mid p-1$, $2^{34} \mid q-1$) while Geppetto's BN and BW6 don't. Later on, [BCTV14a] authors found a new MNT4/6 pair with parameters of 753 bits which they estimated at the 128-bit security level. They shared the parameters of this latter pair with the Coda protocol [BMRS20] developers (now Mina) who used it to build a recursive SNARK-based light blockchain. They updated the pre-print with this pair of parameters only recently (2020).

$$\begin{tikzcd}
\boxed{E'(\mathbb{F}_q)} \\
\boxed{E(\mathbb{F}_p)} \arrow[u, "\#E'(\mathbb{F}_q) = h \cdot p"']
\end{tikzcd}$$

Fig. 2. A 2-chain of elliptic curves.

Few years after [BCTV14a], motivated by a new proof composition application (decentralized private computation), ZEXE [BCG+20] proposed a new chain of curves. It is similar to Geppetto's one although the citation was missing. The first curve is a BLS12-377 defined over a 377-bit field with a subgroup r of 253 bits ($2^{47} \mid r-1$) and, using the more rudimentary Cocks-Pinch method (CP), the second curve is a CP6-782 of embedding degree 6, defined over a 782-bit field and having a group order divisible by BLS12-377's base field size p. Based on this inner BLS12-377, El Housni and Guillevic [EHG20] proposed and alternative outer curve to CP6-782 that is much faster for a recursive SNARK [Gro16] generation and verification. They used a variant of the Brezing–Weng method to construct a BW6-761 curve of embedding degree 6 defined over a 761-bit field and enjoying properties for efficient implementation. Then the same authors [EHG22a] generalized the BW6-761 curve construction, the pairing formulas and the group operations to any BW6 curve defined on top of any BLS12 curve, forming a family of 2-chain pairing-friendly curves.

Up to this point, all the pairing-based SNARKs use the three pairing groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ whether for proving, verifying or setting the SNARK up. Therefore, all previously mentioned curves were constructed in order to optimize operations in the three different groups. Note that the most efficient pairing-based SNARKs have a trapdoored setup specific to the statement to prove (e.g. [Gro16]). Recently, a new kind of SNARKs was introduced, where the setup phase is not specific to a given statement but is rather universal in that sense. Groth et al. [GKM+18] proposed a universal SNARK with a single setup to prove all statements of a given bounded size. However, Sonic [MBKM19] is considered to be the first practical universal SNARK. This work inspired many researchers and practitioners who then came up with new and elegant universal constructions (e.g. AuroraLight [Gab19], PLONK [GWC19], Marlin [CHM+20]).

These universal constructions follow a design similar to Sonic. They use as a fundamental building block a **polynomial commitment** (PC) scheme. While there are different PC schemes with trade-offs, the KZG scheme [KZG10] remains the most efficient. It uses pairings and therefore its implementation requires a suitable pairing-friendly elliptic curve. Contrary to the previous pairing-friendly curves used in the SNARK context, KZG-based universal SNARKs need a curve optimized only for $\mathbb{G}_1$ arithmetic and pairings. In fact, on the one hand, in pairing-based SNARKS with a circuit-specific setup phase, the pairing is used to verify that some polynomial identities hold when evaluated at a secret point included in the trapdoored setup. In KZG-based universal, on the other hand, the pairing is used to open a polynomial commitment (and element in $\mathbb{G}_1$) to a field element, and the polynomial identities are verified in the field. This observation inspired us in [EHG22a], to investigate the use of Barreto–Lynn–Scott curves of embedding degree 24 (BLS24) to instantiate KZG-based universal SNARKs. At the 128-bit security level, the coefficient ring of the elements of BLS24 $\mathbb{G}_1$ is much smaller compared to BLS12. We proposed a BLS24-315 curve defined over a 315-bit field with a subgroup order r of 253 bits such that $2^{22} \mid r-1$. Moreover, similarly to the BLS12-BW6 chains, we characterized all 2-chains that can be formed with a BLS24 as an inner curve and BW6 as an outer curve. We short-listed few BW6 curves (e.g. BW6-633 and BW6-672) and looked into Cocks-Pinch curves of higher embedding degrees (CP8 and CP12) for a more conservative security. This work was implemented in the `gnark` ecosystem [BPEH+22] using PLONK.

Actually, the universality of these constructions comes from the nature of the polynomial commitment scheme. By swapping the KZG scheme for other PC schemes [BCC+16,VP19,BFS20], one gets new transparent (no setup) SNARKs [KPV19,BFS20] and new transparent recursive SNARKs as first proposed by Bowe, Grigg and Hopwood in [BGH19] (Halo) and then formalized and generalized in [BCMS20,BDFG21]. These recursive constructions build on the discrete logarithm based PC from [BCC+16] and Bulletproofs [BBB+18] and one might want to instantiate them with a non-pairing-friendly elliptic curve like the standardized Edwards curve `ed25519`. However, having an efficient proof recursion requires a **cycle** of elliptic curves as in [BCTV14a], hence the curves are of prime order, which means Edwards and Montgomery forms are not possible (as they require an even order). Although this time, the curves do not need to be pairing-friendly. To this end, Halo's authors derived an efficient cycle for SNARK implementation, namely the Tweedledum-Tweedledee cycle. Later, Hopwood proposed the more efficient Pasta cycle [Hop20]. Note that finding such cycles is much easier than finding pairing-friendly cycles. It was investigated previously in a different context by Stange and Silverman [SS11]. Finally, as a future work, Halo suggests using the same SNARK techniques with a hybrid cycle where one curve is pairing-friendly and the other is not. Thereafter, Hopwood proposed a hybrid cycle, Pluto-Eris [Hop21]. Such a cycle can be constructed from any prime-order pairing-friendly curve (e.g. BN [BN06], Freeman [Fre10], MNT [MNT01]).

SNARKs enable verifying non-deterministic polynomial time (NP) computations with substantially lower complexity than those required for classical NP verification. In many applications (e.g. privacy-preserving cryptocurrencies, zk-rollups, verifiable computation outsourcing), the NP-computation revolves around proving the knowledge of a hash pre-image or the verification of a cryptographic signature. In CØCØ [KZM+15], Kosba et al. proposed a library of cryptographic primitives that can be efficiently proved in a SNARK. In particular, the authors looked into proving an Elliptic Curve Diffie-Hellman (ECDH) key exchange. They constructed a new elliptic curve to efficiently implement the operation required in key exchanges, i.e. the scalar multiplication (cf. Fig. 3).

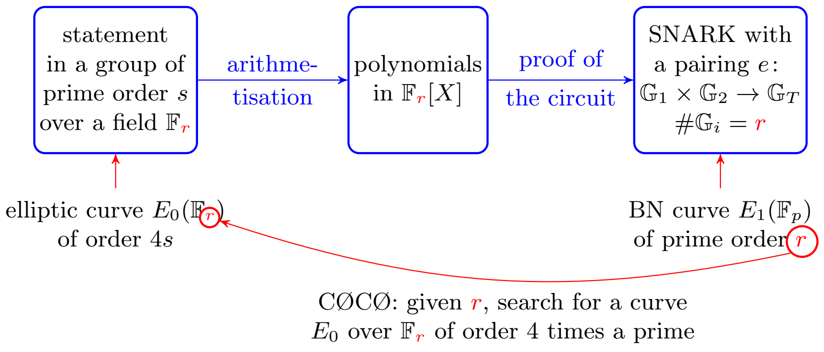
statement in a group of prime order s over a field $\mathbb{F}_r$
arithmetisation
polynomials in $\mathbb{F}_r[X]$
proof of the circuit
SNARK with a pairing $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ $\#\mathbb{G}_i = r$
elliptic curve $E_0(\mathbb{F}_r)$ of order $4s$
BN curve $E_1(\mathbb{F}_p)$ of prime order r
CØCØ: given r, search for a curve E_0 over $\mathbb{F}_r$ of order 4 times a prime

Fig. 3. Kosba et al. construction [KZM+15]

This curve defined over the scalar field $\mathbb{F}_r$ of the BN curve, with the seed `0x44e992b44a6909f1` from [BCTV14a], was given in Montgomery form for further optimizing the arithmetic inside a SNARK. The paper also mentioned converting this associated curve to Edwards form in order to efficiently prove EdDSA signatures. Following this work, the Zcash team introduced the JubJub curve [ZCa21] which is a similar curve in twisted Edwards form associated to BLS12-381, alongside further algebraic optimizations. This curve allowed Zcash to efficiently implement a collision-resistant variant of an EC-based Pedersen hash inside a SNARK. Practitioners who proposed tailored elliptic curves for SNARKs (BN254, BLS12-377, BLS24-315, BW6-761, BW6-633) each time also proposed a twisted Edwards associated curve defined over the scalar field. Finally, Masson et al. [MSZ21] performed an exhaustive search of associated curves over BLS12-381 with small Complex Multiplication (CM) discriminant in order to speed up the curve arithmetic using a fast endomorphism. They found an isolated curve with CM discriminant $D = 8$, called Bandersnatch. A similar associated curve is unlikely to be found for other SNARK curves of interest.

We stress that the problem solved in the CØCØ construction is partway similar to the one solved in the Geppetto construction. Remember that in a proof composition, one requires writing efficiently a pairing computation as a SNARK statement and to this end, researchers came up with new elliptic curves that efficiently encode the pairing in the SNARK (cf. Fig. 4). In CØCØ, the SNARK

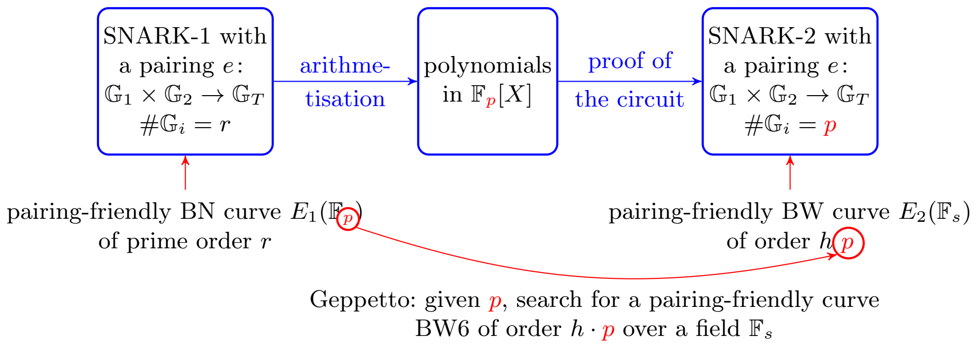
SNARK-1 with a pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ $\#\mathbb{G}_i = r$
arithme-tisation
polynomials in $\mathbb{F}_p[X]$
proof of the circuit
SNARK-2 with a pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ $\#\mathbb{G}_i = p$
pairing-friendly BN curve $E_1(\mathbb{F}_p)$ of prime order r
pairing-friendly BW curve $E_2(\mathbb{F}_s)$ of order $h \cdot p$
Geppetto: given p, search for a pairing-friendly curve BW6 of order $h \cdot p$ over a field $\mathbb{F}_s$

Fig. 4. Geppetto construction [CFH+15]

curve is fixed (BN $E_1(\mathbb{F}_p)$ of order r) and the authors look for a curve $E_0(\mathbb{F}_r)$ for the statement (ECDH). In Geppetto, the statement is "the verification of a previously generated proof" and its curve is fixed (BN $E_1(\mathbb{F}_p)$ of order r) as it was already used to generate that previous proof. The authors look for a new SNARK curve (a BW6 $E_2(\mathbb{F}_s)$ of order $h \cdot p$) to prove "the proof composition" statement.

3 Mathematical preliminaries

3.1 Background on pairings

We give the definition of the Tate pairing and present its computation, and one optimisation called ate pairing used in practice. All elliptic curves discussed thereafter are *ordinary* (i.e. non-supersingular).

Let E be an elliptic curve defined over a field $\mathbb{F}_p$, where p is a prime (or a prime power) of characteristic larger than 3. In this case, E can be given by a short Weierstrass equation $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_p$. The set of points $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$ satisfying this equation is denoted $E(\mathbb{F}_p)$, and E is equipped with a group law given by the chord-and-tangent rule (see e.g. [Cos12]). Denote $\mathcal{O}$ the point at infinity on the curve and neutral element of the group law. Let π_p be the Frobenius endomorphism: $(x, y) \mapsto (x^p, y^p)$. Its minimal polynomial is $\chi(X) = X^2 - tX + p$ where t is called the *trace*, and is bounded by the Hasse–Weil bound $|t| \le 2\sqrt{p}$. The curve order is $\#E(\mathbb{F}_p) = \chi(1) = p + 1 - t$. Let r be a prime divisor of the curve order. A point P on E such that $[r]P = \mathcal{O}$ is a point of r*-torsion.* The r-torsion subgroup of E is denoted $E[r] \coloneqq \{P \in E(\overline{\mathbb{F}_p}), [r]P = \mathcal{O}\}$ and has order r^2. Following Hess and Vercauteren [Ver10], the two eigenspaces of π_p in $E[r]$ define two (orthogonal) subgroups of order r that are useful for pairing applications. The two groups are $\mathbb{G}_1 = E[r] \cap \ker(\pi_p - [1])$ with a generator denoted by G_1, and $\mathbb{G}_2 = E[r] \cap \ker(\pi_p - [p])$ with a generator G_2. The group $\mathbb{G}_2$ is defined over $\mathbb{F}_{p^k}$, where the *embedding degree* k is the smallest integer $k \in \mathbb{N}^*$ such that $r \mid p^k - 1$. It means that a pairing computation involves arithmetic operations over a degree-k extension $\mathbb{F}_{p^k}$, and for efficiency reasons, k should be reasonably small ($k \leq 24$ in our use cases).

We recall the Tate and ate pairing definitions, based on the same two steps: evaluating a function $f_{s,Q}$ at a point P, the Miller loop step, and then raising it to the power $(p^k - 1)/r$, the final exponentiation step. Given P, Q on E of order r, the Tate pairing is expressed in terms of the rational function $f_{r,Q}$ evaluated at the point P. This function lies in $\mathbb{F}_{p^k}[x, y]/(y^2 - x^3 - ax - b)$, the elliptic curve function field. It has a zero of order r at Q and a pole (zero at the denominator) of order r at $\mathcal{O}$, that translates into having a *divisor* $\operatorname{div}(f_{r,Q}) = r(Q) - r(\mathcal{O})$, a formal sum of points $\{Q, \mathcal{O}\}$ on the curve encoding for the zeros and poles with their multiplicities as coefficients (here r). It is impossible to compute directly that function, hence Miller's algorithm is used instead. Miller's algorithm considers an auxiliary rational function $f_{s,Q}$ whose *divisor* is $\operatorname{div}(f_{s,Q}) = s(Q) - ([s]Q) - (s-1)(\mathcal{O})$ that is, a zero of order s at the point Q, a pole of order 1 at sQ and a pole of order $s - 1$ at $\mathcal{O}$. Multiplying, resp. quotienting two functions means adding, resp. subtracting their divisors. One derives the property, 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 intersecting the two points and v the vertical). We compute $f_{s,Q}(P)$ with the Miller loop presented in Algorithm 1. The line $\ell_{S,S}$ corresponds to the tangent line at S involved in doubling S, $\ell_{S,Q}$ is the line through the (distinct) points S, Q needed in the addition $S + Q$, and v_{2S}, v_{S+Q} are the vertical lines through $-2S, 2S$ and $-(S + Q), S + Q$ resp. For the interested reader, we refer to the tutorial [Cos12] on pairings.

```
Algorithm 1: MillerLoop($s, P, Q$)
Output: $m = f_{s,Q}(P)$ for $s = \sum_{i=0}^{t} s_i 2^i$
1  $m \leftarrow 1$; $S \leftarrow Q$;
2  for $b$ from $t - 1$ to $0$ do
3      $\ell \leftarrow \ell_{S,S}(P)$; $S \leftarrow [2]S$;            // DoubleLine
4      $v \leftarrow v_{[2]S}(P)$;                                   // VerticalLine
5      $m \leftarrow m^2 \cdot \ell / v$;                            // Update1
6      if $s_b = 1$ then
7          $\ell \leftarrow \ell_{S,Q}(P)$; $S \leftarrow S + Q$;       // AddLine
8          $v \leftarrow v_{S+Q}(P)$;                                // VerticalLine
9          $m \leftarrow m \cdot \ell / v$;                          // Update2
10 return $m$;
```

The Tate and ate pairings are defined by

$$\begin{aligned}\mathrm{Tate}(P,Q) &\coloneqq f_{r,P}(Q)^{(p^k-1)/r}\\ \mathrm{ate}(P,Q) &\coloneqq f_{t-1,Q}(P)^{(p^k-1)/r}\end{aligned}$$

where $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$. It is also important to recall some results with respect to the 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 $E'[r](\mathbb{F}_{p^{k/d}})$ for some twist E'. Otherwise, in the general case, E admits a single twist (up to isomorphism) and it is of degree 2.

3.2 Some pairing-friendly constructions

We recall some methods and families from the literature related to pairing-friendly constructions that will be of interest in the following sections. A detailed study of these constructions is available in the taxonomy paper by Freeman, Scott and Teske [FST10]. We focus on the Cocks–Pinch [FST10, Sec. 4.1] (CP) and Brezing–Weng [FST10, Sec. 6.1] (BW) methods, and Barreto–Lynn–Scott [FST10, Sec. 6.6] of embedding degrees 12 and 24 (BLS12 and BLS24), Barreto–Naehrig [BN06] (BN), Miyaji–Nakabayashi–Takano [FST10, Sec. 5.1] of embedding degrees 4 and 6 (MNT4 and MNT6), Galbraith–McKee-Valença [FST10, Sec. 5.2] of embedding degree 6 and cofactor 4 families and Kachisa–Schaefer–Scott [KSS08] of embedding degrees 16 and 18 (KSS16 and KSS18) families.

From a high level perspective, there are two ways to obtain pairing-friendly curves. Generic algorithms take as inputs k and r, and output (if it exists) an elliptic curve defined over a field $\mathbb{F}_p$ and of embedding degree k w.r.t. a subgroup of prime order r over $\mathbb{F}_p$. If r is an integer, this is the Cocks–Pinch method, if r is a polynomial, this is the Brezing–Weng method. The alternative is to consider precomputed tables of polynomial families $(k, r(x), D, t(x), p(x))$ as in [FST10]. To rank the families of the same k, the rho-value is defined as the ratio of the sizes of p and r, resp. the ratio of the degrees of the polynomials $p(x)$ and $r(x)$:

$$\rho = \frac{\log p}{\log r}, \quad \text{resp.} \quad \rho = \frac{\deg p(x)}{\deg r(x)} \tag{1}$$

and because $r \mid p + 1 - t$, then $\rho \geq 1$.

Cocks–Pinch is the most flexible method and can be used to construct a curve $E(\mathbb{F}_p)$ with an arbitrary embedding degree and a subgroup order r such that the ratio $\rho = \log_2 p / \log_2 r$ is approximately 2. It works by fixing r and the CM discriminant D and then computing the trace t and the prime p such that the CM equation $4p = t^2 + Dy^2$ (for some $y \in \mathbb{Z}$) is satisfied (cf. Alg. 2).

