CDLS: Proving Knowledge of Committed Discrete Logarithms with Soundness

Sofia Celi[1], Shai Levin[3], and Joe Rowell[3]

[1] Brave Software,
`cherenkov@riseup.net`
[2] University of Auckland,
`shai.levin@auckland.ac.nz`
[3] Royal Holloway, University of London,
`Joe.Rowell.2015@live.rhul.ac.uk`

Abstract. Σ-protocols, a class of interactive two-party protocols, which are used as a framework to instantiate many other authentication schemes, are automatically a proof of knowledge (PoK) given that they satisfy the *special-soundness* property. This fact provides a convenient method to compose Σ-protocols and PoKs for complex relations. However, composing in this manner can be error-prone. While they must satisfy special-soundness, this is unfortunately not the case for many recently proposed composed practical schemes. Here we explore two schemes: *ZKAttest*'s [FLM22] and Agrawal et al.'s [AGM18], and show that their Σ-protocol's suffer from several security misdesigns which invalidate their security proofs, and state a practical cheap attack on *ZKAttest*'s implementation. By exploring and resolving their misdesigns, we propose CDLS, a sound and secure variant of their protocols.

Keywords: Sigma-protocols · Zero-Knowledge · PoK · Elliptic-Curves

1 Introduction

Non-interactive zero knowledge proofs of knowledge (NIZKPoK) [BFM88] are protocols which allow provers to convince verifiers, in a non-interactive setting, that they possess knowledge of a secret witness for a decisional statement without directly revealing such a witness. They are *zero-knowledge*, which means a verifier cannot learn anything from interactions with an honest prover beyond the statement itself. They are often based on Sigma protocols, zk-SNARKs or "MPC-in-the-head" techniques with the Fiat-Shamir transform applied. The paradigm allows for the construction of secure identification protocols and signature schemes, and further gives way to more complex proving mechanisms. In this work, we are interested in proofs of knowledge that attest to the possession of an authentication statement; for example, a digital signature. In the Sigma protocol setting, we focus on the properties that such schemes should provide, and how some constructions fail to do so.

In 1999, Nguyen at al. [NBMV99] proposed the first general construction of zero-knowledge "proofs of possession" for digital signatures. In their proposal, they construct a NIZKPoK which convinces a verifier that a prover possesses a valid signature for a given message and public key. Their construction can be realised in the *committed discrete logarithm* setting (by using a commitment scheme), which means that, given a base element of a cyclic group $g \in \mathbb{G}$, a commitment to a secret exponent α and a commitment to the representation of $h = g^{\alpha}$, one can prove, in zero-knowledge, that the first commitment is the discrete logarithm of the other in regards to the same base g. This idea can be generalised to the following problem: how to prove the equality of a committed value and the discrete logarithm of another committed value. The proof of Nguyen et al., however, requires the representation of $\mathbb{G}$ to be integers modulo a prime and, hence, it is not compatible with elliptic curve groups.

In CRYPTO'18, Agrawal et. al [AGM18] introduced a protocol (which we describe in Section 3.1) that works in the elliptic curve group setting, when looking at NIZPoKs for composite statements (those that consist of ANDs, ORs or function composition, for example). In their protocol, given a public base point P, a commitment to a secret coefficient α, and a commitment to the affine coordinates of the elliptic curve point αP, one can prove, non-interactively, that the committed value of α is the discrete logarithm (to the base P) of the commitments to αP. They use Sigma protocols for polynomial relationships among committed values [CM99] and range proofs [Bou00,CCs08,BBB^{+}18] for their protocol. Internally, they also use an inner 'Proof of Knowledge of the Sum' (as seen in Section 3.1). Following this work, [FLM22] proposed *ZKAttest* (which we describe in Section 3.2), a protocol which modifies the previous underlying commitment scheme to improve efficiency, and includes an additional verification check with the aim of fixing the original proof of security.

For their protocols, both [AGM18] and *ZKAttest* instantiate Pedersen commitment schemes [Ped92] for the point multiplication coefficients in $\mathbb{Z}_p$, and base field elements in $\mathbb{Z}_q$. In the case of [AGM18], $\mathbb{Z}_q$ commitments are made over a message space of size cubic in q, as they claim that it is challenging to generate elliptic curves of prescribed order q. *ZKAttest*, on the contrary, utilises Bröker's method [Bro06,BS07] (as explained in Appendix A) to generate elliptic curves of order q, which is an efficient mechanism for one-time parameter generation. Compared to the first work, which requires larger messages and additional range proofs, *ZKAttest* proved to be more efficient, as their commitments operate over a commitment scheme with a message space of the "correct" order.

Furthermore, *ZKAttest* proposes new applications of these proofs of committed values. In the specific ECDSA setting, for example, a signer can construct a ring signature for "locked" or inaccessible private keys when *only* given access to the ECDSA signatures under that key. This application can be used when cryptographic keys are stored behind hardware or software interfaces which render the private key inaccessible, leaving the user with access only to signing functionality. *ZKAttest* proposes that, for these cases, rings can be constructed from

ECDSA public keys that attest to the identity of specific users even when they do not have direct access to their private key.

What all of these works show is that proofs of committed values are valuable in a different array of applications and can be used to construct many attestation protocols "on top", such as in identity escrow [NBMV99, Sec. 4.2], privacy-preserving web attestation [ZMM+20,CDH+23], privacy-preserving credentials [CL01,CL04,BL12], or, as we saw, ring signatures. However, while these protocols are of interest for their practical usability, their concrete designs suffer from security misdesign.

Prior to discussing their security misdesigns, first note that the above NIZKPoKs are based on the common paradigm of Sigma protocols, a public coin 3-move interactive protocol. A key property of Sigma protocols is *knowledge soundness*, which loosely states that, with negligible probability, a verifier will not accept a statement (or instance) x unless the prover knows a valid witness w for the statement. In order to prove a Sigma protocol is *knowledge sound*, or a Proof of Knowledge (PoK), one relies on proving the 2-special-soundness (or n-special soundness) property. That is, given two (or n) transcripts with identical first rounds and different challenges, one can efficiently recover a witness for the instance. The knowledge error of a Sigma protocol is dependent on the number of transcripts needed to extract a witness, and the size of the challenge set, but can be boosted through parallel composition of protocol instances, and is made non-interactive through a generic transform by Fiat-Shamir [FS87]. However, both *ZKAttest* and [AGM18] suffer from inherent protocol misdesign, which violate their proofs of *special-soundness*. Futhermore, additional security relaxations in their implementation yield a practical attack in the case of *ZKAttest*. In this paper, we will explore this inherent misdesign in Section 3 and propose alternative, provably secure constructions in Section 4.

Contributions. We contribute the following results in our work:

- Attempting to resolve any ambiguity in the works of *ZKAttest* [FLM22] and [AGM18], we show that, in an optimistic interpretation of their protocols, aside from not satisfying perfect completeness, their proofs of special-soundness have gaps in their logic. In particular, *ZKAttest* applies a non-standard approach of protocol composition, which leads to an invalid extractor strategy.
- We show that *ZKAttest*'s choice of only performing verification (from the verifier view) on a random subset of 20 out of the 128 repetitions for the security of their protocol leads to a practical, 'cheap' forgery attack.
- We propose an optimized and sound Sigma protocol for 'Proof of Knowledge of the Sum', which proves an additive relation between the commitments to the coordinates of three elliptic curve points. This protocol can be internally used by any proof of committed discrete logarithm.
- We introduce a new Sigma protocol for proving knowledge of a committed discrete logarithm, CDLs, which is provably perfectly complete, 2-special sound and statistically honest-verifier zero-knowledge. By doing this, we re-

solve the soundness misdesign of prior works. Performing λ repetitions of the protocol and applying the Fiat-Shamir transform, yields a NIZKPoK for elliptic curve committed discrete logarithms with knowledge error $2^{-\lambda}$ and statistical zero-knowledge.

- Using $\mathsf{CDL}s$, in Section 5, we show that one can construct a ring signature for ECDSA key-pairs where the user has access to the signing functionality, but not to their private key. We further show that if private keys are accessible to the user, an existing scheme [GK15] is sufficient for this application, which removes additional work for the signer and verifier.
- We provide an efficient, open-source Rust implementation for our scheme and our interpretation of *ZKAttest*, with benchmarks comparing their performance. Aside from resolving security issues, our implementation yields an order of magnitude improvement to performance when compared to the Typescript implementation of *ZKAttest* [FHLM21].

Acknowledgements.

We would like to thank Ward Beullens, Alex Davidson, Hamed Haddadi and Steven Galbraith for their helpful discussions and insight.

2 Preliminaries

2.1 Notation and General Definitions

As a small note first, in this section, we refer to group operations multiplicatively, but due to the concrete instantiation of groups as elliptic curves, we opt to refer to group operations additively in the sections after this one.

We refer to the set $\{1, \ldots, n\}$ as $[n]$. We denote negligible functions in a security parameter λ as $\mathsf{negl}(\lambda)$. We use the standard Landau notation $O(\cdot)$ for asymptotics and we use $\tilde{O}(n)$, which hides logarithmic factors in the parameter n. For any $q \in \mathbb{Z}$, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denotes the ring of integers modulo q. We refer to a prover as $\mathcal{P}$ and a verifier as $\mathcal{V}$, which are probabilistic polynomial time machines (PPTs); to a polynomial-time simulator as Sim and to polynomial-time algorithm extractor as Ext. We say that two probability ensembles (X_n, Y_n), which are families of distributions over a finite set of size n, are:

– *perfectly indistinguishable*, written as $X_n \equiv Y_n$, if they are identically distributed.

– *statistically indistinguishable*, with negligible advantage ϵ, if for any computationally unbounded distinguisher D,

$$\big| \Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1] \big| \leq \epsilon(n)$$

which is written as $X_n \overset{s}{\equiv} Y_n$.

2.2 NP Relations and Proofs of Knowledge

The concept of proof of knowledge (PoK) was initially formalised by Feige, Fiat and Shamir [FFS88,FS87]. The construction allows for $\mathcal{P}$ to convince $\mathcal{V}$ that they know a witness, w, which is computationally related to a common input, x. That is, given an NP language $\mathcal{L}$, w allows a decision algorithm to determine if $x \in \mathcal{L}$ in a polynomial number of steps. We define the relation of instance-witness pairs for a given language as a set $\mathcal{R}$. Loosely, a proof of knowledge with *knowledge error* κ has probability $\kappa(\lambda)$ of a cheating $\mathcal{P}$ successfully convincing $\mathcal{V}$ that $x \in \mathcal{L}$ without knowing the corresponding w. The correlated language $\mathcal{L}(\mathcal{R})$ is the set of all x such that there exists a w, where $(x, w) \in \mathcal{R}$. Fiat and Shamir [FS87] showed that one can transform a 3-round interactive PoK into a non-interactive one, preserving knowledge soundness.

2.3 Sigma Protocols

A sigma protocol (Σ)[4], in its inception, is a three-move protocol between two parties: $\mathcal{P}$ and $\mathcal{V}$. Σ-protocols are Arthur-Merlin protocols [Bab85], which means that $\mathcal{V}$'s randomness is public. We give a brief description below of them, but for more details, see [HL10,Dam10,KO21].

Given a common input, where both $\mathcal{P}$ and $\mathcal{V}$ have x, and $\mathcal{P}$ has a value w (the witness) such that $(x, w) \in \mathcal{R}$, a Σ-protocol for a relation $\mathcal{R}$ works as follows:

Round 1 (Commit): $\mathcal{P}$ sends a message Comm to $\mathcal{V}$.
Round 2 (Challenge): On receiving Comm, $\mathcal{V}$ sends a random t-bit string Chall.
Round 3 (Response): $\mathcal{P}$ receives the challenge and sends a reply Resp to $\mathcal{V}$.
Verification: At the end of the interaction, $\mathcal{V}$ receives Resp and decides to accept or reject based solely on seen data: $(\mathsf{Comm}, \mathsf{Chall}, \mathsf{Resp})$. $\mathcal{V}$ outputs 1 if it accepts; otherwise, 0.

We assume that $\mathcal{P}$'s only advantage over $\mathcal{V}$ is that they know the private witness w. We stress that the parties use independent randomness for generating their messages in every execution. Given this, Σ-protocols must satisfy the following properties:

Completeness If $(x, w) \in \mathcal{R}$, and $\mathcal{P}$ and $\mathcal{V}$ follow the protocol honestly, on common input x and $\mathcal{P}$'s private input w, then $\mathcal{V}$ accepts with probability 1.

Special Honest Verifier Zero-Knowledge (HVZK). Given Sim which, when given a statement x and a challenge, outputs a valid transcript $(\mathsf{Comm}, \mathsf{Chall}, \mathsf{Resp})$ that is (perfectly) indistinguishable from a real transcript (with the same probability distribution as those between a honest $\mathcal{P}$ and a honest $\mathcal{V}$ on common input x). Note that the challenge being randomly sampled implies that $\mathcal{V}$ behaves honestly. If Sim's output distribution is only statistically indistinguishable with advantage ϵ from the real transcript distribution, the protocol is said to be *statistically* HVZK with advantage ϵ.

[4] The name is given as: the first part refers to "zig-zag" for its three moves and the last part is an abbreviation of "Merlin-Arthur".

Special Soundness. Given Ext which, when given any x and any pair of accepting distinct transcripts $[(\mathsf{Comm}, \mathsf{Chall}, \mathsf{Resp}), (\mathsf{Comm}, \mathsf{Chall}', \mathsf{Resp}')]$ for x, where $\mathsf{Chall} \neq \mathsf{Chall}'$, outputs w so that $(x, w) \in \mathcal{R}$.

Special soundness [Cra97] is restricted to the case where *two* colliding transcripts are necessary and sufficient for extracting w. However, this property can be relaxed to "n-special soundness" [GK14,BCC$^+$15,Wik18,Wik21,AAB$^+$21], where Ext needs $n > 2$ colliding transcripts to extract w. We state this relaxed definition below.

n-Special Soundness [AFK22, Defn. 7]. Given Ext which, when given a statement x and n valid distinct transcripts $[(\mathsf{Comm}_i, \mathsf{Chall}_i, \mathsf{Resp}_i)_{i\in[n]}]$ where $\mathsf{Comm}_i = \mathsf{Comm}_j$ (with a common first message), $\mathsf{Chall}_i \neq \mathsf{Chall}_j$ and $\mathsf{Resp}_i \neq \mathsf{Resp}_j$ for all $1 \leq i \leq j \leq n$, outputs w such that $(x, w) \in \mathcal{R}$. An n-special sound Σ-protocol with a t-bit challenge space has knowledge error $\frac{n-1}{2^t}$ [AFK22, Eqn. 1]. It is known that n-special-soundness for Σ-protocols implies knowledge soundness and thus renders PoKs.

The stated properties of Σ-protocols hold under parallel repetition, where the parties run the same protocol multiple times with the same input in parallel. Given a Σ-protocol with challenge length t, $\mathcal{V}$ samples a random challenge of length rt where r is the number of repetitions. $\mathcal{P}$ replies to each repetition using a different t-bit challenge. $\mathcal{V}$ accepts all iff it accepts in each repetition, and it can be proven that all properties hold. Σ-protocols are also invariant under AND-composition. Given two Σ-protocols for different relations, by running parallel executions of the two protocols under the same challenge, one obtains a Σ-protocol for product relation. Furthermore, given a Σ-protocol with challenge length t, one can construct a Σ-protocol for the same relation, with challenge length s for all $s \leq t$, by restricting the challenge space to a subset of the original challenge space of size 2^s [Dam10, Lemma 2].

As briefly touched upon, the Fiat-Shamir transform [FS87] allows for converting a interactive Σ-protocol into a non-interactive one by replacing $\mathcal{V}$'s challenges with the output of a random oracle query on the protocol transcript. In the random oracle model (ROM), the transformation preserves knowledge soundness and the protocol is perfect (resp. statistical) Zero-Knowledge (ZK) if the underlying Σ-protocol is perfect (resp. statistical) HVZK.

2.4 Commitment Schemes

We assume the reader is familiar with the definitions of commitment schemes, Comm, but we remind them of two important properties:

Computationally Binding. Comm is computationally binding if for all PPT adversary's $\mathcal{A}$, $\Pr[(x \neq x') \wedge (\mathrm{Comm}(x, r) = \mathrm{Comm}(x', r')) \mid (x, x', r, r') \leftarrow \mathcal{A}] \leq \mathsf{negl}(\lambda)$, where λ is the security parameter of the commitment scheme.

