Halo: Recursive Proof Composition without a Trusted Setup

Sean Bowe
sean@electriccoin.co
Electric Coin Company

Jack Grigg
jack@electriccoin.co
Electric Coin Company

Daira Hopwood
daira@electriccoin.co
Electric Coin Company

Abstract

Non-interactive proofs of knowledge allow us to publicly demonstrate the faithful execution of arbitrary computations. SNARKs have the additional property of succinctness, meaning that the proofs are short and fast to verify even when the computations involved are large. This property raises the prospect of recursive proof composition: proofs that verify other proofs. All previously known realizations of recursive proof composition have required a trusted setup and cycles of expensive pairing-friendly elliptic curves.

We obtain the first practical example of recursive proof composition without a trusted setup, using only ordinary cycles of elliptic curves. Our primary contribution is a novel technique for amortizing away expensive verification procedures from within the proof verification cycle so that we could obtain recursion using a composition of existing protocols and techniques. We devise a technique for amortizing the cost of verifying multiple inner product arguments which may be of independent interest.

1 Introduction

Proofs of knowledge [GMR89], introduced by Goldwasser, Micali and Rackoff, allow us to demonstrate knowledge of a satisfying witness to some NP statement. If these proofs also do not reveal anything about the witness we refer to them as zero-knowledge proofs of knowledge. Kilian [Kilian92], and later Micali [Micali00], showed that these non-interactive [BFM88] proofs could be smaller than the statement being proved. In the decades since, significant reductions in the size and verification time of these proofs have been made, culminating in succinct non-interactive arguments of knowledge, or SNARKs for short. Today, the most efficient SNARKs require pairing-friendly elliptic curves and trusted setup assumptions [Groth2016].

However, protocols based on standard assumptions still have either linear-time verifiers or large constants. As a recent example of the state of the art in group-theoretic constructions, Bulletproofs [BBBPWM17] have logarithmic

size proofs but have linear-time verifiers, owing this partly to a linear-size group multiscalar multiplication. The alternative of relying on a trusted setup, which for many is unpalatable, still leaves the difficult logistical problems and expenses surrounding such setups as a barrier for the use of pairing-based SNARKs.

Recursive Proof Composition Besides being an efficient mechanism for constructing zero-knowledge proofs, SNARKs imply the practicality of verifiable computation: the ability to outsource large computations to untrusted third parties and receive strong assurances about the correctness of their alleged outputs. In parallel with the development of efficient proof systems has been an interest in recursive proof composition – that is, proofs that are capable of verifying other instances of themselves.

As an introduction to this concept, consider a blockchain network that requires all participants in the network to download the entire history of the blockchain and validate each individual state transition merely in order to validate and process *new* state changes. SNARKs allow us to partially address this scalability problem by outsourcing some of these verification steps to a third party. However, the participant still must download and check each proof.

Valiant [Val08] suggested the idea of "incrementally verifiable computation" by considering proofs that could themselves verify the correctness of other proofs, allowing a single proof to inductively demonstrate the correctness of many previous proofs. This idea resolves our hypothetical problem: the participant in the blockchain network must now only download the current state of the network as well as a single (recursive) proof that this state is correct. Further proofs of state changes need only reference the latest proof, allowing old history to be discarded forever.

Unfortunately there have been numerous practical and theoretical obstacles to achieving recursive proof composition. On the theory side, arbitrary depths of the recursion seem to degrade the security of the construction by progressively increasing the size of the extractor needed to extract the initial witness at the base of the recursion. This can be addressed, as shown in [BCCT2012], by assuming only a constant depth of the recursion. In practice there are no known attacks and we will not make these theoretical problems a focus of our work.

On the practical side, the obstacles to recursive proof composition are daunting. [BCTV2014] provided the first setting in which a proof could recursively attest to the correctness of another proof. They provide a cycle of elliptic curves such that proofs constructed using one of the curves can feasibly reason about proofs constructed using the other. However, their techniques in practice require proofs of knowledge that have very efficient verifiers, of which the only realistic choices require trusted setups and pairing-friendly

curves. These curves must also be built over very large (roughly 780-bit) fields in order to achieve adequate security due to their low embedding degrees.

1.1 Our Contributions

We present the first realization of recursive proof composition without a trusted setup. As in [BCTV2014], we use a cycle of elliptic curves such that proofs constructed with one curve can reason about proofs constructed over the other. However, neither curve is pairing-friendly; the cycle consists of normal 255-bit prime-order curves that are conjectured to approach the 128-bit security level. Such cycles are easy to construct, as discussed in §5 *'Cycles of Curves'*. Our proof sizes and verification times do not increase even as proofs are continually nested.

Our achievement is based on two novel techniques. First, we present a method for amortizing the cost of verifying an inner product argument such that the marginal cost of verifying such an argument is logarithmic in the problem size, without requiring cooperation from the original prover(s). This technique is likely of independent interest. Second, we apply this technique over a cycle of elliptic curves in order to defer the full verification of each nested proof until the end of the cycle, allowing proofs to efficiently verify each other without the need for a fully succinct proving system.

Our intention is to explore the practical application of our amortization technique, and so we do not make any formal security claims about our demonstration protocol. However, we have attempted to construct a plausibly knowledge-sound construction to demonstrate the feasibility of our approach so as to inspire future improvements. Further, we propose many such future avenues for improving on our techniques.

1.2 Related Work

As we discuss in more detail in §5.1 *'Motivation for Cycles'*, a critical design issue for protocols using recursive composition is to efficiently support the operations, including elliptic curve arithmetic, needed for verification of another proof. Several previous works have attempted to address similar issues (not all in a recursive proof context).

CØCØ [KZM+2015] provides a library of cryptographic primitives optimized for use in arithmetic circuits. Its elliptic curve primitives make effective use of the SNARK-friendly curve approach, in which an "embedded" Montgomery curve is defined over the field for which circuit arithmetic is efficient.

The Sapling upgrade of the Zcash cryptocurrency protocol [Zcash] takes this approach further and achieves significant advances in concrete efficiency. It defines a pairing-friendly BLS12 curve [BLS2002], called BLS12-

381 [Bowe2017] that has an embedded curve called Jubjub [Hopw2018]. A birational equivalence between twisted Edwards and Montgomery forms of Jubjub [BL2017] is used to optimize the curve arithmetic in the Sapling circuits [HBHW2019, Appendix A]. This curve structure does not support efficient recursion.