Barreto, Lynn and Scott [FST10, Sec. 6.1] and later Brezing and Weng [FST10, Sec. 6.1] generalized the Cocks–Pinch method by parameterizing t, r and p as polynomials. This led to curves with a ratio $\rho < 2$. In Alg. 3, we sketch the idea of the algorithm that works both for BLS and BW constructions (cf. Alg. 3).

Algorithm 2: Cocks–Pinch method

```
Input: A positive integer $k$ and a positive square-free integer $D$
Output: $E/\mathbb{F}_p$ with an order-$r$ subgroup and embedding degree $k$
1  Choose a prime $r$ such that $k$ 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 $y = (t-2)/\sqrt{-D}$ mod $r$;
4  Lift $t$ and $y$ in $\mathbb{Z}$;
5  Compute $p = (t^2 + Dy^2)/4$ in $\mathbb{Q}$;
6  if $p$ is a prime integer then
7      Use CM method ($D < 10^{20}$) to construct $E/\mathbb{F}_p$ with order-$r$ subgroup;
8  else
9      Go back to 1;
10 return $E/\mathbb{F}_p$ with an order-$r$ subgroup and embedding degree $k$
```

Particular choices of polynomials for $k = 12$ and $k = 24$ yield two families of curves with good security/performance tradeoffs, denoted respectively BLS12 and BLS24. The parameters are given in Table 1.

Algorithm 3: Idea of BLS and BW methods

```
Input: A positive integer $k$ and a positive square-free integer $D$
Output: $E/\mathbb{F}_{p(x)}$ with an order-$r(x)$ subgroup and embedding degree $k$
1  Choose an irreducible polynomial $r(x) \in \mathbb{Z}[x]$ with positive leading coefficient such that
   $\sqrt{-D}$ and the primitive $k$-th root of unity $\zeta_k$ 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 $p(x) = (t^2(x) + Dy^2(x))/4$ in $\mathbb{Q}[x]$;
5  while $p(x)$ is not irreducible do
6      Go back to 1;
7  return $E/\mathbb{F}_{p(x)}$ with an order-$r(x)$ subgroup and embedding degree $k$
```

MNT curves, however, have a fixed embedding degree $k \in \{3, 4, 6\}$ and a variable discriminant D. They are also parameterized by polynomials and form a sparse family (cf. Table 1) because one is required to solve a generalized Pell equation. Karabina and Teske [KT08] showed that there is a prime-order elliptic curve $E/\mathbb{F}_p$ from the MNT6 family if and only if there is a prime-order elliptic curve $E'/\mathbb{F}_q$ from the MNT4 family such that $\#E(\mathbb{F}_p) = q$ and $\#E'(\mathbb{F}_q) = p$.

GMV curves extend MNT curves with cofactors $2 \leq h \leq 5$. This is achieved by following the MNT strategy and substituting $h \cdot r$ for $\#E(\mathbb{F}_p)$ followed by an explicit analysis of cases $h = 2, 3, 4$ and 5. Polynomial parameters for GMV with $k = 6$ and $h = 4$ are given in Table 1.

BN curves form a family of prime-order pairing-friendly elliptic curves with $k = 12$ and $D = 3$ (cf. Table 1). The construction is based on a result from [GMV07] and a lucky try in which the right-hand side of the CM equation happens to be a constant times a perfect square polynomial. However, it was suggested in [FST10, Example 6.8] that the BN construction can be viewed as a complete family on its own.

Another strategy to build pairing-friendly constructions is to pick random small elements and take their minimal polynomials as the subgroup order polynomial $r(x)$. For well chosen embedding degrees $k = 16$ and $k = 18$, this yields the KSS16 and KSS18 families with $\rho = 5/4$ and $\rho = 4/3$ respectively (cf. Table 1).

Table 1. Polynomial parameters of BN, BLS12, MNT4, MNT6, GMV6, KSS16 and KSS18 families. The bold family of GVM06 is used in [BCG$^+$13].

Family	k	D	ρ	$r(x)$	$p(x)$	$t(x)$
BN	12	-3	1	$36x^4 + 36x^3 + 18x^2 + 6x + 1$	$36x^4 + 36x^3 + 24x^2 + 6x + 1$	$6x^2 + 1$
BLS12	12	-3	3/2	$x^4 - x^2 + 1$	$(x^6 - 2x^5 + 2x^3 + x + 1)/3$	$x + 1$
BLS24	24	-3	5/4	$x^8 - x^4 + 1$	$(x^{10} - 2x^9 + x^8 - x^6 + 2x^5 -$ $x^4 + x^2 + x + 1)/3$	$x + 1$
MNT4	4	D	1	$x^2 + 1$ or $x^2 + 2x + 2$	$x^2 + x + 1$	$-x$ or $x + 1$
MNT6	6	D	1	$4x^2 \pm 2x + 1$	$4x^2 + 1$	$1 \pm 2x$
GMV6 $(h = 4)$	6	D	1	$4x^2 + 2x + 1 = \Phi_6(t-1)$ $28x^2 + 10x + 1 = \Phi_6(t-1)/7$ $28x^2 + 18x + 3 = \Phi_6(t-1)/7$ $\boldsymbol{52x^2 + 14x + 1} = \Phi_6(t-1)/13$ $52x^2 + 38x + 7 = \Phi_6(t-1)/13$	$16x^2 + 10x + 5$ $112x^2 + 54x + 7$ $112x^2 + 86x + 17$ $\boldsymbol{208x^2 + 30x + 1}$ $208x^2 + 126x + 19$	$2x + 2$ $14x + 4$ $14x + 6$ $\boldsymbol{-26x - 2}$ $-26x - 8$
KSS16	16	-4	5/4	$(x^9 + 48x^4 + 625)/61550$	$(x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 +$ $240x^4 + 625x^2 + 2398x + 3125)/980$	$(2x^5 + 41x + 35)/35$
KSS18	18	-3	4/3	$(x^6 + 37x^3 + 343)/343$	$(x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 +$ $259x^3 + 343x^2 + 1763x + 2401)/21$	$(x^4 + 16x + 7)/7$

3.3 zk-SNARKs

In this survey, we mainly focus on zkSNARKs for NP languages for which we give a basic overview. Given a public NP program F, public inputs a and b and a private input w, such that the program F satisfies the relation $F(a, w) := b$, a zk-SNARK consists in proving this relation succinctly without revealing the private input w. Given a security parameter λ, it consists of the `Setup`, `Prove` and `Verify` algorithms:

$$\begin{aligned}(\sigma_p, \sigma_v) &\leftarrow \texttt{Setup}(F, 1^\lambda)\\ \pi &\leftarrow \texttt{Prove}(a, b, w, \sigma_p)\\ 0/1 &\leftarrow \texttt{Verify}(a, b, \pi, \sigma_v)\end{aligned}$$

where σ_p is the proving key which encodes the program F for the prover, σ_v the verification key that encodes F for the verifier and π the proof. If the `Setup` algorithm is trapdoored an additional secret input τ is required $(\sigma_p, \sigma_v) \leftarrow \texttt{Setup}(F, \tau, 1^\lambda)$.

Table 2. Cost of `Setup`, `Prove` and `Verify` algorithms for [Gro16] and PLONK. m = number of wires, n = number of multiplication gates, a = number of addition gates and ℓ = number of public inputs. $\mathtt{M}_{\mathbb{G}}$ = multiplication in $\mathbb{G}$ and $\mathtt{P}$=pairing. *Note:* Both Groth16 and PLONK verifiers have a dependency on the number of public inputs ℓ, but for PLONK it is just a polynomial evaluation (FFT).

	`Setup`	`Prove`	`Verify`
[Gro16]	$3n\ \mathtt{M}_{\mathbb{G}_1}$ $m\ \mathtt{M}_{\mathbb{G}_2}$	$(3n+m-\ell)\ \mathtt{M}_{\mathbb{G}_1}$ $n\ \mathtt{M}_{\mathbb{G}_2}$	$3\ \mathtt{P}$ $\ell\ \mathtt{M}_{\mathbb{G}_1}$
PLONK (KZG)	$d\ (\geq n+a)\ \mathtt{M}_{\mathbb{G}_1}$ $1\ \mathtt{M}_{\mathbb{G}_2}$	$9(n+a)\ \mathtt{M}_{\mathbb{G}_1}$	$2\ \mathtt{P}$ $18\ \mathtt{M}_{\mathbb{G}_1}$

4 SNARKs with pairing-friendly elliptic curves

Pairing-based SNARKs can be instantiated with any secure pairing-friendly elliptic curve. For efficiency, we require a fast arithmetic in $\mathbb{F}_r$ (where r is the curve prime subgroup order), in $\mathbb{G}_1$ and in $\mathbb{G}_2$, and a fast pairing computation. For security, we are interested in the 128-bit security level in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.

4.1 Efficiency

The `Setup` and `Prove` algorithms involve solving multiple large instances of tasks about polynomial arithmetic in $\mathbb{F}_r[X]$ and multi-scalar multiplication (MSM) over $\mathbb{G}_1$ and $\mathbb{G}_2$. The `Verify` algorithm involves computing a product of pairings and an evaluation in $\mathbb{G}_T$. Fast arithmetic in $\mathbb{F}_r[X]$, when manipulating large-degree polynomials, is best implemented using the Fast Fourier Transform (FFT) [Pol71] and MSMs of large sizes are best implemented using a variant of Pippenger's algorithm [BDLO12, Section 4]. For example, Table 2 reports the numbers of MSMs required in the `Setup`, `Prove` and `Verify` algorithms in the [Gro16] SNARK and the KZG-based PLONK universal SNARK [GWC19]. The report excludes the number of FFTs as it changes from one implementation to another, but these usually consume less compute time than group operations.

An efficient implementation of the FFT over $\mathbb{F}_r$ requires $r-1$ to split into small primes or ideally to be divisible by a large power of 2. This narrows the search of pairing-friendly elliptic curves to the ones with a subgroup order r such that $2^L \mid r-1$ for a large integer $L \geq 1$. This was suggested first in [BCG$^+$13] and yielded the GMV6-183 curve with $2^{31} \mid r-1$.

Recently, a new algorithm was proposed by Ben-Sasson et al. [BCKL21] that implements a variant of the FFT over arbitrary fields $\mathbb{F}_{p'}$ where $p'-1$ is not smooth (ECFFT). However, this algorithm suffers from two major drawbacks: finding an elliptic curve over $\mathbb{F}_{p'}$ with a large 2^L-torsion and efficiently implementing the algorithm in the canonical basis. Since the new algorithm is asymptotically much slower than the basic FFT using favorable fields, designing a SNARK curve with smooth $r-1$ remains the best approach.

This narrows the search to an elliptic curve with the following requirements:

(i) valid parameters: integers p, r, t $(p(u), r(u) \in \mathbb{N}, t(u) \in \mathbb{Z})$, prime p.
(ii) a subgroup order r such that $2^L \mid r - 1$ for a large integer $L \geq 1$,
(iii) a fast pairing,
(iv) a fast arithmetic in $\mathbb{G}_1$ and
(v) a fast arithmetic in $\mathbb{G}_2$.

For KZG-based universal SNARKs, the last requirement can be dropped.

Elliptic curves proposed in the literature to suit SNARKs belong to one of the families discussed in the preliminaries in section 3. Except for the Cocks-Pinch curve CP6-782, all the curves are in families defined by polynomials. Condition (ii) becomes a congruence condition on the seed x of the polynomial $r(x)$ (cf. Table 3).

Pairing computations take place mainly in $\mathbb{F}_{p^k}$ and it is important to construct efficiently the $\mathbb{F}_{p^k}$ tower. Pairing-friendly tower extensions are built using a sequence of quadratic and cubic sub-extensions. The ideal way is to start with an optimal quadratic extension $\mathbb{F}_{p^2} = \mathbb{F}_p[u]/(u^2 + 1)$ that arises when $p \equiv 3 \bmod 4$. For the discussed families, satisfying condition (iii) boils down then to satisfying a congruence equation in the seed x alongside the previous congruence condition (cf. Table 3). Note that even seeds allow an additional speedup in the pairing computation (final exponentiation) for some families (e.g. BLS12 and BLS24 [GF16]). Finally, as is the custom in pairing-based cryptography, the Miller loop scalar (e.g. $u = t - 1$ for BLS curves) should have small Hamming weight. For the curve families we are interested in, the optimal Miller loop scalar is a low-degree polynomial in the seed x, hence we are additionally looking for a sparse seed x in (signed) binary representation.

Condition (iv) is mainly related to the bitsize of p as the point coordinates are in $\mathbb{F}_p$. The bitsize of p varies for each family and is constrained by the security. One requires r of about 256 bits, and each family has a fixed parameter $\rho = \log_2 p / \log_2 r$, $1 \leq \rho \leq 2$, hence p is usually (but not always) larger than r.

Except for KZG-based universal SNARKs, one requires also a fast arithmetic in $\mathbb{G}_2$. As discussed in the preliminaries, the curves from Table 1 have a twist of degree $d = 2, 4$ or 6 and thus $\mathbb{G}_2$ is isomorphic to $E'[r](\mathbb{F}_{p^{k/d}})$ for an appropriate d-twist E'. Condition (v) is immediately related to the choice of k and d (cf. Tab 3).

Congruence conditions on the seed x to achieve (ii). The subgroup order r is given by an irreducible polynomial $r(x)$ in $\mathbb{Q}[x]$. Computing the congruence conditions on x_0 such that $2^L \mid r(x_0) - 1$ given L is equivalent to finding the roots x_0 of $r(x) - 1$ modulo 2^L. Because 2 is a prime, we define in Alg. 4 a Hensel-like root-lifting technique inspired by [GS21, §A], with auxiliary functions in Algs. 5 and 6. Hensel lifting is well-known for simple roots x_0 of a polynomial $f(x)$ (where x_0 is a zero of multiplicity one of $f(x)$ modulo a prime), but it becomes intricate when dealing with *multiple roots*, that are zeros of higher multiplicities, in other words, when $f(x_0) = f'(x_0) = 0$ modulo a prime.

For BN curves, one has $r(x) - 1 = 6x(6x^3 + 6x^2 + 3x + 1)$. With $2^{L-1} \mid x$, then immediately $2^L \mid r - 1$. The other option is $2^{L-1} \mid 6x^3 + 6x^2 + 3x + 1$

[1] For GMV6 and KSS18, one needs to construct respectively the $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{18}}$ towers starting from $\mathbb{F}_{p^3}$. In this case, one looks for the smaller cubic non-residue in $\mathbb{F}_p$.

and we run Alg. 4 (we were not able to derive a generic formula). Note that $\gcd(p(x)-1, r(x)-1) = 6x$. For BLS12 and BLS24 curves, $\gcd(p(x)-1, r(x)-1) = x-1$. For BLS12, $r(x)-1 = x^2(x^2-1)$ and for BLS24, $r(x)-1 = x^4(x^4-1)$ and the results of Table 3 follow. For KSS16, we did not obtain a generic formula and we derived the condition for $L = 64$ in Table 3 with Alg. 4. Note that for curves with $D = 1$, $p \equiv 1 \bmod 4$ is required otherwise the curve would be supersingular. For KSS18 curves, Alg. 4 outputs another congruence but p is always even in that case so we discarded it. For BN and KSS16 we picked $L = 64$ and obtained the conditions $x_0 \equiv u_0 \bmod c \cdot 2^{L-1}$ but for a smaller $L_0 < L$, the reduction $u_0 \bmod c \cdot 2^{L_0-1}$ gives the answer. Algorithms 4, 5 and 6 are implemented in SageMath at `https://gitlab.inria.fr/tnfs-alpha/alpha`, in the file `sage/tnfs/gen/generate_curve_utils.py`.

4.2 Security

We look at the security of all the pairing-friendly families of the proposed curves both from a generic point of view (TNFS attack) and a SNARK-specific point of view (Cheon's attack).

TNFS attack. In 2015, [BGK15] revisited the Tower-NFS algorithm (TNFS) to compute discrete logarithms in $\mathbb{F}_{p^k}$. Then in 2016, Kim with Barbulescu combined it with other variants of NFS and exploited the extension fields to improve the TNFS algorithm. This resulted in an expected asymptotic complexity $L_Q(\alpha, c) = \exp((c+o(1))(\log Q)^{\alpha}(\log\log Q)^{1-\alpha})$ to be $L_{p^k}(1/3, (48/9)^{1/3} \approx 1.747)$ instead of $L_{p^k}(1/3, (96/9)^{1/3} \approx 2.201)$ with the NFS-HD algorithm of [BGGM15]. More important, the complexity of TNFS is lower than the general NFS algorithm in prime fields: $L_{p^k}(1/3, (64/9)^{1/3} \approx 1.923)$. The keysizes should be enlarged, and several papers deal with security estimates of TNFS [MSS16,BD19,GS21,Gui20,DGP20]. However these papers *estimate* the security level, but they do not scale with respect to a record computation. This is not yet possible as the first record computation with the TNFS algorithm was published after them, in [DGP21]: it runs the TNFS in a field $\mathbb{F}_{p^6}$ but new security estimates are not extrapolated. In Table 4 we reproduce the keysizes from [GS21,Gui21].

For pairing-friendly curves with parameters given by polynomials evaluated at some seed, the *Special*-TNFS algorithm applies. It exploits the special form of the prime p, resulting in an asymptotic complexity of $L_{p^k}(1/3, (32/9)^{1/3} \approx 1.526)$ in the most favorable case. Is means that compared to prime fields $\mathbb{F}_q$, the total size $k \log p$ should be twice larger to ensure the same level of security: $k \log p = 2 \log q$. For MNT and GMV curves, p is given by a quadratic polynomial and the special variant of TNFS is not better than the generic methods (the degree of $p(x)$ should be at least 3 to make a difference). By the way, a change of variables $v = -26x - 3$ gives $t(v) = v + 1$, $r(v) = (v^2 - v + 1)/13$ and $p(v) = 4r(v) + t(v) - 1 = (4v^2 + 9v + 4)/13$ with smaller coefficients. We can reasonably assume that the sizes required for GMV6 curves are the same as for MNT6 curves. Such sizes were given in [Gui21, § MNT Curves]. We run the

Table 3. Conditions (i), (ii), (iii), and (v) for Table 1 families. For BN curves with $p \equiv 3 \bmod 4$ and KSS16 curves, it was not possible to obtain a general rule. The residue of $x \bmod 2^L$ is computed by Alg. 4 with input $L = 64$ but any L can be given. For KSS18 curves, the other residues x do not give a prime p, and condition (iii) is not possible.