Perfectly Hiding. A commitment scheme Comm is perfectly hiding if for all distinct messages (x, x'), $\{\text{Comm}(x, R)\} \equiv \{\text{Comm}(x', R)\}$, where R is the uniform distribution on the randomness set sampled for the commitments.

Pedersen commitments (PC) [Ped92]. Let (g, h) be generators of some group $\mathbb{G}$ of order q and let α be a secret value. We denote a *Pedersen commitment* to α with randomness r as $C = \mathsf{Com}_q(\alpha, r) = g^\alpha h^r$. In settings where it is convenient to ignore the random value r, we shall simply omit it and write $C = \mathsf{Com}_q(\alpha)$. They are perfectly hiding, additively homomorphic and computationally binding assuming the hardness of the underlying group's discrete logarithm problem [Ped92].

2.5 Σ-protocols for proving arithmetic relations between Pedersen commitments.

The heart of many PoK constructions rely on Σ-protocols proving arithmetic relations between different PC. Related protocols (e.g Schnorr [Sch91], Chaum-Pedersen [CP93] and Fujisaki-Okamoto [FO97]) can be used to construct the schemes. In this paper, we use the proofs of [WTs+18, App. A] due to their efficiency.

Proving knowledge of an opening of a PC. Given commitment C to a message x with randomness r, $\mathcal{P}$ may convince $\mathcal{V}$ that they possess knowledge of x and r by engaging in the following Construction 2.1.

Construction 2.1: Opening Proof(C)

Public parameters: $g, h \in \mathbb{G}$ where $\text{ord}(g) = \text{ord}(h) = q$ for a prime q.
Inputs: C such that $C = g^x h^r$, and $\mathcal{P}$ knows $x, r \in \mathbb{Z}_q$.

1. $\mathcal{P}$ samples $\alpha_1, \alpha_2 \leftarrow\!\!\$\ [q]$ and sends $t \leftarrow g^{\alpha_1} h^{\alpha_2}$.
2. $\mathcal{V}$ sends challenge $c \leftarrow\!\!\$\ [q]$.
3. $\mathcal{P}$ sends $s_1 \leftarrow xc + \alpha_1$ and $s_2 \leftarrow rc + \alpha_2$.
4. $\mathcal{V}$ verifies that $g^{s_1} h^{s_2} = C^c t$.

Theorem 1. *Construction 2.1 is a Σ-protocol for the relation:*

$$\mathcal{R} = \{((C, g, h, q), (x, r)) \mid C = g^x h^r\}.$$

Proof. It follows from [Sch91].

Proving equality of PCs under the same base. The proof works to attest the equality of PCs under the same base group elements. Given commitments (C_1, C_2) to the same message x under different randomness (r_1, r_2), $\mathcal{P}$ may convince $\mathcal{V}$ of this fact by engaging in Construction 2.2.

Construction 2.2: EqualityProof(C_1, C_2)

Public parameters: $g, h \in \mathbb{G}$ where $\mathrm{ord}(g) = \mathrm{ord}(h) = q$ for a prime q.
Inputs: (C_1, C_2) such that $C_1 = g^x h^{r_1}$, $C_2 = g^x h^{r_2}$, and $\mathcal{P}$ knows $x, r_1, r_2 \in \mathbb{Z}_q$.

1. $\mathcal{P}$ samples $\alpha \leftarrow\$ [q]$ and sends $t \leftarrow h^\alpha$.
2. $\mathcal{V}$ sends challenge $c \leftarrow\$ [q]$.
3. $\mathcal{P}$ sends $s \leftarrow c(r_1 - r_2) + \alpha$.
4. $\mathcal{V}$ verifies that $h^z \overset{?}{=} t(C_1 C_2^{-1})^c$.

Theorem 2 (Folklore). *Construction 2.2 is a Σ-protocol for the relation:*

$$\mathcal{R} = \{((C_1, C_2, g, h, q), (x, r_1, r_2)) \mid C_1 = g^x h^{r_1},\ C_2 = g^x h^{r_2}\}.$$

Proving multiplicative relationships between committed values. Given $C_1 = Com(x) = g^x h^{r_1}$, $C_2 = Com(y) = g^y h^{r_2}$, $C_3 = Com(z) = g^z h^{r_3}$, $\mathcal{P}$ may engage in Construction 2.3 to convince $\mathcal{V}$ that $z = xy$.

Theorem 3. *Construction 2.3 is a Σ-protocol for the relation:*

$$\mathcal{R} = \{((C_1, C_2, C_3, g, h, q), (x, y, r_1, r_2, r_3)) \mid C_1 = g^x h^{r_1},\ C_2 = g^x h^{r_2}, C_3 = g^{xy} h^{r_3}\}.$$

Proof. It follows from [WTs+18, Thm. 10].

Construction 2.3: MulProof(C_1, C_2, C_3)

Public parameters: $g, h \in \mathbb{G}$ where $\mathrm{ord}(g) = \mathrm{ord}(h) = q$ for a prime q.
Inputs: $C_1 = g^x h^{r_1}, C_2 = g^y h^{r_2}, C_3 = g^{xy} h^{r_3}$ where $\mathcal{P}$ knows x, y, r_1, r_2, r_3.

1. $\mathcal{P}$ samples $\alpha_1, \ldots, \alpha_5 \leftarrow\$ [q]$ and sends
$$t_1 \leftarrow g^{\alpha_1} h^{\alpha_2} \qquad t_2 \leftarrow g^{\alpha_3} h^{\alpha_4} \qquad t_3 \leftarrow C_1^{\alpha_3} h^{\alpha_5}$$
2. $\mathcal{V}$ sends challenge $c \leftarrow\$ [q]$
3. $\mathcal{P}$ sends response:
$$s_1 \leftarrow \alpha_1 + cx \qquad s_2 \leftarrow \alpha_2 + cr_1 \qquad s_3 \leftarrow \alpha_3 + cy$$
$$s_4 \leftarrow \alpha_4 + cr_2 \qquad s_5 \leftarrow \alpha_5 + c(r_3 - r_1 y)$$
4. $\mathcal{V}$ checks that
$$t_1 C_1^c \overset{?}{=} g^{s_1} h^{s_2} \qquad t_2 C_2^c \overset{?}{=} g^{s_3} h^{s_4} \qquad t_3 C_3^c \overset{?}{=} g^{s_3} h^{s_5}$$

Constructing Sigma Protocols for arithmetic circuits. Given protocols to prove equality and multiplication (as given in Constructions Construction 2.2 and Construction 2.3) of Pedersen commitments, it is possible to construct Σ-protocols for proofs of arbitrary arithmetic relations of committed values. These may be viewed as arithmetic circuits or polynomial systems. Whilst in many cases, for larger circuits, it is more practical to use *succinct* non-interactive proofs, in some applications, such as performing a proof that an elliptic curve point addition has been honestly computed, the approach presented here may be suitable.

Given an arbitrary polynomial system of n variables $(x_1, \ldots, x_n)$ over $\mathbb{F}_q$, one can translate the original system into one of M quadratic constraints where each constraint $j \in M$ is an equation of the form:

$$\left(\sum_{i \in N} a_{i,j} x_i,\right) \circ \left(\sum_{i \in N} b_{i,j} x_i\right) = \sum_{i \in N} c_{i,j} x_i$$

,where $N = n + k$, $(x_{n+1}, \ldots, x_{n+k})$ are the intermediate variables needed to expand the polynomial system[5], and all variables and coefficients are defined over $\mathbb{F}_q$.

Observe that $\mathcal{V}$ can compute linear combinations of committed values by applying the group operation to each comitted values by the linearity of Pedersen commitments. So, given commitments $(C_i = \mathrm{Com}_q(x_i))_{i \in [N]}$ and the coefficients $(a_{i,j}, b_{i,j}, c_{i,j})_{i \in [N], j \in [M]}$, $\mathcal{P}$ can convince $\mathcal{V}$ that the polynomial system holds by performing the following:

1. $\mathcal{V}$ computes the commitments to the linear combinations:
$$A_j = \prod_{i \in [N]} a_{i,j} x_i, \quad B_j = \prod_{i \in [N]} b_{i,j} x_i, \quad C_j = \prod_{i \in [N]} c_{i,j} x_i \quad \text{for } j \in [M],$$
where taking the product correponds to the group operation on the commitments.
2. $\mathcal{P}$ engages in multiplication proofs $\mathrm{MulProof}(A_j, B_j, C_j)$ for $j \in [M]$.
3. $\mathcal{P}$ engages in opening proofs[6] $\mathrm{OpeningProof(C_i)}$ for $i \in [N]$.

There are several other solutions to this problem, many of which involve the use of additional equality proofs [NBMV99,CM99,AGM18,FHLM21,CS97a,Bra97,Cam99]. However, our approach yields more optimisation, and is equivalent to *Rank-1 Constraint Systems*, the relation used in many zk-SNARKs, such as [Set20,BCR+19]. In the applications of the latter, this methodology of representing polynomial systems is common as seen in, for instance, [Hou22,CLL23].

[5] Or, equivalently, an arithmetic circuit with n inputs and k intermediate variables with respect to multiplication gates.

[6] Note that this step is sufficient, but not necessary. In many cases it is possible to satisfy special soundness without running opening proofs on all inputs, such as the protocol in Section 4.1.

2.6 Ring Signatures

A ring signature provides a mechanism for an authorised set of users to sign messages on behalf of a set. This user-set is called a ring.

A ring signature consists of the PPT algorithms (SetUp, KeyGen, Sign, Verify) which behave in the following manner:

SetUp$(1^\lambda) \rightarrow pp$: Generates parameters for a given security parameter λ that are public to all users.

KeyGen$(pp) \rightarrow (pk, sk)$: A non-deterministic algorithm that outputs a public-private key pair for an individual user.

Sign$(m, R; pp, sk) \rightarrow \sigma$: Given a message $m \in \{0,1\}^*$, ring $R = (pk_1, ..., pk_N)$ of N users, sign message with respect to R. We note that a valid signature must have that pk, the corresponding public key to sk, is contained in R.

Verify$(m, R, \sigma; pp) \rightarrow 0, 1$: Verification outputs 1 if σ is an accepted signature for ring R, and 0 otherwise.

A crucial property of ring signatures is anonymity, which guarantees that no information about the individual who signed the message is learned, beyond that they are a member of the ring.

Definition 1 (Anonymity). *A ring signature scheme* (SetUp, KeyGen, Sign, Verify) *has perfect anonymity, if, for any computationally unbounded adversary $\mathcal{A}$,*

$$\Pr\left[\mathcal{A}(\sigma) = b \;\middle|\; \begin{array}{c} pp \leftarrow \mathrm{SetUp}(1^\lambda);\;\; (m, i_0, i_1, R) \leftarrow \mathcal{A}^{\mathrm{KeyGen}}(pp) \\ b \leftarrow\!\!\$\; \{0,1\};\;\; \sigma \leftarrow \mathrm{Sign}(m, R; pp, sk_{i_b}) \end{array}\right] = \frac{1}{2}$$

In loose terms, this means that it is (perfectly) impossible to distinguish two signatures on the same message signed by different users in a ring.

As with standard signatures, ring signatures also possess a notion of unforgeability and correctness.

Definition 2 (Perfect Correctness). *A ring signature scheme* (SetUp, KeyGen, Sign, Verify) *has perfect correctness if, for any λ, pp obtained from* SetUp(λ)*, $\{sk_j, pk_j\}_{j=1}^N$ output by* KeyGen(pp)*, every $i \in [N]$, and message m,*

$$\Pr\left[\mathrm{Verify}(m, R, \sigma; pp) = 1 \;\middle|\; \sigma \leftarrow \mathrm{Sign}(m, R; pp, sk_i)\right] = 1,$$

where $R = (pk_1, ..., pk_N)$.

Definition 3 (Unforgeability). *A ring signature scheme* (SetUp, KeyGen, Sign, Verify) *has unforgeability (with respect to insider corruption) if, for any PPT adversary $\mathcal{A}$,*

$$Pr\left[\mathrm{Verify}(m, R, \sigma; pp) = 1 \;\middle|\; \begin{array}{c} pp \leftarrow \mathrm{SetUp}(1^\lambda); \\ (m, R, \sigma) \leftarrow \mathcal{A}^{\mathrm{PKGen}, \mathrm{OSign}, \mathrm{Corrupt}}(pp) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

– PKGen *on the ith query picks randomness r_i and runs* $(pk_i, sk_i) \leftarrow \text{KeyGen}(pp, r_i)$ *and outputs* pk_i*.*
– $\text{OSign}(\cdot,\cdot,\cdot)$ *is a signing oracle where* $\text{OSign}(m, S, i)$ *outputs* $\sigma \leftarrow \text{Sign}(m, S; pp, sk_i)$*, provided* (pk_i, sk_i) *has been generated by* PKGen*, otherwise outputs* $\perp$*.*
– $\text{Corrupt}(\cdot)$ *is a corruption oracle where* $\text{Corrupt}(i)$ *outputs* sk_i*.*
– $\mathcal{A}$ *outputs* (m, R, σ) *such that* OSign *has not been queried with* $(\star, m, R)$ *and* R *only contains keys* pk_i *generated by* PKGen *where* $\mathcal{A}$ *has not queried* Corrupt *with input* i*.*

A protocol (SetUp, KeyGen, Sign, Verify) is a ring signature scheme with perfect anonymity if it is correct, unforgeable and perfectly anonymous.

3 Proofs of Knowledge and Soundness Misdesign

ZKAttest (henceforth referred to as ZKA) is a protocol designed in [FLM22]. The authors introduce the scheme as a way to build both a privacy preserving ECDSA PoK and ring signature, which require both *pre-existing* signatures and their associated public key as input. The core primitive in ZKA is a NIZKPoK for proving knowledge of a valid *ECDSA signature* under a committed public key. Additional properties can be attested to; such as proving, for example, that the commitment to the public key is a value on a list of valid public keys, essentially obtaining a ring signature.

Under the hood, ZKA builds on top of a Proof of Knowledge of Double Discrete Logarithm (PKDLog) scheme given by [AGM18] (referred to as PKDL, see Section 3.1), which aims to prove *the equality of a committed value and the discrete logarithm of another committed value* when working in elliptic curve groups (the known techniques for double discrete logarithm proofs do not work for this case [CS97b,MGGR13], as a group element cannot be naturally interpreted as a field element). For this, [AGM18] internally proves that the sum of two elliptic curve points that are committed to is another public point on the curve. This last proof, as seen in Section 3.1, is referred to as Proof of Knowledge of the Sum —PKS–.

A visual representation of a PKDLog can be seen in Appendix B (instantiated in the ZKA manner), where solid boxes represent the internal PKS.

In the subsections below, we explain the underlying PoKs that are used to build the outer ECDSA PoK from ZKA [FLM22]. We first explore the schemes given by [AGM18] and, then, how they are extended for ZKA. Note that internally all these protocols rely on Constructions 2.1 to 2.3 for opening, equality and multiplication proofs, respectively. We introduce, and explain misdesigns and attacks to the constructions designed by both papers.

3.1 Proofs of Knowledge in [AGM18]

Proof of Knowledge of the Sum (PKS) in [AGM18]: This PoK aims to prove that the commitments to three elliptic curve points P, Q, T satisfy a valid point addition such that $P + Q = T$. We state the point addition formulae in Theorem 3.1 given by [Sil09]. Given the family of curves $\mathcal{E}$ defined by $y^2 = x^3 + ax + b$ (where a, $b \in \mathbb{F}_t$), we have a point addition relation of the form shown in 3.1.

Theorem 3.1: Point addition

Let $P = (a_x, a_y)$, $Q = (b_x, b_y)$, $(P, Q) \in E(\mathbb{F}_t)$ where E is in the family $\mathcal{E}$. For non-identity elements (P, Q) where $P \neq \pm Q$, $(t_x, t_y) = P + Q$ is given by:

$$t_x = \left(\frac{b_y - a_y}{b_x - a_x}\right)^2 - a_x - b_x \tag{1}$$

$$t_y = \left(\frac{b_y - a_y}{b_x - a_x}\right)(a_x - t_x) - a_y \tag{2}$$

The above relations given for point addition can be proven by using Σ-protocol techniques for arithmetic relationships; but, as the point addition formulae is over $\mathbb{F}_t$, the commitments to the coordinates have to be in a group of order t, which is not necessarily the same as the order of the group $E(\mathbb{F}_t)$. [AGM18] solve this problem by rearranging the point addition formulae so that Σ-protocols for polynomial relationships among committed values [CM99] and range proofs [Bou00,CCs08,BBB$^+$18] can be used on the intermediate commitments. The proof is expanded to handle all the cases for point addition by using OR composition [CDS94], and it is referred to as PKS, which we will not describe in detail but we refer to the reader to Section 3.1 of [AGM18].

