# ZKAttest: Ring and Group Signatures for existing ECDSA keys

Armando Faz-Hernández[1], Watson Ladd[1], and Deepak Maram[2]

[1] Cloudflare, Inc.
{armfazh,watson}@cloudflare.com
[2] Cornell Tech
sm2686@cornell.edu

**Abstract.** Cryptographic keys are increasingly stored in dedicated hardware or behind software interfaces. Doing so limits access, such as permitting only signing via ECDSA. This makes using them in existing ring and group signature schemes impossible as these schemes assume the ability to access the private key for other operations. We present a $\Sigma$-protocol that uses a committed public key to verify an ECDSA or Schnorr signature on a message, without revealing the public key. We then discuss how this protocol may be used to derive ring signatures in combination with Groth–Kohlweiss membership proofs and other applications. This scheme has been implemented and source code is freely available.

**Keywords:** ring signature · zero-knowledge proof · $\Sigma$-protocol

## 1 Introduction

A *ring signature* scheme allows a signer to sign a message without revealing their identity, thereby providing anonymity behind a ring. A verifier can check the validity of the signature, but cannot know who among the ring members generated the signature. This notion was introduced by Rivest *et al.* [28]. For example, a whistleblower belonging to an organization could reveal sensitive documents using a ring of all members within the organization, demonstrating the authenticity of their claim to be a member while still remaining anonymous.

Existing schemes typically presuppose that every member of the ring has ready access to a suitable key. But establishing such keys across a large group of members purely for the purpose of whistleblowing may attract unwanted attention. Furthermore it is unlikely that organizations would cooperate in issuing keys for use with ring signatures. This problem can be avoided if a ring signature could use *already existing keys*, such as those stored in a hardware security module. A ring signature scheme would then be usable without requiring changes to the hardware or the distribution of new keys across an organization.

For example, WebAuthn [21], a popular standard for hardware authenticators, provides support only for signing messages through a standardized algorithm such as ECDSA [23]. These signatures, and the certificate chain that shows

the authenticity of the public key, reveal which authenticator was used when carrying out attestation, creating a trade-off between confidence in authentication and privacy. Changing the install base of such modules to enhance privacy, for example by supporting an existing ring signature scheme, will take years. Therefore a privacy enhancing form of WebAuthn attestation that works with existing keys is highly desirable: this is the core motivation for ZKAttest.

The core primitive in ZKAttest is *proof of valid ECDSA signature under a committed public key.* A verifier can check signature validity by verifying this proof but learns nothing about the public key, thanks to the hiding commitment. Zero–knowledge proofs then enable us to show additional properties of the public key. For instance we can show that $C$ is a commitment to a value on a public list of keys, thereby producing a ring signature.

**Outline:** Section 2 contains definitions and recalls facts about $\Sigma$-protocols, Pedersen commitments, ring signatures and other things we use. Section 3 lists parameters of the groups used. Section 4 discusses proofs of point addition that we then use in Section 5 to prove knowledge of scalar multiplications. We then apply these scalar multiplication proofs to proof of signature under a committed key in Section 6, and use that proof to derive ring and group signatures in Section 7. We discuss an implementation and application to WebAuthn privacy in Section 8.

## 2 Preliminaries

We will write $G$ for the group of points on an elliptic curve and $g, h, g_1, g_2, \ldots$ be generators of the group. In our construction, we will use two different sets of elliptic curves, and pick one from each. One set, denoted $G_{\text{NIST}}$ are standardized elliptic curves, and the other $G_{\text{Tom}}$ is a set of elliptic curves such that the group order equals the size of the base field of $G_{\text{NIST}}$.

We write $\text{Com}(x; r) = xg + rh$ to denote the computation of a Pedersen commitment to $x$ with randomness $r$. Pedersen commitments are unconditionally hiding, computationally binding, and additively homomorphic [26].

We recall some basic $\Sigma$-protocols and their definition and properties [16,12]. A protocol is a $\Sigma$-protocol if it is a three round protocol satisfying Special Honest-Verifier Zero-Knowledge (SHVZK) and Special soundness. SHVZK implies that we are able to sample from the distribution of transcripts without any knowledge of the witness. Special soundness means given several transcripts with identical commitment phases and different challenges a witness can be extracted.