Family, (i) $r, p \in \mathbb{N},\ t \in \mathbb{Z}$	(ii) $r \equiv 1 \bmod 2^L$	(iii) $p \equiv 3$ mod 4	(v) $\mathbb{G}_2$ coord. in
BN	$x \equiv 2570880382155688433 \bmod 2^{63} \Rightarrow 2^{64} \mid r-1$	✓	$\mathbb{F}_{p^2}$
any x	$x \equiv 0 \bmod 2^{L-1} \Rightarrow 2^L \mid r-1,\ 2^L \mid p-1$	✗	
BLS12	$x \equiv 1 \bmod 3 \cdot 2^{L-1} \Rightarrow 2^L \mid r-1,\ 2^{L-1} \mid p-1$	✗	
$x \equiv 1$	$x \equiv 2^{L-1} - 1 \bmod 3 \cdot 2^{L-1} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	$\mathbb{F}_{p^2}$
mod 3	$x \equiv 2^{L/2} \bmod 3 \cdot 2^{L/2} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	
BLS24	$x \equiv 1 \bmod 3 \cdot 2^{L-2} \Rightarrow 2^L \mid r-1,\ 2^{L-2} \mid p-1$	✗	
$x \equiv 1$	$x \equiv 2^{L-1} - 1 \bmod 3 \cdot 2^{L-2} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	$\mathbb{F}_{p^4}$
mod 3	$x \equiv 2^{L/4} \bmod 3 \cdot 2^{L/4} \Rightarrow 2^L \mid r-1,\ 6 \mid p-1$	✓	
MNT4, $t = x + 1$	$x \equiv 0 \bmod 2^{L/2} \Rightarrow 2^L \mid r-1, 2^{L/2} \mid p-1$	✗	$\mathbb{F}_{p^2}$
MNT6	$x \equiv 0 \bmod 2^{L-1} \Rightarrow 2^L \mid r-1, 2^{2L} \mid p-1$	✗	$\mathbb{F}_{p^3}$
GMV6$(h = 4)$ any x	$x \equiv 0 \bmod 2^{L-1} \Rightarrow 2^L \mid r-1,\ 2^{L-1} \mid p-1$	NA[1]	$\mathbb{F}_{p^3}$
KSS16 $(x \equiv \pm 25 \bmod 70)$	$\pm 14398186520986421885, \pm 37456616613123361405 \bmod 35 \cdot 2^{62} \Rightarrow 2^{64} \mid r-1,\ p \equiv 1 \bmod 4$	✗	$\mathbb{F}_{p^4}$
KSS18 $(x \equiv 14 \bmod 42)$	$x = 14 \cdot 2^{L/3} \bmod 42 \cdot 2^{L/3} \Rightarrow 2^L \mid r-1,\ 12 \mid p-1$	NA[1]	$\mathbb{F}_{p^3}$

```
Algorithm 4: Congruence condition
Input: polynomial $s(x) \in \mathbb{Q}[x]$, modulus $m$ and congruence conditions $\{a_i\}_{i \geq 0}$ such
       that $x_0 \equiv a_i \bmod m \Rightarrow s(x_0) \in \mathbb{Z}$, prime integer $\ell$, integer $L > 0$
Output: Residues $u_j$, integers $L_j$ and moduli $m_j$ s.t. for all $x_j \equiv u_j \bmod m_j$,
        $s(x_j) \in \mathbb{Z}$ and $s(x_j) \equiv 0 \bmod \ell^{L_j}$, $L_j \geq L$
for $a_i \in \{a_i\}_{i \geq 0}$:
    $s_i(x) \leftarrow s(m \cdot x + a_i) \in \mathbb{Z}[x]$ (this ensure that $s_i(x)$ has integer coefficients)
    $v_i \leftarrow \mathrm{valuation}_\ell(\mathrm{content}(s_i(x)))$
    $s_i(x) \leftarrow s_i(x)/\ell^{v_i}$
    for $r_j \in \mathbb{Z}/\ell\mathbb{Z}$ a simple root of $s_i(x)$ modulo $\ell$:
        $r_j \leftarrow \mathrm{lift}_\mathbb{N}(r_j)$
        $(u_j, m_j, L_j) \leftarrow$ lift_simple_root$(s_i, r_j, a_i + r_j \cdot m, m \cdot \ell, 1 + v_i, \ell, L)$
        $S \leftarrow S \cup \{(u_j, m_j, L_j)\}$
    for $r_j \in \mathbb{Z}/\ell\mathbb{Z}$ a multiple root of $s_i(x)$ modulo $\ell$:
        $r_j \leftarrow \mathrm{lift}_\mathbb{N}(r_j)$
        $S_j \leftarrow$ lift_multiple_root$(s_i, r_j, a_i + r_j \cdot m, m \cdot \ell, 1 + v_i, \ell, L)$
        $S \leftarrow S \cup S_j$
return $S$
```

Algorithm 5: Hensel lifting of simple roots

Input: polynomial $s_i(x) \in \mathbb{Z}[x]$, modulus m_i, residue u_i mod m_i, root $r_i \in \mathbb{N}$ s.t. $s_i(r_i) \equiv 0 \bmod \ell$, prime integer ℓ, integers L_i and bound $L > 0$

Output: Residue u_j, integer L_j and modulus m_j s.t. for all $x_j \equiv u_j \bmod m_j$, $s_i(x_j) \equiv 0 \bmod \ell^{L_j - L_i}$

```
def lift_simple_root($s_i, r_i, u_i, m_i, L_i, \ell, L$):
    while $L_i < L$:
        $s_i(x) \leftarrow s_i(\ell \cdot x + r_i)/\ell$
        $L_i \leftarrow L_i + 1$
        $r_i \leftarrow \mathrm{root}_{\mathbb{Z}/\ell\mathbb{Z}}(s_i(x))$        #(by Hensel, a simple root lifts to one root only)
        $u_i \leftarrow u_i + \mathrm{lift}_{\mathbb{N}}(r_i) \cdot m_i$
        $m_i \leftarrow \ell \cdot m_i$
    return $(u_i, m_i, L_i)$
```

Algorithm 6: Hensel lifting of multiple roots

Input: polynomial $s_i(x) \in \mathbb{Q}[x]$, modulus m_i, residue u_i mod m_i, multiple root r_i mod ℓ, prime integer ℓ, integer L_i and bound $L > 0$

Output: Residues u_j, integers L_j and moduli m_j s.t. for all $x_j \equiv u_j \bmod m_j$, $s_i(x_j) \equiv 0 \bmod \ell^{L_j - L_i}$

```
def lift_multiple_root($s_i, r_i, u_i, m_i, L_i, \ell, L$):
    $S \leftarrow [(s_i, r_i, u_i, m_i, L_i)]$ a linked list
    $R \leftarrow \{\}$
    while $S$ is not empty:
        $(s_i, r_i, u_i, m_i, L_i) \leftarrow$ pop($S$)
        $s_i(x) \leftarrow s_i(\ell \cdot x + r_i)/\ell$
        $v_i \leftarrow \mathrm{valuation}_\ell(\mathrm{content}(s_i(x)))$
        $s_i(x) \leftarrow s_i(x)/\ell^{v_i}$
        $L_i \leftarrow L_i + 1 + v_i$
        if $L_i \geq L$:
            $R \leftarrow R \cup \{(u_i, m_i, L_i)\}$
        else:
            for $r_i \in \mathbb{Z}/\ell\mathbb{Z}$ a simple root of $s_i(x)$ modulo $\ell$:
                $r_i \leftarrow \mathrm{lift}_{\mathbb{N}}(r_i)$
                $(u_j, m_j, L_j) \leftarrow$ lift_simple_root($s_i, r_i, u_i + r_i \cdot m_i, m_i \cdot \ell, L_i, \ell, L$)
                $R \leftarrow R \cup \{(u_j, m_j, L_j)\}$
            for $r_i \in \mathbb{Z}/\ell\mathbb{Z}$ a multiple root of $s_i(x)$ modulo $\ell$:
                $r_i \leftarrow \mathrm{lift}_{\mathbb{N}}(r_i)$
                $S \leftarrow$append($S, (s_i, r_i, u_i + r_i \cdot m_i, m_i \cdot \ell, L_i)$)
    return $R$
```

SageMath code of [GS21] and obtain a security estimate of 71 bits in GF(p^6) for the GMV6 curve (Table 4). Security levels for BLS12-377, CP6-782 and BW6-761 were provided in [EHG20, § C] (Table 5).

Table 4. Security level estimates of MNT curves from [Gui21], and of the GMV curve obtained with the software from [GS21]. The curves MNT-298 and GMV6-183 offer a very low security level and should be avoided.

curve	k	D	ref	r bits	p bits	p^k bits	DL cost in $\mathbb{F}_{p^k}$
MNT4-298	4	614144978799019	[BCTV14a]	298	298	1192	2^{77}
MNT6-298	6	614144978799019	[BCTV14a]	298	298	1788	2^{87}
MNT4-753	4	241873351932854907	[BCTV14a]	753	753	3012	2^{113}
MNT6-753	6	241873351932854907	[BCTV14a]	753	753	4517	2^{137}
MNT4-992	4	95718723	[Gui21]	992	992	3966	2^{126}
MNT6-992	6	95718723	[Gui21]	992	992	5948	2^{156}
GMV6-183	6	21048712401	[BCG+13]	181	183	1093	2^{71}

Table 5. Security level estimates of BLS curves, CP6-782, BW6 and CP8, CP12 curves from [EHG20,EHG22a], with seeds u_{377} = `0x8508c00000000001`, u_{379} = `0x9b04000000000001`, $u_{315} = -$`0xbfcfffff`, u_{317} = `0xe19c0001`.

curve	k	D	ref	r bits	p bits	p^k bits	DL cost in $\mathbb{F}_{p^k}$
BLS12-377, u_{377}	12	3	[BCG+20]	253	377	4521	2^{126}
BLS12-379, u_{379}	12	3	[EHG22a, Tab. 9]	254	379	4537	2^{126}
BLS24-315, u_{315}	24	3	[EHG22a, Tab. 10]	253	315	7543	2^{160}
BLS24-317, u_{317}	24	3	[EHG22a, Tab. 10]	255	317	7599	2^{160}
outer curve with a BLS12 curve							
CP6-782, u_{377}	6	339	[BCG+20]	377	782	4691	2^{138}
BW6-761, u_{377}	6	3	[EHG20]	377	761	4566	2^{126}
BW6-764, u_{379}	6	3	[EHG22a, Tab. 11]	379	764	4584	2^{126}
outer curves with the BLS24-315 curve							
BW6-633, u_{315}	6	3	[EHG22a, Tab. 11]	315	633	3798	2^{124}
BW6-672, u_{315}	6	3	[EHG22a, Tab. 11]	315	672	4032	2^{128}
CP8-632, u_{315}	8	4	[EHG22a, Tab. 7]	315	632	5056	2^{140}
CP12-630, u_{315}	12	3	[EHG22a, Tab. 7]	315	630	7560	2^{166}

Cheon's attack. Cheon [Che10] showed that given $G, [\tau]G$ and $[\tau^{\mathrm{T}}]G$, with G a point in a subgroup $\mathbb{G}$ of order r with $\mathrm{T} \mid r-1$, it is possible to recover τ in $2(\lceil\sqrt{(r-1)/\mathrm{T}}\rceil + \lceil\sqrt{\mathrm{T}}\rceil) \times (\mathtt{Exp}_{\mathbb{G}}(r) + \log r \times \mathtt{Comp}_{\mathbb{G}})$ where $\mathtt{Exp}_{\mathbb{G}}(r)$ stands for the cost of one exponentiation in $\mathbb{G}$ by a positive integer less than r and

Table 6. Security level estimates of BN curves and outer curves with the software shipped with [GS21].

curve	k	D	ref	r bits	p bits	p^k bits	DL cost in $\mathbb{F}_{p^k}$
BN-256 $u = 1868033^3$ Pinocchio	12	3	[PHGR13]	256	256	3063	2^{103}
BN-254 $u = 2^{62} - 2^{54} + 2^{44}$ Pantry	12	3	[BFR+13]	256	256	3038	2^{102}
BN-254 $u = -(2^{62} + 2^{55} + 1)$ Geppetto	12	3	[CFH+15]	254	254	3038	2^{102}
BN-254 $u =$ `0x44e992b44a6909f1` Ethereum	12	3	[BCTV14b]	254	254	3044	2^{103}
outer curve with the Geppetto BN curve							
BW6-509	6	3	[CFH+15]	254	509	3051	2^{102}

$\texttt{Comp}_{\mathbb{G}}$ for the cost to determine if two elements are equal in $\mathbb{G}$. According to [Che10, Theorem 2], 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. Sakemi et al. reported an implementation on a 160-bit elliptic curve in [SHI+12].

In SNARKs such as [Gro16] and KZG-based schemes, the `Setup` keys include elements $[\{\tau^i\}_{i=0}^{\mathrm{T}}]G \in \mathbb{G}$ where $\mathrm{T} \in \mathbb{N}^*$ is at least the size of the arithmetic circuit related to the statement to prove, and τ is the secret trapdoor. The property $\mathrm{T} \mid r - 1$ also holds since we need $r - 1$ to be highly *2-adic* (condition (ii)). So, given these auxiliary inputs, an attacker can recover the secret using Cheon's algorithm in time $O(\sqrt{r/\mathrm{T}})$, hence breaking the SNARK soundness. We stress that this attack vector is not inherent to the curve design but to the SNARK design. Given a curve where $r \equiv 1 \bmod 2^L$ used in a SNARK requiring a setup of size $2^{L'}$ where $L' < L$, Cheon's attack runs in $O(\sqrt{r/L'})$ and not $O(\sqrt{r/L})$. To the best of our knowledge, the Filecoin circuit ($L'_{\mathbb{G}_1} = 28, L'_{\mathbb{G}_2} = 27$) is the biggest application circuit of public interest.

While taking into account this attack at the curve design level might limit the attack vector, this prevents a nice speed up in the pairing computation. As observed in [EHG22a], having a large L s.t. $r(x) \equiv 1 \bmod 2^L$ is often entangled to having a large number of consecutive zeroes in the seed x. This allows mixing efficiently the Karabina [Kar13] and Granger-Scott [GS10] cyclotomic squaring algorithms, hence speeding up significantly the final exponentiation. That is said, Cheon's attack must be taken into account when implementing a SNARK circuit with a given elliptic curve.

4.3 Examples

In Table 7, we recall the literature curves presented in the state-of-the-art section 2 and summarize their SNARK-friendliness properties and security levels.

We see that only the BLS12-381 satisfies conditions (ii), (iii), (iv), (v) and has almost a 128-bit security level. KSS16 and KSS18 families were not investigated in the SNARK literature. BLS24 family was considered in [EHG22a] in the context of 2-chains, hence the proposed BLS24-315 does not satisfy the condition (iii) by definition (cf. 5.2). In table 8, we propose new BN and BLS24 that satisfy all the requirements. We also propose the first KSS16 and KSS18 SNARK curves and

Table 7. Properties of SNARK curves from the literature. GMV6-183 being insecure, it should be avoided.

Curve	x	L	$r = \#\mathbb{G}_1$ (bits)	p, $\mathbb{G}_1$ (bits)	$p^{k/d}$, $\mathbb{G}_2$ (bits)	$p \equiv 3 \bmod 4$	security (bits)	
							$\mathbb{G}_1$	$\mathbb{F}^*_{p^k}$
BN-256 [PHGR13]	1868033^3 $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 19$	5	256	256	512	✓	128	103
BN-254 [BFR+13]	$2^{62} - 2^{54} + 2^{44}$ $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 7$	45	254	254	508	$\times$	127	102
GMV6-183 [BCG+13]	`0x8eed757d90615e40000000` $\mathrm{HW}(-26x-2) = 16$	31	181	183	549	NA[2]	90	71
BN-254 [BCTV14b]	`0x44e992b44a6909f1` $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 22$	28	254	254	508	✓	127	103
BLS12-381 [Bow17]	`-0xd201000000010000` $\mathrm{HW}(x) = 6$	32	255	381	762	✓	127	126

compare them to the existing curves. We omit to revisit the GMV6 family as a 128-bit secure curve would be defined over a large field (around 704 bits).

Table 8. New SNARK curves from the BN, BLS24, KSS16 and KSS18 families.

Curve	x	L	$r = \#\mathbb{G}_1$ (bits)	p, $\mathbb{G}_1$ (bits)	$p^{k/d}$, $\mathbb{G}_2$ (bits)	$p \equiv 3 \bmod 4$	security (bits)	
							$\mathbb{G}_1$	$\mathbb{F}^*_{p^k}$
BN383	`0x49e69d16fdc80216226909f1` $\mathrm{HW}_{\text{2-NAF}}(6x+2) = 30$	44	383	383	766	✓	191	123
BLS24-317	`0xd9018000` $\mathrm{HW}_{\text{2-NAF}}(x) = 6$	60	255	317	1268	✓	127	160
KSS16-329	`0x38fab7583` $\mathrm{HW}(x) = 12$	19	255	329	1316	✓	127	140
KSS18-345	`0xc0c44000000` $\mathrm{HW}(x) = 6$	78	254	345	690	NA[3]	127	150

KSS16 and KSS18. The KSS family was not investigated previously in the SNARK context. KSS16 and KSS18 defined respectively over fields of size 328-bit and 348-bit offer 128 bits of security. We suggest the KSS16-329 and KSS18-345 that fulfill all conditions except $p \equiv 3 \mod 4$ (condition (iii) does not apply to KSS18 and is not possible for KSS16).

BLS24. The BLS family with embedding degree $k = 24$ was previously investigated in [EHG22a] in the recursive SNARK context. However, the authors only considered lifting of simple roots and proposed the BLS24-315 curve with

[2] For GMV6-183, 3 is the smallest cubic non-residue on $\mathbb{F}_p$.
[3] For KSS18-345, 2 is the smallest cubic non-residue on $\mathbb{F}_p$.

$2^{22} \mid r-1$. Here we consider multiple roots and propose the BLS24-317 curve with $2^{60} \mid r-1$ that fulfills all SNARK-friendliness conditions. This curve is particularly suitable for KZG-based SNARKs compared to previous known curves. As a reference example, compared to the widely used BLS12-381, it takes 14% less time to generate a PLONK proof of a circuit of 40000 constraints (implemented in `gnark` [BPEH+22]). The setup generation is also 23% faster but the verification is 30% slower, although this can be likely amortized with a batch verification.

5 Recursive SNARKs with elliptic curves

5.1 Cycles of pairing-friendly elliptic curves

A *cycle* of elliptic curves is a list of curves defined over finite fields in which the number of points in one curve is equal to the size of the field of definition of the next curve, in a cyclic manner.

Definition 1. *An m-cycle of elliptic curves is a list of m distinct elliptic curves*

$$E_1/\mathbb{F}_{p_1}, \ldots, E_m/\mathbb{F}_{p_m},$$

for primes $p_1, \ldots, p_m$, such that the numbers of points on these curves satisfy

$$\#E_1(\mathbb{F}_{p_1}) = p_2, \ldots, \#E_i(\mathbb{F}_{p_i}) = p_{i+1}, \ldots, \#E_m(\mathbb{F}_{p_m}) = p_1.$$

This notion was initially introduced under different names, for example *amicable pairs* (or equivalently *dual elliptic primes* [Mih07]) for 2-cycles of ordinary curves, and *aliquot cycles* for the general case [SS11]. A *pairing-friendly m-cycle* is an m-cycle composed of ordinary pairing-friendly curves.