In Zexe [BCG+2019], two pairing-friendly curves are used in order to support recursive proof verification. The larger curve is constructed "on top of" the smaller curve using the Cocks–Pinch method [FST2009, §4.1]. However, only one layer of recursion is used in Zexe.

[BCTV2014] aims to support "scalable zero-knowledge", using two pairing-friendly MNT curves (of embedding degrees 4 and 6) that form a cycle. The Coda block chain [MS2018] is an example of a deployed protocol using this approach. However, due to the low embedding degrees (and recent progress on the Discrete Logarithm Problem in extension fields used as the target group of pairings) the MNT4/MNT6 construction requires curves of size approaching 800 bits for the 128-bit security level. To date there is no other known construction for pairing-friendly cycles, and there is some evidence that more efficient constructions either do not exist or will be difficult to find [CCW2018].

All of the systems mentioned above use proof systems that require a trusted setup [Groth2010; PHGR2013; BCTV2014a; Groth2016], i.e. there must exist a trusted party (or a simulation of such a party via an MPC protocol) to generate the system parameters. This trusted party (or all of the MPC participants colluding together) would be able to break soundness of the proof system. In recent years new proof systems have appeared without this drawback [BBHR2018; WTS+2017; Setty2019]. These systems also avoid pairing-based cryptography, which eliminates potential concerns about improvements in cryptanalysis of pairings. Unfortunately, although asymptotically succinct, these systems all have large constants.

Our approach is similar to [BCTV2014], except that we do not require either curve to be pairing-friendly. This allows a significant reduction in the size of curves needed.

2 Preliminaries

We use the notation $\mathbb{G}$ for a prime-order abelian group (in practice instantiated as the group of points on a prime-order elliptic curve), and $\mathbb{F}$ for its scalar field.

We use uppercase letters to denote group elements, and lowercase letters to denote scalars. We write group operations in additive notation; scalar multiplication is denoted by aG for $a \in \mathbb{F}$ and $G \in \mathbb{G}$. We use boldface variable names for vectors, such that $\mathbf{a}$ is a vector of scalars and $\mathbf{G}$ is a vector of group elements.

We write the inner product $a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ of scalar vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$, as $\langle \mathbf{a}, \mathbf{b} \rangle$. Similarly we write the multiscalar multiplication $a_1 G_1 + a_2 G_2 + \cdots + a_n G_n$ of a scalar vector $\mathbf{a} \in \mathbb{F}^n$ with a vector of group elements $\mathbf{G} \in \mathbb{G}^n$, as $\langle \mathbf{a}, \mathbf{G} \rangle$.

3 Amortized Succinctness

Consider a prover that wishes to convince a verifier that they know a witness w for a statement ϕ_1, such that $(w, \phi_1) \in \mathcal{R}$ for some polynomial-time decidable relation $\mathcal{R}$. The prover will send a non-interactive proof of knowledge π_1 to the verifier such that given ϕ_1 the verifier accepts with high probability only if the proof demonstrates knowledge of the witness.

Suppose that the verifier must perform some linear-time operation f to check the proof, but that the verifier expects to receive another proof π_2 for a (different) statement ϕ_2 in the future. Instead of performing this procedure, the verifier could ask the prover to supply $v_1 = f(x_1)$ (for some input x_1 derived from the verification of π_1) so that the verifier can perform the remaining (sublinear-time) operations for checking π_1. This initially requires the verifier to take the prover's word. After the next proof π_2 is supplied, the verifier does not evaluate f but again asks the prover to provide $v_2 = f(x_2)$ similarly. Now, instead of checking the correctness of the prover's supplied values v_1, v_2 by evaluating f, the verifier and prover will engage in a public coin argument that convinces the verifier that v_1 and v_2 are correct with high probability so long as $v_3 = f(x_3)$ for some new values v_3 and x_3. The verifier can now check this by evaluating f only once.

This strategy for amortizing the cost of verifying proofs is central to our technique for achieving recursive proof composition. Consider a proof π_2 that additionally verifies a proof π_1, but again does not perform some requisite procedure f to fully check π_1; instead, the claimed inputs and outputs for f are embedded in the statement that π_2 proves. The verifier of π_2, which may, due to the recursion, actually be simulated by a prover of another proof π_3, can use the aforementioned technique to encode one instance (not two!) of checking an evaluation of f into the statement for π_3. This acts to continually amortize the cost of evaluating f such that it is never evaluated directly by any particular proof, but instead once at the end of the cycle. We refer to this concept as nested amortization.

Of course, in order to exploit this to achieve recursive proof composition, we first must actually devise a proving system employing procedures that can be amortized in this fashion.

3.1 Amortized Polynomial Commitments

Polynomial commitment schemes are a central tool of several recent proving systems [MBKM2019; WTS+2017; Setty2019]. We will focus on univariate

polynomial commitment schemes. These schemes allow a prover to commit to a polynomial $p(X) = \sum_{i=0}^{n-1} \mathbf{a}_i X^i$, and then later open this committed polynomial at arbitrary points such that a verifier will be convinced that the evaluation is correct; that the prover knows the polynomial; and that the polynomial is of degree at most $n - 1$.

We will leverage the inner product argument from [BCC+2016], which allows a prover to commit to vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ and then to provably evaluate their inner product $\langle \mathbf{a}, \mathbf{b} \rangle$ for a verifier. This is a generalization of a polynomial commitment scheme, if we fix $\mathbf{b} = (x^0, x^1, x^2, ..., x^{n-1})$ for some point x at which we wish a prover to evaluate the committed polynomial defined as before by the coefficients $\mathbf{a}$. The inner product argument can be easily adapted to a fixed $\mathbf{b}$, and in fact this variant of the argument can be made zero-knowledge [WTS+2017, Appendix A.3].