On complexity of PKS. The efficiency of PKS is roughly dominated by the cost of the range proofs used to prove that the sides of the rearranged addition formulae are equal modulo t, and that every coordinate is in the correct range. Using the work of [BBB$^+$18] for their range proofs, they prove that a committed value is in a range using a number of field elements that is logarithmic in the bit length of the range. The proof size, which is largely underestimated, is claimed to be roughly $75 + \log\log t$ elements, and prover's work is $60 + \log t$ group exponentiations. This estimation does not take into account that their proofs operate over commitments modulo q which is cubic in t. This would concretely triple their proof lengths (and proof computations).

Proof of DLog (PKDL) in [AGM18]: Internally using PKS, [AGM18] builds a proof, called PKDL, of the equality of a committed value ω and the discrete logarithm to the public base point P of another committed value ωP. For ease of explanation, we describe in detail how the PKDL protocol works (with some clarifications, which are highlighted, that are missing in the original text) in Appendix C.

This proof, however, has several issues. As pointed out by [FLM22], the proof fails to verify C_1 —which is $\omega P + r_1 Q$—, the commitment to the secret value s. This validity check should be done after $\mathcal{V}$ receives Resp, and, without without this check, $\mathcal{P}$ fails to demonstrate knowledge of the opening of C_1. [FLM22] presents a corrected version of this proof which adds the verification of C_1. But even with this correction, the proof does not achieve the special-soundness that it needs. First, [AGM18] does not specify the type of special-soundness that the proof provides: stating informally that the soundness proof is over "two accepting transcripts". Second, it is unclear from the text what challenge is used for the inner PKS(as seen in Section 3.1, which instantiation is referred to as π). Without this, it is not clear as to what extractor strategies may yield an valid witness. Their intent may have been to include an additional challenge round, which is not a Σ-protocol.

On complexity of PKDL. The efficiency of PKDL is roughly dominated by the cost of the range proofs used in the internal PKS, which can be instantiated with [BBB+18]. At soundness level 2^{-60}, which is a relaxed level of security for real-world applications, the proof has an underestimated size of $2370 + \log\log t$ elements, the prover's work is $30 \log t + 1800$ group exponentiations, and the verifier's work is $10 \log t$ group exponentiations. Again, this estimation should be considered in the light that each commitment's computation is modulo q.

3.2 Proofs of Knowledge in ZKA [FLM22]

Proof of Knowledge of the Sum (ZKAPointAddition) in ZKA: ZKA introduces a different strategy to create a PKS. They resolve the need for range proofs by proving relations among commitments which have a message space of matching order. Note that the point addition computation is done over the base field, so the commitments to the coordinates of each point can be done in $\mathbb{F}_q \cong \mathbb{Z}_q$, which is not the same as the order p of $E(\mathbb{F}_q)$. The authors rely on the method given by Bröker's (see Appendix A) to find elliptic curve groups of prescribed order q. Recall that the method takes $\tilde{O}(\log N)^3$ steps, and since it need only be run *once* to fix the curve parameters, with this consideration in mind, it is sufficiently practical.

With elliptic curves of prescribed order q, the authors prove the relations of Theorem 3.1 (note that their paper incorrectly stated the addition formulae for elliptic curve points: we use the correct formulae here) and create the proof seen in Construction 3.1.

We highlight some corrections we made to this proof:

- We fix several equations as highlighted in Construction 3.1. These mistakes stem from either incorrectly copying the point addition formulae or from arithmetic mistakes.
- We emphasise that all "internal" proofs have to be run in parallel, as is standard in the parallel composition of Σ-protocols. The implementation [FLM22] of ZKAPointAddition composes them sequentially, which does not preserve special-soundness [FS90,GK90].

Construction 3.1: ZKAPointAddition

Given $C_1 = \mathrm{Com}_q(a_x)$, $C_2 = \mathrm{Com}_q(a_y)$, $C_3 = \mathrm{Com}_q(b_x)$, $C_4 = \mathrm{Com}_q(b_y)$, $C_5 = \mathrm{Com}_q(t_x)$, $C_6 = \mathrm{Com}_q(t_y)$, prove that $T = A+B$, where $A = (a_x, a_y)$, $B = (b_x, b_y)$, $T = (t_x, t_y)$, $(A, B, T) \in E(\mathbb{F}_q)$.

1. $\mathcal{P}$ computes:

$$C_7 = C_3 - C_1 = \mathrm{Com}_q(b_x - a_x) \quad C_8 = \mathrm{Com}_q((b_x - a_x)^{-1}),$$
$$C_9 = C_4 - C_2 = \mathrm{Com}_q(b_y - a_y) \quad C_{10} = \mathrm{Com}_q\left(\frac{b_y - a_y}{b_x - a_x}\right),$$
$$C_{11} = \mathrm{Com}_q\left(\left(\frac{b_y - a_y}{b_x - a_x}\right)^2\right) \quad C_{12} = \mathrm{Com}_q(a_x - t_x),$$
$$C_{13} = \mathrm{Com}_q\left(\left(\frac{b_y - a_y}{b_x - a_x}\right)(a_x - t_x)\right).$$

2. $\mathcal{P}$ engages with $\mathcal{V}$ in the following Σ-protocols *in parallel* (note that $\mathcal{V}$ can compute C_7 and C_9 from the public values of C_1, C_2, C_3, C_4):
   - ⋆ Multiplication proofs via Construction 2.3:

$$\mathrm{MulProof}(C_7, C_8, \mathrm{Com}_q(1)), \qquad \mathrm{MulProof}(C_8, C_9, C_{10}),$$
$$\mathrm{MulProof}(C_{10}, C_{10}, C_{11}), \qquad \mathrm{MulProof}(C_{10}, C_{12}, C_{13}).$$

   - ⋆ Equality proofs via Construction 2.2:

$$\mathrm{EqualityProof}(C_5, C_{11} - C_1 - C_3), \quad \mathrm{EqualityProof}(C_6, C_{13} - C_2).$$

Note that this proof does not handle exceptional cases, where the public input is commitment to points A, B, T such that $A = \pm B$. It also does not guarantee that the given coordinates correspond to valid curve points, but if A and B are on the curve, so is T, as we note in Remark 1. If the points A, B are randomly chosen, the probability that $A = \pm B$ is $2/|E(\mathbb{F}_q)|$. If this occurs when ZKAPointAddition is invoked in ZKADlog (see Section 3.2), $\mathcal{P}$ can run the entire protocol again, but this admits an issue with perfect completeness. We stress that in ZKADlog, this issue can be resolved without the need for extending the point addition proof. Nevertheless, the ZKA paper (but not in its implementation) does propose an extension to handle exceptional cases at the cost of efficiency. Note that the protocol guarantees that $((a_x - b_x) \neq 0)$ by verifying it has an inverse in the first inner multiplication proof. This can be extended to the case when $((a_x - b_x) = 0)$, which corresponds to $P = \pm Q$, and is either a case of point doubling or addition to the point at infinity (observe below, that the latter is still disregarded). Theoretically, the authors propose handling of exceptional cases by using AND and OR composition of Σ protocols: $(a_x - b_x \neq 0 \wedge t = a + b) \vee (a_x = b_x \wedge a_y = b_y \wedge T = 2P)$ if represented in affine coordinates. This protocol, which would require a subroutine for proving the satisfiability of the doubling formulae (which is unspecified), would be complete provided $A \neq -B$, but still does not account for when $A = -B$, since the points are represented in affine coordinates.

The PKS from [AGM18] does take care of all cases by using AND and OR composition. It gives: $(P \neq Q \wedge P \neq -Q \wedge T = \mathsf{PointAddition}(P, Q)) \vee (P = Q \wedge T = \mathsf{PointDouble}(P)) \vee (P \neq -Q \wedge T = 0)$ by creating a proof, PointDouble, that proves the doubling formulae. It is not clear how they represent the point $T = 0$.

On complexity of ZKAPointAddition*.* The cost of the ZKAPointAddition proof is dominated by the number of field multiplications and field inversions, which are as seen in Construction 2.3. The proof uses 5 initial commitments, 4 multiplication proofs and 2 equality proofs. Considering point multiplications as the dominating cost for complexity, a single execution of the protocol incurs in 36 point multiplications for $\mathcal{P}$, 36 point multiplications for $\mathcal{V}$, and a proof size of 19 elliptic curve points and 22 base field elements.

Remark 1. Note that in ZKAPointAddition above, and later in Section 4.1, it is assumed that at least two of $\mathcal{P}$'s input point coordinates (A, B, T) are indeed valid points on a given curve E. If at least two points are valid, then following the point formulae, it follows that the third must be as well. In its applications, since $\mathcal{V}$ should be convinced of this fact in outer protocols, this seems to be sufficient.

Proof of DLog (ZKADlog) in ZKA: Following the same approach as PKDL, the ZKADlog proof works by internally using the ZKAPointAddition proof. The protocol can be seen in Construction 3.2, where solid boxes represent the values sent by a party. Note that we highlight the steps that were omitted or missing in the original publication, which we incorporate for clarity and completeness.

For a given instance (C_1, C_2', C_3'), the witness is the tuple (ω, r_1, r_2, r_3) such that $(x, y) = \omega P$, $C_1 = \mathrm{Com}_p(\omega, r_1)$, $C_2' = \mathrm{Com}_q(x, r_2)$ and $C_3' = \mathrm{Com}_q(y, r_3)$. Below, we discuss the issues with the original proof's security for this PoK. We note that the PoK is indicated to be a Σ-protocol and it is specified (by the number of repetitions) to have a knowledge error of $\frac{1}{2}$, and perfect HVZK. We explore the failures and limitations of the properties that ZKADlog, as a Σ-protocol, should provide.

Completeness. We remark that the protocol described in Construction 3.2 does not satisfy perfect completeness. If $\alpha \in \{0, \omega, 2\omega\}$, then the inner ZKAPointAddition will fail, since it does not handle point doubling, inverse addition (without its extension), and addition by the identity. This failure, in turn, means that the outer protocol will fail, and we note that these exceptional cases occur with probability $\frac{3}{p}$. A solution to this completeness issue would be to have an honest prover sample from $\mathbb{Z}_p \backslash \{0, \omega, 2\omega\}$. As a trade-off, doing so leads to a protocol that is honest verifier *statistical* zero-knowledge.

Construction 3.2: ZKADlog

Given $C_1 = \mathrm{Com}_p(\omega) = \omega P + r_1 Q$, $C'_2 = \mathrm{Com}_q(x) = xP' + r_2 Q'$, $C'_3 = \mathrm{Com}_q(y) = yP' + r_3 Q'$, for q equal to the modulus of the base field of E, prove that $S = (x, y)$ is equal to ωP, where $P, Q \in E$ are public elements of prime order p, and (P', Q') are points in E' of prime order q.

1. The **prover**::
   - $\star$ chooses a random $\alpha, \beta_1 \in \mathbb{Z}_p$, and β_2, β_3, β_4, β_5 $\in \mathbb{Z}_q$,
   - $\star$ sets $(\gamma_1, \gamma_2) = \alpha P$, and
   - $\star$ sets $(u, v) = (\alpha - \omega)P$

   They, then, compute the following values:
$$a_1 = Com_p(\alpha) = \alpha P + \beta_1 Q,$$
$$a_2 = Com_q(\gamma_1) = \gamma_1 P' + \beta_2 Q', \quad a_3 = Com_q(\gamma_2) = \gamma_2 P' + \beta_3 Q',$$
$$C'_4 = Com_q(u) = uP' + \beta_4 Q', \quad C'_5 = Com_q(v) = vP' + \beta_5 Q'.$$

   sending $\boxed{a_1, a_2, a_3, C'_4, C'_5, \mathsf{Comm}'}$ to $\mathcal{V}$ as Comm,

   where Comm$'$ is for ZKAPointAddition on $(C'_2, C'_3, C'_4, C'_5, a_2, a_3)$.
2. The **verifier**:
   - – chooses a challenge string $c = (c_0, c_1)$, where c_0 is a single random bit $\in \{0, 1\}$ and $c_1 \in \mathbb{Z}_q$ is a challenge for the ZKAPointAddition.

   They send $\boxed{c}$ as Chall.
3. The **prover** receives c:
   - $\star$ If $c_0 = 0$, computes $z_1 = \alpha$, $z_2 = \beta_1$, $z_3 = \beta_2$, $z_4 = \beta_3$.
     Sends the tuple $\boxed{(z_1, z_2, z_3, z_4)}$ as Resp.
   - $\star$ If $c_0 = 1$, computes $z_1 = \alpha - \omega$, $z_2 = \beta_1 - r_1$, $z_3 = \beta_4$, $z_4 = \beta_5$.
     Then, they compute the response for ZKAPointAddition:
     - Given $T = z_1 P = (u, v)$,
     - Compute Resp$'$ with Chall$'$ $= c_1$. which verifies that $T = (\gamma_1, \gamma_2) - (x, y)$ $(T = \alpha P - S)$.

     Sends the tuple $\boxed{(z_1, z_2, z_3, z_4, \mathsf{Resp}')}$ as Resp.
4. Upon receiving Resp, the **verifier** performs the following:
   - $\star$ If $c_0 = 0$, computes $(t_1, t_2) = z_1 P$. Then, verifies that $a_1 \overset{?}{=} z_1 P + z_2 Q$, $a_2 \overset{?}{=} Com_q(t_1, z_3)$ and $a_3 \overset{?}{=} Com_q(t_2, z_4)$.
   - $\star$ If $c_0 = 1$, computes $(t_1, t_2) = z_1 P$. Then, verifies that $a_1 \overset{?}{=} z_1 P + z_2 Q + C_1$, that $C'_4 \overset{?}{=} Com_q(t_1, z_3)$ and $C'_5 \overset{?}{=} Com_q(t_2, z_4)$, and sequentially verifies the point addition proof $\pi = (\mathsf{Comm}', c_1, \mathsf{Resp}')$.

Special Honest Verifier Zero-Knowledge. In order to prove this property, the authors of ZKA construct a simulator, ZKSim, such that its output is perfectly indistinguishable from a honest transcript of the protocol. On input challenge $c = (c_0, c_1)$, the simulator does the following:

– If $c_0 = 0$, ZKSim randomly samples $z_1, z_2 \in \mathbb{Z}_p$, $z_3, z_4, u, v \in \mathbb{Z}_q$, and sets $(t_1, t_2) = z_1 P$ (the x and y coordinate). ZKSim, then, computes $a_1 = z_1 P + z_2 Q$, $a_2 = t_1 P' + z_3 Q'$ and $a_3 = t_2 P' + z_4 Q'$, and sets C'_4, C'_5 as commitments to the random values u, v. Lastly, ZKSim invokes the commitment phase Comm' of the simulator for $\mathsf{ZKAPointAddition}$, with no challenge or response phase, and input $(C'_2, C'_3, C'_4, C'_5, a_2, a_3)$. ZKSim outputs the transcript:

$$((a_2, a_3, C'_4, C'_5, \mathsf{Comm}'), (0, c_1), (z_1, z_2, z_3, z_4)).$$

– If $c_0 = 1$, ZKSim randomly samples $z_1, z_2 \in \mathbb{Z}_p$ and $z_3, z_4, \gamma_1, \gamma_2 \in \mathbb{Z}_q$, and sets $(t_1, t_2) = z_1 P$. ZKSim, then, computes $a_1 = z_1 P + z_2 Q + C_1$, $C'_4 = t_1 P' + z_3 Q'$, $C'_5 = t_2 P' + z_4 Q'$ and a_2, a_3 as the commitments to the random values γ_1, γ_2. Lastly, ZKSim invokes the full simulator for $\mathsf{ZKAPointAddition}$ with input $(C'_2, C'_3, C'_4, C'_5, a_2, a_3)$ and with challenge c_1 which outputs $(\mathsf{Comm}', c_1, \mathsf{Resp}')$. ZKSim outputs the transcript:

$$((a_2, a_3, C'_4, C'_5, \mathsf{Comm}'), (0, c_1), (z_1, z_2, z_3, z_4, \mathsf{Resp}')).$$

Due to the perfect hiding property of PCs [Ped92], simulated transcripts are perfectly indistinguishable from real transcripts. However, resolving the issues described with the completeness of the construction in the manner described would result in *statistical* indistinguishability between simulated and real transcripts, similar to the proof described in Section 4.2.