The state of the art. Chiesa, Chu and Weidner studied the existence of pairing-friendly cycles, beyond the previously known cycles of curves with embedding degrees 4 and 6, and arrived at a number of negative results which indicate that having a small embedding degree is a *strong* restriction on constructing cycles [CCW19]:

- There are no 2-cycles of elliptic curves with embedding degrees $(5, 10), (8, 8)$ or $(12, 12)$, which means that there are no *optimal* (in terms of parameter sizes) pairing-friendly 2-cycles at the 128-bit security level. The notation (k, ℓ) here means a cycle of two elliptic curves with embedding degrees k and ℓ, respectively.
- There are no pairing-friendly cycles with more than 2 curves with the same CM discriminant $D > 3$, which implies that elliptic curves from families of varying discriminants must be used to construct cycles.
- There are no cycles of prime-order pairing-friendly curves only from the Freeman and Barreto-Naehrig families; or cycles of composite-order elliptic curves. This motivates the search for new constructions of prime-order pairing-friendly curves.

On the positive side, the authors characterize all cycles of MNT curves as consisting of curves with alternating embedding degrees of 4 or 6, and construct a new 4-cycle of MNT curves which consists of the union of 2-cycles. Table 9 collects the possibilities to generate MNT cycles, for which a specific cycle can be generated by substituting the possible choices of parameter x and checking that all polynomials $p(x)$ and $r(x)$ evaluate to prime numbers. Concrete parameters used in practice can be found in Table 4.

Open problems. The above restrictions and current constructions of cycles suggest that pairing-friendly 2-cycles will always have the inconvenience of including two elliptic curves at different security levels, which means that at least one curve would have sub-optimal performance, but there are several *open problems* for which a satisfying solution would potentially increase the available options:

– Are there cycles of elliptic curves with the same embedding degree, and possibly the same discriminant?
– Are there pairing-friendly cycles of embedding degrees greater than 6?
– Are there pairing-friendly cycles combining MNT, Freeman and Barreto-Naehrig curves?

Table 9. Parameterized (6,4) 2-cycles and (6,4,6,4) 4-cycles of MNT curves, where 4-cycles are constructed as the union of the 2-cycles.

	(6,4,6,4) 4-cycle			
	(6,4) 2-cycle		(6,4) 2-cycle	
	E_1	E_2	E_3	E_4
k	6	4	6	4
$p(x)$	$4x^2+1$	$4x^2+2x+1$	$4x^2+1$	$4x^2-2x+1$
$r(x)$	$4x^2+2x+1$	$4x^2+1$	$4x^2-2x+1$	$4x^2+1$
$t(x)$	$-2x+1$	$2x+1$	$2x+1$	$-2x+1$

We already know that *optimal* 2-cycles for recursive SNARKs cannot be found, but it is possible that better alternatives with higher embedding degrees and smaller parameter sizes than the inefficient MNT (6,4)-cycles exist. However, reconciling the conditions for pairing-friendly cycles (prime order, varying families and discriminants) with the requirements for SNARK-friendliness to construct efficient recursive SNARKs, i.e. conditions (i)–(iv), further restricts the choice of parameters and increase the challenge. Since no methods are known to find prime-order pairing-friendly curves with embedding degree larger than 12, novel ideas are severely needed. In particular, finding 2-cycles of pairing-friendly curves with embedding degrees $(16, 16)$ or higher would already be a significant contribution to improving performance of constructions at current 128-bit and higher security levels.

Because of these difficulties in finding parameters, current efforts have focused on findings *chains* of pairing-friendly elliptic curves, *hybrid cycles* containing one pairing-friendly curve, or amicable pairs that allow alternative recursive proofs without pairings.

5.2 Chains of pairing-friendly elliptic curves

We have seen in Section 5.1 that constructing a *cycle* of two pairing-friendly curves is a very difficult task, and results in a very slow pairing computation, and large parameters. Relaxing the conditions on the curves, one can look for a chain of pairing-friendly elliptic curves, as in Definition 3 (following [EHG22a]).

Definition 2. *An m-chain of elliptic curves is a list of distinct curves*

$$E_1/\mathbb{F}_{p_1}, \ldots, E_m/\mathbb{F}_{p_m}$$

where $p_1, \ldots, p_m$ are large primes and

$$p_1 = r_2 \mid \#E_2(\mathbb{F}_{p_2}), \ldots,\ p_{i-1} = r_i \mid \#E_i(\mathbb{F}_{p_i}), \ldots,\ p_{m-1} = r_m \mid \#E_m(\mathbb{F}_{p_m})\ . \quad (2)$$

Definition 3. *An m-chain of SNARK-friendly elliptic curves is an m-chain where each of the $\{E_i/\mathbb{F}_{p_i}\}_{1\le i\le m}$ curves*

- *is pairing-friendly;*
- *has a highly 2-adic subgroup, i.e. $r_i - 1 \equiv 0 \mod 2^L$ for a large $L \ge 1$.*

In a 2-chain, the first curve is denoted the **inner curve**, while the second curve whose order is the characteristic of the inner curve, is denoted the **outer curve**.

Previous work on chains of pairing-friendly curves. Approximately at the same time as [BCTV14a] released the first cycle of MNT curves, Costello et al. [CFH+15] presented a 2-chain of pairing-friendly curves to obtain a bounded recursive proof composition (of depth 2). Their inner curve is the popular BN curve from [NAS+08] with seed $x = -(2^{62}+2^{55}+1)$ and their outer curve is a new Brezing–Weng curve of embedding degree 6 and 509 bits (just under the machine-word-aligned size of 512 bits). However, the curves do not satisfy Condition (i) that is, no large power of 2 divides neither $r-1$ nor $p-1$. It is surprising that they did not use another widespread BN curve with seed $x = 2^{62} - 2^{54} + 2^{44}$ for example (Beuchat et al. [BGM+10], in the TEPLA library), which does satisfy Condition (i): $2^{45} \mid r-1$ and $2^{45} \mid p-1$. Nevertheless, the Geppetto construction is the first example of a 2-chain of pairing-friendly curves (with parameters having a polynomial form) that we found in the literature, to the best of our knowledge.

Probably because Geppetto appeared *before* the new TNFS algorithm and the re-evaluation of all key sizes of pairing-friendly curves, the MNT cycle of 298 bits was a milestone construction, while the Geppetto 2-chain was not remembered in the subsequent construction named ZEXE [BCG+20]. In light of the need of larger keysizes, Bowe et al. released a 2-chain of pairing-friendly curves based on the BLS12 family. The inner curve is a BLS12-377 curve with $2^{46} \mid r-1$ and $2^{46} \mid p-1$. However the outer curve was a Cocks–Pinch curve that did not benefit from all possible speed-ups of pairing-friendly curves.

In [EHG20], El Housni and Guillevic introduced a 2-chain of curves made of the previous BLS12-377 and a new BW6-761 curve, a Brezing–Weng curve of embedding degree 6 defined over a 761-bit prime field, which they demonstrated to be orders of magnitude faster than CP6-782. Indeed many optimizations are now available, from a twist of degree 6 to an optimal ate pairing, and 761 bits fit in one less machine word of 64 bits compared to 782 bits. The most recent work to date (in 2022) is [EHG22a]. The authors investigate many different possibilities of 2-chains with BLS12 and BLS24 inner curves, and Brezing–Weng outer curves of embedding degree 6, but also Cocks–Pinch curves of the larger embedding degrees 8 and 12.

Inner curves from polynomial families. The inner curves are either BN or BLS curves (BLS12 and BLS24). For efficiency reasons, the condition (i) of $2^L \mid r-1$ for a large power L applies to the two curves of the 2-chain: the inner curve and the outer curve. That is, the inner curve should have $2^L \mid r-1$ and $2^L \mid p-1$ at the same time. Table 3 shows that BN curves with $x \equiv 0 \bmod 2^{L-1}$, BLS12 and BLS24 curves with $x \equiv 1 \bmod 3 \cdot 2^L$, satisfy the 2-valuation condition on $p-1$ and $r-1$. We will then focus on these three families of curves: BN, BLS12 and BLS24.

Outer curves with the Brezing–Weng and Cocks-Pinch methods. The papers [EHG20,EHG22a] consider 2-chains from a BLS12 and a BLS24 curve. The paper [EHG20] introduces the curve BW6-761, an outer curve with BLS12-377. The paper [EHG22a] describes a general framework for all 2-chains made with a BW6 curve from a BLS12 curve, resp. all 2-chains of a BW6 curve with a BLS24 curve (Figure 5).

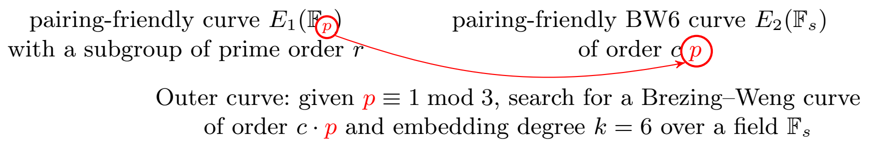
pairing-friendly curve $E_1(\mathbb{F}_p)$
with a subgroup of prime order r
pairing-friendly BW6 curve $E_2(\mathbb{F}_s)$
of order $c \cdot p$
Outer curve: given $p \equiv 1 \bmod 3$, search for a Brezing–Weng curve
of order $c \cdot p$ and embedding degree $k = 6$ over a field $\mathbb{F}_s$

Fig. 5. Brezing–Weng outer curve construction for BN and BLS inner curves

The families of 2-chains are parameterized by the lifting cofactors h_t, h_y of $t(x), y(x)$ in Alg. 7. Non-zero lifting cofactors produce curves with a rho-value at

Algorithm 7: Brezing–Weng outer curve construction

```
Input: Polynomial $p(x)$ s.t. $\zeta_6 \in K = \mathbb{Q}[x]/(p(x))$ where $\zeta_6^2 - \zeta_6 + 1 = 0$, seed $u$ of an
       inner curve s.t. $p(u)$ is a prime integer
Output: $E_2/\mathbb{F}_s$ with an order-$p(u)$ subgroup and embedding degree $k = 6$
1 Set $z_6(x) \in \mathbb{Q}[x]$ a polynomial mapping to $\zeta_6$ in $K$ ;
2 Choose $t(x) \in \mathbb{Q}[x]$ be a polynomial representing $\zeta_6 + 1$ in $K$;
3 Set $y(x) \in \mathbb{Q}[x]$ be a polynomial mapping to $(\zeta_6 - 1)/\sqrt{-3}$ in $K$;
4 Compute $s(x) = (t^2(x) + 3y^2(x))/4$ in $\mathbb{Q}[x]$;
5 $h_t = 0$; $h_y = 0$ ;
6 while $s(x)$ is not irreducible or $s(u)$ is not prime:
7     Increment one of $h_t, h_y$ ;
8     $s(x) = ((t(x) + h_t \cdot p(x))^2 + 3(y(x) + h_y \cdot p(x))^2)/4$;
9 return $E_2/\mathbb{F}_{s(u)}$ with an order-$p(u)$ subgroup and embedding degree $k = 6$
```

least 2, but otherwise the polynomial defining $s(x)$ is not irreducible. Non-zero lifting cofactors appear in [FK19,GMT20] for other pairing-friendly curves.

We reproduce the tables of parameters [EHG22a, Tables 3, 4] in Tables 13 and 14. Table 13 gives the parameters of the BW6-BLS12 curves in terms of the seed x, and the two lifting cofactors h_t, h_y. Table 14 gives the parameters of the BW6-BLS24 curves with the same parameters x, h_t, h_y. The data is also available at [EHG22b].

Optimal ate pairing and optimal twisted ate pairing on outer curves. As can be seen in Tables 13 and 14, the Miller loop of the optimal ate pairing on the BW6 outer curves is not as short as for the well-known BN:

$$f_{6u+2,Q}(P)\ell_{[6u+2]Q,\pi(Q)}(P)\ell_{[6u+2]Q+\pi(Q),\pi^2(-Q)}(P)$$

or BLS12, BLS24 curves ($f_{u,Q}(P)$). It comprises two parts, $f_{a_0,Q}(P)f_{a_1,\pi(Q)}(P)$ for the optimal ate pairing, and $f_{a_0+a_1\lambda,P}(Q)$ for the optimal twisted ate pairing, with $\lambda = t_0 - 1$ the eigenvalue of the endomorphism $(x, y) \mapsto (\omega x, y)$ on the appropriate subgroup, where $\omega^2 + \omega + 1 = 0$. Indeed, all our BW curves E_2 have j-invariant 0 and discriminant $D = -3$. Two optimization strategies are developed in [EHG20,EHG22a]. First, the Miller loop can be shared thanks to the formula

$$f_{uv,Q}(P) = f_{u,Q}^v(P) \cdot f_{v,[u]Q}(P) \tag{3}$$

and the exponentiation $f_{u,Q}^v(P)$ is implicitly performed by initializing the second Miller function $f_{v,[u]Q}(P)$ at $f = f_{u,Q}(P)$ instead of $f = 1$. This is [EHG20, Alg. 5]. The second optimization applies to the ate and twisted ate pairings and consists in computing a multi-scalar Miller function $f_{a_0+a_1\lambda,P}(Q)$ with shared doublings and squarings, this is [EHG22a, Alg. 2] reproduced in Alg .8.

When using a twisted curve of E_2 to compress $\mathbb{G}_2$-elements, the coordinates are *sparse* elements of $\mathbb{F}_{p^k}$, that is, some of the coefficients are always zero. As a consequence, the line and vertical functions ℓ, v are sparse elements of $\mathbb{F}_{p^k}$.

Algorithm 8: [EHG22a, Alg. 2] Miller loop for optimal pairing with endomorphism ϕ on $\mathbb{G}_1$ (Tate), resp. $\mathbb{G}_2$ (ate) of eigenvalue λ and degree 2.

```
Input: $P \in \mathbb{G}_i$, $Q \in \mathbb{G}_{1-i}$, end. $\phi$ on $\mathbb{G}_i$ of eigenvalue $\lambda$, scalars $a_0, a_1$
         s. t. $a_0 + a_1\lambda = 0 \bmod r$
Output: $f_{a_0+a_1\lambda,P}(Q)$
1  $P_0 \leftarrow P$; $P_1 \leftarrow \phi(P)$
2  if $a_0 < 0$: $a_0 \leftarrow -a_0$; $P_0 \leftarrow -P_0$
3  if $a_1 < 0$: $a_1 \leftarrow -a_1$; $P_1 \leftarrow -P_1$
4  $P_{1+\lambda} \leftarrow P_0 + P_1$; $\ell_{1,\lambda} \leftarrow \ell_{P_0,P_1}(Q)$
5  $l_0 \leftarrow \mathrm{bits}(a_0)$; $l_1 \leftarrow \mathrm{bits}(a_1)$
6  if $\#l_0 = \#l_1$:  $S \leftarrow P_{1+\lambda}$; $f \leftarrow \ell_{1,\lambda}$; $n \leftarrow \#l_0$
7  elif $\#l_0 < \#l_1$:  $S \leftarrow P_1$; $f \leftarrow 1$; $n \leftarrow \#l_1$; pad $l_0$ with 0 s.t. $\#l_0 = n$
8  else:  $S \leftarrow P_0$; $f \leftarrow 1$; $n \leftarrow \#l_0$; pad $l_1$ with 0 s.t. $\#l_1 = n$
9  for $i = n-2$ downto 0:
10     $f \leftarrow f^2$ ; $\ell_t \leftarrow \ell_{S,S}(Q)$; $S \leftarrow [2]S$
11     if $l_0[i] = 0$ and $l_1[i] = 0$:  $f \leftarrow f \cdot \ell_t$                    // $m_{\text{full-sparse}}$
12     elif $l_0[i] = 1$ and $l_1[i] = 1$:
13         $S \leftarrow S + P_{1+\lambda}$; $\ell \leftarrow \ell_{S,P_{1+\lambda}}(Q)$
14         $f \leftarrow (f \cdot \ell_t) \cdot (\ell \cdot \ell_{1,\lambda})$       // $m_k + m_{\text{full-sparse}} + m_{\text{sparse-sparse}}$
15     elif $l_0[i] = 1$:
16         $S \leftarrow S + P_0$; $\ell \leftarrow \ell_{S,P_0}(Q)$
17         $f \leftarrow f \cdot (\ell_t \cdot \ell)$                               // $m_k + m_{\text{sparse-sparse}}$
18     else: $(l_1[i] = 1)$
19         $S \leftarrow S + P_1$; $\ell \leftarrow \ell_{S,P_1}(Q)$
20         $f \leftarrow f \cdot (\ell_t \cdot \ell)$                               // $m_k + m_{\text{sparse-sparse}}$
21 return $f$
```

Scott noted that it is more efficient to first multiply together the sparse lines and then multiply the result with the accumulated value f, rather than multiplying each line to the result one by one. For that reason, Alg. 8 has *sparse-sparse* multiplications, and *full-sparse* multiplications.

Our optimized final exponentiation on outer curves. The final exponentiation raises the result f of the Miller loop to the power $(s^k - 1)/p$ so that the final result is well-defined. It is usually divided into an easy part made of Frobenius powers: $x \mapsto x^s$ in $\mathbb{F}_{s^k}$, and a hard part, as follows:

$$\frac{s^k - 1}{p} = \underbrace{\frac{s^k - 1}{\Phi_k(s)}}_{\text{easy part}} \cdot \underbrace{\frac{\Phi_k(s)}{p}}_{\text{hard part}} . \tag{4}$$

For $k = 6$, the easy part is $f^{(s^3-1)(s+1)}$. We improve the hard part inspired by the work of Hayashida, Hayasaka and Teruya [HHT20], and the work of Cai, Hu and Zhao [CHZ22]. The key-ingredient is to factor as many terms as possible. The cost summary is in Tab. 10. The SageMath implementation is at [EHG22b].