In our adaptation of the inner product argument a polynomial commitment is constructed with the vector commitment $\langle \mathbf{a}, \mathbf{G} \rangle$ for some vector $\mathbf{G} \in \mathbb{G}^n$ of random group elements. At the conclusion of the argument, in order to convince the verifier of the correct evaluation of a committed polynomial, the verifier must evaluate $G = \langle \mathbf{s}, \mathbf{G} \rangle$ and $b = \langle \mathbf{s}, \mathbf{b} \rangle$ where for some $k = \log_2(n)$ challenges $u_1, u_2, ..., u_k \in \mathbb{F}$ we have

$$\begin{aligned} \mathbf{s} = (\, & u_1^{-1} \; u_2^{-1} \cdots u_k^{-1}, \\ & u_1 \;\; u_2^{-1} \cdots u_k^{-1}, \\ & u_1^{-1} \; u_2 \;\; \cdots u_k^{-1}, \\ & u_1 \;\; u_2 \;\; \cdots u_k^{-1}, \\ & \qquad \vdots \\ & u_1 \;\; u_2 \;\; \cdots u_k \,) \end{aligned}$$

Observe that the verifier can compute b as $\prod_{i=1}^{k} (u_i + u_i^{-1} x^{2^{i-1}})$, with work only logarithmic in n. This reveals that G itself can be interpreted as a commitment to a polynomial that the verifier can efficiently evaluate at arbitrary points. Thus, the verifier could check the correctness of multiple values of G (constructed with different challenges) by evaluating them at a random point using our polynomial commitment scheme. Given a large enough field, a prover that provides a dishonest value of G is unlikely to satisfy this test due to the degree bound on the committed polynomials. Additionally, the polynomial commitments can be opened simultaneously by using the fact that the commitments are additively homomorphic, or by running two openings in parallel to share challenges. In any case this allows the verifier to reduce the cost of evaluating two values of G to the cost of computing just one, and as described before this technique can be continued repeatedly across the cycle such that only one computation of G is needed for checking all nested proofs.

Batch Verification It is worth contrasting this technique with the conventional technique of "batch verification". Batch verification takes advantage of the fact that multiple equations involving (mostly) the same fixed group generators can be checked simultaneously by probabilistically combining the checks into a single equation; this can vastly reduce the marginal cost of verifying additional inner product arguments. However, this technique still requires a marginal amount of work (scalar arithmetic) that is linear in the size of the problem. Our technique requires only a logarithmic marginal amount of work and communication, and so it can be seen as a kind of probabilistically accelerated batch verification. Furthermore, it does so without requiring the original provers of each individual inner product argument to cooperate.

3.2 Amortized Commitment Checks

In the protocol we describe later the verifier will need to evaluate a multivariate polynomial $s(X, Y)$ at some point (x, y_{cur}). Instead, we will have the prover commit to $s(X, y_{\text{cur}})$ and then evaluate it (using our polynomial commitment scheme) at x. This requires the verifier to check that the commitment is to the correct polynomial. We leverage the same technique described before where the verifier does not immediately check the correctness of the commitment but rather continually folds this check together with another check of the commitment $s(X, y_{\text{old}})$ for some other y_{old}. In particular, we use a technique from Sonic [MBKM2019] where the prover is asked to provide a commitment to $s(x', Y)$ for some random challenge x'. The commitments are checked for consistency, which shows that if the commitment to $s(x', Y)$ was correct then it holds (with high probability) that the former commitments were also to the correct polynomials, as low-degree polynomials cannot agree at most points. This subprotocol is played again with a new challenge y_{new} to obtain a purported commitment to $s(X, y_{\text{new}})$, restoring the problem to its original form.

4 Recursive Arguments of Knowledge

We will now use the techniques described in §3 *'Amortized Succinctness'* to obtain a recursive argument of knowledge. Our protocol is a variation of Sonic [MBKM2019] that is adjusted to leverage our nested amortization technique and for the polynomial commitment scheme we described earlier. Due to the similarity of our protocol with Sonic and our only minor adjustments to the inner product argument of [BCC+2016] we claim our protocol is plausibly knowledge sound, but we do not argue this formally. Our focus is entirely on the practicality of the nested amortization technique, so we leave these details for future work.

Observe that in Sonic the only operations the verifier must perform are polynomial commitment openings and a signature of correct computation. As we described, a polynomial commitment scheme amenable to nested amortization can be constructed using the inner product argument of [BCC+2016], and this can serve as a substitute for the commitment scheme used in Sonic. Further, we leverage a technique described in Section 8 of [MBKM2019] (which we reviewed in §3.2 *'Amortized Commitment Checks'*) to amortize the cost of multivariate polynomial commitment checks.

Briefly, let us review the Sonic proof of knowledge. The prover will demonstrate knowledge of a witness $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}^N$ that satisfies a system of constraints that consists of N multiplication constraints where the ith such constraint is of the form

$$\mathbf{a}_i \cdot \mathbf{b}_i = \mathbf{c}_i$$

and Q linear constraints where the qth such constraint is of the form

$$\Big(\sum_{i=1}^{N} \mathbf{a}_i \cdot (\mathbf{u}_q)_i\Big) + \Big(\sum_{i=1}^{N} \mathbf{b}_i \cdot (\mathbf{v}_q)_i\Big) + \Big(\sum_{i=1}^{N} \mathbf{c}_i \cdot (\mathbf{w}_q)_i\Big) = \mathbf{k}_q$$

for some fixed $\mathbf{u}_q, \mathbf{v}_q, \mathbf{w}_q \in \mathbb{F}^N$ and for the statement $\mathbf{k} \in \mathbb{F}^Q$. This system of constraints generalizes arithmetic circuits and so by demonstrating a satisfying assignment the prover will demonstrate the faithful execution of some circuit.