Special-Soundness. The proof of special-soundness in [FLM22] is flawed due to inherent misdesign. First note that in the original protocol specification and implementation, $\mathcal{P}$ does not construct the commitment phase for the point addition protocol until after the challenge has been received[7], and their implementation computes challenges for the subroutines of the point addition proof on the fly. Sending a challenge before the commitment phase leads to a trivial attack where one can forge a valid point addition proof in the same way the HVZK simulator for $\mathsf{ZKAPointAddition}$ behaves. For the purpose of our argument, we assume that their intent was to send the commitment phase for $\mathsf{ZKAPointAddition}$ in the first round, and, hence, we can compute the challenges for the point addition correctly. Our diagram (Construction 3.2) provides the corrected version.

The authors of ZKA claim the protocol is 3-special-sound, and construct an extractor which takes as input three transcripts, $(\mathsf{Comm}_i, \mathsf{Chall}_i, \mathsf{Resp}_i)_{i\in[3]}$, where $\mathsf{Chall}_1 = (0, a)$, $\mathsf{Chall}_2 = (1, b)$, $\mathsf{Chall}_3 = (1, c)$ with $b \neq c$ and $a, b, c \in \mathbb{Z}_q$.

[7] This is likely also the case in [AGM18], however the specification of the protocol is not sufficiently detailed.

Such an accepting transcript would indeed allow for the extraction of the witness since:

- Given that $\mathsf{ZKAPointAddition}$ is 2-special sound, the extractor can invoke the extractor for this internal proof with transcripts for the challenges b, c. This would yield x, y, r_2, r_3 as output.
- The extractor may take $\mathsf{Resp}_1 = (z_1, z_2, z_3, z_4)$ and $\mathsf{Resp}_2 = (z_1', z_2', z_3', z_4', \pi)$, and deduce $\omega = z_1 - z_1'$ and $r_1 = z_2 - z_2'$.

We note that for this specific input (the stated three transcripts), the witness extracted above is valid. For further justification, see the proof of the fixed protocol in Section 4.2. However, this extractor is still flawed. In particular, there is no way to extract the witness given the following cases with these transcript triples (with $a, b, c \in \mathbb{Z}_q$):

1. $(\mathsf{Comm}_i, \mathsf{Chall}_i, \mathsf{Resp}_i)_{i\in[3]}$ where $\mathsf{Chall}_1 = (0, a)$, $\mathsf{Chall}_2 = (0, b)$, $\mathsf{Chall}_3 = (1, c)$ for $a \neq b$. We explain this extractor fault first since it is the easiest to correct. In this case, the extractor may recover the values ω, x, y, r_1, but not the randomness r_2, r_3. In particular, if we modify the protocol to run $\mathsf{ZKAPointAddition}$ independently of c_0, such that $\mathcal{P}$ engages in it for both $c_0 \in \{0, 1\}$, then, this case can yield a valid witness extraction.
2. $(\mathsf{Comm}_i, \mathsf{Chall}_i, \mathsf{Resp}_i)_{i\in[3]}$ where $\mathsf{Chall}_1 = (1, a)$, $\mathsf{Chall}_2 = (1, b)$, $\mathsf{Chall}_3 = (1, c)$ for $a \neq b \neq c$. In this case, the extractor may recover part of the witness by invoking the extractor of $\mathsf{ZKAPointAddition}$, but the extractor cannot recover ω, only the openings of C_2', C_3'. There is no clear solution that allows the extractor to recover ω in this setting.
3. $(\mathsf{Comm}_i, \mathsf{Chall}_i, \mathsf{Resp}_i)_{i\in[3]}$ where $\mathsf{Chall}_1 = (0, a)$, $\mathsf{Chall}_2 = (0, b)$, $\mathsf{Chall}_3 = (0, c)$ for $a \neq b \neq c$. In this case, it follows that the extractor can learn nothing about the witness, since $\mathsf{Resp}_1 = \mathsf{Resp}_2 = \mathsf{Resp}_3$. If we perform the same modification to the proof as in the first case, we still remain with the same issue as in the second case, where it is not possible to extract ω.

Note that the definition of 3-special soundness requires an extractor to succeed in extracting the witness for *any* 3 accepting transcripts. Therefore, the scheme is not 3-special sound. Furthermore, the claim that the protocol above has soundness error $\frac{1}{2}$ is left unjustified. Recall that an n-special sound protocol of challenge space C has knowledge error $\frac{n-1}{|C|}$. If the scheme was in fact 3-special sound, it would have knowledge error $\frac{2}{2q}$ (since $C = \mathbb{Z}_2 \times \mathbb{Z}_q$), which is unrealistic for this construction.

On complexity of $\mathsf{ZKADlog}$. $\mathsf{ZKAPointAddition}$ dominates the cost of the construction. Since the prover only provides the response to the point addition protocol with probability $\frac{1}{2}$, the expected cost for a single execution of $\mathsf{ZKADlog}$ is 46 point multiplications for $\mathcal{P}$, 24 point multiplications for $\mathcal{V}$, and a proof size of length 24 elliptic curve points and 15 field elements. Note that the scheme uses two curves of different order over different fields which have approximately the same bit-lengths.

3.3 A practical attack on ZKA's Implementation

In addition to the above concerns, the authors of ZKA implemented the non-interactive protocol with the Fiat-Shamir transform applied on 128 repetitions. However, as an efficiency measure (as stated in Section 8 of their paper), the verification is only performed on a random subset of 20 of the repetitions. This ad-hoc choice reduces the probability of a forged proof being accepted to at least 2^{-20} (we note this bound is not tight due to the soundness issues above), but may also allow for a further reduction to security. We sketch out a possible attack strategy below.

A malicious $\mathcal{P}$ may construct the commitment phase for $c_0 = 0$ in the same fashion as the HVZK simulator, ZKSim, for the 128 repetitions, and compute the resultant challenge, which is a hash of the concatenation of these commitments. Then, they arbitrarily select the transcript of a single repetition. They will replace the commitment to C'_4 in this repetition in order to change the output of the resulting challenge hash. This can be done as this commitment is never opened for the verifier and should be uniformly distributed in the point-set of $E'(\mathbb{F}_{q'})$. For i steps, the malicious prover sets C'_4 to a random point[8] in $E'(\mathbb{F}_{q'})$ until the resultant hash yields a challenge string which contains at least m challenges which have $c_0 = 0$. Call these the 'good' challenges. For the 'good' challenges, they complete their proofs as in the ZKSim, and for 'bad' challenges they output uniformly random values as response.

For $m = 115$, the probability that a verifier chooses repetitions which have 'good' challenges, and accepts the forged proof, is roughly 2%. We make some heuristic assumptions as to the practicality of this attack. The expected number of attempts to find a hash which has at least 115 zeroes is $i \approx 2^{70}$. This takes 2^{70} hash computations. Now, assuming each increment's hash only requires a single SHA-256 execution and basing the cost of SHA-256 computations on the revenue of Bitcoin mining[9], the approximate cost of the attack is 1500 USD. [10] Even with the added counter-measure of rate-limiting, this "cheap" attack is likely sufficiently practical to be successful if launched in a distributed fashion. Alongside the other issues, the authors of ZKA have been made aware and acknowledged this attack.

[8] Using the common compressed representation for elliptic curve points, which is the x-coordinate along with a parity bit, one does not need to evaluate the curve equation each time.

[9] See `https://charts.woobull.com/bitcoin-hash-price/`, which places the value of 10^{12} SHA-256 hashes at approximately 10^{-6} USD, assuming modern ASICs can be set up to handle arbitrary fixed length input.

[10] For the calculations of these estimates, see the attached Sage script as seen in Appendix E.

4 Sound Protocols for ZK Verification of ECDSA Signatures under a Committed Public Key: CDLS

In this section, we consider several solutions to the issues faced with the flawed security proof of ZKA's ZKADlog for proving knowledge of an elliptic curve discrete logarithm. Recall that in our context, we are interested in protocols which operate in the elliptic curve setting, where [NBMV99] does not apply.

In Section 4.1, we propose an optimized Σ-protocol for proving that the coordinates of a committed elliptic curve point is the sum of two others, which we call CDLSS. In Section 4.2, we propose a Σ-protocol with knowledge error $\frac{1}{2}$, for a proof of committed DLog, which we call CDLSD. This can be used instead of ZKA to construct a secure NIZKPoK of a valid ECDSA signature verification under a committed public key. Further, in the same fashion as ZKADlog, the NIZKPoK can be composed with a Σ-protocol for proving set membership of a committed key in order to construct a ring signature, described in Section 5.1.

4.1 Proof of Knowledge of the Sum: CDLSS

A key insight that yields optimisation beyond [CM99,FLM22,AGM18], as discussed in Section 2.5,is that only multiplication and a small number of opening proofs are necessary in the process of proving a polynomial relation. In particular, linear combinations of commitments can be obtained without any interaction from $\mathcal{P}$. As an example, consider a $\mathcal{P}$ who wishes to convince $\mathcal{V}$ that taking the product of the opening of two commitments $X = \mathrm{Com}(x)$, $Y = \mathrm{Com}(y)$ is a linear combination of n other committed values, such that for $Z_i = \mathrm{Com}(z_i)$, $i \in [n]$,

$$xy = \sum_{i \in [n]} a_i z_i$$

In this example, $\mathcal{P}$ may verify the relationship holds by having $\mathcal{V}$ compute $T = \prod_{i \in [n]} Z_i^{a_i}$, and engaging them with $\mathrm{MulProof}(X, Y, T)$. This holds since T is a valid commitment for the linear combination. The opening proofs are needed to allow an extractor to recover the openings for the individual commitments given an opening for the linear combination.

Proving knowledge of the sum. Given $C_1 = \mathrm{Com}_q(a_x)$, $C_2 = \mathrm{Com}_q(a_y)$, $C_3 = \mathrm{Com}_q(b_x)$, $C_4 = \mathrm{Com}_q(b_y)$, $C_5 = \mathrm{Com}_q(t_x)$, $C_6 = \mathrm{Com}_q(t_y)$, prove that $T = A + B$, where $A = (a_x, a_y)$, $B = (b_x, b_y)$, $T = (t_x, t_y)$, $(A, B, T) \in E(\mathbb{F}_q)$.

Recall the elliptic curve addition formulae (stated in Theorem 3.1). We may rearrange this formula into a system of equations, by adding an additional variable τ (note that $\tau = \frac{b_y - a_y}{b_x - a_x}$):

$$(b_x - a_x)\tau = b_y - a_y \tag{3}$$

$$\tau^2 = a_x + b_x + t_x \tag{4}$$

$$\tau(a_x - t_x) = a_y + t_y \tag{5}$$

With this rearrangement, along with the previously defined commitments $(C_1, \ldots, C_6)$, $\mathcal{P}$ will also send the commitment $C_7 = \mathrm{Com}_q(\tau)$ in the commitment phase of CDLSS. $\mathcal{P}$ engages $\mathcal{V}$ on the instance $(C_1, \ldots, C_6)$. Note that $\mathcal{V}$ can compute the commitment to any linear combination of known commitments (including C_7) due to the linearity of Pedersen commitments. They perform the following multiplication proof interactions in parallel:

– $\mathrm{MulProof}(C_3 - C_1, C_7, C_4 - C_2)$ which verifies that Eq. (3) holds,
– $\mathrm{MulProof}(C_7, C_7, C_1 + C_3 + C_5)$ which verifies that Eq. (4) holds,
– $\mathrm{MulProof}(C_7, C_1 - C_5, C_2 + C_6)$, which verifies that Eq. (5) holds.
– $\mathrm{OpeningProof}(C_2)$, which allows the extractor to fully recover a witness.

As an abuse of notation, we write the elliptic curve group operations above additively, instead of multiplicatively as in Constructions 2.1 and 2.3. $\mathcal{V}$ accepts if all of the above protocols are accepting.

Theorem 4. *The protocol described above is a Σ-protocol for the relation $\mathcal{R} =$*

$$\left\{ \left(\begin{pmatrix} C_1, C_2, \\ C_3, C_4, \\ C_5, C_6 \end{pmatrix}, \begin{pmatrix} a_x, a_y, b_x, b_y, \\ c_x, c_y, r_1, r_2, \\ r_3, r_4, r_5, r_6 \end{pmatrix} \right) \middle| \begin{array}{c} A + B = T \textit{ where } A \neq \pm B, A, B \neq \mathcal{O} \\ A = (a_x, a_y), B = (b_x, b_y),\ T = (t_x, t_y) \\ \textit{Each commitment } C_i \textit{ is valid} \\ \textit{with randomness } r_i \end{array} \right\}$$

assuming the coordinates of at least two of the points correspond to valid points on an elliptic curve.

Proof. We show the protocol satisfies *completeness*, *honest verifier zero-knowledge* and *special-soundness.*

Completeness. Due to the correctness of the point addition formulae, the multiplication proofs will be accepted, since the coordinates of the points must satisfy Eqs. (3) to (5), given that $A \neq \pm B$, and neither points correspond to the identity element.

Honest Verifier Zero-Knowledge. On input $(C_1, \ldots, C_6)$, and challenge c, the simulator samples random $(a, b) \leftarrow\$ [q]$, and sets $C_7' = \mathrm{Com}_q(a, b)$, adding it to the commitment phase of the transcript. The simulator then invokes the simulators for the three inner multiplication proofs and opening proof where C_7' is used as input in lieu of C_7. The scheme is perfect honest verifier zero-knowledge due to the perfect hiding property of the commitment C_7', and the perfect HVZK of the underlying inner multiplication and opening proofs.

2-Special Soundness. Given two accepting transcripts for the protocol, the extractor invokes the sub-extractor for the three multiplication proofs, respectively. In particular, the extractor learns the quantities $b_x - a_x$, $a_x + b_x + t_x$ and $a_x - t_x$. Note that the extractor also recovers $b_y - a_y$, $a_y + t_y$, $r_4 - r_2$ and $r_2 + r_6$ from the extractors of the multiplication proofs.

To recover the x-coordinates and associated randomness, the extractor solves a system of 3 linear equations in three unknowns and recovers a_x, b_x, and t_x. The randomness r_1, r_3, r_5, which satisfies the same system of equations, is extracted similarly.

To recover the y-coordinates and associated randomness, the extractor invokes the sub-extractor for the opening proof of C_2, learning a_y and r_2. Having the y coordinate and the associated randomness, the extractor can recover b_y, t_y and r_4, r_6 by substituting the known values for a_y and r_2.[11] $\square$

On the complexity of our point addition protocol. CDLSS incurs in 22 point multiplications for $\mathcal{P}$, 26 point multiplications for $\mathcal{V}$, and proofs contain 11 elliptic curve points and 17 field elements.

4.2 Fixing ZKA: Σ-Protocol for Committed Discrete Logarithms: CDLSD

Due to the flawed extractor in ZKADlog, we propose reducing the challenge space of the inner point addition protocol. The intuition behind this choice, is that the outer ZKADlog has knowledge error of at least $\frac{1}{2}$, and thus the inner ZKAPointAddition (which has knowledge error of $\approx \frac{1}{q}$) cannot be utilised properly. Moreover, instead of running the point addition proof conditionally on the response of the verifier, we choose to run it in parallel. Lastly, we also differentiate between the base of the discrete logarithm, R, and the parameter P used as a parameter in the Pedersen commitments, as these need not necessarily be equal.

CDLSD: Sigma Protocol with Binary Challenges. We show CDLSD in Construction 4.1.

Theorem 5. CDLSD *(Construction 4.1) is a Σ-protocol for the relation $\mathcal{R} =$*

$$\left\{ ((C_1, C_2', C_3'), (\omega, r_1, r_2, r_3)) \;\middle|\; \begin{matrix} (x, y) = \omega R,\ C_1 = Com_p(\omega, r_1), \\ C_2' = Com_q(x, r_2),\ C_3' = Com_q(y, r_3) \end{matrix} \right\}$$

assuming the instantiations of Pedersen commitments Com_p and Com_q are perfectly hiding and computationally binding.

Proof. We aim to prove *completeness*, *HVZP* and *Special-Soundness*.