Algorithm 9: Hard part of final exp. m^{e_0}, BN-BW6, trace $t_0 + h_t p$ (Tab. 15)

```
Input: $x$, $m$, $h_1 = (h_t - h_y)/2$,
       $h_2 = (h_t^2 + 3h_y^2)/4$
Output: $m^{2x\Phi_6(s_0)/p}$
$A \leftarrow m^x$
$B \leftarrow A^2 \cdot A \cdot m \cdot m^{s_0}$                $\triangleright m^{a_0}$
$A \leftarrow \overline{B}$
$B \leftarrow (B^2)^x \cdot m$                              $\triangleright m^{b_0}$
$C \leftarrow (B^x \cdot B)^x$
$C \leftarrow C^2 \cdot C \cdot B$                          $\triangleright B^d$
$D \leftarrow (C^2 \cdot B)^x$
$D \leftarrow D^2 \cdot D$                                  $\triangleright B^{-t_0}$
$F \leftarrow (D^x \cdot C)^2 \cdot \overline{F}$                     $\triangleright B^r$
$F \leftarrow F^{h_2} \cdot \overline{D}^{h_1} \cdot C$
return $F \cdot A$
```

Algorithm 10: Hard part of final exp. $m^{e'_3}$, BN-BW6, trace $t_3 + h_t p$ (Tab. 15)

```
Input: $x$, $m$, $h_1 = (h_t + h_y)/2$,
       $h_2 = (h_t^2 + 3h_y^2)/4$
Output: $m^{(6x^2+2x+1)\Phi_6(s_3)/p}$
$B \leftarrow m^{s_3}$
$A \leftarrow B^x$
$A \leftarrow (A^2 \cdot A \cdot B) \cdot m$                  $\triangleright m^{a'_3}$
$B \leftarrow (A^2)^x \cdot B$                              $\triangleright m^{b'_3}$
$C \leftarrow B^x$
$D \leftarrow C^2 \cdot C$
$C \leftarrow D^x \cdot D \cdot B$                          $\triangleright B^d$
$E \leftarrow (C^2 \cdot B)^x \cdot B$
$E \leftarrow E^2 \cdot E$                                  $\triangleright B^{t_3}$
$F \leftarrow ((E \cdot D)^x)^2 \cdot B$                      $\triangleright B^p$
$F \leftarrow F^{h_2} \cdot E^{h_1} \cdot C$
return $F \cdot A$
```

Algorithm 11: Hard part of final exp. $m^{e'_0}$, BLS12-BW6, trace $t_0 + h_t p$ (Tab. 13)

```
Input: $x$, $m$, $h_1 = (h_t - h_y)/2$,
       $h_2 = (h_t^2 + 3h_y^2)/4$
Output: $m^{(x^3-x^2-x)\Phi_6(s_0)/p}$
$Q \leftarrow m^{s_0}$
$A \leftarrow (Q^{x-1})^{x-1} \cdot m$                        $\triangleright m^{a'_0}$
$B \leftarrow A^{x+1} \cdot \overline{Q}$                       $\triangleright m^{b'_0}$
$A \leftarrow \overline{A^2 \cdot A}$                             $\triangleright m^{-3a'_0}$
$C \leftarrow B^{(x-1)/3}$
$D \leftarrow C^{x-1}$
$E \leftarrow (D^{x-1})^{x-1} \cdot D$                  $\triangleright B^{(d-1)/3}$
$F \leftarrow \overline{E^{x+1} \cdot C} \cdot D$                 $\triangleright B^{t_0/3}$
$G \leftarrow \overline{(F \cdot D)^{x+1}} \cdot C \cdot B$                $\triangleright B^p$
$H \leftarrow F^{h_1} \cdot E$
$H \leftarrow H^2 \cdot H \cdot B \cdot G^{h_2}$
return $A \cdot H$
```

Algorithm 12: Hard part of final exp. m^{e_3}, BLS12-BW6, trace $t_3 + h_t p$ (Tab. 13)

```
Input: $x$, $m$, $h_1 = (h_t + h_y)/2$,
       $h_2 = (h_t^2 + 3h_y^2)/4$
Output: $m^{(x+1)\Phi_6(s_3)/p}$
$A \leftarrow m^{s_3} \cdot (A^{x-1})^{x-1}$                  $\triangleright m^{a_3}$
$B \leftarrow \overline{m} \cdot A^{x+1}$                         $\triangleright m^{b_3}$
$A \leftarrow A^2 \cdot A$                                  $\triangleright m^{3a_3}$
$C \leftarrow B^{(x-1)/3}$
$D \leftarrow C^{x-1}$
$E \leftarrow (D^{x-1})^{x-1} \cdot D$                  $\triangleright B^{(d-1)/3}$
$D \leftarrow \overline{D}$
$F \leftarrow D \cdot B$
$G \leftarrow E^{x+1} \cdot F$
$H \leftarrow G \cdot C$                                    $\triangleright B^{t_3/3}$
$I \leftarrow (G \cdot D)^{x+1} \cdot \overline{F}$                   $\triangleright B^p$
$J \leftarrow H^{h_1} \cdot E$
$K \leftarrow J^2 \cdot J \cdot B \cdot I^{h_2}$
return $A \cdot K$
```

Algorithm 13: Hard part of final exp. $m^{e_0'}$, BLS24-BW6, trace $t_0 + h_t p$ (Tab. 14)

```
Input: $x$, $m$, $h_1 = (h_t - h_y)/2$,
       $h_2 = (h_t^2 + 3h_y^2)/4$
Output: $m^{(x^5-x^4-x)\Phi_6(s_0)/p}$
$m_s \leftarrow m^{s_0}$
$A \leftarrow (m_s^{u-1})^{u-1}$
$A \leftarrow A^{u^2+1}$                      $\triangleright$or $(A^u)^u \cdot A$
$A \leftarrow m \cdot A$
$B \leftarrow A^{u+1} \cdot \overline{m_s}$                 $\triangleright m^{b_0}$
$A \leftarrow \overline{A^2 \cdot A}$                      $\triangleright m^{-3a_0}$
$C \leftarrow B^{(u-1)/3}$
$D \leftarrow C^{u-1}$
$D \leftarrow D^{u^2+1}$                      $\triangleright$or $(D^u)^u \cdot D$
$E \leftarrow (D^{u-1})^{u-1}$
$E \leftarrow E^{u^2+1}$                      $\triangleright$or $(E^u)^u \cdot E$
$E \leftarrow E \cdot D$                        $\triangleright B^{(d-1)/3}$
$F \leftarrow \overline{E^{u+1} \cdot C} \cdot D$               $\triangleright B^{t_0/3}$
$G \leftarrow \overline{F \cdot D}$
$H \leftarrow G^{u+1} \cdot C \cdot B$                 $\triangleright B^p$
$I \leftarrow F^{d_1} \cdot E$
$I \leftarrow I^2 \cdot I \cdot B \cdot H^{d_2}$
return $A \cdot I$
```

Algorithm 14: Hard part of final exp. m^{e_3}, BLS24-BW6, trace $t_3 + h_t p$ (Tab. 14)

```
Input: $x$, $m$, $h_1 = (h_t + h_y)/2$,
       $h_2 = (h_t^2 - 3h_y^2)/4$
Output: $m^{(x+1)\Phi_6(s_3)/p}$
$A \leftarrow (m^{u-1})^{u-1}$
$A \leftarrow A^{u^2+1}$                      $\triangleright$or $(A^u)^u \cdot A$
$A \leftarrow A \cdot m^{s_3}$
$B \leftarrow A^{u+1} \cdot \overline{m}$                  $\triangleright m^{b_3}$
$A \leftarrow A^2 \cdot A$                       $\triangleright m^{3a_3}$
$C \leftarrow B^{(u-1)/3}$
$D \leftarrow C^{u-1}$
$D \leftarrow D^{u^2+1}$                      $\triangleright$or $(D^u)^u \cdot D$
$E \leftarrow (D^{u-1})^{u-1}$
$E \leftarrow E^{u^2+1}$                      $\triangleright$or $(E^u)^u \cdot E$
$E \leftarrow E \cdot D$                        $\triangleright B^{(d-1)/3}$
$D \leftarrow \overline{D}$
$F \leftarrow E^{u+1} \cdot D \cdot C$
$G \leftarrow F \cdot B$                        $\triangleright B^{t_3/3}$
$H \leftarrow (F \cdot D)^{u+1} \cdot C \cdot B$            $\triangleright B^p$
$I \leftarrow G^{d_1} \cdot E$
$I \leftarrow I^2 \cdot I \cdot B \cdot H^{d_2}$
return $A \cdot I$
```

A lattice reduction (with Magma on polynomials) gives the formulas for the exponents e_0, e_0', e_3, e_3' in Tab. 15, Tab. 13, and Tab. 14, the Magma script being available at [EHG22b]. The well-known strategy [FKR12] first simplifies the ratio $\Phi_k(s(x))/p(x)$ then finds a small multiple of that exponent such that when expressed in the Frobenius basis $\{1, s(x), s^2(x), \ldots, s^{\varphi(k)-1}(x)\}$, the coefficients have the lowest possible degree in x. To finish, that exponent is factored so as to minimize further the powering cost. In our case, $k = 6$ and the exponent is spanned over the basis $\{1, s(x)\}$ where $s(x)$ has degree twice the degree of $p(x)$. We only give the optimized algorithms of exponentiation and refer to the appendix C for the technical details.

The limitation of the order of the twist with the Brezing–Weng method, and the Cocks–Pinch construction. The authors of [EHG22a] also considered the Cocks–Pinch construction for the outer curves. While this construction allows less efficient pairings, it provides curves with more conservative security when the embedding degree of the inner curve is large (e.g. BLS24). BW and CP constructions follow the same recipe, but CP deals with integers, while BW deals with polynomials [FST10, §4.1,§6]. They start from the subgroup order

Table 10. Cost of hard part of final exponentiation. $\mathbf{e}^y$ stands for exponentiation to y, $\mathbf{m}$ is multiplication, $\mathbf{s}$ is squaring, $\mathbf{f}$ is Frobenius (x^p), $\mathbf{cj}$ is conjugation, all in $\mathbb{F}_{p^6}$. For BLS24-BW6 curves, $m^{x^2+1} = (m^x)^x \cdot m$. If x^2+1 has a lower Hamming weight than $2\,\mathrm{Hw}(x)+1$, it is faster to do the former.

Curve	Cost in terms of operations in $\mathbb{F}_{p^6}$
BN-BW6, t_0, Alg. 9	$6\mathbf{e}^x + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 6\mathbf{s} + 14\mathbf{m} + \mathbf{f} + 3\mathbf{cj}$
BN-BW6, t_0, alternative	$6\mathbf{e}^x + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 6\mathbf{s} + 14\mathbf{m} + \mathbf{f} + 4\mathbf{cj}$
BN-BW6, t_3, alternative	$6\mathbf{e}^x + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 6\mathbf{s} + 15\mathbf{m} + \mathbf{f} + 2\mathbf{cj}$
BN-BW6, t_3, Alg. 10	$6\mathbf{e}^x + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 6\mathbf{s} + 15\mathbf{m} + \mathbf{f}$
BLS12-BW6, t_0, Alg. 11	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 14\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 4\mathbf{cj}$
BLS12-BW6, t_0, alt.	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 15\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 4\mathbf{cj}$
BLS12-BW6, t_3, Alg. 12	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 14\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 3\mathbf{cj}$
BLS12-BW6, t_3, alt.	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 15\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 4\mathbf{cj}$
BLS24-BW6, t_0, Alg. 13	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x^2+1} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 14\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 4\mathbf{cj}$
BLS24-BW6, t_0, alt.	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x^2+1} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 15\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 4\mathbf{cj}$
BLS24-BW6, t_3, Alg. 14	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x^2+1} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 15\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 2\mathbf{cj}$
BLS24-BW6, t_3, alt.	$5\mathbf{e}^{x-1} + \mathbf{e}^{(x-1)/3} + 3\mathbf{e}^{x^2+1} + 3\mathbf{e}^{x+1} + \mathbf{e}^{h_2} + \mathbf{e}^{h_1} + 16\mathbf{m} + 2\mathbf{s} + \mathbf{f} + 3\mathbf{cj}$

$p(x)$, and look for k-th roots of unity $\zeta_k \bmod p$ to set the trace value $t = \zeta_k + 1$. For CP, the existence of ζ_k requires $p(u) \equiv 1 \bmod k$: for $k = 6, 12, 8$ resp., this means $u \equiv 1 \bmod 3$, $u \equiv 1, 10 \bmod 12$, and $u \equiv 1, 10 \bmod 24$ resp. For BW, the number field defined by $Q[x]/(p(x))$ only contains $\zeta_k(x)$ for $k \mid 6$, limiting the BW construction to $k = 6$ at most.

5.3 Cycles of plain elliptic curves

Given the difficulties in finding optimal cycles and chains of pairing-friendly curves, a recent alternative is employing inner-product argument proofs together with amicable pairs (2-cycles) of *plain* elliptic curves, i.e., curves not equipped with efficient bilinear maps, where just the hardness of the discrete logarithm holds. The advantages of this approach are higher performance, since parameters can scale just like traditional Elliptic Curve Cryptography, and conservativeness against advances in discrete logarithm computation over finite fields, observing that advances in solving elliptic curve discrete logarithms would impact pairing-friendly curves anyway. On the other hand, these advantages come at mild increase in proof sizes and verification times.

Silverman and Stange [SS11] showed that there exist elliptic curves with arbitrary long cycles, and conjectured that for any elliptic curve $E/\mathbb{Q}$, the number $\mathcal{Q}_E(X)$ of prime pairs (p, q) with $p < q$ and $p \leq X$ such that reducing E modulo p and q gives an amicable pair is such that:

1. $\mathcal{Q}_E(X) = \Theta\left(\frac{\sqrt{X}}{(\log X)^2}\right)$ as $X \to \infty$ if E does not have complex multiplication;
2. $\mathcal{Q}_E(X) \approx A_E \frac{X}{(\log X)^2}$ for a constant $A_E > 0$, otherwise.

In other words, there is a surprising difference between the CM and non-CM cases, and amicable pairs are asymptotically common. The reason for why CM curves have so many amicable pairs is attributed to the fact that if $E/\mathbb{Q}$ has CM and $\#E_p(\mathbb{F}_p) = q$ then there are generally only two possible values for $\#E_q(\mathbb{F}_q)$: p and $2q + 2 - p$, while non-CM curves have $\#E_q(\mathbb{F}_q)$ ranging throughout the interval given by the Hasse condition $p - 2\sqrt{p} + 1 \leq r \leq p + 2\sqrt{p} + 1$. Further research in [Jon13,Par19] has refined and proven these estimates on average.

Regarding security, one notes that the extreme case of an 1-cycle, i.e. $\#E(\mathbb{F}_p) = p$ leads to anomalous curves for which the discrete logarithm can be computed in linear time [Sma99]. For longer cycles, a set of requirements for choosing parameters can be derived from the SafeCurves [BL] criteria for ECC security:

1. Hardness of the ECDLP against Pollard's rho method on the curve and twist.
2. Large embedding degree and CM discriminant D.
3. Rigidity for generating parameters in a reproducible manner.
4. Availability of Montgomery ladder and complete addition laws.
5. Efficient bijective maps (encodings) between uniform random strings and a large set of rational points on the curve. Such an encoding can also be used for hashing efficiently to the curve, as required by some protocols. For more information in this topic, see [FHSS$^+$22] for a comprehensive treatment and [CSRHT22] for recent advances.

A conservative choice of parameters attempts to finds curves that maximize the set of satisfied security requirements.

`sec(p|q)k256k1` **curves** . Probably the earliest 2-cycle of elliptic curves with application in zero-knowledge proofs was based on the elliptic curve adopted for signatures in Bitcoin. The initial motivation was being able to build zero-knowledge proofs over already deployed cryptosystems by using the closely related curve obtained with just swapping base field and order [Poe18]:

$$\begin{aligned}
&E_p/\mathbb{F}_p : y^2 = x^3 + 7 \text{ of order } q, \text{ called } \texttt{secp256k1} \\
&E_q/\mathbb{F}_q : y^2 = x^3 + 7 \text{ of order } p, \text{ called } \texttt{secq256k1}, \text{ with} \\
&\quad p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 \text{ and} \\
&\quad q = 2^{256} - 432420386565659656852420866394968145599.
\end{aligned}$$

These curves satisfy SafeCurves criteria 1 and 3, but have CM discriminant $D = -3$ for the efficient endomorphism [GLV01]; and prime order which does not allow the original Montgomery ladder or Elligator encoding [BHKL13]. However, an optimized version of the Elligator Squared [Tib14] encoding can be used [Wui21]. The endomorphism can be written down as $\psi : (x, y) \to (\beta x, y)$ with β a primitive third root of unity in the base field, corresponding to a multiplication by λ a third root of unity in the scalar field.

Tweedle curves. One of the first 2-cycles of elliptic curves suitable for recursive proofs was the Tweedledum-Tweedledee curves in Halo [BGH19]:

$$
\begin{aligned}
&E_p/\mathbb{F}_p : y^2 = x^3 + 5 \text{ of order } q\text{, called Tweedledum;}\\
&E_q/\mathbb{F}_q : y^2 = x^3 + 5 \text{ of order } p\text{, called Tweedledee, with}\\
&\quad p = 2^{254} + 4707489545178046908921067385359695873 \text{ and}\\
&\quad q = 2^{254} + 4707489544292117082687961190295928833 4.
\end{aligned}
$$

These curves satisfy some of the SafeCurves criteria at the 126-bit security, but have CM discriminant $D = -3$ as a consequence of how they were constructed, and do not benefit from the Montgomery ladder as prime-order curves. On the other hand, they have high 2-valuation, are equipped with fast endomorphisms for efficient scalar multiplication, and somewhat efficient encoding/hashing algorithms.

Pasta curves. Like the Tweedle curves, Pallas and Vesta form a 2-cycle of elliptic curves for recursive proofs in Halo 2 and beyond [Hop20]:

$$
\begin{aligned}
&E_p/\mathbb{F}_p : y^2 = x^3 + 5 \text{ of order } q\text{, called Pallas;}\\
&E_q/\mathbb{F}_q : y^2 = x^3 + 5 \text{ of order } p\text{, called Vesta, with}\\
&\quad p = 2^{254} + 45560315531419706090280762371685220353 \text{ and}\\
&\quad q = 2^{254} + 45560315531506369815346746415080538113.
\end{aligned}
$$

These curves provide several of the attractive properties of the Tweedle curves, while relaxing twist security to 100 bits for Pallas, and otherwise improving over them on a number of ways. Both curves have:

- Lower-degree isogenies (3 instead of 23) from curves with a nonzero j-invariant, which accelerates hashing.
- The same 2-valuation of 32 (instead of 33 and 34) that accelerates square root extraction and simplifies point compression.
- Sparse moduli to accelerate base field arithmetic in the Montgomery representation.
- A reproducible generation method based on a search utility that is publicly available and should facilitate searching for similar 2-cycles in the future, satisfying rigidity concerns.

The generation method specializes the CM method for $D = -3$ to produce curves of form $E(\mathbb{F}) : y^2 = x^3 + b$ with $b \neq 0$. It looks for values (V, T) within a certain range such that $p = (3V^2 + T^2)/4$ is a prime with the desired length and $p \equiv 1 \pmod 6$ to equip the curve with a fast endomorphism. By requiring $(V - 1)/2$ and $(T - 1)/2$ to be multiples of 2^L, $p - 1$ be a multiple of 2^L and $q - 1$ will be a multiple of 2^L for $q \in \{p + 1 - T, p + 1 + (T - 3V)/2\}$ among the six possible orders given by the CM method. Algorithm 15 summarizes the generation strategy.