The prover accomplishes this by committing to a polynomial $r(X, Y)$ with their witness, obtaining a random challenge y from the verifier, and then demonstrating that a second polynomial $t(X, y)$ has a zero constant term.

$$\begin{aligned}
r(X,Y) &= \sum_{i=1}^{N} \mathbf{a}_i X^i Y^i + \sum_{i=1}^{N} \mathbf{b}_i X^{-i} Y^{-i} + \sum_{i=1}^{N} \mathbf{c}_i X^{-i-N} Y^{-i-N} \\
s(X,Y) &= \sum_{i=1}^{N} u_i(Y) X^{-i} + \sum_{i=1}^{N} v_i(Y) X^i + \sum_{i=1}^{N} w_i(Y) X^{i+N} \\
s'(X,Y) &= Y^N s(X,Y) - \sum_{i=1}^{N} (Y^i + Y^{-i}) X^{i+N} \\
t(X,Y) &= r(X,1)(r(X,Y) + s'(X,Y)) - Y^N k(Y)
\end{aligned}$$

Given that the prover's commitment to $r(X, Y)$ is degree bounded at N, then if the constant term of $t(X, y)$ is zero then with high probability the constraint system is satisfied. Note that we differ from Sonic in that we have rearranged the definition of $s(X, Y)$ (and some other polynomials) to reduce the degree of $s(X, Y)$ and to leverage the fact that the verifier can efficiently compute $\sum_{i=1}^{N} (Y^i + Y^{-i}) X^{i+N}$ itself, though this has no effect on

the argument. We compensate by redefining the polynomials

$$\begin{aligned} u_i(Y) &= \sum_{q=1}^{Q} Y^q (\mathbf{u}_q)_i \quad v_i(Y) = \sum_{q=1}^{Q} Y^q (\mathbf{v}_q)_i \\ w_i(Y) &= \sum_{q=1}^{Q} Y^q (\mathbf{w}_q)_i \quad k(Y) = \sum_{q=1}^{Q} Y^q \mathbf{k}_q \end{aligned}$$

Observe that $r(X, Y)$ is designed such that $r(X, Y) = r(XY, 1)$, and so the prover can commit to this polynomial with only a univariate polynomial commitment scheme. However, observe also that these polynomials are Laurent polynomials. We will address this by scaling the polynomials before committing to them, and rescaling the openings to obtain the correct results.

4.1 Basic Protocol

In all of the following, let k be an integer such that our polynomial commitment scheme bounds the degree of our committed polynomials at $2^k - 1$, and let $N = 2^{k-2}$. Further, let $Q < 2^k - 1$.

The prover begins by sending the commitment

$$R = \text{Commit}(r(X, 1)X^{3N-1})$$

which allows us later to demonstrate our degree bound for $r(X, Y)$. The verifier samples random challenge $y_{\text{cur}} \in \mathbb{F}$ and asks the prover to commit to $t(X, y_{\text{cur}})$ while establishing that $t(X, y_{\text{cur}})$ has a zero constant term. We accomplish this by having the prover send two commitments T^+ and T^- to polynomials $t^+(X)$ and $t^-(X)$ such that

$$t(X, y_{\text{cur}}) = t^+(X)X + t^-(X)X^{-4N}$$

which, due to the degree bound for the polynomial commitment scheme gives us that the committed polynomial has a zero constant term. The verifier now selects random challenge x and asks the prover to open their commitments to $v_1 = r(x, 1)$, $v_2 = r(x, y_{\text{cur}})$, $v_3 = t(x, y_{\text{cur}})$ and checks that

$$v_3 = v_1 \cdot (v_2 + s'(x, y_{\text{cur}})) - k(y_{\text{cur}})$$

which demonstrates that the commitment to $t(X, y)$ was correct with high probability, and so the polynomial $t(X, y)$ has a zero constant term, and therefore that the constraint system is satisfied with high probability.

4.2 Amortizing the Evaluation of $s'(x, y_{\text{cur}})$

In order to avoid the need for the verifier to evaluate $s'(x, y_{\text{cur}})$ itself, which is a multivariate polynomial with a linear number of terms (in the size of the circuit) we will instead borrow a technique described in Section 8

of [MBKM2019]. The prover will supply, prior to the sampling of x, the commitment

$$S_{\text{cur}} = \text{Commit}(s'(X, y_{\text{cur}})X^N)$$

which the verifier will ask the prover to also open at x. This requires the verifier to check that S_{cur} is a commitment to the correct polynomial. Consider that a previous proof (in our recursive cycle) produced a similar claimed S_{old} for some y_{old} but for the same polynomial $s(X, Y)$. (The polynomial $s(X, Y)$ is fixed for a given circuit.) The verifier will now ask the prover to supply a commitment

$$C = \text{Commit}(s'(x, Y)x^N)$$

and check that it opens at y_{old} and y_{cur} to the same values that S_{old} and S_{cur} open at x. Since $S_{\text{old}}, S_{\text{cur}}$ were committed prior to the choice of x, if C is a commitment to the correct polynomial then we have that S_{old} and S_{cur} are as well with high probability. The verifier now samples y_{new} and asks the prover to commit to

$$S_{\text{new}} = \text{Commit}(s'(X, y_{\text{new}})X^N)$$

for which we check that S_{new} opens at x to the same value that C opens at y_{new}. These new values $(S_{\text{new}}, y_{\text{new}})$ play the role of $(S_{\text{old}}, y_{\text{old}})$ in the next proof. Thus, the verifier only checks the correctness of S_{new} once due to nested amortization.

4.3 Commitments to $k(Y)$

It is necessary for the prover (and verifier) to compute a commitment $K = \text{Commit}(k(Y))$ prior to the choice of y_{cur} for two reasons. First, the statement $\mathbf{k}$ that derives $k(Y)$ contains values such as $S_{\text{old}}, y_{\text{old}}$ which we must commit to prior to the choice of x for the soundness of our amortization to hold anyway. More importantly, however, is that the verifier for our protocol will be simulated by the (adversarial) prover, who could otherwise choose a statement $\mathbf{k}$ after learning y_{cur} to falsely convince the verifier that their commitment to $t(X, y_{\text{cur}})$ was correct, violating soundness.

Because we are already evaluating the commitment C at y_{cur}, the evaluation of the commitment K is nearly free due to the fact that these commitments are additively homomorphic.

4.4 Commitments to G

Recall that during the polynomial opening argument the verifier must compute a group element $G \in \mathbb{G}$ given challenges $u_1, u_2, ..., u_k \in \mathbb{F}$ in order to fully verify the proof. As described before, these values will be brought in as part of the statement and we will engage in the technique described in §3 '*Amortized Succinctness*' to amortize away the cost of checking this value.

The previous proof (in the cycle) will be said to have produced G_{old} from challenges $u_1, u_2, ..., u_k$. We will compute the expected opening of the polynomial commitment that G_{old} represents, using the challenges, and open it at the point x to see that it is correct.