$\Sigma$-protocols compose in parallel. In this paper we apply this result to prove multiple properties of a common set of commitments, without being forced to spell out each proof. Given commitments $C_1 = xg + r_1h$, $C_2 = yg + r_2h$, and a commitment $C_3$ to the sum, product, difference, or quotient of $x$ and $y$, there is a $\Sigma$-protocol that proves that $C_3$ is in fact a commitment to the sum, product, difference, or quotient of $x$ and $y$. $\Sigma$-protocols can be combined to prove that two statements are simultaneously true, or that one of two statements is true [14].

From a $\Sigma$-protocol we can apply the Fiat-Shamir transform to achieve a *non-interactive zero-knowledge proof* (NIZK) [6,9]. The security definitions of a NIZK in the random oracle model (ROM) involve the existence of a simulator, that can interact with a verifier and produce true seeming proofs by programming the oracle, and an extractor, that interacts with a prover and obtains witnesses.

We recall ring signatures following definitions given by Bender, Katz, and Morselli [8]. Intuitively ring signatures permit signers to hide among a group of their choice, generating only a proof that one among the ring singed without revealing who. Ideally even if all members of the ring have their private keys exposed, the signer will not be discovered. Thus in addition to unforgeability we have a requirement of anonymity.

**Definition 2.1.** *A ring signature scheme is a triple of algorithms* (**Gen**, **Sign**, **Verify**) *such that* $\mathbf{Gen}(1^k)$ *outputs keys* $(sk, pk)$, $\mathbf{Sign}(m, sk_i, R)$ *produces a signature* $\sigma$ *on the message* $m$ *with respect to the ring* $R = \{pk_1, \ldots, pk_n\}$, *and* $\mathbf{Verify}(\sigma, m, R)$ *accepts or rejects the signature.*

**Definition 2.2.** *A ring signature scheme is* complete *if* **Verify** *accepts when run on signatures generated by* **Sign**.

**Definition 2.3.** *A ring signature scheme is* anonymous against full key exposure *if the adversarial advantage in the following game is negligible:*

1. *A set* $S$ *of public keys* $pk_i$ *is generated with* **Gen**.
2. *The adversary is given access to an oracle* **Corrupt** *that on input* $i$ *returns the randomness used to generate* $pk_i$.
3. *The adversary is given access to an oracle* **OSign** *to sign messages of their choice with rings of their choice.*
4. *The adversary outputs a message* $m$, *a ring* $R$ *and a pair of indices* $i_0$ *and* $i_1$ *such that* $pk_{i_0}$ *and* $pk_{i_1}$ *are both in the ring. The game picks a random bit* $b$ *and sends back* $\mathbf{Sign}(m, sk_{i_b}, R)$. *The adversary then attempts to guess who signed. There are no restrictions on picking corrupted indices in this step.*

**Definition 2.4.** *A ring signature scheme is* unforgeable against insider corruption *if the adversarial advantage in the following game is negligible:*

1. *A set* $S$ *of public keys* $PK_s$ *is generated by* **Gen** *and given to the adversary.*
2. *The adversary is given a signing oracle* $\mathbf{OSign}(s, M, R) = \mathbf{Sign}(m, sk_s, R)$ *for* $PK_s \in S$, *where* $PK_s \in R$.
3. *The adversary is also given an oracle* **Corrupt** *such that* $\mathbf{Corrupt}(i)$ *returns* $sk_i$.
4. *The adversary outputs* $(R^*, M^*, \sigma^*)$. *The adversary wins if* $\mathbf{Verify}(\sigma^*, M^*, R^*)$ *accepts,* $R^*$ *does not contain corrupted users and is a subset of* $S$ *and the adversary did not ask the signature oracle to sign* $M^*$ *with the ring* $R^*$.