Algorithm 15: FindAmicablePair(ℓ, L)

Input: designed length ℓ for prime p, 2-valuation L, k-bit security

Output: 2-cycle of curves E_p, E_q.

```
1  while true:
2      Find $(V, T)$ such that:
3      • $4p = 3V^2 + T^2$, $2^L \mid (V-1)/2$, $2^L \mid (T-1)/2$;
4      • $p$ is prime, $|p| = \ell \wedge p \equiv 1 \pmod 6$;
5      for $q \in \{p+1-T, p+1+(T-3V)/2\}$:
6          if $q \notin \{p, p+1, p-1\} \wedge q \equiv 1 \pmod 6 \wedge 2^L|(q-1) \wedge q$ is prime:
7              Find $b_p$ incrementally such that $E_p(\mathbb{F}_p) : y^2 = x^3 + b_p$ has order $q$;
8              Find $b_q$ incrementally such that $E_q(\mathbb{F}_q) : y^2 = x^3 + b_q$ has order $p$;
9              if ECDLPSecurity($E_p$) $\geq k$ $\wedge$ ECDLPSecurity($E_q$) $\geq k$:
10                 return $(E_p, E_q)$;
```

Generalizing the method for constructing 2-cycles described above gives the immediate corollary that every curve generated with the CM method using a small discriminant forms a cycle with another CM curve generated with the prescribed order given by the characteristic of the base field. In particular, if $q = p + 1 - t$ with CM discriminant D (such that $T^2 - 4p = V^2D$), then taking $S = 2 - T$ automatically gives $p = q + 1 - S$ and $S^2 - 4q = T^2 - 4p = V^2D$.

5.4 Hybrid cycles of elliptic curves

A middle-ground alternative to cycle of pairing-friendly curves and cycles of plain elliptic curves is *half-pairing cycles*, where only one of the curves in a 2-cycle is pairing-friendly. This can be potentially useful to combine recursive proofs with protocols based on cryptographic pairings.

Using the results from previous sections, it is possible to generate half-pairing cycles where the pairing-friendly curve is of prime order by just using the CM method. Hence this strategy applies to generating 2-cycles where the pairing-friendly curve is either from the BN, Freeman or MNT families. One such example is the Pluto-Eris curve.

Pluto-Eris curves. Pluto and Eris form a 2-cycle of elliptic curves [Hop21]:

$E_p/\mathbb{F}_p : y^2 = x^3 + 57$ is a BN curve of order q, called Pluto;

$E_q/\mathbb{F}_q : y^2 = x^3 + 57$ is a plain curve of order p, called Eris, with

$$p = 36x^4 + 36x^3 + 24x^2 + 6x + 1 \text{ and}$$

$$q = 36x^4 + 36x^3 + 18x^2 + 6x + 1, \text{for } x = -(2^{110} + 2^{60} + 2^{39} + 2^{35} - 2^{31}).$$

The field size of 446 bits for Pluto is chosen to satisfy 128 bits of security for pairings [Gui20], although is naturally leaves a larger security margin for Eris. Regarding performance, both curves have the high 2-valuation of 32, are again equipped with fast endomorphisms for efficient scalar multiplication due to the choice of small CM discriminant and efficient encoding/hashing algorithms. The choice of seed for the BN curve gives a particularly fast pairing computation due to its low Hamming weight.

Another such example is the hybrid cycle based on a BN382 curve that was briefly used by the Mina testnet in early 2020 [Mec20].

BN382-Plain curves. BN382 forms, with a related plain curve, a 2-cycle of elliptic curves [Mec20]:

$E_p/\mathbb{F}_p : y^2 = x^3 + 14$ is a BN curve of order q;
$E_q/\mathbb{F}_q : y^2 = x^3 + 7$ is a plain curve of order p, with
$p = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ and
$q = 36x^4 + 36x^3 + 18x^2 + 6x + 1,$ for $x = 2^{94} + 2^{81} + 2^{74} + 2^{66}$.

The fields sizes of 382 bits are chosen to satisfy slightly less than 128 bits of security for pairings, while allowing an efficient 6-limb arithmetic on 64-bit architectures. Similarly to Pluto-Eris, both curves have the high 2-valuation of 64, are equipped with fast endomorphisms for efficient scalar multiplication due to the choice of small CM discriminant and efficient encoding/hashing algorithms. The choice of seed for the BN curve gives a particularly fast pairing computation due to its low Hamming weight.

6 Elliptic curves inside SNARKs

While SNARKs allow proving general-purpose computations, in many applications these computations revolve around proving some cryptographic operations such as hashings, encryptions, key exchanges or signatures. For example, in Zerocash [BCG+14], the authors consider proving the SHA256 hash computation to build a privacy-preserving cryptocurrency protocol. Zcash (ZEC) is a cryptocurrency that first implemented the Zerocash protocol with many improvements over the years. Among these, they replaced the SHA256 hash by a variant of the Pedersen hash based on elliptic curves. Gyges [JKS16] and Hawk [KMS+16] consider proving the RSA encryption for privacy-preserving blockchain smart contracts. Cinderella [DFKP16] considered proving the validity and compliance of X.509 certificates by proving the RSA signature verification. Later, CØCØ's authors observed that the RSA scheme is not a well suited computation to be proven in a SNARK and considered replacing it by a hybrid encryption. They first proved an ECDH key exchange and then a lightweight symmetric encryption (Speck [BSS+13] and Chaskey-LTS [MMV+14]). More recently, zk-rollups were proposed to solve the scalability problem of the Ethereum blockchain. A zk-rollup operator validates a batch of transactions by verifying a proof of the correct verifications of many signatures (e.g. ConsenSys rollup and Polygon Hermez rollup verify EdDSA signatures on a twisted Edwards curve associated to BN254).

All those computations are elliptic curve based cryptographic primitives. They require an elliptic curve E_0 to express the computation, independently of the

SNARK elliptic curve E_1 to prove the computation. CØCØ's authors were the first to propose to construct a customized elliptic curve E_0 to efficiently express this kind of computations. Proving systems use different models to translate a generic-purpose computation into an equivalent arithmetized statement. The Rank-1 Constraint System (R1CS) [BCC+16] is one of the most used arithmetization models. A key property of this model is that multiplications, squarings and inversions have the same cost while additions and subtractions are free. Therefore, an implementor should express the statement they want to prove with the least multiplicative complexity and, counter-intuitively, they should bear in mind that inverses are not costly (they cost about two multiplications). The arithmetization step takes place in $\mathbb{F}_r$ where r is the prime subgroup order of the SNARK curve, i.e. $r \mid \#E_1(\mathbb{F}_p)$. CØCØ constructs a custom curve E_0 over precisely $\mathbb{F}_r$ to avoid emulating non-native field operations (cf. Fig. 3). Following the guidelines described in constructing `curve25519`, the authors proposed $E_0(\mathbb{F}_r)$ in Montgomery form $y^2 = x^3 + 126932x^2 + x$ with order 8 times a 251-bit prime and with a twist order 4 times a 252-bit prime. Implementing affine-coordinate scalar multiplication using this curve is very efficient in the R1CS model. Later, Zcash engineers proposed the use of a twisted Edwards curve instead that is more efficient than the CØCØ Montgomery curve. It was built on top of the BLS12-381 curve and named Jubjub. In fact, conditional statements are costly in R1CS and one resorts to a *double-and-always-add*-like algorithm (as in a side-channel resistant implementation) instead of the classical *double-and-add* algorithm to express a scalar multiplication. Instinctively, it would seem that windowed multiplications would be expensive in this context but it turns out that constant-time windowed methods can be achieved efficiently in R1CS using a twisted Edwards curve. The affine group law on these curves is *complete* meaning that one can use a lookup table to select the precomputed points (x_i, y_i) or the zero point $(x_0, y_0) = (0, 1)$ to add in the algorithm. This is done through polynomials which vanish at the inputs that are not being selected. As an example, we show how to perform a 2-bit windowed scalar multiplication in R1CS in Algorithms 17 and 16.

Algorithm 16: `Lookup2`: 2-bit lookup table in R1CS

Input: bits (b_0, b_1), and constants (c_0, c_1, c_2, c_3)

Output: $r = \begin{cases} c_0, & \text{if } b_0 = 0, b_1 = 0 \\ c_1, & \text{if } b_0 = 0, b_1 = 1 \\ c_2, & \text{if } b_0 = 1, b_1 = 0 \\ c_3, & \text{if } b_0 = 1, b_1 = 1 \end{cases}$

```
1 $t_1, t_2 \leftarrow$ temporary variables;
2 $(c_3 - c_2 - c_1 + c_0) \times b_1 = t_1 - c_1 + c_0$;
3 $t_1 \times b_0 = t_2$;
4 $(c_2 - c_0) \times b_1 = r - t_2 - c_0$;
5 return $r$;
```

Algorithm 17: 2-bit windowed scalar multiplication in R1CS

```
Input: a scalar $s$ and a point $P \in E_0$
Output: $R = [s]P \in E_0$
1  $(b_0, \cdots, b_{n-1}) \leftarrow$ the binary decomposition of $s$ ($b_0$ being the least significant bit and
   assuming $n$ is even) ;                          // This is also a statement to prove
2  $R, T \in E_0$; $A \leftarrow [2]P$; $B \leftarrow A + P$;
3  $R_x \leftarrow$ Lookup2$(b_{n-1}, b_{n-2}, 0, P_x, A_x, B_x)$;
4  $R_y \leftarrow$ Lookup2$(b_{n-1}, b_{n-2}, 1, P_y, A_y, B_y)$;
5  for $b_i$ from $b_{n-3}$ downto $b_0$, $i \leftarrow i - 2$:
6      $R \leftarrow [4]R$;
7      $T_x \leftarrow$ Lookup2$(b_i, b_{i-1}, 0, P_x, A_x, B_x)$;
8      $T_y \leftarrow$ Lookup2$(b_i, b_{i-1}, 1, P_y, A_y, B_y)$;
9      $R \leftarrow R + T$;
10 return $R$;
```

In [MSZ21] Masson et al. performed an exhaustive search of a twisted Edwards curve defined over the BLS12-381 with small Complex Multiplication discriminant in order to speed up the scalar multiplication using a fast endomorphism. They found an isolated curve with the discriminant $D = 8$, called Bandersnatch. This curve has $a \neq -1$ and is not a square, hence the formulae are incomplete. However, all infinity points are of even order and checking that the points are of the right odd prime order is enough to rule out exceptions [HWCD08, Th. 1]. All SNARK curves can have associated twisted Edwards curves but it is unlikely to find such a curve with a small CM discriminant. In table 11, we gather the parameters of the proposed twisted Edwards associated to some SNARK curves discussed in this paper. Note that the multiplication by the curve coefficients is free in the R1CS model as they are constants with respect to the statement.

7 Conclusion

The application of elliptic curves in modern zero-knowledge proof systems has energized research efforts well beyond traditional elliptic curve cryptography. The initiative already developed practical solutions to several challenging problems: how to instantiate recursive proof systems with 2-cycles of MNT pairing-friendly curves, how to use 2-chains to trade-off generality (as in the number of composition layers) for efficiency, and how to use 2-cycles of plain curves to further trade-off proof performance for deployability (transparent setup).

In this survey, we present an overview of elliptic curves generated for SNARKs belonging to one of the three classes above. We summarize the mathematical background and revisit the efficiency and security requirements for (pairing-friendly) elliptic curves within SNARKs. We then collect results and examples from the literature of 2-cycles and 2-chains of elliptic curves, while introducing novel improvements for several families in the latter case. In particular, we generalize the 2-chain framework to include BN curves, improve final exponentiation for BW6 curves, and propose a way to obtain higher 2-valuation for BLS24 curves.

Table 11. Associated twisted Edwards curves to some SNARK curves.

	associated twisted Edwards curve $ax^2 + y^2 = 1 + dx^2y^2$			
	a=-1?	curve order	twist cofactor	fast endomorphism?
BN254 (Ethereum)	✓	$8 \times$ (251-bit prime)	4	✗
BLS12-381	✓ (Jubjub)	$8 \times$ (252-bit prime)	4	✗
	✗($a = -5$) (Bandersnatch)	$4 \times$ (253-bit prime)	$2^7 \cdot 3^3$	✓
BLS24-317	✓	$8 \times$ (250-bit prime)	1	✗
BLS12-377	✓	$4 \times$ (251-bit prime)	8	✗
BW6-761	✓	$8 \times$ (374-bit prime)	4	✗
BLS24-315	✓	$8 \times$ (250-bit prime)	4	✗
BW6-633	✓	$8 \times$ (312-bit prime)	4	✗
MNT4-753	✓	$8 \times$ (750-bit prime)	4	✗
MNT4-298	✓	$4 \times$ (296-bit prime)	8	✗

Still, a number of questions linger: Are there more efficient cycles of pairing-friendly curves? How to generate them? What are the optimal choices for 2-chains at higher security levels? Are there more efficient constructions of 2-chains with smaller outer curves? We hope that our comprehensive overview attracts new interest towards this topic, and that further developments translate into practical impact through one of the many frameworks for theoretical research or practical deployment in this space.

A Implementations

We report in Table 12 some libraries that implement different SNARK curves, 2-chains and 2-cycles. We only cite implementations that are used in zero-knowledge proofs based projects and we omit to cite forks that improve independently over the original work. The libraries are implemented in different languages and some use more assembly acceleration than others. Besides the different algorithmic and software optimizations used across them, it should also be noted that some libraries target constant-time implementations for some or all the operations.

Note. Libraries in Table 12 provide the classical implementation of elliptic curves. Few of these libraries provide also implementations of curves as SNARK computations. That is, the arithmetic of fields and groups of elliptic curves as statements to be proved in a SNARK using another elliptic curve (e.g. Alg. 17 for twisted Edwards). For example arkworks [aC22], gnark [BPEH+22], libsnark [BSCT+b] and zcash [BS] provide such implementations within different proving systems.

[4] gnark-crypto and arkworks implement for each SNARK curve an associated twisted Edwards curve, including Jubjub and Bandersnatch.

Table 12. Some implementations of SNARK curves.

Library	curves implemented	programming language	license	zero-knowledge projects
arkworks-rs [aC22]	BN254 BLS12-381 BLS12-377/BW6-761 chain MNT753 cycle MNT298 cycle Pasta cycle (and all associated curves[4])	Rust	MIT Apache-2.0	arkworks-rs Celo Aleo
Barretenberg [Wil]	BN254	C++		Aztec rollup
blst [Sup]	BLS12-381	C	Apache-2.0	Filecoin
constantine [AR]	BN254 BLS12-381 BLS12-377 Jubjub, Bandersnatch Curve25519	Nim	MIT Apache-2.0	Status-Ethereum (ongoing adoption)
Dalek [dVYA22]	Ed25519 Curve25519	Rust	BSD-3-Clause	Dalek-bulletproofs Spartan
Geth (Cloudflare)	BN254	Go	LGPL-3.0	Geth Erigon
gnark-crypto[4] [BPEH$^+$]	BN254 BLS12-381 BLS24-317 BLS12-377/BW6-761 chain BLS24-315/BW6-633 chain (and all associated curves[4])	Go	Apache-2.0	gnark ConsenSys Rollup Baseline protocol geth (Fuzzing)
Kilic [Kil]	BN254 BLS12-381 BLS12-377/BW6-761 chain	Go	Apache-2.0	Geth Celo
libff [BSCT$^+$a]	BN254 BLS12-381 GMV6-183 MNT298 cycle	C++	MIT	Libsnark Loopring
mcl [Shi]	BN254 BLS12-381	C++	BSD-3-Clause	DFINITY
RELIC [AGM$^+$]	BN254 BLS12-377 BLS12-381 BLS24-315 Tweedle/Pasta cycles	C	Apache-2.0 LGPL-2.1	Chia Network
wasmcurves [Bay]	BN254 BLS12-381	JavaScript, WASM	GPL-3	Circom/snarkjs Polygon Hermez rollup
zcash [BS,Zca]	BN254 BLS12-381 Jubjub Pasta cycle	Rust	MIT Apache-2.0	Zcash Dusk Algorand OpenEthereum zkSync rollup

B Parameter Tables

Table 13. Parameters of a BW6 outer curve with a BLS12 inner curve, $x \equiv 1 \bmod 3$.

parameter	value	property
Barreto–Lynn–Scott $k = 12$ curve (BLS12) $E_1/\mathbb{F}_p,\ r \mid \#E_1(\mathbb{F}_p)$		
r	$x^4 - x^2 + 1$	$p(x),\ r(x)$ generate prime,
p	$(x-1)^2/3(x^4 - x^2 + 1) + x$	$p(u),\ r(u)$ are prime
Brezing–Weng $k = 6$ curve (BW6) $E_2/\mathbb{F}_{s_i},\ p \mid \#E_2(\mathbb{F}_{s_i})$		
parameters modulo the subgroup order p		
ζ_6	$x^5 - 3x^4 + 3x^3 - x + 2$	
$\text{conjugate}(\zeta_6) = 1 - \zeta_6$	$-x^5 + 3x^4 - 3x^3 + x - 1$	
$1/\sqrt{-3}$	$-(2x^5 - 6x^4 + 6x^3 - 2x + 3)/3$	
$t_0 = \text{conj}(\zeta_6) + 1$	$-x^5 + 3x^4 - 3x^3 + x$	$6 \mid t_0$
$t_3 = \zeta_6 + 1$	$x^5 - 3x^4 + 3x^3 - x + 3$	$3 \mid t_3, 2 \nmid t_3$
$y_0 = (t_0 - 2)/\sqrt{-3}$	$(x^5 - 3x^4 + 3x^3 - x)/3 = -t_0/3$	$2 \mid y_0$
$y_3 = (t_3 - 2)/\sqrt{-3}$	$(x^5 - 3x^4 + 3x^3 - x + 3)/3 = t_3/3$	$2 \nmid y_3$
with lifting parameters h_t, h_y		
s_0	$((t_0 + h_t p)^2 + 3(y_0 + h_y p)^2)/4$	generates prime
s_3	$((t_3 + h_t p)^2 + 3(y_3 + h_y p)^2)/4$	generates prime
$\mathbb{G}_1$-cofactor c_i such that $\#E_2(\mathbb{F}_{s_i}) = c_i \cdot p$		
$\Phi_6(t_i - 1)$	$3p(x^4 - 4x^3 + 7x^2 - 6x + 3)$	
c_0	$(h_t^2 + 3h_y^2)/4p + (h_t - h_y)/2t_0 + x^4 - 4x^3 + 7x^2 - 6x + 3 - h_t$	
c_3	$(h_t^2 + 3h_y^2)/4p + (h_t + h_y)/2t_3 + x^4 - 4x^3 + 7x^2 - 6x + 3 - h_t$	
$\mathbb{G}_2$-cofactor c'_i such that $\#E'_2(\mathbb{F}_{s_i}) = c'_i \cdot p$		
$c'_0\ (\mathbb{G}_2)$	$c_0 + (h_t + 3h_y)/2$	
$c'_3\ (\mathbb{G}_2)$	$c_3 + (h_t - 3h_y)/2$	
Optimal ate Miller loop $f_{a_0,Q}(P) \cdot f^p_{a_1,Q}(P)$, Optimal twisted ate $f_{a_0+a_1\lambda,P}(Q)$		
$\lambda = t_0 - 1$	$a_0 = x^3 - x^2 - x,\ a_1 = x + 1 \quad a'_0 = -(x+1),\ a'_1 = x^3 - x^2 + 1$	
$\lambda = t_3 - 1$	$a_0 = x + 1,\ a_1 = x^3 - x^2 - x \quad a'_0 = x^3 - x^2 + 1,\ a'_1 = -(x+1)$	
Final exponentiation in $\mathbb{F}_{s_i}$, exponent e_i multiple of $\Phi_6(s_i)/p$		
$e_0 = (x+1)\Phi_6(s_0)/p$	$((x+1)s_0 - x^3 + x^2 - 1)(c_0 + h_t) + 3(s_0 - x^2 + 2x - 2)$	
$e'_0 = (x^3 - x^2 - x)\Phi_6(s_0)/p$	$((x^3 - x^2 - x)s_0 + x + 1)(c_0 + h_t) - 3(1 + (x^2 - 2x + 1)s_0)$	
$e_3 = (x+1)\Phi_6(s_3)/p$	$(x^3 - x^2 - x + (x+1)s_3)(c_3 + h_t) + 3(x^2 - 2x + 1 + s_3)$	
$e'_3 = (x^3 - x^2 + 1)\Phi_6(s_3)/p$	$((x^3 - x^2 + 1)s_3 - x - 1)(c_3 + h_t) - 3(1 - (x^2 - 2x + 2)s_3)$	

C Optimized final exponentiation for our BW outer curves with inner BN, BLS12, and BLS24

Remember from Fig. 4 that the inner curves BN, BLS12 and BLS24 are called E_1, defined over a prime field $\mathbb{F}_p$, and have a subgroup of prime order r (the scalar field). The BW6 outer curves are called E_2, have a subgroup of prime order p (the scalar field of the second SNARK is the coefficient field of the first SNARK), and are defined over a new prime field $\mathbb{F}_s$.