4.5 Shared Evaluations

After the various commitments are sent to the verifier, the prover and verifier will engage in our modified inner product argument. The prover begins by providing the claimed openings consistent with the descriptions above. Then, leveraging the fact that the commitments are additively homomorphic the verifier will sample a random $z \in \mathbb{F}$ and both parties will compute

$$P = R + zS_{\text{old}} + z^2 S_{\text{cur}} + z^3 T^+ + z^4 T^- + z^5 S_{\text{new}} + z^6 G_{\text{old}}$$

which will be opened at x, and similarly they will compute

$$U = C + zK$$

which will be opened at y_{cur}. Also, C will be opened at y_{old} and y_{new}, and R will be opened at $x \cdot y_{\text{cur}}$. These 5 polynomial opening protocols are run in parallel to share challenges so that only a single value of G (from the inner product argument) must be witnessed. This value of G serves the role of G_{old} for the next proof in the cycle.

4.6 Delegated Computations

Due to our use of curve cycles, the simulated verifier will not be able to efficiently perform the required scalar arithmetic to check e.g. Equation 4.1, and so our proofs will expose the openings of the various commitments involved as part of their statements so that proofs on the other curve (for which these equations operate over the native scalar field) can efficiently perform the checks for us.

5 Cycles of Curves

5.1 Motivation for Cycles

Statements for recent efficient proving systems are usually expressed in some variation of arithmetic circuits; that is, circuits using operations over a large field $\mathbb{F}_q$ (of size suitable for a discrete-logarithm-based cryptosystem).

Simulating field arithmetic in a different similarly large field, $\mathbb{F}_p$, in a circuit over $\mathbb{F}_q$, is highly inefficient. The main difficulty is with reduction of products modulo p. The best approach known to the authors is a "sum-of-residues" algorithm [Zcash-4093]: convert the value needing to be reduced

from a multilimb representation to binary, and then for each set bit of weight 2^k, add up the values $2^k \bmod p$ expressed in the desired multilimb representation for the output. With careful attention to the range of each limb, the result will be suitable for another multiplication or squaring even though it is not fully reduced. However, for each reduction, the conversion to binary would require a number of boolean constraints at least equal to the total width, in bits, of the partial products. Even taking into account possibilities for delayed reduction, the overhead is considerable. Well-known techniques such as Barett or Montgomery reduction do not provide any improvement over this sum-of-residues algorithm, because they are designed under the assumption that division is costly compared to multiplication, but that splitting a value into ranges of bits is cheap. Neither of these assumptions hold in arithmetic circuits.

So, while it may be feasible to implement a small number of operations in the "wrong" field, we must ensure that the vast majority of operations are in the "right" field in order to obtain circuits of practical sizes. This motivates the use of amicable pairs of elliptic curves, as in [BCTV2014].

5.2 Constructing amicable pairs

Given primes p and q, we call the elliptic curves $E_p/\mathbb{F}_p$ and $E_q/\mathbb{F}_q$ an amicable pair [SS2011] if it holds that $\#E_p = q$ and $\#E_q = p$.

Due to the Hasse bound, curves that form an amicable pair are necessarily prime-order (this is proven independently of pairing-friendliness in [CCW2018, Proposition 7]). This limits the curve types that can be used. Nevertheless, it turns out that such cycles are common and easy to find at all cryptographically relevant curve sizes.

The security of the overall system is limited by the hardness of the Discrete Logarithm Problem on each curve. Because both curves are prime-order, the field sizes and group orders will in practice all have the same bit length, and for k-bit security, this bit length must be at least $2k$ bits in order to resist Pollard rho and Pollard lambda attacks.

We also wish to use primes that allow for efficient FFT-based polynomial multiplications, as proposed in [BCG+2013, Appendix E.2]. This requires that the multiplicative groups $\mathbb{F}_p^\times$ and $\mathbb{F}_q^\times$ have high 2-adicity. Let z be a desired lower bound on this 2-adicity, i.e. we will search for curves such that $p, q \equiv 1 \pmod{2^z}$.

Any prime $p > 3$ is congruent to either 1 or 5 $\pmod 6$. We will use primes that are congruent to 1 $\pmod 6$, since that allows for a straightforward and efficient algorithm to construct amicable pairs via Complex Multiplication, as follows.

Let $D = 3$ be the absolute value of the CM discriminant of E_p. The norm equation of E_p is $4p = DV^2 + t^2$ for integers V and t.

Choose V and t so that $\frac{DV^2}{4}$ is approximately the desired bit length for p and q, and so that $\frac{V-1}{2}$ and $\frac{t-1}{2}$ are both multiples of 2^z. We also choose t to be 1 (mod 6). Then we have

$$\begin{aligned} 4p &= 3(V-1)^2 + 6(V-1) + (t-1)^2 + 2(t-1) + 4 \\ p &= 3\left(\frac{V-1}{2}\right)^2 + 3\frac{V-1}{2} + \left(\frac{t-1}{2}\right)^2 + \frac{t-1}{2} + 1 \end{aligned}$$

So $p-1$ will be a multiple of 2^z, and so will $(p+1-t)-1 = (p-1)-(t-1)$.

$p+1-t$ is one of six possible orders for a curve satisfying the norm equation $3V^2 = 4p-t^2$ (the others are $p+1+t$ and $p+1\pm\frac{t\pm 3V}{2}$) [BN2005, §2] [IEEE2000, §A.14.2.3, item 6]. Thus, we only need to check that $q = p+1-t$ is prime (which occurs with high enough probability to make the search for suitable V and t efficient), find parameters for the resulting curves, and verify that they form a cycle. At this point we can also check other desired criteria such as large embedding degree.

As a consequence of p and q being congruent to 1 (mod 6), the cube maps $x \mapsto x^3$ are not permutations on $\mathbb{F}_p$ and $\mathbb{F}_q$. This means that we cannot use the most efficient choice of $\alpha = 3$ for Rescue [AABDS2019], which we use to instantiate the hash needed for the Fiat-Shamir heuristic (see §6 *'Implementation'*). Instead we search for curves for which the next most efficient choice, $\alpha = 5$, can be used. That is, we ensure that $\gcd(p-1, 5) \neq 1$ and $\gcd(q-1, 5) \neq 1$, so that the maps $x \mapsto x^5$ are permutations on $\mathbb{F}_p$ and $\mathbb{F}_q$.

The cycle found by this method that is used by our demonstration software is described in §6.1 *'Tweedledum and Tweedledee'*.