## 3 Tom Curves

Digital signatures are generated using standardized elliptic curves defined over prime fields [25,30]. These curves are defined by $E/\mathbb{F}_p\colon y^2 = x^3 - 3x + b$, which in turn leads to a prime order group $G_{\mathrm{NIST}} = E(\mathbb{F}_p)$.

In order to prove relations among commitments to values in $\mathbb{F}_p$, (for example, commitments of the coordinates of a point), it is convenient to operate in a group that has order $p$. If we had to use a group that did not have this order, then to prove knowledge of openings of $C_1, C_2, C_3$ to $x, y, xy \bmod p$, we would need to have a group of size at least $p^2$ so that we could prove knowledge of $C_4$ opening to $xy$ over the integers then prove that there was a number $r$ of size at most $p$ such that $t = xy - rp$. This would require an additional range proof and many auxiliary commitments. Furthermore we cannot make use of homomorphisms to carry out addition of commitments.

Instead, we follow the method of Bröker [11] to generate elliptic curves with the desired order, and we call these curves the *Tom curves*. Hence, for a prime $q$, a Tom curve $E'/\mathbb{F}_q\colon y^2 = x^3 + a_4x + a_6$ results in a group $G_{\mathrm{Tom}} = E'(\mathbb{F}_q)$ of prime order equal to $p$. Correspondingly, we use $\mathrm{Com}_{\mathrm{Tom}}$ and $\mathrm{Com}_{\mathrm{NIST}}$ to distinguish between the two commitment functions.

To generate these curves we used complex multiplication. The starting point is searching through (negative) discriminants $d$ for one that allows an integral solution to the problem $x^2 - dy^2 = 4p$. Given such a solution, $q = p + 1 - x$ or $p + 1 + x$ are possible base fields if they are prime. Once the base field is found, creating the curve requires computing the Hilbert class polynomial of $d$ and taking a root modulo $q$. It is possible that the curve found is the twist of the one that is sought, and this can be determined by point-counting.

In the smaller cases, curve generation took few seconds on a commodity laptop, and up to two minutes for the largest instance. The main complexity is computation of the Hilbert class polynomial, which is built into PARI/GP [32]. For a one-time computation our script runs efficiently enough, taking a few minutes wall clock time.

Generating curves whose order is a small multiple of $p$ to have a faster arithmetic (such as in Edwards curves) is also possible. However, it requires a slightly more complicated search for elements of small norm, and one must apply the Decaf group interface to deal with cofactors. To simplify the presentation we focus on prime order Tom curves.

Table 1 shows Tom curves associated to commonly used curves from the FIPS 186-2 [25] and SEC 2 [31] standards.

## 4 Proof of Point Addition

Let $a, b, t$ be points on $G_{\mathrm{NIST}}$ with coordinates $a_x, a_y, b_x, b_y, t_x, t_y$ and $C_1, \ldots, C_6$ be the corresponding commitments computed on the Tom curve (i.e., $C_1 = \mathrm{Com}_{\mathrm{Tom}}(a_x)$, and so on). Now suppose we wish to prove the relation $a + b = t$. There are several special cases.

**Table 1.** Tom curves and their associated parameters.

| Standard curve $E(\mathbb{F}_p)$ | **Tom curve** $E'(\mathbb{F}_q)$ |
|---|---|
| P-256 | **T-256**<br>$d = -4155$<br>$q =$ `ffffffff00000001000000000000000017e72b42b30e7317793135661b1c4b117`<br>$a_4 =$ `776679e96d94aff61a5fb6d256dece8a9162868d9a3fcbcead27946509c31405`<br>$a_6 =$ `c0450ed15e63704c6dfdd9be22fd6bbdfe5f4ccbc43c4d88a2ec905a2af4fef7` |
| P-384 | **T-384**<br>$d = -619$<br>$q =$ `ffffffffffffffffffffffffffffffffffffffffffffffffeaf5f689f8669fb41b08d5f5edffd26599c434bbd978917c5`<br>$a_4 =$ `821dfdc940e7f074ac481f8b2870c48962cce56abd72dfc42813a944cea15df78dc0a2d97fbf031ed26c9076826940ba`<br>$a_6 =$ `9b5b584b655fdcb087d37f8c4fee893c0499223db5e004c674ea0dee48a4ec0c9e9f684099f2a51c62a2cce400cb1e4b` |
| P-521 | **T-521**<br>$d = -28243$<br>$q =$ `2000000000000000000000000000000000000000000000000000000000000002c54be78524c33584f734a266748b2063accf5028e6778dc5056476d0690853249`<br>$a_4 =$ `ef6432c21701cc48c63fb9263e14ba76d4a94ba14d173b134e3032b0e2e543180eb6725125992a7d00162a5f57d21918b0766364eeb53c53bb12f405dac1d527e2`<br>$a_6 =$ `3cbc65d1e0245d79703b18e9aaea1ac6d67f87a2cd4bd84b9e6df6a45a979c481825ca5a857270fc890352f9fac7fd6020deaabb28d099718f0f77a4eec222871d` |