Table 14. Parameters of a BW6 outer curve with a BLS24 inner curve, $x \equiv 1 \bmod 3$.

Barreto–Lynn–Scott $k = 24$ curve (BLS24) $E_1/\mathbb{F}_p,\ r \mid \#E_1(\mathbb{F}_p)$		
r	$x^8 - x^4 + 1$	
p	$(x-1)^2/3(x^8 - x^4 + 1) + x$	prime
	$= (x^{10} - 2x^9 + x^8 - x^6 + 2x^5 - x^4 + x^2 + x + 1)/3$	
Brezing–Weng $k = 6$ curve (BW6) $E_2/\mathbb{F}_{s_i},\ p \mid \#E_2(\mathbb{F}_{s_i})$		
parameters modulo the subgroup order p		
ζ_6	$x^9 - 3x^8 + 4x^7 - 4x^6 + 3x^5 - 2x^3 + 2x^2 - x + 2$	
$\mathrm{conj}(\zeta_6)$	$-x^9 + 3x^8 - 4x^7 + 4x^6 - 3x^5 + 2x^3 - 2x^2 + x - 1$	
$1/\sqrt{-3}$	$(2x^9 - 6x^8 + 8x^7 - 8x^6 + 6x^5 - 4x^3 + 4x^2 - 2x + 3)/3$	
$t_0 = \mathrm{conj}(\zeta_6) + 1$	$-x^9 + 3x^8 - 4x^7 + 4x^6 - 3x^5 + 2x^3 - 2x^2 + x$	$6 \mid t_0$
$t_3 = \zeta_6 + 1$	$x^9 - 3x^8 + 4x^7 - 4x^6 + 3x^5 - 2x^3 + 2x^2 - x + 3$	$3 \mid t_3, 2 \nmid t_3$
$y_0 = (t_0 - 2)/\sqrt{-3}$	$(x^9 - 3x^8 + 4x^7 - 4x^6 + 3x^5 - 2x^3 + 2x^2 - x)/3 = -t_0/3$	$2 \mid y_0$
$y_3 = (t_3 - 2)/\sqrt{-3}$	$(x^9 - 3x^8 + 4x^7 - 4x^6 + 3x^5 - 2x^3 + 2x^2 - x + 3)/3 = t_3/3$	$2 \nmid y_3$
with lifting parameters h_t, h_y		
s_0	$((t_0 + h_t p)^2 + 3(y_0 + h_y p)^2)/4$	prime
s_3	$((t_3 + h_t p)^2 + 3(y_3 + h_y p)^2)/4$	prime
$\mathbb{G}_1$-cofactor c_i such that $\#E_2(\mathbb{F}_{s_i}) = c_i \cdot p$		
$\Phi_6(t_i - 1)$	$(x^8 - 4x^7 + 8x^6 - 12x^5 + 15x^4 - 14x^3 + 10x^2 - 6x + 3) \cdot 3 \cdot p$	
c_0	$(h_t^2 + 3h_y^2)/4p + (h_t - h_y)/2t_0 + \Phi_6(t_0 - 1)/(3p) - h_t$	
c_3	$(h_t^2 + 3h_y^2)/4p + (h_t + h_y)/2t_3 + \Phi_6(t_3 - 1)/(3p) - h_t$	
$\mathbb{G}_2$-cofactor c_i' such that $\#E_2'(\mathbb{F}_{s_i}) = c_i' \cdot p$		
c_0' ($\mathbb{G}_2$)	$c_0 + (h_t + 3h_y)/2$	
c_3' ($\mathbb{G}_2$)	$c_3 + (h_t - 3h_y)/2$	
Optimal ate Miller loop $f_{a_0,Q}(P) \cdot f_{a_1,Q}^p(P)$, Optimal twisted ate $f_{a_0+a_1\lambda,P}(Q)$		
$\lambda = t_0 - 1$	$a_0 = -(x+1),\ a_1 = x^5 - x^4 + 1 \quad a_0' = x^5 - x^4 - x,\ a_1' = x + 1$	
$\lambda = t_3 - 1$	$a_0 = x + 1,\ a_1 = x^5 - x^4 - x \quad a_0' = x^5 - x^4 + 1,\ a_1' = -(x+1)$	
Final exponentiation in $\mathbb{F}_{s_i}$, exponent e_i multiple of $\Phi_6(s_i)/p$		
e_0	$(x+1)\Phi_6(s_0)/p =$	
	$(-x^5 + x^4 - 1 + (x+1)s_0)(c_0 + h_t) + 3(x^4 + 2(-x^3 + x^2 - x + 1) - s_0)$	
e_0'	$(x^5 - x^4 - x)\Phi_6(s_0)/p =$	
	$((x+1) + (x^5 - x^4 - x)s_0)(c_0 + h_t) - 3(1 + (x^4 - 2x^3 + 2x^2 - 2x + 1)s_0)$	
e_3	$(x+1)\Phi_6(s_3)/p =$	
	$(x(x^4 - x^3 - 1) + (x+1)s_3)(c_3 + h_t) + 3(x^4 + 2(-x^3 + x^2 - x) + 1 + s_3)$	
e_3'	$(x^5 - x^4 + 1)\Phi_6(s_3)/p =$	
	$((-x - 1) + (x^5 - x^4 + 1)s_3)(c_3 + h_t) - 3(1 - (x^4 - 2x^3 + 2x^2 - 2x + 2)s_3)$	

Table 15. Parameters of a BW6 outer curve with a BN inner curve, any integer x.

Barreto–Naehrig $k = 12$ curve (BN) $E_1/\mathbb{F}_p,\ r \mid \#E_1(\mathbb{F}_p)$		
r	$36x^4 + 36x^3 + 18x^2 + 6x + 1$	prime
p	$36x^4 + 36x^3 + 24x^2 + 6x + 1$	prime
Brezing–Weng $k = 6$ curve (BW6) $E_2/\mathbb{F}_{s_i},\ p \mid \#E_2(\mathbb{F}_{s_i})$		
parameters modulo the subgroup order p		
ζ_6	$18x^3 + 18x^2 + 9x + 2$	
$\mathrm{conj}(\zeta_6)$	$-18x^3 - 18x^2 - 9x - 1$	
$1/\sqrt{-3}$	$12x^3 + 12x^2 + 6x + 1$	
$t_0 = \mathrm{conj}(\zeta_6) + 1$	$-18x^3 - 18x^2 - 9x$	$9 \mid t_0$
$t_3 = \zeta_6 + 1$	$18x^3 + 18x^2 + 9x + 3$	$3 \mid t_3$
$y_0 = (t_0 - 2)/\sqrt{-3}$	$-t_0/3 = 6x^3 + 6x^2 + 3x$	$3 \mid y_0$
$y_3 = (t_3 - 2)/\sqrt{-3}$	$t_3/3 = 6x^3 + 6x^2 + 3x + 1$	
with lifting parameters h_t, h_y		
s_0	$((t_0 + h_t p)^2 + 3(y_0 + h_y p)^2)/4$	prime
s_3	$((t_3 + h_t p)^2 + 3(y_3 + h_y p)^2)/4$	prime
$\mathbb{G}_1$-cofactor c_i such that $\#E_2(\mathbb{F}_{s_i}) = c_i \cdot p$		
$\Phi_6(t_{\mathrm{bw},i} - 1)$	$(9x^2 + 9x + 3) \cdot p$	
c_0	$(h_t^2 + 3h_y^2)/4p + (h_t - h_y)/2t_0 + \Phi_6(t_0 - 1)/(3p) - h_t$	
c_3	$(h_t^2 + 3h_y^2)/4p + (h_t + h_y)/2t_3 + \Phi_6(t_3 - 1)/(3p) - h_t$	
$\mathbb{G}_2$-cofactor c'_i such that $\#E'_2(\mathbb{F}_{s_i}) = c'_i \cdot p$		
c'_0 ($\mathbb{G}_2$)	$c_0 + (h_t + 3h_y)/2$	
c'_3 ($\mathbb{G}_2$)	$c_3 + (h_t - 3h_y)/2$	
Optimal ate Miller loop $f_{a_0,Q}(P) \cdot f^p_{a_1,Q}(P)$, Optimal twisted ate $f_{a_0+a_1\lambda,P}(Q)$		
$\lambda = t_0 - 1$	$a_0 = 2x,\ a_1 = 6x^2 + 2x + 1 \quad a'_0 = 6x^2 + 4x + 1,\ a'_1 = -2x$	
$\lambda = t_3 - 1$	$a_0 = 6x^2 + 2x + 1,\ a_1 = 2x \quad a'_0 = -2x,\ a'_1 = 6x^2 + 4x + 1$	
Final exponentiation in $\mathbb{F}_{s_i}$, exponent e_i multiple of $\Phi_6(s_i)/p$		
$e_0 = 2x\Phi_6(s_0)/p = (6x^2 + 2x + 1 + 2xs_0)(c_0 + h_t) - (3x + 1 + s_0)$		
$e'_0 = (6x^2 + 4x + 1)\Phi_6(s_0)/p = (-2x + (6x^2 + 4x + 1)s_0)(c_0 + h_t) + 1 - (3x + 2)s_0$		
$e_3 = 2x\Phi_6(s_3)/p = (-(6x^2 + 4x + 1) + 2xs_3)(c_3 + h_t) - (3x + 2) + s_3$		
$e'_3 = (6x^2 + 2x + 1)\Phi_6(s_3)/p = (2x + (6x^2 + 2x + 1)s_3)(c_3 + h_t) + 1 + (3x + 1)s_3$		

BN-BW6 curves. A lattice reduction (with Magma on polynomials) gives the formulas in Tab. 15, where $d = \Phi_6(t_i - 1)/(3p) = 3x(x+1)+1$. We highlight with an underbrace the similar parts that can be shared. The sequential steps are $t_0 = -3x(2d+1)$ then $p = 2(-xt_0 + d) - 1$ in Alg. 9 to compute m^{e_0}, and $t_3 = 3(x(2d+1)+1)$, then $p = 2x(t_3 + 3x) + 1$ in Alg. 10 to compute $m^{e_3'}$.

$$e_0 = (2x)\tfrac{\Phi_6(s_0)}{p} = \overbrace{\left(1 + 2x\underbrace{(3x+1+s_0)}_{a_0}\right)}^{b_0}\left(\tfrac{h_t^2+3h_y^2}{4}p + \tfrac{h_t-h_y}{2}t_0 + d\right) - \underbrace{(3x+1+s_0)}_{a_0}$$

$$e_0' = (6x^2+4x+1)\tfrac{\Phi_6(s_0)}{p} = \overbrace{\left(s_0 - 2x\underbrace{(1-(3x+2)s_0)}_{a_0'}\right)}^{b_0'}\left(\tfrac{h_t^2+3h_y^2}{4}p + \tfrac{h_t-h_y}{2}t_0 + d\right) + \underbrace{1-(3x+2)s_0}_{a_0'}$$

$$e_3 = (2x)\tfrac{\Phi_6(s_3)}{p} = \overbrace{\left(2x\underbrace{(s_3-(3x+2))}_{a_3}-1\right)}^{b_3}\left(\tfrac{h_t^2+3h_y^2}{4}p + \tfrac{h_t+h_y}{2}t_3 + d\right) + \underbrace{s_3-(3x+2)}_{a_3}$$

$$e_3' = (6x^2+2x+1)\tfrac{\Phi_6(s_3)}{p} = \overbrace{\left(2x\underbrace{(1+(3x+1)s_3)}_{a_3'}+s_3\right)}^{b_3'}\left(\tfrac{h_t^2+3h_y^2}{4}p + \tfrac{h_t+h_y}{2}t_3 + d\right) + \underbrace{1+(3x+1)s_3}_{a_3'}$$

BLS12-BW6 curves. Parameters are in Tab. 13, where $d = \Phi_6(t_i - 1)/(3p) = (x^4 - 4x^3 + 7x^2 - 6x + 3)$. We highlight with an underbrace the exponent a_i, a_i' that can be shared.

$$e_0 = (x+1)\tfrac{\Phi_6(s_0)}{p} = \overbrace{\left((x+1)\underbrace{(s_0-(x-1)^2-1)}_{a_0}+1\right)}^{b_0}\left(\tfrac{h_t^2+3h_y^2}{4}p+3\left(\tfrac{h_t-h_y}{2}\tfrac{t_0}{3}+\tfrac{d-1}{3}\right)+1\right)-3\underbrace{(s_0-(x-1)^2-1)}_{a_0}$$

$$e_0' = (x^3-x^2-x)\tfrac{\Phi_6(s_0)}{p} = \overbrace{\left((x+1)\underbrace{((x-1)^2s_0+1)}_{a_0'}-p\right)}^{b_0'}\left(\tfrac{h_t^2+3h_y^2}{4}p+3\left(\tfrac{h_t-h_y}{2}\tfrac{t_0}{3}+\tfrac{d-1}{3}\right)+1\right)-3\underbrace{((x-1)^2s_0+1)}_{a_0'}$$

$$e_3 = (x+1)\tfrac{\Phi_6(s_3)}{p} = \overbrace{\left((x+1)\underbrace{((x-1)^2+s_3)}_{a_3}-1\right)}^{b_3}\left(\tfrac{h_t^2+3h_y^2}{4}p+3\left(\tfrac{h_t+h_y}{2}\tfrac{t_3}{3}+\tfrac{d-1}{3}\right)+1\right)+3\underbrace{((x-1)^2+s_3)}_{a_3}$$

$$e_3' = (x^3-x^2+1)\tfrac{\Phi_6(s_3)}{p} = \overbrace{\left((x+1)\underbrace{(((x-1)^2+1)s_3-1)}_{a_3'}-s_3\right)}^{b_3'}\left(\tfrac{h_t^2+3h_y^2}{4}p+3\left(\tfrac{h_t+h_y}{2}\tfrac{t_3}{3}+\tfrac{d-1}{3}\right)+1\right)+3\underbrace{(((x-1)^2+1)s_3-1)}_{a_3'}$$

The steps in Alg. 11 for $m^{e_0'}$ correspond to the exponents

$$\begin{array}{lll}
a = (x-1)/3 & & \\
b = a(x-1) & & = (x-1)^2/3 \\
c = b((x-1)^2+1) & = (d-1)/3 & = (x-1)^2/3(x^2-2x+2) \\
e = -(u+1)c + b - a & = \quad t_0/3 & = (x-1)^2/3(-x^3+x^2-1) - (x-1)/3 \\
f = -(u+1)(e+b) + a + 1 = & \quad p & = (x-1)^2/3(x^4-x^2+1) + x
\end{array}$$

The steps in Alg. 12 for m^{e_3} correspond to this sequence deduced from the former, with $t_3/3 = -t_0 + 1$.

$$\begin{aligned}
a &= (x-1)/3 && \\
b &= a(x-1) && = (x-1)^2/3 \\
c &= b((x-1)^2+1) && = (d-1)/3 \\
b' &= -b && \\
e &= b'+1 && \\
f &= c(x+1)+e && = (x-1)^2/3(x^3-x^2+1)+1 \\
g &= f+a && = t_3/3 \\
h &= (f+b')(x+1)-e = p &&
\end{aligned}$$

BLS24-BW6 curves. The exponents of the hard part of the final exponentiation for BW6-BLS24 curves are the following, with $i_0 = (x-1)^2(x^2+1)$, $i'_0 = (x-1)^2(x^2+1)+1 = i_0+1$, $i_3 = (x-1)^2(x^2+1)$, and $i'_3 = (x-1)^2(x^2+1)+1 = i_3+1$.

$$e_0 = (x^5-x^4-x)\tfrac{\Phi_6(s_0)}{p} = \Big(\overbrace{(x+1)\underbrace{(1+i_0s_0)}_{a_0}-s_0}^{b_0}\Big)\Big(\tfrac{h_t^2+3h_y^2}{4}p + 3\big(\tfrac{h_t-h_y}{2}\tfrac{t_0}{3}+\tfrac{d-1}{3}\big)+1\Big) - 3\underbrace{(1+i_0s_0)}_{a_0}$$

$$e'_0 = (x+1)\tfrac{\Phi_6(s_0)}{p} = \Big(\overbrace{(x+1)\underbrace{(s_0-i'_0)}_{a'_0}+1}^{b'_0}\Big)\Big(\tfrac{h_t^2+3h_y^2}{4}p + 3\big(\tfrac{h_t-h_y}{2}\tfrac{t_0}{3}+\tfrac{d-1}{3}\big)+1\Big) - 3\underbrace{(s_0-i'_0)}_{a_0}$$

$$e_3 = (x+1)\tfrac{\Phi_6(s_3)}{p} = \Big(\overbrace{(x+1)\underbrace{(i_3+s_3)}_{a_3}-1}^{b_3}\Big)\Big(\tfrac{h_t^2+3h_y^2}{4}p + 3\big(\tfrac{h_t+h_y}{2}\tfrac{t_3}{3}+\tfrac{d-1}{3}\big)+1\Big) + 3\underbrace{(i_3+s_3)}_{a_3}$$

$$e'_3 = (x^5-x^4+1)\tfrac{\Phi_6(s_3)}{p} = \Big(\overbrace{(x+1)\underbrace{(i'_3s_3-1)}_{a'_3}-p}^{b'_3}\Big)\Big(\tfrac{h_t^2+3h_y^2}{4}r + 3\big(\tfrac{h_t+h_y}{2}\tfrac{t_3}{3}+\tfrac{d-1}{3}\big)+1\Big) + 3\underbrace{(i'_3s_3-1)}_{a'_3}$$

The parameters are given in Tab. 14, and we set $d = \Phi_6(t_i-1)/(3p) = x^8-4x^7+8x^6-12x^5+15x^4-14x^3+10x^2-6x+3$. Part of the exponent is

$$\tfrac{h_t^2+3h_y^2}{4}p + \tfrac{h_t+h_y}{2}t_3 + d \quad ; \quad \tfrac{h_t^2+3h_y^2}{4}p + \tfrac{h_t-h_y}{2}t_0 + d$$

We compute the exponents p, $t_0/3$, $t_3/3$ and $(d-1)/3$ as follows and obtain Alg. 13 and Alg. 14.