5.3 A note on CM discriminants

Let D be the absolute value of a curve's Complex Multiplication discriminant. The above constructions always produce curves with $D = 3$ (this is mainly for simplicity of ensuring the 2-adicity requirement, and is not a necessary condition for finding a cycle).

Given that the influential "Safe Curves" criteria prohibit curves with low D [BL2013] (and a similar condition on the class number, that indirectly rules out low D, is included in the "Brainpool" criteria [RFC-5639]), this may raise a question in practitioners' minds over the security of the curves so constructed.

The reason why Safe Curves and Brainpool prohibit curves with low D, is that these curves have additional endomorphisms that may be used for optimization of the Pollard rho algorithm [BL2013] [DGM1999]. The improvement to the cost of Pollard rho may be precisely calculated and taken into account when choosing curve sizes. A conservative estimate of the available improvement is that on a group of prime order q with an

endomorphism ring of order 3, the cost of Pollard rho is $\sqrt{\frac{\pi q}{12}}$, as compared to $\sqrt{\frac{\pi q}{4}}$ using only the negation map as described in [BLS2011]. That is, the maximum speed-up is only a factor of $\sqrt{3} \approx 1.732$ (for a given success probability) [DGM1999].

To the authors' knowledge, there is no other reason not to use curves with $D = 3$. Such curves are in common use in deployed protocols that do not use pairings (for example, secp256k1 in Bitcoin [BitcoinCore]); also, all BN curves have $D = 3$ [BN2005].

5.4 Completeness and side-channel attacks

All curves used in our protocols are short Weierstrass curves of the form $y^2 = x^3 + b$, and are prime-order. Use of prime-order curves simplifies protocols and security analysis, avoiding error-prone techniques such as cofactor multiplication that may be applied incorrectly. However, the most efficient addition formulae for these curves are incomplete: they do not work correctly when adding two points with the same x-coordinate.

In our circuits, we pay careful attention to this issue and specify the necessary additional checks. In curve arithmetic performed outside the circuit, or if the same curves are used elsewhere in an application protocol, close attention to this issue is needed from implementors. Suitable complete, constant-time formulae for prime-order short Weierstrass curves are given in [RCB2016] or [SM2017].

6 Implementation

In order to demonstrate the practicality of our techniques, we have written an implementation of the protocol from Section 4. We sampled a cycle of ordinary (non-pairing friendly) elliptic curves which have high 2-adicity, meaning that their scalar fields are equipped with a large 2^k root of unity for accelerating the computation of $t(X, y)$ in our protocol.

We instantiate the protocol non-interactively by appling the Fiat-Shamir heuristic [FS1986] with a duplex sponge construction [BDPV2012]. We absorb openings of the protocol commitments into the transcript, and sample challenges as the low 128 bits of squeezed field elements. (We set bit 128 of each challenge because that is useful for optimizing scalar multiplication in the circuit.) We instantiate the duplex sponge with the Rescue algebraic symmetric primitive for prime-order groups [AABDS2019].

6.1 Tweedledum and Tweedledee

Our implementation uses a specific amicable pair of curves found using the algorithm give in Section 5.2, which we call Tweedledum and Tweedledee [Carroll1872]:

- $E_p/\mathbb{F}_p : y^2 = x^3 + 5$ of order q is called Tweedledum;
- $E_q/\mathbb{F}_q : y^2 = x^3 + 5$ of order p is called Tweedledee;

where p and q are 255-bit primes:

- $p = 2^{255} - 45523545521311825785582641903761044275 1$;
- $q = 2^{255} - 45523545521313402807170283885575746355 1$.

The software used to generate these curves and to test various security properties is available at [Hopw2019]. Its documentation describes how to reproduce this generation.
The following additional properties hold:

- $p - 1, q - 1 \equiv 1 \pmod 6$;
- $p - 1, q - 1 \equiv 1 \pmod{2^{34}}$ (that is, both curves have multiplicative 2-adicity of at least 34);
- $\gcd(p - 1, 5) = \gcd(q - 1, 5) = 1$;
- E_p and E_q have large embedding degrees:– $(q - 1)/4$ and $(p - 1)/2$ respectively.

We use the base points $(p - 1, 2)$ on E_p and $(q - 1, 2)$ on E_q.

7 Future Work

Another possible approach, not explored in this paper, would be to use a "half-pairing cycle" in which one curve is pairing-friendly and the other is not. This would be at the expense of requiring a trusted setup (for currently available choices of pairing-based proof systems), but could potentially provide greater concrete efficiency than the use of MNT curves. Such cycles can be found easily by adapting the procedure for searching for a Barreto-Naehrig curve in [BN2005].

8 Acknowledgements

This work was sponsored by the Electric Coin Company.

References

[AABDS2019] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. *Efficient Symmetric Primitives for Advanced Cryptographic Protocols (A Marvellous Contribution)*. Cryptology ePrint Archive: Report 2019/426. Last revised May 20, 2017. URL: `https://eprint.iacr.org/2019/426` (visited on 2019-09-09) (↑ p 13, 14).

[BCC+2016] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. *Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting*. Cryptology ePrint Archive: Report 2016/263. Received March 8, 2016. URL: `https://eprint.iacr.org/2016/263` (visited on 2019-09-04) (↑ p 6, 7, 8).

[BCCT2012] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. *From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again*. Cryptology ePrint Archive: Report 2011/443. Last revised November 30, 2011. URL: `https://eprint.iacr.org/2011/443` (visited on 2019-08-28). Also published in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS '12), 326–349 (2012). DOI: 10.1145/2090236.2090263 (↑ p 2).

[BCG+2013] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. *SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge*. Cryptology ePrint Archive: Report 2013/507. Last revised October 7, 2013. URL: `https://eprint.iacr.org/2013/507` (visited on 2016-08-31). An earlier version appeared in *Proceedings of the 33rd Annual International Cryptology Conference, CRYPTO 2013*, pages 90–108; IACR, 2013. (↑ p 12).

[BCG+2019] Sean Bowe, Alessandro Chiesa, Matthew Green, Ian Miers, Pratyush Mishra, and Howard Wu. *Zexe: Enabling Decentralized Private Computation*. Cryptology ePrint Archive: Report 2019/962. Last revised February 21, 2017. URL: `https://eprint.iacr.org/2018/962` (visited on 2019-09-10) (↑ p 4).