[11] Note that while it is possible to recover the y-coordinates given only the extracted witnesses for the multiplication proofs: it requires evaluating the x-coordinates through the curve equation and deducing the correct choices of sign by the known quantities. This does not allow for the recovery of associated randomness, which is why we include the opening proof for C_2.

Completeness. If $\mathcal{P}$ knows the witness w, and samples an α such that

$$\alpha - \omega \notin \{0, \omega, -\omega\},$$

then the inner CDLSS will accept with probability 1, the equalities in Step 4 of Construction 4.1 will hold, and $\mathcal{V}$ will always accept.

Honest Verifier Zero-Knowledge. We construct a simulator for an accepting transcript which is *statistically indistinguishable* from a real accepting transcript. On input challenge c, $\mathcal{P}$ does the following:

– If $c_0 = 0$, the simulator randomly samples $z_1 \in \mathbb{Z}_p \backslash \{0\}$, $z_2 \in \mathbb{Z}_p$, $z_3, z_4, u, v \in \mathbb{Z}_q$, and sets $(s, t) = z_1 R$. The simulator computes $C_4 = z_1 P + z_2 Q$, $C'_5 = sP' + z_3 Q'$ and $C'_6 = tP' + z_4 Q'$, and sets C'_7, C'_8 as commitments to the random values (u, v).
– If $c_0 = 1$, the simulator randomly samples $z_1 \in \mathbb{Z}_p \backslash \{0\}$, $z_2 \in \mathbb{Z}_p$ and $z_3, z_4, s, t \in \mathbb{Z}_q$, and sets $(u, v) = z_1 R$. The simulator computes $C_4 = z_1 P + z_2 Q + C_1$, $C'_7 = uP' + z_3 Q'$, $C'_8 = vP' + z_4 Q'$ and C'_5, C'_6 as the commitments to the random values (s, t).

In both cases, the simulator then invokes the inner simulator for CDLSS with input $(C'_2, C'_3, C'_7, C'_8, C'_5, C'_6)$ and the binary challenge c.

We show that real and simulated transcripts are *statistically indistinguishable* given that the simulator of CDLSS outputs a transcript that is perfectly indistinguishable from a real transcript and taking into consideration the outer messages sent by $\mathcal{P}$. First, observe that if $c = 0$ (resp. $c = 1$), z_1 in the real transcript is a uniformly random value in $\mathbb{Z}_p \backslash \{0, \omega, 2\omega\}$ (resp. $\mathbb{Z}_p \backslash \{0, -\omega, \omega\}$) and z_1 in the simulated transcript is a uniformly random value in $\mathbb{Z}_p \backslash \{0\}$. Call the real sampling distribution X_q and the simulated sampling distribution Y_q (resp. Y'_q). Note that the statistical distance:

$$\begin{aligned}
\Delta(X_q, Y_q) &= \frac{1}{2} \sum_{x \in \mathbb{Z}_q \backslash \{0\}} |\Pr[X_q = x] - \Pr[Y_q = x]| \\
&= \frac{1}{2}\Big(2\big|0 - \frac{1}{q-1}\big| + \sum_{x \in \mathbb{Z}_q \backslash \{0, \omega, 2\omega\}} \big|\frac{1}{q-3} - \frac{1}{q-1}\big|\Big) \\
&= \frac{1}{q-1} + \frac{1}{q-1} \\
&= \frac{2}{q-1} \quad (= \Delta(X_q, Y'_q) \text{ by a similar argument})
\end{aligned}$$

is statistically indistinguishable, since we require that $q = \exp(\omega)$, which is negligible in ω.

Furthermore, both in the real and simulated proofs, z_2, z_3, z_4 is uniformly random in their respective domains ($z_2 \in \mathbb{Z}_p$ and $z_3, z_4 \in \mathbb{Z}_q$). If $c = 0$ (resp $c = 1$), the verification equations uniquely determine C_4, C'_5, C'_6 (resp. C_4, C'_7, C'_8) conditioned on $(z_1, z_2, z_3, z_4, C_1)$ and that the remaining commitments are to uniformly random inputs in $\mathbb{Z}_q$. Since the commitment scheme is perfectly hiding, the commitments in the real and simulated transcripts are perfectly indistinguishable. Hence, the distribution of the real and simulated transcripts are statistically indistinguishable, where an unbounded distinguisher has at most negligible advantage.

2-Special Soundness. Given two accepting transcripts for the protocol for challenges $c = 0$ and $c' = 1$, the extractor invokes the extractor for the inner CDLSS, which renders r_2, r_3 as openings to the commitments C'_2, C'_3.

Let $(z_1, z_2, z_3, z_4), (z'_1, z'_2, z'_3, z'_4)$ be the responses for c and c', respectively. By the verification equations, we know that C'_5, C'_6 and C'_7, C'_8 are valid commitments to the coordinates of points $z_1 P$ and $z'_1 P$, respectively. Furthermore, since both the transcripts are accepting, CDLSS must correspond to a valid instance, and thus we know that the commitments to the coordinates of ωR, $z_1 R$ and $z'_1 R$ must satisfy the equation $\omega R + z'_1 R = z_1 R$. Lastly, we know by the verification equations, that $C_4 - C_1 = \mathrm{Com}_p(z'_1, z'_2)$ and $C_4 = \mathrm{Com}_p(z_1, z_2)$, and hence we can recover the opening to C_1 as $\omega = z_1 - z'_1$, and $r_1 = z_2 - z'_2$. Hence, the extractor recovers ω, r_1, r_2, r_3. By the point addition proof, and the consistency of the commitments of $z_1 = \alpha, z'_1 = \alpha - \omega$, we must have that for $(x, y) = (z_1 - z'_1)P$, $C'_2 = \mathrm{Com}_q(x, r_2)$ and $C'_3 = \mathrm{Com}_q(y, r_3)$. □

On the complexity of Construction 4.1. The protocol above incurs 32 point multiplications for $\mathcal{P}$, 22 point multiplications for $\mathcal{V}$, and a proof length of 15 elliptic curve points and 21 field elements. Note that $\mathcal{V}$'s complexity is reduced in CDLSS as binary challenges reduce the number of point multiplications required in their checks.

Remark 2. Constructing a 5-round protocol was considered, where in the first round, $\mathcal{P}$ would open one of the points and verify the consistency of the point multiplication commitments. In the second, $\mathcal{P}$ would run the point addition protocol (conditional on the first challenge being 1). We believe such an interactive protocol would be secure, with the techniques described in [AFK22]. However, in the analysis of [AFK22], the authors claim knowledge error loss when Fiat-Shamir transform is applied to parallel repetitions of a multi-round protocol. In this case, the security loss would be quadratic in the number of random oracle queries of an attacker. Since the 5 round protocol would offer very few benefits if provably secure as a $(2, q)$-special-sound protocol, with expected proof lengths roughly $\frac{1}{3}$ shorter than the protocol above in the best case, we opt to remain in the more secure setting of 3-round protocols.

Construction 4.1: CDLSD

Given $C_1 = \mathrm{Com}_p(\omega) = \omega P + r_1 Q$, $C'_2 = \mathrm{Com}_q(x) = xP' + r_2 Q'$, $C'_3 = \mathrm{Com}_q(y) = yP' + r_3 Q'$, for q equal to the modulus of the base field of E, prove that $S = (x, y)$ is equal to ωR, where $R, P, Q \in E(\mathbb{F}_q)$ are public points of prime order p; $P', Q' \in E'(\mathbb{F}_{q'})$ are public points of prime order q and (P, Q), (P', Q') instantiate Com_p and Com_q respectively.

1. **The prover**:
   (a) chooses a random $\alpha, \beta_1 \in \mathbb{Z}_p$, and $\beta_2, \beta_3, \beta_4, \beta_5 \in \mathbb{Z}_q$, such that $\alpha \notin \{0, \omega, 2\omega\}$.
   (b) sets $(s, t) = \alpha R$ (the x and y coordinates of αP),
   (c) sets $(u, v) = (\alpha - \omega)R$ (the x and y coordinates of $((\alpha - \omega)R)$),
   Then, they compute commitments to α, and to the coordinates of αR and $((\alpha - \omega)R)$ $((s, t), (u, v)$, respectively) in the following way:

$$C_4 = \mathrm{Com}_p(\alpha) = \alpha P + \beta_1 Q,$$
$$C'_5 = \mathrm{Com}_q(s) = sP' + \beta_2 Q', \qquad C'_6 = \mathrm{Com}_q(t) = tP' + \beta_3 Q',$$
$$C'_7 = \mathrm{Com}_q(u) = uP' + \beta_4 Q', \qquad C'_8 = \mathrm{Com}_q(v) = vP' + \beta_5 Q'.$$

   Then, they send $C_4, C'_5, C'_6, C'_7, C'_8$ to $\mathcal{V}$. In parallel, they send the commitments for the inner CDLSS (as in Section 4.1), with binary challenge space, and input $(C'_2, C'_3, C'_7, C'_8, C'_5, C'_6)$ that will be used to verify that $\alpha R = \omega R + ((\alpha - \omega)R)$.
2. **The verifier** chooses a random challenge $c \in \{0, 1\}$ and sends it to $\mathcal{P}$.
3. **The prover** receives c and performs the following:
   (a) If $c = 0$, sends $(z_1, z_2, z_3, z_4) = (\alpha, \beta_1, \beta_2, \beta_3)$.
   (b) If $c = 1$, sends $(z_1, z_2, z_3, z_4) = (\alpha - \omega, \beta_1 - r_1, \beta_4, \beta_5)$.
   (c) They send the response to the inner CDLSS with challenge c.
4. Upon response, **the verifier** performs the following:
   (a) If $c = 0$, computes $(s', t') = z_1 R$ and check that $C_4 \stackrel{?}{=} \mathrm{Com}_p(z_1, z_2)$, $C'_5 \stackrel{?}{=} \mathrm{Com}_q(s', z_3)$ and $C'_6 \stackrel{?}{=} \mathrm{Com}_q(t', z_4)$.
   (b) If $c = 1$, computes $(u', v') = z_1 R$ and check that $C_4 - C_1 \stackrel{?}{=} \mathrm{Com}_p(z_1, z_2)$, $C'_7 \stackrel{?}{=} \mathrm{Com}_q(u', z_3)$ and $C'_8 \stackrel{?}{=} \mathrm{Com}_q(v', z_4)$.
   (c) In parallel, verifies the response for the point addition proof with binary challenge c.

4.3 Transforming to Non-Interactive Zero-Knowledge Proof of Knowledge

To boost the soundness of CDLSD, we run λ parallel repetitions of it, and apply[12] the Fiat-Shamir transform [FS87] to obtain a NIZKPoK with knowledge error of

[12] The Fiat-Shamir transformation should be implemented in the standard manner, such as including all public parameters in the challenge oracle query, in order to prevent any weak Fiat-Shamir attacks [DMWG23].

$2^{-\lambda}$. There is still a healthy margin for the statistical zero-knowledge parameter. Given λ repetitions of the protocol, a real and simulated transcript can be distinguished by an unbounded distinguisher with advantage at most $\frac{2\lambda}{2^{2\lambda}-1} \leq 2^{-\lambda}$ (for $\lambda \geq 3$).

Let the commitments for $\mathbb{Z}_q$ (resp. $\mathbb{Z}_p$) correspond to points on an elliptic curve E' (resp E) defined over a field of q' (resp. q) elements. Assume[13] $\log_2 p \approx \log_2 q \approx \log_2 q' \approx 2\lambda$. Then, concretely for $\lambda = 128$, and operating over fields and elliptic curves of order 2λ, this protocol incurs 4096 point multiplications for $\mathcal{P}$ and 2816 point multiplications for $\mathcal{V}$, with proof size of roughly 151 kB.

Remark 3. As a side note, the implementation of ZKA's protocols misses several important considerations in the application of the Fiat-Shamir transform. In particular, public parameters should be included as input to the challenge computation, and challenges should be derived from all information which would be available to the verifier at the time of computation. Without these considerations, one can stage various attacks (described in [DMWG23]) which render the scheme insecure.

5 Constructions on top of our CDLS protocols

We can use the protocols designed in Section 4 to build others "on top". We explore how to instantiate ring signatures (which properties are introduced in Section 2.6) "on top" of the previous protocols and compare our approach to that of ZKA.

5.1 Ring Signatures in ZKA

ZKA proposes a ring signature over ECDSA that is comprised of two parts: a Zero-Knowledge Proof of Knowledge (ZKPoK) and a Groth-Kohlweiss proof of membership (PoM). We describe both below.

Zero-Knowledge Proof of Knowledge (ZKPoK) of an ECDSA signature: Let $t = h(m)$ be a hash of a message, m, to be signed, $C'_{S_x} = \mathrm{Com}_q(S_x)$, $C'_{S_y} = \mathrm{Com}_q(S_y)$ be commitments to a public ECDSA key $S = (S_x, S_y)$, and (r, s) be a valid signature of m under S. Recall the ECDSA verification equation involves computing:

$$R = ts^{-1}P + rs^{-1}S \tag{6}$$

and checking that the x-coordinate of R matches r. In ZKA's construction, $\mathcal{P}$ defines R as above, sets $z = sr^{-1}$ and computes $zR = (u, v)$. Then, they send the values m and R along with commitments $C_1 = \mathrm{Com}_p(z)$, $C'_2 = \mathrm{Com}_q(u)$, $C'_3 = \mathrm{Com}_q(v)$ to $\mathcal{V}$. Then, they perform:

[13] This is a reasonable assumption, since common parameters, such as the ones for `secp256k1`, implement fields of bit length equal to the elliptic curve group order, and Bröker's algorithm heuristically returns an elliptic curve whose order and base field are the same bit length.

– ZKADlog on input (C_1, C'_2, C'_3) to verify that the commitments to the coordinates of zR are valid.
– ZKAPointAddition with input $(C'_4, C'_5, C'_{S_x}, C'_{S_y}, C'_2, C_3)$, which verifies that Eq. (6), multiplied by z on both sides, holds:

$$zR \overset{?}{=} tr^{-1}P + S, \tag{7}$$

where C'_4, C'_5 are commitments to coordinates of $tr^{-1}P$ (with zero randomness). Note that $\mathcal{V}$ can compute these commitments offline since t and r can be determined from m and R.

Groth-Kohlweiss Proof of Membership [GK15] (GKPoM): $\mathcal{P}$ sends the GKPoM, which is used to prove that $\mathcal{P}$'s Pedersen commitment is a valid commitment to an ECDSA public key contained in a given (public) signing ring. This can be achieved with a slight modification to the protocol discussed in the next section. See [GK15] for details.

We can instantiate both the NIZKPoK with our protocols CDLSD and CDLSS, and via straightforward deduction, the ring signature is provably secure. Intuitively, besides perfectly hiding commitments and two zero-knowledge proofs, $\mathcal{P}$ sends only a clear value to a random point R, which is independent of the ECDSA key-pair. This prevents any of the misdesigns previously noted. We stress that the construction's (instantiated in either the ZKA or our manner) anonymity is nullified if the used signature, (r, s), is seen by an adversary. A system must have the precaution of discarding the signature after use.

Remark 4. There is a substantial divergence between the NIZKPoK for an ECDSA signature described in ZKA's paper and in their implementation. Their implementation attempts to consolidate the proof of the sum of Eq. (7) with the internal proof in the call of ZKADlog on zR. For the sake of brevity, and since these modifications are not specified in the security proof, we avoid this approach in our analysis and recommend avoiding it until further consideration of the security implications.

Recall that ZKA's primary motivation for this construction is the ability to create ring signatures for *pre-existing* signing keys, which can be locked behind dedicated hardware or software interfaces. In these cases, the signer only has access to the signing functionality and not directly to the private key. ZKA's ideas can also be used in privacy-preserving authentication mechanisms in order to maintain backwards compatibility with, for example, web standards [WMK+22]. However, there are more practical ways to achieve this functionality if we relax the restriction of using pre-existing signing keys with access to only the signing functionality.

In the scenario where a user has access to the private key, depending on the application, one can perform the following simpler protocols instead:

1. If one wishes to implement a PoK of an ECDSA signature under a committed public key, it is sufficient to construct an identification protocol which proves

knowledge of a discrete logarithm of the committed public key. Using standard techniques, the message is tied into the randomness of the Fiat-Shamir transform (the message is included in the challenge hash computation). This avoids the complications which require modification to a pre-existing signature and an additional proof of the sum.