$a = (x-1)/3$	$a = (x-1)/3$
$b = a(x-1)(x^2+1)$	$b = a(x-1)(x^2+1)$
$c = b((x-1)^2(x^2+1)+1) \quad = (d-1)/3$	$c = b((x-1)^2(x^2+1)+1) = (d-1)/3$
	$e = (x+1)c-b+a$
$f = -(x+1)c+b-a \quad = t_0/3$	$f = e+1 \quad = t_3/3$
$g = -(x+1)(f+b)+a+1 = p$	$g = (x+1)(e-b)+a+1 \quad = p$

Data availability (Data Deposition Information) Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

References

aC22. arkworks Contributors. `arkworks` zkSNARK ecosystem. `https://arkworks.rs`, 2022.

AGM$^+$. D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. `https://github.com/relic-toolkit/relic`.

AR. Mamy André-Ratsimbazafy. Constant time pairing-based or elliptic curve based cryptography and digital signatures. `https://github.com/mratsim/constantine`.

Bay. Jordi Baylina. Web assembly low level implementation of pairing friendly curves. `https://github.com/iden3/wasmcurves`.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BCC$^+$16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

BCCT12. Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In Shafi Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, January 2012.

BCG$^+$13. Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 90–108. Springer, Heidelberg, August 2013.

BCG$^+$14. Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

BCG$^+$20. Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. ZEXE: Enabling decentralized private computation. In *2020 IEEE Symposium on Security and Privacy*, pages 947–964. IEEE Computer Society Press, May 2020.

BCKL21. Eli Ben-Sasson, Dan Carmon, Swastik Kopparty, and David Levit. Elliptic curve fast fourier transform (ECFFT) part I: fast polynomial algorithms over all finite fields. *CoRR*, abs/2107.08473, 2021.

BCMS20. Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive proof composition from accumulation schemes. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 1–18. Springer, Heidelberg, November 2020.

BCTV14a. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 276–294. Springer, Heidelberg, August 2014.

BCTV14b. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 781–796. USENIX Association, August 2014.

BD19. Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. *Journal of Cryptology*, 32(4):1298–1336, October 2019.

BDFG21. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 649–680, Virtual Event, August 2021. Springer, Heidelberg.

BDL$^+$12. Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012.

BDLO12. Daniel J. Bernstein, Jeroen Doumen, Tanja Lange, and Jan-Jaap Oosterwijk. Faster batch forgery identification. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 454–473. Springer, Heidelberg, December 2012.

BFR$^+$13. Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. Verifying computations with state. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 341–357, New York, NY, USA, 2013. Association for Computing Machinery. ePrint with major differences at `ePrint 2013/356`.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

BGGM15. Razvan Barbulescu, Pierrick Gaudry, Aurore Guillevic, and François Morain. Improving NFS for the discrete logarithm problem in non-prime finite fields. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 129–155. Springer, Heidelberg, April 2015.

BGH19. Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021, 2019. `https://eprint.iacr.org/2019/1021`.

BGJT14. Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 1–16. Springer, Heidelberg, May 2014.

BGK15. Razvan Barbulescu, Pierrick Gaudry, and Thorsten Kleinjung. The tower number field sieve. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 31–55. Springer, Heidelberg, November / December 2015.

BGM$^+$10. Jean-Luc Beuchat, Jorge E. González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal Ate pairing over Barreto-Naehrig curves. In Marc Joye, Atsuko Miyaji, and Akira Otsuka, editors, *PAIRING 2010*, volume 6487 of *LNCS*, pages 21–39. Springer, Heidelberg, December 2010.

BGN05. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, Heidelberg, February 2005.

BHKL13. Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 967–980. ACM Press, November 2013.

BL. Daniel J. Bernstein and Tanja Lange. Safecurves: choosing safe curves for elliptic-curve cryptography. `https://safecurves.cr.yp.to`, accessed 28 February 2022.

BLS01. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

BLS04. Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 17–25. Springer, Heidelberg, August 2004.

BMRS20. Joseph Bonneau, Izaak Meckler, Vanishree Rao, and Evan Shapiro. Coda: Decentralized cryptocurrency at scale. Cryptology ePrint Archive, Report 2020/352, 2020. `https://eprint.iacr.org/2020/352`.

BN06. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006.

Bow17. Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction. Zcash blog, March 11 2017. `https://blog.z.cash/new-snark-curve/`.

BPEH$^+$. Gautam Botrel, Thomas Piellard, Youssef El Housni, Arya Tabaie, and Ivo Kubjas. Go library for finite fields, elliptic curves and pairings for zero-knowledge proof systems. `https://doi.org/10.5281/zenodo.6092968`.

BPEH$^+$22. Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas, and Arya Tabaie. Consensys/gnark. `https://doi.org/10.5281/zenodo.6093969`, February 2022.

BS. Sean Bowe and Str4d. Zero-knowledge Cryptography in Rust. `https://github.com/zkcrypto`.

BSCT$^+$a. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, Madars Virza, Howard Wu, and Contributors. C++ library for finite fields and elliptic curves. `https://github.com/scipr-lab/libff`.

BSCT$^+$b. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, Madars Virza, Howard Wu, and Contributors. C++ library for zksnark. `https://github.com/scipr-lab/libsnark`.

BSS$^+$13. Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. `https://eprint.iacr.org/2013/404`.

CCW19. Alessandro Chiesa, Lynn Chua, and Matthew Weidner. On cycles of pairing-friendly elliptic curves. *SIAM Journal on Applied Algebra and Geometry*, 3(2):175–192, 2019.

CFH$^+$15. Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 253–270. IEEE Computer Society, 2015. ePrint 2014/976.

Che10. Jung Hee Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, July 2010.

CHM$^+$20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

CHZ22. Shi Ping CAI, Zhi HU, and Chang An ZHAO. Faster final exponentiation on the kss18 curve. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E105.A(8):1162–1164, 2022.

Cos12. Craig Costello. Pairings for beginners. https://www.craigcostello.com.au/s/PairingsForBeginners.pdf, 2012.

CSRHT22. Jorge Chávez-Saab, Francisco Rodríguez-Henríquez, and Mehdi Tibouchi. Swiftec: Shallue–van de woestijne indifferentiable function to elliptic curves. Cryptology ePrint Archive, Paper 2022/759, 2022. To appear in ASIACRYPT 2022.

DFKP16. Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. Cinderella: Turning shabby X.509 certificates into elegant anonymous credentials with the magic of verifiable computation. In *2016 IEEE Symposium on Security and Privacy*, pages 235–254. IEEE Computer Society Press, May 2016.

DGP20. Gabrielle De Micheli, Pierrick Gaudry, and Cécile Pierrot. Asymptotic complexities of discrete logarithm algorithms in pairing-relevant finite fields. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 32–61. Springer, Heidelberg, August 2020.

DGP21. Gabrielle De Micheli, Pierrick Gaudry, and Cécile Pierrot. Lattice enumeration for tower NFS: a 521-bit discrete logarithm computation. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, volume 13090 of *LNCS*, pages 67–96. Springer, 2021. ePrint `2021/707`.

dV21. Henry de Valence. The ristretto group. `https://ristretto.group`, 2021.

dVYA22. Henry de Valence, Cathie Yun, and Oleg Andreev. dalek cryptography: Fast, sage, pure-rust elliptic curve cryptography, 2022. `https://github.com/dalek-cryptography/bulletproofs`.

EHG20. Youssef El Housni and Aurore Guillevic. Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings*, volume 12579 of *LNCS*, pages 259–279. Springer, 2020.

EHG22a. Youssef El Housni and Aurore Guillevic. Families of SNARK-friendly 2-chains of elliptic curves. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022*, volume 13276 of *LNCS*, pages 367–396. Springer, 2022. ePrint `2021/1359`.

EHG22b. Youssef El Housni and Aurore Guillevic. Families of SNARK-friendly 2-chains of elliptic curves. `https://gitlab.inria.fr/zk-curves/snark-2-chains`, 2022. SageMath/Python and Magma implementation.

ES10. Andreas Enge and Andrew V. Sutherland. Class invariants by the CRT method. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *Algorithmic Number Theory Symposium*, pages 142–156, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

FHSS$^+$22. Armando Faz-Hernández, Sam Scott, Nick Sullivan, Riad S. Wahby, and Christopher A. Wood. Hashing to Elliptic Curves. Technical report, IETF Secretariat, 2022. Working Draft available at `https://datatracker.ietf.org/doc/draft-irtf-cfrg-hash-to-curve/`.

FK19. Georgios Fotiadis and Elisavet Konstantinou. TNFS resistant families of pairing-friendly elliptic curves. *Theoretical Computer Science*, 800:73–89, 31 December 2019.

FKR12. Laura Fuentes-Castañeda, Edward Knapp, and Francisco Rodríguez-Henríquez. Faster hashing to $\mathbb{G}_2$. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 412–430. Springer, Heidelberg, August 2012.

FNK12. Fujitsu Laboratories, NICT, and Kyushu University. DL record in $\mathbb{F}_{3^{6\cdot 97}}$ of 923 bits (278 dd). NICT press release, June 18, 2012. `http://www.nict.go.jp/en/press/2012/06/18en-1.html`.

Fre10. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, Heidelberg, May / June 2010.

FST10. David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, April 2010.

Gab19. Ariel Gabizon. AuroraLight: Improved prover efficiency and SRS size in a sonic-like system. Cryptology ePrint Archive, Report 2019/601, 2019. `https://eprint.iacr.org/2019/601`.

GF16. Loubna Ghammam and Emmanuel Fouotsa. On the computation of the optimal ate pairing at the 192-bit security level. Cryptology ePrint Archive, Report 2016/130, 2016. `https://eprint.iacr.org/2016/130`.

GGPR13. Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

GKL$^+$21. Robert Granger, Thorsten Kleinjung, Arjen K. Lenstra, Benjamin Wesolowski, and Jens Zumbrägel. Computation of a 30750-bit binary field discrete logarithm. *Math. Comp.*, 90(332):2997–3022, 2021. ePrint `2020/965`.

GKM$^+$18. Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

GKZ14. Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. Breaking '128-bit secure' supersingular binary curves - (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$). In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 126–145. Springer, Heidelberg, August 2014.

GLV01. Robert P. Gallant, Robert J. Lambert, and Scott A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 190–200. Springer, Heidelberg, August 2001.

GM16. Aurore Guillevic and François Morain. *Pairings for Engineers*, chapter 9 – Discrete Logarithms, pages 203–242. CRC Press Taylor and Francis group, Spring 2016. N. ElMrabet and M. Joye (eds), `https://www.crcpress.com/Guide-to-Pairing-Based-Cryptography/El-Mrabet-Joye/p/book/9781498729505` `https://hal.inria.fr/hal-01420485v2`.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

GMT20. Aurore Guillevic, Simon Masson, and Emmanuel Thomé. Cocks–Pinch curves of embedding degrees five to eight and optimal ate pairing computation. *Des. Codes Cryptogr.*, 88:1047–1081, March 2020.

GMV07. Steven D. Galbraith, James F. McKee, and P. C. Valença. Ordinary abelian varieties having small embedding degree. *Finite Fields Their Appl.*, 13(4):800–814, 2007.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 97–111. Springer, Heidelberg, August 2006.

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

Gro10. Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

GS10. Robert Granger and Michael Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 209–223. Springer, Heidelberg, May 2010.

GS21. Aurore Guillevic and Shashank Singh. On the alpha value of polynomials in the tower number field sieve algorithm. *Mathematical Cryptology*, 1(1), Feb. 2021.

Gui20. Aurore Guillevic. A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 535–564. Springer, Heidelberg, May 2020.

Gui21. Aurore Guillevic. Pairing-friendly curves. `https://members.loria.fr/AGuillevic/pairing-friendly-curves/`, 2021.

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

GWC19. Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. `https://eprint.iacr.org/2019/953`.

Ham15. Mike Hamburg. Decaf: Eliminating cofactors through point compression. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 705–723. Springer, Heidelberg, August 2015.

HHT20. Daiki Hayashida, Kenichiro Hayasaka, and Tadanori Teruya. Efficient final exponentiation via cyclotomic structure for pairings over families of elliptic curves. Cryptology ePrint Archive, Report 2020/875, 2020. `https://eprint.iacr.org/2020/875`.

Hop20. Daira Hopwood. The pasta curves for halo 2 and beyond. `https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/`, 2020.

Hop21. Daira Hopwood. Pluto-eris hybrid cycle of elliptic curves, 2021. `https://github.com/daira/pluto-eris`.

HWCD08. Hüseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted Edwards curves revisited. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 326–343. Springer, Heidelberg, December 2008.

JKS16. Ari Juels, Ahmed E. Kosba, and Elaine Shi. The ring of Gyges: Investigating the future of criminal smart contracts. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 283–295. ACM Press, October 2016.

Jon13. Nathan Jones. Elliptic aliquot cycles of fixed length. *Pacific Journal of Mathematics*, 263(2):353–371, 2013.

Kar13. Koray Karabina. Squaring in cyclotomic subgroups. *Math. Comput.*, 82(281):555–579, 2013.

KB16. Taechan Kim and Razvan Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 543–571. Springer, Heidelberg, August 2016.

Kil. Onur Kilic. High-speed implementation of curves in Go. `https://github.com/kilic/bn254`, `https://github.com/kilic/bls12-381`, `https://github.com/kilic/bls12-377` and `https://github.com/kilic/bw6`.

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992.

KMS$^+$16. Ahmed E. Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858. IEEE Computer Society Press, May 2016.

KPP$^+$14. Ahmed E. Kosba, Dimitrios Papadopoulos, Charalampos Papamanthou, Mahmoud F. Sayed, Elaine Shi, and Nikos Triandopoulos. TRUESET: Faster verifiable set computations. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 765–780. USENIX Association, August 2014.

KPV19. Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. RedShift: Transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400, 2019. `https://eprint.iacr.org/2019/1400`.

KSS08. Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 126–135. Springer, Heidelberg, September 2008.

KT08. Koray Karabina and Edlyn Teske. On prime-order elliptic curves with embedding degrees k = 3, 4, and 6. In Alfred J. van der Poorten and Andreas Stein, editors, *Algorithmic Number Theory, 8th International Symposium, ANTS-VIII, Banff, Canada, May 17-22, 2008, Proceedings*, volume 5011 of *Lecture Notes in Computer Science*, pages 102–117. Springer, 2008.

KW22. Thorsten Kleinjung and Benjamin Wesolowski. Discrete logarithms in quasi-polynomial time in finite fields of fixed characteristic. *J. Amer. Math. Soc.*, 35(02):581–624, Jan 2022. ePrint `2019/751`.

KZG10. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

KZM$^+$15. Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. `https://eprint.iacr.org/2015/1093`.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Mec20. Izaak Meckler. O(1) labs fork of zexe: implementation of bn382-plain, 2020. `https://github.com/o1-labs/zexe/tree/master/algebra/src/bn_382`.

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

Mih07. Preda Mihailescu. Dual elliptic primes and applications to cyclotomy primality proving. arXiv `0709.4113`, 2007.

MMV$^+$14. Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 306–323. Springer, Heidelberg, August 2014.

MNT01. Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under FR-reduction. In Dongho Won, editor, *ICISC 00*, volume 2015 of *LNCS*, pages 90–108. Springer, Heidelberg, December 2001.

MSS16. Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. In Raphael C.-W. Phan and Moti Yung, editors, *Mycrypt Conference*, volume 10311 of *LNCS*, pages 83–108, Kuala Lumpur, Malaysia, December 1-2 2016. Springer. `https://ia.cr/2016/1102`.

MSZ21. Simon Masson, Antonio Sanso, and Zhenfei Zhang. Bandersnatch: a fast elliptic curve built over the bls12-381 scalar field. Cryptology ePrint Archive, Report 2021/1152, 2021. `https://ia.cr/2021/1152`.

NAS$^+$08. Yasuyuki Nogami, Masataka Akane, Yumi Sakemi, Hidehiro Katou, and Yoshitaka Morikawa. Integer variable chi-based Ate pairing. In Steven D.

Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 178–191. Springer, Heidelberg, September 2008.

NNS10. Michael Naehrig, Ruben Niederhagen, and Peter Schwabe. New software speed records for cryptographic pairings. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT 2010*, volume 6212 of *LNCS*, pages 109–123. Springer, Heidelberg, August 2010.

Par19. James Parks. An asymptotic for the average number of amicable pairs for elliptic curves. *Mathematical Proceedings of the Cambridge Philosophical Society*, 166(1):33–59, 2019.

PHGR13. Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society Press, May 2013.

Poe18. Andrew Poelstra. Curve with group order $2^{255}-19$. `https://moderncrypto.org/mail-archive/curves/2018/000992.html`, accessed 28 February 2022, 2018.

Pol71. J. M. Pollard. The Fast Fourier Transform in a finite field. *Math. Comp.*, 25(114):365–374, April 1971.

Shi. MITSUNARI Shigeo. A portable and fast pairing-based cryptography library. `https://github.com/herumi/mcl`.

SHI$^+$12. Yumi Sakemi, Goichiro Hanaoka, Tetsuya Izu, Masahiko Takenaka, and Masaya Yasuda. Solving a discrete logarithm problem with auxiliary input on a 160-bit elliptic curve. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 595–608. Springer, Heidelberg, May 2012.

Sma99. Nigel P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12(3):193–196, June 1999.

SS11. Joseph H. Silverman and Katherine E. Stange. Amicable Pairs and Aliquot Cycles for Elliptic Curves. *Experimental Mathematics*, 20(3):329 – 357, 2011.

Sup. Supranational. Multilingual BLS12-381 signature library. `https://github.com/supranational/blst`.

Sut11. Andrew V. Sutherland. Computing Hilbert class polynomials with the chinese remainder theorem. *Math. Comp.*, 80(273):501–538, 2011. arXiv 0903.2785.

Tib14. Mehdi Tibouchi. Elligator squared: Uniform points on elliptic curves of prime order as uniform random strings. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 139–156. Springer, Heidelberg, March 2014.

Ver10. F. Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, Jan 2010.

VP19. Alexander Vlasov and Konstantin Panarin. Transparent polynomial commitment scheme with polylogarithmic communication complexity. Cryptology ePrint Archive, Report 2019/1020, 2019. `https://eprint.iacr.org/2019/1020`.

Wil. Zachary Williamson. An optimized elliptic curve library for the BN128 curve, and PLONK SNARK prover. `https://github.com/AztecProtocol/barretenberg/tree/master/barretenberg`.

WTs$^+$18. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE*

Symposium on Security and Privacy, pages 926–943. IEEE Computer Society Press, May 2018.

Wui21. Peter Wuille. Elligator Squared for BN-like curves. `https://github.com/sipa/writeups/tree/main/elligator-square-for-bn`, 2021.

Zca. Zcash. Rust implementation for the Pasta cycle in Rust. `https://github.com/zcash/pasta_curves`.

ZCa21. ZCash. What is jubjub? `https://z.cash/technology/jubjub/`, 2021.