We start with the general case of $a_x \neq b_x$ in which the following affine addition formula [29, Chapter III, Section 2] holds:

$$t_x = \left(\frac{b_y - a_y}{b_x - a_x}\right)^2 - a_x - b_x, \quad \text{and } t_y = \left(\frac{b_y - a_y}{b_x - a_x}\right)(b_x - t_x) - a_y\,.$$

To prove $a + b = t$, the prover computes auxiliary commitments

$$C_7 = \mathrm{Com}_{\mathrm{Tom}}(b_x - a_x) \qquad C_8 = \mathrm{Com}_{\mathrm{Tom}}\left((b_x - a_x)^{-1}\right)$$
$$C_9 = \mathrm{Com}_{\mathrm{Tom}}(b_y - a_y)$$
$$C_{10} = \mathrm{Com}_{\mathrm{Tom}}\left(\frac{b_y - a_y}{b_x - a_x}\right) \quad C_{11} = \mathrm{Com}_{\mathrm{Tom}}\left(\left(\frac{b_y - a_y}{b_x - a_x}\right)^2\right)$$
$$C_{12} = \mathrm{Com}_{\mathrm{Tom}}(b_x - t_x) \qquad C_{13} = \mathrm{Com}_{\mathrm{Tom}}\left(\left(\frac{b_y - a_y}{b_x - a_x}\right)(b_x - t_x)\right)$$

and then proves in parallel that each of these auxiliary commitments opens to the proper value as a sum or product of previous one, and then finally that

$$C_5 = C_{11} - C_1 - C_3, \text{ and } \quad C_6 = C_{13}C_{12} - C_2\,.$$

Each of these auxiliary proofs can be done through either proving a correct multiplication or addition, via known techniques [14]. Note that a proof of proper inversion can be achieved through a proof of multiplication as proving $C_8 = {C_7}^{-1}$ is equivalent to proving $C_8 C_7 = \mathrm{Com}_{\mathrm{Tom}}(1)$.

For our application, the points we must add are unlikely to be in exceptional cases. And if ever they are, the prover can simply try the entire protocol again, as the probability of an exceptional case $a = -b$ or $a = b$ is just $2/|G|$. This proof does not demonstrate that the points are on the curve, but if $a$ and $b$ are then $t$ is guaranteed to be.

To handle the special cases in our addition law we extend our proof to show that $a_x - b_x$ is not zero via showing it has an inverse. Then the two remaining cases possibilities are that $a_y = b_y$ in which case we have a point doubling, or the result is the point at infinity, which cannot be represented in affine coordinates. Since the result is represented in affine coordinates, it cannot be the point at infinity.

Therefore it suffices to prove the following statement, using the OR and AND compositions of $\Sigma$-protocols: $(a_x - b_x \neq 0 \wedge t = a + b) \vee (a_x = b_x \wedge a_y = b_y \wedge t = 2a)$. This demonstrates that $t = a + b$ provided the output can be written in affine coordinates.

The cost of this proof is dominated by the number of field multiplications and field inversions; each of them amounts to a proof of multiplication. Therefore, in the proof-space, the affine formulas are the most efficient formulas to use. A more complicated complete formulas such as the ones from Renes *et al.* [27] would require significantly more operations. A cheaper unified formula that applies to both addition and multiplication in affine coordinates is unknown to the authors.