2. Given the results described in [GK15, Sec. 4], it is possible to construct ring signatures directly from Groth-Kohlweiss proofs of membership whose rings correspond to sets of ECDSA public keys. We remark this is achieved without the need for any novel techniques. Rather than running both a PoM and an ECDSA PoK for a committed public key, the signer only needs to run one PoM. We discuss this idea in the next section.

5.2 Groth-Kohlweiss Ring Signatures for Existing ECDSA Keys

Recall that an ECDSA private-public key pair is the tuple (sk, skP) for some public point P on an elliptic curve $E(\mathbb{F}_q)$. We discuss how to construct a ring signature scheme via [GK15] using a ring $\mathcal{R}$, which is a set of existing ECDSA public keys. In order to construct this ring signature, we introduce a protocol, PMZ, to prove knowledge of the randomness to a commitment to 0 in an ordered list of N commitments $(C_1, \dots, C_N)$.

In PMZ, Construction 5.1, $\mathcal{P}$ possesses the tuple (ℓ, r) such that $C_\ell = \mathrm{Com}_p(0, r)$. $\mathcal{P}$ first commits to the n-bit decomposition of k, written as $(k_1, \dots k_n)$, where, without loss of generality[14], $N = 2^n$. $\mathcal{P}$ engages in n-parallel compositions of a protocol to prove that these commitments lie in the set $\{0, 1\}$, where, in the response, $\mathcal{P}$ reveals evaluations of polynomials $f_1, \dots f_n$ of the form $f_j(c) = \ell_j c + a_j$ (where c is the challenge and the a_j's are randomly sampled elements in the commitment phase). For each $i \in [N]$, $\mathcal{P}$ computes the binary decomposition of i as $(i_1, \dots i_n)$. Then, they define the following polynomials (to be used by both $\mathcal{P}$ and $\mathcal{V}$):

$$f_{j,i_j}(c) = \begin{cases} f_j(c) & \text{if } i_j = 1 \\ c - f_j(c) & \text{if } i_j = 0 \end{cases} \tag{8}$$

Given Eq. (8) and since $\ell_j, i_j \in \{0, 1\}$, it follows that $f_{j,1}(c) = \delta_{\ell_j,1} c + a_j$ and $f_{j,0} = \delta_{\ell_j,0} - a_j$. Notice that $\delta_{a,b}$ is the *Kronecker delta function*, which determines that the function is 1 when $a = b$ and 0 otherwise.

For each i, $\mathcal{P}$ computes the polynomial $p_i(x) = \prod_{j=1}^{n} f_{j,i_j}$. Observe that:

$$p_i(c) = \prod_{j=1}^{n} f_{j,i_j} = \delta_{i,\ell} c^n + \sum_{k=0}^{n-1} p_{i,k} c^k. \tag{9}$$

which means that p_i is a degree n polynomial if $i = \ell$ and a degree $n-1$ otherwise. Upon computing p_i's, $\mathcal{P}$ records the lower degree term coefficients $p_{i,k}$, and uses them to compute the commitments $(C_{d_0}, \dots, C_{d_n})$. The intuition

[14] If $N < 2^n$, one can simply duplicate some of the commitments in the list.

for the construction is that these commitments will cancel out the lower degree terms during the verification step. We refer the reader to [GK15, Sec. 3] for further details.

Construction 5.1: $\mathsf{PMZ}(S)$

Public Parameters: Public parameters for commitment scheme over $\mathbb{Z}_p$.
Inputs: Ordered list $S = (C_i)_{i\in[N]}$ where $\mathcal{P}$ knows a pair (k, ℓ) such that $\ell \in [N]$, $r \in \mathbb{Z}_p$ and $C_\ell = \mathrm{Com}_p(0, r)$. Write $(\ell_1, \dots \ell_n)$ and $(r_1, \dots, r_n)$ as the binary expansion of k and r, where $n = \log_2(N)$.

1. For $j \in [n]$, $\mathcal{P}$ computes the following:
$$r_j, a_j, s_j, t_j, \rho_{j-1} \leftarrow\!\!\$\ \mathbb{Z}_p$$
$$C_{\ell_j} \leftarrow \mathrm{Com}_p(\ell_j, r_j), \qquad C_{a_j} \leftarrow \mathrm{Com}_p(a_j, s_j)$$
$$C_{b_j} \leftarrow \mathrm{Com}_p(\ell_j a_j, t_j), \qquad C_{d_{j-1}} \leftarrow \prod_{i=1}^{N} C_i^{p_{i,(j-1)}} \mathrm{Com}_p(0, \rho_{j-1})$$
(where $p_{i,j}$ is computed as in Eq. (9))
$\mathcal{P}$ sends $C_{\ell_1}, C_{a_1}, C_{b_1}, C_{d_0}, \dots, C_{\ell_n}, C_{a_n}, C_{b_n}, C_{d_{n-1}}$.
2. $\mathcal{V}$ sends challenge $c \leftarrow\!\!\$\ \mathbb{Z}_p$
3. $\mathcal{P}$ computes:
$$z_d \leftarrow rc^n - \Sigma_{k=0}^{n-1} \rho_k c^k$$
and for $j \in [n]$ computes:
$$f_j \leftarrow \ell_j c + a_j, \qquad z_{a_j} \leftarrow r_j c + s_j, \qquad z_{b_j} \leftarrow r_j(c - f_j) + t_j$$
$\mathcal{P}$ sends response $z_d, f_1, z_{a_1}, z_{b_0}, \dots, f_n, z_{a_n}, z_{b_n}$.
4. $\mathcal{V}$ checks that:
$$\prod_{i=1}^{N} C_i^{\prod_{j=1}^{n} f_{j,i_j}} \cdot \prod_{k=0}^{n-1} C_{d_k}^{-c^k} \overset{?}{=} \mathrm{Com}_p(0, z_d)$$
(where f_{j,i_j} is computed as in Eq. (8))
and for $j \in [n]$:
$$C_{a_j} C_{\ell_j}^{c} \overset{?}{=} \mathrm{Com}_p(f_j, z_{a_j})$$
$$C_{b_j} C_{\ell_j}^{c-f_j} \overset{?}{=} \mathrm{Com}_p(0; z_{b_j})$$

Theorem 6. PMZ*(Construction 5.1) is a* $(\lceil \log_2(N) \rceil + 1)$*-special sound* Σ*-protocol for the relation:*

$$\mathcal{R} = \left\{ ((C_1, \ldots, C_N), (\ell, r)) \;\middle|\; \begin{array}{c} C_\ell = Com_p(0, r), r \in \mathbb{Z}_p, \ell \in [N], \\ C_1, \ldots C_N \text{ are valid outputs of } Com_p(\cdot, \cdot) \end{array} \right\}$$

where $Com_p(\cdot, \cdot)$ *is a fixed instantiation of Pedersen commitments over a group of prime order* p*.*

Proof. Follows from [GK15, Thm. 3]

Ring Signature with Existing ECDSA Public Keys. Observe that, given an ECDSA key, we may interpret it as a Pedersen commitment to zero. Consider the public parameters for ECDSA, (p, q, E, Q), where $P \in E(\mathbb{F}_q)$ is a point of prime order p. Let $(sk, pk := skQ)$ be a key-pair of this parameter set. By computing an additional generator P on $E(\mathbb{F}_q)$, we can define a Pedersen commitment parameter set (p, q, E, P, Q) where $P, Q \in E(\mathbb{F}_q)$ and $ord(P) = ord(Q) = p$. Let $\mathrm{Com}_p(m, r) = mP + rQ$. Note that now the ECDSA key-pair above is equivalent to a commitment to zero (with randomness sk): $(sk, \mathrm{Com}_p(0, sk))$.

We instantiate the ring signature described in [GK15, Fig. 3] with the above paradigm in Construction 5.1. $\mathcal{P}$ takes first an instance witness relation pair as Comm, and Chall as second input (which is made non-interactive via Fiat-Shamir). $\mathcal{V}$ is a non-interactive verifier which, when given the Comm, Chall and Resp phases of the protocol as input, performs the verification phase (Step 4.) and outputs 1 if all the equalities hold or 0, otherwise. We define this ring signature scheme as follows:

$pp \leftarrow$ **SetUp(1^λ)**: Outputs public parameters $pp = (p, q, E, Q, P, H)$ for Com_p, the commitment scheme, which is defined as $\mathrm{Com}_p(m, r) = mP + rQ$, where p is a prime chosen in the interval $[2^{2\lambda-1}, 2^{2\lambda}]$, E is an elliptic curve of prime order p over $\mathbb{F}_p$, $(P, Q) \in E(\mathbb{F}_q)$, and H is a random oracle which outputs elements in $\mathbb{Z}_p$.

$(sk, pk) \leftarrow$ **KeyGen(pp)**: Samples $sk \leftarrow\!\!\$\, \mathbb{Z}_p$ and computes $pk \leftarrow \mathrm{Com}_p(0, sk)$.

$\sigma \leftarrow$ **Sign($m, R; pp, sk$)**: Parses R as $(C_1, \ldots, C_N)$ and determines ℓ such that $C_\ell = \mathrm{Com}_p(0, sk)$. Computes Comm $\leftarrow \mathcal{P}(R, (\ell, sk))$ and response Resp $\leftarrow \mathcal{P}(H(m, R, \mathsf{Comm}))$. Outputs $\sigma = (\mathsf{Comm}, \mathsf{Resp})$.

$b \leftarrow$**Verify($m, R, \sigma; pp$)**: Parses $R = (C_1, \ldots, C_N)$ and $\sigma = (\mathsf{Comm}, \mathsf{Resp})$. Computes Chall $\leftarrow H(m, r, \mathsf{Comm})$ and outputs $b \leftarrow \mathcal{V}(R, \mathsf{Comm}, \mathsf{Chall}, \mathsf{Resp})$.

Theorem 7. *The above scheme (SetUp, KeyGen, Sign, Verify) is a ring signature scheme with perfect correctness, perfect anonymity and unforgeability (with respect to insider corruption) given that the instantiation of Pedersen commitments with parameters* (p, q, E, P, Q) *is perfectly hiding and computationally binding (assuming the hardness of the discrete logarithm problem). Moreover, these properties are preserved for signing and verification when using a ring* $\mathcal{R}$ *which consists of ECDSA public keys for parties who possess the corresponding private keys over compatible ECDSA public parameters* (p, q, E, Q)*.*

Proof. See [GK15, Thm. 4]. For compatibility with ECDSA key-pairs, see Section 5.2

$\mathcal{P}$'s cost for the Σ-protocol in Construction 5.1 is dominated by the n multi-exponentiations of N group elements when computing $(C_{d_0}, \ldots, C_{d_{n-1}})$. This computation is claimed to be reduced to roughly $2N$ single exponentiations using optimized multi-exponentiation techniques. Similarly, $\mathcal{V}$'s computations should roughly take $\frac{2N}{\log(N)}$ exponentiations. Note these group exponentiations are realised as point multiplications in the curve used in the Pedersen commitment scheme.

We now give a precise size for the signatures. $\mathcal{P}$'s messages in the membership proof are $4\lceil \log_2(N) \rceil$ curve points and $3\lceil \log_2(N) \rceil$ field elements. Assuming the elliptic curve is defined over a field of the same bit length as the curve's order, 2λ, the signature scheme will yield messages of length $22\lambda \cdot \lceil \log_2(N) \rceil$ bits.

6 Implementation and Benchmarks

In this section, we discuss the implementation details and evaluation of the $\mathsf{CDL}s$ protocols. All timings and results in this section were produced using our prototype Rust implementation [15] on a Macbook M1 with 8 GB of memory. The timings presented in this section were produced as the average of 100 iterations.

We implemented both the PoKs of CDLS and the ones in ZKA in around 4700 lines of Rust code. We note that our re-implementation of ZKA, which still suffers from soundness issues, is primarily for the sake of fair comparison: ZKA's original implementation is written in Typescript, which suffers from a lack of direct hardware access making it less efficient than Rust. We additionally stress that there are some differences between the protocols used in the ZKA implementation and the ZKA paper: the implementation of ZKAPointAddition, for example, does not appear to perform the OR composition specified in the paper. In order to resolve these ambiguities, we based our re-implementation on the original implementation, and not on the paper. We note that we did not re-implement the subset checking algorithm used in ZKA due to security concerns: the verifier in our implementation checks all repetitions.

Table 1 and Table 2 summarise $\mathcal{P}$ and $\mathcal{V}$'s time for PoK of the Sum, PoK of DLog and repeated PoK of DLog for 256-bit and 384-bit curves, respectively. Notably, both Table 1 and Table 2 show that $\mathsf{CDL}s$ performs favourably compared to ZKA, achieving a speed-up for $\mathcal{P}$'s times and with similar times for verification. These timings also show that the subset checking technique used in ZKA is entirely unnecessary, as, even when checking all commitments, our verification timings are comparable to those originally given by ZKA. Note that our $\mathcal{P}$ times are around a factor of ten faster than those reported in ZKA.

We report on the sizes of the proofs in Table 3. Our PoK of the Sum (CDLSS) is around two-thirds the size of those produced by ZKA due to the use of fewer inner proofs. However, we note that the difference is less pronounced for the

[15] `https://github.com/brave-experiments/CDLS`

Table 1: $\mathcal{P}$ and $\mathcal{V}$ timings for the PoK of the Sum, DLog and repeated DLog for 256-bit curves.

	ZKA		CDLS	
Protocol	*Prover (ms)*	*Verifier (ms)*	*Prover (ms)*	*Verifier (ms)*
Sum PoK.	4.20	3.07	2.71	2.39
DLog PoK.	5.75	3.79	4.21	3.11
DLog PoK. (128 rep.)	737	298	539	410

Table 2: $\mathcal{P}$ and $\mathcal{V}$ timings for the PoK of the Sum, DLog and repeated DLog for 384-bit curves.

	ZKA		CDLS	
Protocol	*Prover (ms)*	*Verifier (ms)*	*Prover (ms)*	*Verifier (ms)*
Sum PoK.	10.24	7.68	6.65	5.88
DLog PoK.	14.31	7.90	10.48	7.92
DLog PoK. (192 rep.)	2756	1124	2017	1535

repeated PoK of the DLog (CDLSD). Intuitively, this difference is due to the fact that ZKA only conditionally generates the inner PoK of the Sum, whereas ours always contains the inner proof. Nevertheless, our proof sizes are always approximately the same size as those produced by ZKA. We give additional sizes that showcase the difference between the ammount of inner PoK of the Sum for both ZKA and CDLS in Appendix D.

Table 3: Proof sizes for ZKA and CDLs on 256-bit and 384-bit curves, where the security parameter λ is 128 and 192 respectively.

	256-*bit curve*		384-*bit curve*	
Protocol	ZKA(kB)	CDLS(kB)	ZKA(kB)	CDLS(kB)
Sum PoK.	1.50	1.11	2.23	1.65
DLog PoK.	1.62	1.30	2.52	1.94
DLog PoK. (λ rep.)	162.7	153.7	364.0	344.2

7 Conclusion

In this work, we have highlighted and resolved various security misdesigns in the Σ-protocols presented in ZKA [FLM22] and, to an extent, in Agrawal et al. [AGM18]. These issues stem from implementations which are not faithful to their source paper, and security proofs which miss cases, both of which can be prevented when a protocol description is sufficiently detailed.

It is worth noting that these protocols can be reformulated into multi-round protocols, where $\mathcal{P}$ and $\mathcal{V}$ interact over multiple challenge-response rounds. But these are no longer Σ-protocols, and will require more care with respect to their security proofs and loss factor when Fiat-Shamir is applied. Several 5-round identification protocols, such as [CHR+16,BFK+19], have suffered from attacks due to improper design [KZ20]. However, recent works have published results bounding the knowledge error of NIZKPoKs obtained from the Fiat-Shamir transformation of multi-round interactive proofs [AFK22], and such an approach may be viable.

Future Work. Whilst in this work we focus on the construction of NIZKPoKs from Σ-protocols, modern zkSNARK protocols, such as Spartan [Set20] offer promising performance benefits. These platforms are *succinct*, meaning that proof sizes and verification times scale sub-linearly with the size of the witness. They allow $\mathcal{P}$ to prove generic statements which correspond to proving knowledge of arithmetic circuit satisfiability. The trade-off with zkSNARKs is that they often come at the cost of slower prover times, which may be undesirable in the context of privacy-preserving credentials. There is some work towards an ECDSA-PoK-based ring signature via Spartan with `Spartan-ecdsa` [TSB23], with some promising results. However, there remains work to be done towards proving security of such constructions.