[BCTV2014] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. "Scalable Zero Knowledge via Cycles of Elliptic Curves (extended version)". In: *Advances in Cryptology - CRYPTO 2014*. Vol. 8617. Lecture Notes in Computer Science. Springer, 2014, pages 276–294. URL: `https://www.cs.tau.ac.il/~tromer/papers/scalablezk-20140803.pdf` (visited on 2016-09-01) (↑ p 2, 3, 4, 12).

[BCTV2014a] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. *Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture*. Cryptology ePrint Archive: Report 2013/879. Last revised February 5, 2019. URL: `https://eprint.iacr.org/2013/879` (visited on 2019-02-08) (↑ p 4).

[BDPV2012] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". In: *Selected Areas in Cryptography*. Ed. by Ali Miri and Serge Vaudenay. Berlin, Heidelberg: Springer, 2012, pages 320–337. ISBN: 978-3-642-28496-0 (↑ p 14).

[BitcoinCore] The Bitcoin Core Developers. *Bitcoin Core implementation*. URL: `https://github.com/bitcoin/bitcoin` (visited on 2019-09-04) (↑ p 14).

[BL2013] Daniel Bernstein and Tanja Lange. *SafeCurves: choosing safe curves for elliptic-curve cryptography*. URL: `https://safecurves.cr.yp.to` (visited on 2018-01-29) (↑ p 13, 18).

[BL2017] Daniel Bernstein and Tanja Lange. *Montgomery curves and the Montgomery ladder*. Cryptology ePrint Archive: Report 2017/293. Received March 30, 2017. URL: `https://eprint.iacr.org/2017/293` (visited on 2017-11-26) (↑ p 4).

[BLS2002] Paulo Barreto, Ben Lynn, and Michael Scott. *Constructing Elliptic Curves with Prescribed Embedding Degrees*. Cryptology ePrint Archive: Report 2002/088. Last revised February 22, 2005. URL: `https://eprint.iacr.org/2002/088` (visited on 2018-04-20) (↑ p 3).

[BLS2011] Daniel Bernstein, Tanja Lange, and Peter Schwabe. "On the correct use of the negation map in the Pollard rho method". In: *PKC '11 — Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography (Taormina, Italy, March 6–9, 2011)*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. Lecture Notes in Computer Science. International Association for Cryptologic Research. Springer, 2011, pages 128–146. ISBN: 978-3-642-19378-1. DOI: `10.1007/978-3-642-19379-8_8`. URL: `https://www.iacr.org/archive/pkc2011/65710132/65710132.pdf` (visited on 2019-08-26) (↑ p 14).

[BN2005] Paulo Barreto and Michael Naehrig. *Pairing-Friendly Elliptic Curves of Prime Order*. Cryptology ePrint Archive: Report 2005/133. Last revised February 28, 2006. URL: `https://eprint.iacr.org/2005/133` (visited on 2018-04-20) (↑ p 13, 14, 15).

[Bowe2017] Sean Bowe. *BLS12-381: New zk-SNARK Elliptic Curve Construction*. Zcash blog. March 11, 2017. URL: `https://electriccoin.co/blog/new-snark-curve/` (visited on 2019-08-27) (↑ p 4).

[Carroll1872] Lewis Carroll. *Through the Looking-Glass, and What Alice Found There*. Macmillan and Co., London, 1872. URL: `https://archive.org/details/throughlooking00carr` (visited on 2019-09-09) (↑ p 15).

[CCW2018] Alessandro Chiesa, Lynn Chua, and Matthew Weidner. *On cycles of pairing-friendly elliptic curves*. arXiv:1803.02067 [math.NT]. Last revised November 26, 2018. November 5, 2018. URL: `https://arxiv.org/abs/1803.02067` (visited on 2019-08-26) (↑ p 4, 12).

[DGM1999] Iwan Duursma, Pierrick Gaudry, and François Morain. "Speeding up the discrete log computation on curves with automorphisms". In: *Advances in Cryptology - ASIACRYPT 1999. Proceedings, International Conference on the Theory and Application of Cryptology and Information Security (Singapore, November 14–18, 1999)*. Ed. by Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing. Vol. 1716. Lecture Notes in Computer Science. Springer, 1999, pages 103–121. ISBN: 978-3-540-48000-6. DOI: 10.1007/b72231. URL: https://hal.inria.fr/inria-00511639 (visited on 2019-09-06) (↑ p 13, 14).

[FS1986] Amos Fiat and Asaph Shamir. "How to Prove Yourself: Practical Solutions to Identification and Signature Problems". In: *CRYPTO*. 1986 (↑ p 14).

[FST2009] David Freeman, Michael Scott, and Edlyn Teske. *A Taxonomy of Pairing-friendly Elliptic Curves*. Cryptology ePrint Archive: Report 2006/372. Last revised November 20, 2009. URL: https://eprint.iacr.org/2006/372 (visited on 2019-08-26) (↑ p 4).

[Groth2010] Jens Groth. *Short Pairing-based Non-interactive Zero-Knowledge Arguments*. URL: https://www.iacr.org/archive/asiacrypt2010/6477323/6477323.pdf (visited on 2019-09-09) (↑ p 4).

[Groth2016] Jens Groth. *On the Size of Pairing-based Non-interactive Arguments*. Cryptology ePrint Archive: Report 2016/260. Last revised May 31, 2016. URL: https://eprint.iacr.org/2016/260 (visited on 2017-08-03) (↑ p 1, 4).

[HBHW2019] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. *Zcash Protocol Specification*. URL: https://zips.z.cash/protocol/protocol.pdf (visited on 2019-09-09) (↑ p 4).

[Hopw2018] Daira Hopwood. *GitHub repository 'daira/jubjub': Supporting evidence for security of the Jubjub curve to be used in Zcash*. URL: https://github.com/daira/jubjub (visited on 2018-02-18). Based on code written for SafeCurves [BL2013] by Daniel Bernstein and Tanja Lange. (↑ p 4).

[Hopw2019] Daira Hopwood. *GitHub repository 'daira/tweedle': Generator and supporting evidence for security of the Tweedledum/Tweedledee pair of elliptic curves*. URL: https://github.com/daira/tweedle (visited on 2019-09-09) (↑ p 15).

[IEEE2000] IEEE Computer Society. *IEEE Std 1363-2000: Standard Specifications for Public-Key Cryptography*. IEEE, August 29, 2000. DOI: 10.1109/IEEESTD.2000.92292. URL: https://perso.telecom-paristech.fr/guilley/recherche/cryptoprocesseurs/ieee/00891000.pdf (visited on 2019-09-06) (↑ p 13).

[KZM+2015] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, T-H. Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. *CØCØ: A Framework for Building Composable Zero-Knowledge Proofs*. Cryptology ePrint Archive: Report 2015/1093. Last revised April 9, 2017. URL: https://eprint.iacr.org/2015/1093 (visited on 2019-09-09) (↑ p 3).

[MBKM2019] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. *Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updateable Structured Reference Strings*. Cryptology ePrint Archive: Report 2019/099. Last revised July 8, 2019. URL: https://eprint.iacr.org/2019/099 (visited on 2019-09-04) (↑ p 5, 7, 8, 10).

[BBBPWM17] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. *Bulletproofs: Short Proofs for Confidential Transactions and More*. Cryptology ePrint Archive, Report 2017/1066. https://eprint.iacr.org/2017/1066. 2017 (↑ p 1).

[BBHR2018] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive: Report 2018/046. Last revised March 5, 2018. URL: https://eprint.iacr.org/2018/046 (visited on 2019-08-28) (↑ p 4).

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. "Non-interactive Zero-knowledge and Its Applications". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC '88. Chicago, Illinois, USA: ACM, 1988, pages 103–112. ISBN: 0-89791-264-0. DOI: 10.1145/62212.62222. URL: http://doi.acm.org/10.1145/62212.62222 (↑ p 1).

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems". In: *SIAM J. Comput.* 18.1 (February 1989), pages 186–208. ISSN: 0097-5397. DOI: 10.1137/0218012. URL: http://dx.doi.org/10.1137/0218012 (↑ p 1).

[Kilian92] Joe Kilian. "A Note on Efficient Zero-knowledge Proofs and Arguments (extended abstract)". In: *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*. STOC '92. Victoria, British Columbia, Canada: ACM, 1992, pages 723–732. ISBN: 0-89791-511-9. DOI: 10.1145/129712.129782. URL: http://doi.acm.org/10.1145/129712.129782 (↑ p 1).

[Micali00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM J. Comput.* 30.4 (October 2000), pages 1253–1298. ISSN: 0097-5397. DOI: 10.1137/S0097539795284959. URL: https://doi.org/10.1137/S0097539795284959 (↑ p 1).

[Setty2019] Srinath Setty. *Spartan: Efficient and general-purpose zkSNARKs without trusted setup*. Cryptology ePrint Archive: Report 2019/550. Last revised June 3, 2019. URL: https://eprint.iacr.org/2019/550 (visited on 2019-08-28) (↑ p 4, 5).

[Val08] Paul Valiant. "Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency". In: *Theory of Cryptography*. Ed. by Ran Canetti. Berlin, Heidelberg: Springer, 2008, pages 1–18. ISBN: 978-3-540-78524-8 (↑ p 2).

[MS2018] Izaak Meckler and Evan Shapiro. *Coda: Decentralized cryptocurrency at scale*. O(1) Labs whitepaper. May 10, 2018. URL: https://cdn.codaprotocol.com/v2/static/coda-whitepaper-05-10-2018-0.pdf (visited on 2019-09-10) (↑ p 4).

[PHGR2013] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. *Pinocchio: Nearly Practical Verifiable Computation*. Cryptology ePrint Archive: Report 2013/279. Last revised May 13, 2013. URL: https://eprint.iacr.org/2013/279 (visited on 2016-08-31) (↑ p 4).

[RCB2016] Joost Renes, Craig Costello, and Lejla Batina. *Complete addition formulas for prime order elliptic curves*. Cryptology ePrint Archive: Report 2015/1060. Last revised March 8, 2016. URL: https://eprint.iacr.org/2015/1060 (visited on 2019-08-26) (↑ p 14).

[RFC-5639] M. Lochter and J. Merkle. *Request for Comments 5639: Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*. Internet Engineering Task Force (IETF). March 2010. URL: https://tools.ietf.org/html/rfc5639 (visited on 2019-09-10). See also errata at https://www.rfc-editor.org/errata_search.php?rfc=5639. (↑ p 13).

[SM2017] Ruggero Susella and Sofia Montrasio. "A Compact and Exception-Free Ladder for All Short Weierstrass Elliptic Curves". In: *Smart Card Research and Advanced Applications: 15th International Conference, CARDIS 2016 (Cannes, France, November 7–9, 2016), Revised Selected Papers*. Ed. by Kerstin Lemke-Rust and Michael Tunstall. Vol. 10146. Security and Cryptology. Springer, 2017, pages 156–173. ISBN: 978-3-319-54669-8. DOI: 10.1007/978-3-319-54669-8_10. URL: https://sci-hub.tw/10.1007/978-3-319-54669-8_10 (visited on 2019-08-26) (↑ p 14).

[SS2011] Katherine E. Stange and Joseph H. Silverman. *Amicable pairs and aliquot cycles for elliptic curves*. Submitted to arXiv.org on 9 December, 2009. DOI: 10.1080/10586458.2011.565253. URL: https://arxiv.org/abs/0912.1831 (visited on 2019-08-26). This paper also appeared in *Experimental Mathematics*, Vol. 20(3), pages 329–357; Taylor & Francis, 2011. (↑ p 12).

[WTS+2017] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. *Doubly-Efficient zkSNARKs Without Trusted Setup*. Cryptology ePrint Archive: Report 2017/1132. Last revised April 19, 2018. URL: https://eprint.iacr.org/2017/1132 (visited on 2019-09-04). A version of this paper (with major differences) appeared in *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pages 926–943; IEEE, 2018. (↑ p 4, 5, 6).

[Zcash] Electric Coin Company. *Zcash website*. URL: https://z.cash/ (visited on 2019-09-09) (↑ p 3).

[Zcash-4093] Daira Hopwood. *GitHub repository 'zcash/zcash': Issue 4093 – Implementing Fp arithmetic in an Fq circuit.* URL: `https://github.com/zcash/zcash/issues/4093` (visited on 2019-08-26) (↑ p 11).