References

AAB$^+$21. Masayuki Abe, Miguel Ambrona, Andrej Bogdanov, Miyako Ohkubo, and Alon Rosen. Non-interactive composition of sigma-protocols via share-then-hash. Cryptology ePrint Archive, Report 2021/457, 2021. https://eprint.iacr.org/2021/457.

AFK22. Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. pages 113–142, 2022.

AGM18. Shashank Agrawal, Chaya Ganesh, and Payman Mohassel. Non-interactive zero-knowledge proofs for composite statements. pages 643–673, 2018.

AKS04. Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

Bab85. L Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 421–429, New York, NY, USA, 1985. Association for Computing Machinery.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. pages 315–334, 2018.

BCC$^+$15. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, Jens Groth, and Christophe Petit. Short accountable ring signatures based on DDH. pages 243–265, 2015.

BCR$^+$19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. pages 103–128, 2019.

BFK$^+$19. Ward Beullens, Jean-Charles Faugère, Eliane Koussa, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. PKP-based signature scheme. pages 3–22, 2019.

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, page 103–112, New York, NY, USA, 1988. Association for Computing Machinery.

BL12. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. Cryptology ePrint Archive, Report 2012/298, 2012. https://eprint.iacr.org/2012/298.

Bou00. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. pages 431–444, 2000.

Bra97. Stefan Brands. Rapid demonstration of linear relations connected by Boolean operators. pages 318–333, 1997.

Bro06. Reinier Broker. *Constructing elliptic curves of prescribed order*. PhD thesis, 2006.

BS07. Reinier Broker and Peter Stevenhagen. Constructing elliptic curves of prime order, 2007.

Cam99. J. Camenisch. Group signature schemes and payment systems based on the discrete logarithm problem, 1999.

CCs08. Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. pages 234–252, 2008.

CDH$^+$23. Sofía Celi, Alex Davidson, Hamed Haddadi, Gonçalo Pestana, and Joe Rowell. Distefano: Decentralized infrastructure for sharing trusted encrypted facts and nothing more. Cryptology ePrint Archive, Paper 2023/1063, 2023. https://eprint.iacr.org/2023/1063.

CDS94. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. pages 174–187, 1994.

CHR$^+$16. Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass MQ-based identification to MQ-based signatures. pages 135–165, 2016.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. pages 93–118, 2001.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. pages 56–72, 2004.

CLL23. Kelong Cong, Yi-Fu Lai, and Shai Levin. Efficient isogeny proofs using generic techniques. In Mehdi Tibouchi and XiaoFeng Wang, editors, *Applied Cryptography and Network Security*, pages 248–275, Cham, 2023. Springer Nature Switzerland.

CM99. Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. pages 107–122, 1999.

CP93. David Chaum and Torben P. Pedersen. Wallet databases with observers. pages 89–105, 1993.

Cra97. Ronald Cramer. *Modular Design of Secure yet Practical Cryptographic Protocols*. PhD thesis, jan 1997.

CS97a. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms, 1997.

CS97b. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). pages 410–424, 1997.

Dam10. Ivan Damgård. On sigma-protocols, 2010. https://www.cs.au.dk/~ivan/Sigma.pdf.

DMWG23. Quang Dao, Jim Miller, Opal Wright, and Paul Grubbs. Weak fiat-shamir attacks on modern proof systems. Cryptology ePrint Archive, Paper 2023/691, 2023. https://eprint.iacr.org/2023/691.

FFS88. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. 1(2):77–94, June 1988.

FHLM21. Armando Faz-Hernández, Watson Ladd, and Deepak Maram. Zkattest: Ring and group signatures for existing ecdsa keys. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography*, pages 68–83, Cham, oct 2021. Springer International Publishing. Available at https://github.com/cloudflare/zkp-ecdsa. v0.2.5 Accessed Nov 2022.

FLM22. Armando Faz-Hernández, Watson Ladd, and Deepak Maram. ZKAttest: Ring and group signatures for existing ECDSA keys. pages 68–83, 2022.

FO97. Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. pages 16–30, 1997.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. pages 186–194, 1987.

FS90. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. pages 416–426, 1990.

GK90. Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 268–282, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

GK14. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. Cryptology ePrint Archive, Report 2014/764, 2014. https://eprint.iacr.org/2014/764.

GK15. Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. pages 253–280, 2015.

Has36. Helmut Hasse. Zur theorie der abstrakten elliptischen funktionenkörper iii. die struktur des meromorphismenrings. die riemannsche vermutung. *Journal für die reine und angewandte Mathematik*, 175:193–208, 1936.

HL10. Carmit Hazay and Yehuda Lindell. *Sigma Protocols and Efficient Zero-Knowledge1*, pages 147–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

Hou22. Youssef El Housni. Pairings in rank-1 constraint systems. Cryptology ePrint Archive, Report 2022/1162, 2022. https://eprint.iacr.org/2022/1162.

KO21. Stephan Krenn and Michele Orrù. Proposal: σ-protocols, 2021.

KZ20. Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. pages 3–22, 2020.

MGGR13. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed E-cash from Bitcoin. pages 397–411, 2013.

Mor05. François Morain. Implementing the asymptotically fast version of the elliptic curve primality proving algorithm, 2005.

NBMV99. Khanh Quoc Nguyen, Feng Bao, Yi Mu, and Vijay Varadharajan. Zero-knowledge proofs of possession of digital signatures and its applications. pages 103–118, 1999.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. pages 129–140, 1992.

Sch84. René Schoof. Elliptic curves over finite fields and the computation of the square roots modulo p. *Mathematics of Computation*, 44, 1984.

Sch87. René Schoof. Nonsingular plane cubic curves over finite fields. *J. Comb. Theory, Ser. A*, 46:183–211, 1987.

Sch91. Claus-Peter Schnorr. Efficient signature generation by smart cards. 4(3):161–174, January 1991.

Sch95. René Schoof. Counting points on elliptic curves over finite fields. *Journal de théorie des nombres de Bordeaux*, 7(1):219–254, 1995.

Set20. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. pages 704–737, 2020.

Sil09. Joseph H. Silverman. *The Geometry of Elliptic Curves*, pages 41–114. Springer New York, New York, NY, 2009.

Tat74. John T. Tate. The arithmetic of elliptic curves. *Inventiones mathematicae*, 23:179–206, 1974.

TSB23. Daniel Tehrani, Lakshman Sankar, and Vivek Bhupatiraju. Spartan-ecdsa. https://github.com/personaelabs/spartan-ecdsa, 2023.

Wik18. Douglas Wikström. Special soundness revisited. Cryptology ePrint Archive, Report 2018/1157, 2018. https://eprint.iacr.org/2018/1157.

Wik21. Douglas Wikström. Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265, 2021. https://eprint.iacr.org/2021/1265.

WMK$^+$22. Tara Whalen, Thibault Meunier, Mrudula Kodali, Alex Davidson, Marwan Fayed, Armando Faz-Hernández, Watson Ladd, Deepak Maram, Nick Sullivan, Benedikt Christoph Wolters, Maxime Guerreiro, and Andrew Galloni. Let the right one in: Attestation as a usable CAPTCHA alternative.

In *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*, pages 599–612, Boston, MA, August 2022. USENIX Association.

WTs$^+$18. Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. pages 926–943, 2018.

ZMM$^+$20. Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: Liberating web data using decentralized oracles for TLS. pages 1919–1938, 2020.

Appendix

A Elliptic Curves of Prescribed Order: Bröker's method

This work considers proving statements about points on elliptic curves $E/\mathbb{F}_q$. In order to prove these statements efficiently, it is convenient to operate using a group that has prime order q, which mitigates the need for range-proofs and provides additional algebraic structure. Concretely, we require a method that accepts an elliptic curve $E/\mathbb{F}_q$ and produces a new elliptic curve $E/\mathbb{F}_p$ with order q, where for the sake of efficiency, $p \approx q$. We stress that this method only needs to be carried out once per curve, and thus a highly efficient method is not mandatory. Nevertheless, we briefly describe how to arrive at efficient methods for computing such curves. We claim no novelty here but rather present an sketch of the techniques that exist in the literature.

We first recall Hasse's theorem. Hasse's theorem on elliptic curves [Has36] states that the order of an elliptic curve $E/\mathbb{F}_q$ is an integer in the Hasse interval $\mathcal{H}_q = \left[q+1-2\sqrt{q}, q+1+2\sqrt{q}\right]$ (a lower and upper bound: the bounds depend only on the finite field and not on the curve) around $q+1$. Notably, the relation $N \in \mathcal{H}_q$ is actually symmetric in N (an integer) and q (i.e $N \in \mathcal{H}_q \iff q \in \mathcal{H}_N$), and thus if q is a prime in $\mathcal{H}_N$, then an elliptic curve $E/\mathbb{F}_q$ with order N always exists. Whilst Hasse's theorem only gives a range for the order of the elliptic curve, we can check $\#E(\mathbb{F}_q)$ in Weierstrass form by using Schoof's algorithm [Sch84] (that is, to count the number of points in an elliptic curve). Schoof's algorithm is a deterministic polynomial time algorithm to compute $\#E(\mathbb{F}_q)$ from a standard representation of E. The problem of finding elliptic curves of prescribed order N, which is of our interest, is an "inverse" problem to the counting problem of Schoof. However, Schoof's algorithm is polynomially bounded in the input size $\log q$, making its immediate use in our problem impractical.

The most basic form of this problem is stated as: if given a finite field $\mathbb{F}_q$ and an integer $N \in \mathcal{H}_q$, find an elliptic curve $E/\mathbb{F}_q$ for which $E(\mathbb{F}_q)$ has order N. There is no known algorithm to solve this problem that is faster than a run-time of $\tilde{O}(N^{1/2})$ (disregarding logarithmic factors): the known algorithm is referred to as "the naïve algorithm". But, we can relax this problem by giving the following correction: given an integer $N \in \mathbb{Z}_{\geq 1}$, find a finite field $\mathbb{F}_q$ and an elliptic curve $E/\mathbb{F}_q$ for which $E(\mathbb{F}_q)$ has order N. The "relaxed" problem can work for applications in cryptography as it is often not needed which finite field the curve E is defined, but rather that the order is prime.

A simple algorithm to find a curve that can work for cryptography is to select a prime p and try random elliptic curves over $\mathbb{F}_p$ until we find a curve of prime order N. This simple algorithm resembles the naïve algorithm for constructing a curve with N points. Using this algorithm, however, is heuristically polynomial time as point counting is polynomial time. The prime number theorem tells us that one out of every $\log p$ integers of size p is prime. If we treat the group orders of the random curves as random integers of size $\log p$, we will have to try about $\log p$ curves until we find the curve of prime order we seek.

Algorithm 1: Naïve algorithm

```
Data: A integer $N > 4$
Result: A prime $p \in [N + 1 - \sqrt{N}, N + 1 + \sqrt{N}]$
Set $a \leftarrow \lceil N + 1 - \sqrt{N} \rceil$                // Step 1
if $a > (N + 1 + \sqrt{N})$ then
    Return and abort
end
if $N$ is odd then
    $p \leftarrow a$, $t \leftarrow p + 1 - N$ and go to line 10.
end
$a \leftarrow a + 1$, and go to to line 2.

Pick a random $b \in F_p^* \backslash \{\frac{-27}{4}\}$                // Step 2
Define $E_b : Y^2 = X^3 + bX - b$ and $P = (1, 1) \in E_b(F_p)$
if $(p + 1 - t)P = O_{E_b}$ then
    Compute the trace of Frobenius $tr$ for $E_b$.
    If $tr = t$, return $E_b$.
end
if $t \neq 0$ and $(p + 1 + t)P = O_{E_b}$ then
    Compute the trace of Frobenius $tr$ for $E_b$.
    If $tr = -t$, return the quadratic twist of $E_b$.
end
Go to line 10.
```

Given Schoof's algorithm and Hasse's theorem we can construct a naïve algorithm to compute elliptic curves of the desired order. First, choose a prime $p \in H_N$(for $N > 4$ as a restriction, so we ensure that $p \geq 5$) and construct random curves over $\mathbb{F}_p$ until a curve of the desired order is found. Recall that Hasse's theorem can be rewritten as $\#E(\mathbb{F}_q) = q + 1 - t$, for $|t| \leq 2\sqrt{q}$, which is called the "trace of Frobenius"[16]. Schoof's basic strategy is simple: compute the trace of Frobenius t modulo many small primes l and use the Chinese remainder theorem to uniquely determine t, which then determines $\#E(\mathbb{F}_q) = q + 1 - t$. Given that

[16] The trace of $\alpha \in End(E)$ is the integer $Tr\alpha = \alpha + \tilde{\alpha}$. Recall here that the Frobenius endomorphism (π_E) is inseparable, but $\pi_E + 1$ is separable.

computing the trace of Frobenius of an elliptic curve $E/\mathbb{F}_p$ takes time polynomial in $\log p$ (given Schoof's algorithm), a following idea is to choose a prime $p \in \mathcal{H}_N$ and construct random curves over $\mathbb{F}_p$ until we have found a correct one. The high level idea of the algorithm is described in Algorithm 1 as given by [Bro06, Alg. 2.3].

Algorithm 2: CM algorithm

```
   Data: A integer $N > 6$ and a prime $p \in H_N$
   Result: An elliptic curve $E\mathcal{F}_p$ with $|E(F_p) = N|$
 1 Compute the Hilbert class polynomial $P_\Delta \in Z[X]$ for
     $\Delta = (p+1-N)^2 - 4p$.
 2 Compute a root $j \in F_p$ of $\tilde{P}_\Delta \in F_p[X]$.
 3 Set $a \leftarrow 27j/(4(1728-j))$ and $E : Y^2 = X^3 + aX - a$ for $j \neq (0, 1728)$.
 4 if $j = 0$ then
 5     Set $E : Y^2 = X^3 + 1$.
 6 end
 7 if $j = 1728$ then
 8     Set $E : Y^2 = X^3 + X$.
 9 end
10 Return a twist of $E$ with $N$ points.
```

The algorithm works as follows: if we find a prime p in Step 1, we know there exists an elliptic curve $E/\mathbb{F}_p$ with $|E(F_p)| = N$ (as stated in [Bro06, Theo. 2.5] with the caveat of [Sch87]). Notice that we look for primes in an interval smaller than the entire Hasse's interval so that the associated discriminant $\Delta = t^2 - 4p$ is not small in absolute value, which gives us a "bigger range" in which to find a curve in Step 2. In Step 2, we assume that there exists a curve $E/\mathbb{F}_p$ with N points and with a j-invariant $j(E) \neq (0, 1728)$. For b ranging over $F_p^* \backslash \{\frac{-27}{4}\}$, the j-invariant reaches every value of $F_p^* \setminus \{1728\}$. For $j \neq (0, 1728)$, there are two non-isomorphic curves E, E' with the consideration that if E has $p + 1 - t$ points, then E' has $p + 1 + t$ points. Said possibilities are tested in lines 12 and 16, respectively. In this case, N satisfies $p + 1 - t$ where t denotes the trace of the Frobenius morphism $\mathbb{F}_p : E \leftarrow E$. The quadratic ring $\mathbb{Z}[\mathbb{F}_p]$ has discriminant $\Delta = t^2 - 4p < 0$, and the endomorphism ring $End_{\mathbb{F}_p}(E)$ contains a subring isomorphic to the imaginary quadratic order O_Δ. Let $E'/\mathbb{F}_p$ be a curve with $\mathbb{Z}[\mathbb{F'}_p] \cong O_\Delta$, where $\mathbb{F'}_p$ is the Frobenius morphism of E'. Following this, we know that one of the twists of E' has trace t and, hence, N points. This argument, then, shows that finding an elliptic curve E with $End_{\mathbb{F}_p}(E) \subseteq O_\Delta$ is equivalent to finding a twist of a curve that has $N = p + 1 + t$ points, where $\Delta = t^2 - 4p$. Said curve can be lifted from a curve in characteristic 0 via the Deuring lifting.

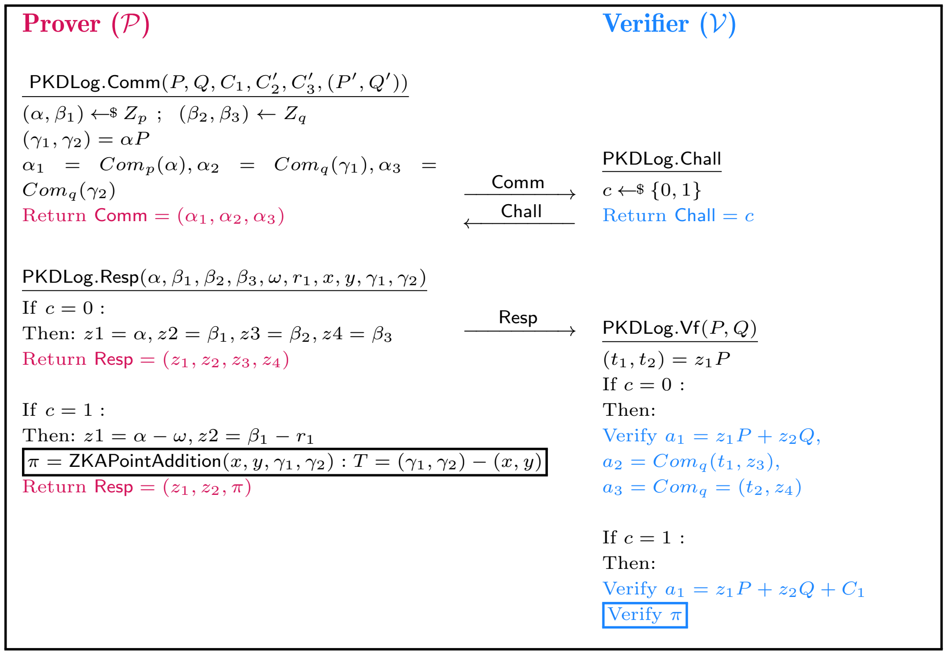
Whilst this algorithm will work, the running time of it is exponential in $\log N$, and thus it is unlikely to be practical. This can be explained as follows. Primality testing is polynomial in time [AKS04], so the total running time will be $\tilde{O}(N)^{1/2}$ (heuristically, polynomial in $\log N$). This primality testing is needed in order to find a prime p that satisfies the $\mathcal{H}_q$ interval ($\left\lceil N + 1 - \sqrt{N}, N + 1 + \sqrt{N} \right\rceil$).

Then, in Step 2, we compute the twist of an elliptic curve (which boils down to finding a representative for $\mathbb{F}_p^*/\mathbb{F}_p^{*2}$), which can be done in time polynomial in $\log p$. Once we have the twists, we have to compute their group orders: using Schoof's algorithm [Sch84], this takes time $\tilde{O}(\log q)$.

We could also solve our original problem by solving the related problem of constructing a curve with $N = p + 1 - t$ (where t is the trace of the Frobenius morphism) points in characteristic zero with a prescribed endomorphism ring, as previously explained. It is known [Tat74], that elliptic curves in characteristic 0 have endomorphism rings of rank at most 2 over $\mathbb{Z}$. If the rank equals 2, the curve is said to be a CM-curve, where CM is an abbreviation for "complex multiplication". The deterministic algorithm that we will see is referred to as the "complex multiplication" (CM) method, as the theory of complex multiplication allows us to construct a curve in characteristic zero with prescribed endomorphism ring. The algorithm can be seen in Algorithm 2 as given by [Bro06, Theo. 3.6]. However, the running time of this algorithm is determined by the time needed to compute the Hilbert class polynomial P_Δ, in line 1, with $\Delta = t^2 - 4q < 0$ which is $O(|\Delta|^{1+o(1)})$. Since $\Delta = O(N)$, we have the run time of $O(N^{1+\epsilon})$ for every $\epsilon > 0$, which is far worse than a probabilistic version of the naïve algorithm. This time can be improved with a multi-evaluation approach in a complex analytic algorithm (a run time of $O(|\Delta|^{(3/2+\epsilon)})$.

A better approach is to use the algorithm proposed by Bröker and Stevenhagen [BS07,Bro06], which uses the "traditional CM-algorithm" but replaces Δ with the fundamental discriminant $D = disc(\mathbb{Q}(\sqrt{\Delta}))$. Note that the polynomial P_D is much smaller than P_Δ. In our problem we usually have many primes $p \in \mathcal{H}_N$ to choose from, and every prime p leads to a field discriminant $D(p) = disc(\mathbb{Q}\sqrt{\Delta})$ with $\Delta = \Delta(p, N) = t^2 - 4p = (p + 1 - N)^2 - 4p$. We need to find the 'minimal' imaginary quadratic fundamental discriminant D, which comes down to solving the following equation: $x^2 - Df^2 = 4N$ in integers x and f in such a way that the number $p = N + 1 - x$ is prime. A heuristic analysis can be given to show that it is reasonable to expect a solution to this problem of size $((logN)^2)$, and we can use the 1908 algorithm of Cornacchia [Sch95] if the factorization of N is given (which allows us to find D in $\tilde{O}((\log N)^{4+\epsilon})$: this can be improved as we will see).

The algorithm itself consists of multiple "search rounds" for a suitable discriminant with a heuristic running time of $\tilde{O}((\log N)^3)$. In more practical terms, the algorithm is able to construct a curve having a large prime number N of points in all cases where the curve, as described in [Mor05], can prove primality of a number of the same size. The starting point of the algorithm is searching through (negative) discriminants D for one that allows an integral solution to the equation $x^2 - Dy^2 = 4N$ (which determines the "minimal" imaginary quadratic fundamental discriminant D —where the imaginary quadratic field is K— containing an element α of norm N —with the property that $N_{K/\mathbb{Q}} = (1-\alpha) = N+1-Tr_{K/\mathbb{Q}}(\alpha)$ is twice a probable prime number N'— which can be used to construct an elliptic curve of order N) with $q = N + 1 - x$. If $\alpha \in K$ is found, N becomes the order of the finite field $\mathbb{F}$ and $2N'$ the number of points of an elliptic curve over $\mathbb{F}$. Solving



Prover ($\mathcal{P}$)

```
PKDLog.Comm$(P, Q, C_1, C_2', C_3', (P', Q'))$
$(\alpha, \beta_1) \leftarrow\$ Z_p$ ; $(\beta_2, \beta_3) \leftarrow Z_q$
$(\gamma_1, \gamma_2) = \alpha P$
$\alpha_1 = Com_p(\alpha), \alpha_2 = Com_q(\gamma_1), \alpha_3 = Com_q(\gamma_2)$
Return Comm $= (\alpha_1, \alpha_2, \alpha_3)$

PKDLog.Resp$(\alpha, \beta_1, \beta_2, \beta_3, \omega, r_1, x, y, \gamma_1, \gamma_2)$
If $c = 0$ :
Then: $z1 = \alpha, z2 = \beta_1, z3 = \beta_2, z4 = \beta_3$
Return Resp $= (z_1, z_2, z_3, z_4)$

If $c = 1$ :
Then: $z1 = \alpha - \omega, z2 = \beta_1 - r_1$
$\boxed{\pi = \mathsf{ZKAPointAddition}(x, y, \gamma_1, \gamma_2) : T = (\gamma_1, \gamma_2) - (x, y)}$
Return Resp $= (z_1, z_2, \pi)$
```

$\xrightarrow{\text{Comm}}$ $\xleftarrow{\text{Chall}}$ $\xrightarrow{\text{Resp}}$

Verifier ($\mathcal{V}$)

```
PKDLog.Chall
$c \leftarrow\$ \{0, 1\}$
Return Chall $= c$

PKDLog.Vf$(P, Q)$
$(t_1, t_2) = z_1 P$
If $c = 0$ :
Then:
Verify $a_1 = z_1 P + z_2 Q$,
$a_2 = Com_q(t_1, z_3)$,
$a_3 = Com_q = (t_2, z_4)$

If $c = 1$ :
Then:
Verify $a_1 = z_1 P + z_2 Q + C_1$
$\boxed{\text{Verify } \pi}$
```

Fig. 1: Visual representation for the PKDLog construction. $\boxed{\text{Solid box}}$ represent the PKS that is run as a inner procedure. The figure omits the fixes and clarifications we introduce in Section 3.2, and serves only for visual purposes.

for the "smallest" D (for which we can use the 1908 algorithm of Cornacchia, with time $\tilde{O}((\log N)^{4+\epsilon})$, which can be lowered to $\tilde{O}((\log N^3))$ via [Mor05]) that admits to a solution, we have that $q = N + 1 - x$ or $q = N + 1 + 1$ are possible base fields if they are prime. Notice that the first n digits of both q and N will be the same, which follows from Hasse's theorem: it differs at most $2\sqrt{q}$ from the size q of the finite field. Once the base field is found, creating the curve requires both computing the Hilbert class polynomial of D (which will split completely) and a root modulo q. The complexity of this algorithm is, hence, $\tilde{O}((\log N)^3)$, given a prime input N. Note that the finite field $\mathbb{F}$ will be of prime order p for some p that is close to N, but if the input N is a prime, E will be over a prime field $\mathbb{F}_p$ with exactly N points over $\mathbb{F}_p$.

We provide a minimal open-source Sagemath script [17] that uses the method of [BS07,Bro06] to compute curves of a prescribed order. This script is only for visualization and testing purposes: we leave the full implementation of the algorithm and improvements as the subject of future work. For 256-bit curves, this script produces a valid curve in a few seconds, with larger (and far less typical) curves being found in around a few minutes.

[17] `https://github.com/brave-experiments/CDLS/tree/main/sage`

B PKDLog from [FLM22] in a visual representation

The visual representation of PKDLog can be seen in Fig. 1. The figure omits the fixes and clarifications we introduce in Section 3.2, and serves only for visual purposes.

Construction 2.1: PKDL from [AGM18]

Given $C_1 = \mathrm{Com}_p(\omega) = \omega P + r_1 Q$, $C_2' = \mathrm{Com}_q(x) = xP' + r_2 Q'$, $C_3' = \mathrm{Com}_q(y) = yP' + r_3 Q'$, for $q > 2t^3$, prove that $S = (x, y)$ is equal to ωP, where $P \in E$ are public elements of prime order p, and (P', Q') are points in E' of prime order q.

1. The **prover**:
   ⋆ chooses a random $\alpha, \beta_1 \in \mathbb{Z}_p$, and $\beta_2, \beta_3 \in \mathbb{Z}_q$,
   ⋆ sets $(\gamma_1, \gamma_2) = \alpha P$
   They then compute the following values:
$$a_1 = Com_p(\alpha) = \alpha P + \beta_1 Q,$$
$$a_2 = Com_q(\gamma_1) = \gamma_1 P' + \beta_2 Q', \quad a_3 = Com_q(\gamma_2) = \gamma_2 P' + \beta_3 Q'.$$
   sending $\boxed{a_1, a_2, a_3}$ to $\mathcal{V}$ as Comm.
2. The **verifier**:
   – chooses a challenge string $c \in \{0, 1\}$.
   They send $\boxed{c}$ as Chall.
3. The **prover** receives c:
   ⋆ If $c = 0$, computes $z_1 = \alpha$, $z_2 = \beta_1$, $z_3 = \beta_2$, $z_4 = \beta_3$.
   Sends the tuple $\boxed{(z_1, z_2, z_3, z_4)}$ as Resp.
   ⋆ If $c = 1$, computes $z_1 = \alpha - \omega$, $z_2 = \beta_1 - r_1$. Then, in parallel, constructs a proof of addition (as in Section 3.1) of the form $\pi = \mathsf{PKS}\{(x, y, \gamma_1, \gamma_2) : T = (\gamma_1, \gamma_2) - (x, y)\}$:
   - Given $T = z_1 P = (t_1, t_2)$,
   - Calculates the proof of addition of $T = (\gamma_1, \gamma_2) - (x, y)$ $(T = \omega P - S)$ with some other challenge bit and calls it π.

   Sends the tuple $\boxed{(z_1, z_2, \pi)}$ as Resp.
4. Upon receiving Resp, the **verifier** performs the following:
   – If $c = 0$, computes $(t_1, t_2) = z_1 P$. Then, verifies that $a_1 \stackrel{?}{=} z_1 P + z_2 Q$, $a_2 \stackrel{?}{=} Com_q(t_1, z_3)$ and $a_3 \stackrel{?}{=} Com_q(t_2, z_4)$.
   – If $c = 1$, computes $(t_1, t_2) = z_1 P$. Then, verifies that
   $a_1 \stackrel{?}{=} z_1 P + z_2 Q + C_1$, and, in parallel,
   verifies π with the proof of addition challenge bit.

C PKDL from [AGM18] in detail

We render now our interpretation of the proof given by [AGM18] informally in Construction 2.1, adding fixes and clarifications. The solid boxes represents the values that are sent by a party.

D DLog PoK sizes and times depending on number of inner PoKs of the Sum in ZKA

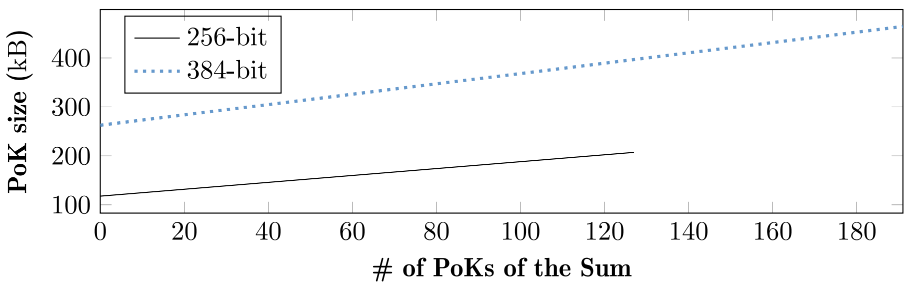


Fig. 2: DLog PoK sizes (in kB) for ZKA depending on the number of included PoKs of the Sum.

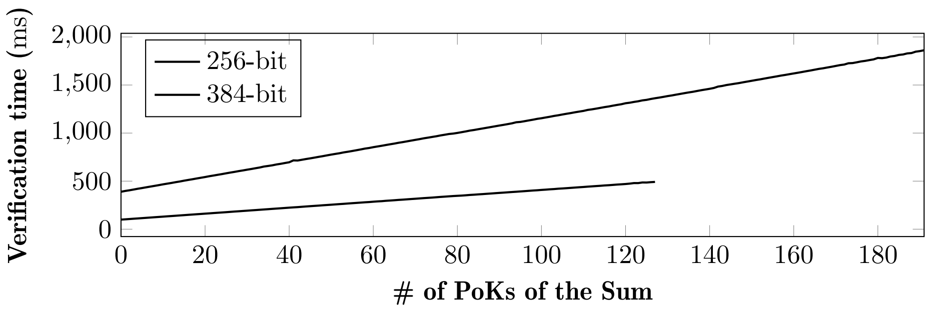


Fig. 3: DLog PoK verification times (in ms) for ZKA depending on the number of included PoKs of the Sum.

E Code to compute the estimates of ZKA's practical attack

```
#Binomial distribution function
#Probability that you get m heads out of n coin flips
def f(n,m):
        return binomial(n,m)*(1/2)^n

#Probability that you get at least m heads out of n coin flips
def g(n,m):
        pr = 0
        for i in range(m,n+1):
                pr += f(n,i)
        return pr

#Probability that you select k heads out of a set of n flipped
# coins of which m are heads.
def h(n,m,k):
        pr = 1
        for i in range(k):
                pr *= (m-i)/n
        return pr


#Input: probability of a verifier accepting if they only check
#m out of n repetitions
#Output: Expected number of times a Prover needs to send a Verifier
#the proof before a Verifier accepts at least once with
#probability 0.99
def numberofrequests(pr):
        return ceil(log(0.01, 1-pr))


#Cost to compute 2^n SHA-256 hashes,
#see https://charts.woobull.com/bitcoin-hash-price/
#(approx 1 microUSD per terahash)
#Note this is a heuristic and oversimplified
def costtohash(n):
        return (2^n) * (10^(-18))

#Parameters
n = 128
m = 115
k = 20

#Expected number of hashes to get at least m good challenge bits
print("Expected number of hashes to get at least {} good
        challenge bits:\n2^{}\n".format(m, -float(log(g(n,m),2))))

#The cost
print("Cost to compute the number of hashes above (assuming you
        only need to run 1 sha-256 which is a very relaxed
        heuristic):\nUSD${}\n".format(float(costtohash(-log(g(n,m),2)))))

#Probability that a verifier selects the good repetitions
print("Probability that a verifier selects the 'good' {} r
        epetitions:\n{}\n".format(m,float(h(n,m,k))))

print("Number of times a malicious prover needs to send a proof (or
        ring signature) to the verifier before they accept with
        probability >0.99:\n{}\n".format(numberofrequests(h(n,m,k))))
```