## 5 Proof of Scalar Multiplication

To make the notation nicer, group elements and commitments of $G_{\mathrm{NIST}}$ will be unprimed while those on $G_{\mathrm{Tom}}$ primed. For example $g, h$ refer to points on $G_{\mathrm{NIST}}$ while $g', h'$ to points on $G_{\mathrm{Tom}}$.

In this protocol, the prover starts with commitments $C_1 = \mathrm{Com}_{\mathrm{NIST}}(\lambda)$, $C_2' = \mathrm{Com}_{\mathrm{Tom}}(x)$ and $C_3' = \mathrm{Com}_{\mathrm{Tom}}(y)$. Their goal is to prove knowledge of opening of these commitments $C_1, C_2', C_3'$ to values $x, y$ and $\lambda$ such that $(x, y) = \lambda g$ where $g$ is a point of $G_{\mathrm{NIST}}$.

This proof is a corrected form of one that appears in Agrawal *et al.* [1]. We discuss the correction towards the end of this section. We now describe a round of the $\Sigma$-protocol. For 128-bit security, we do 128 parallel instances of the below protocol.

1. The prover picks $\alpha, \beta_1, \beta_2, \beta_3$ at random and computes $(\gamma_1, \gamma_2) = \alpha g$, $a_1 = \alpha g + \beta_1 h$, $a_2' = \gamma_1 g' + \beta_2 h'$, and $a_3' = \gamma_2 g' + \beta_3 h'$.

2. The prover lets $C'_4, C'_5$ be commitments to the $x$ and $y$ coordinates of $(\alpha-\lambda)g$. It sends $a_1, a'_2, a'_3, C'_4, C'_5$ to the verifier. It also sends the commitments for a point addition proof showing that $(a'_2, a'_3)$ and $(C'_2, C'_3)$ sum to the point $(C'_4, C'_5)$.
3. The verifier responds with a challenge string $c = (c_0, c_1)$ where $c_0$ is a single bit, and $c_1$ a challenge for the point addition protocol.
4. If $c_0 = 0$ the prover computes $z_1 = \alpha, z_2 = \beta_1, z_3 = \beta_2, z_4 = \beta_3$ and responds with $(z_1, z_2, z_3, z_4)$.
5. If $c_0 = 1$ the prover computes $z_1 = \alpha - \lambda$, $z_2 = \beta_1 - r$. The prover uses the point addition protocol (Sec. 4) to prove knowledge of an opening of $a'_2, a'_3$, $C'_2, C'_3$ and $C'_4, C'_5$ to $\gamma_1, \gamma_2, x, y, u, v$ respectively such that $(u, v) = (\gamma_1, \gamma_2) - (x, y)$ using challenge $c_1$. Call the response $\pi$. The prover sends $(\pi, z_1, z_2)$ to the verifier as well as the opening of $C'_4$ and $C'_5$.
6. If $c_0 = 0$, the verifier simply verifies that the commitments $a_1, a'_2, a'_3$ are opened correctly by $(\alpha, \beta_1, \beta_2, \beta_3)$. If $c_0 = 1$, then the verifier validates $\pi$ using $c_1$ and checks that $z_1 g + z_2 h + C_1 = a_1$, as well as verifying the opening of $C'_4$ and $C'_5$ to $z_1 g$.

We added the verification step $z_1 g + z_2 h + C_1 = a_1$ to the verification algorithm in [1]. Without this the verification ignored $C_1$ entirely, and therefore did not demonstrate knowledge of an opening of $C_1$.

**Theorem 5.1.** *The above protocol is a $\Sigma$-protocol.*

*Proof.* It is easy to see that the protocol is three rounds and that the challenge is a random string of bits. What is not clear is the Special Honest-Verifier Zero-Knowledge and Soundness properties. We prove each one at a time.

*Special Honest-Verifier Zero-Knowledge:* Our goal is to construct a simulator such that the output of the simulator is statistically indistinguishable from the transcript of the protocol with a prover. On the input of challenge $c$, the simulator does the following: If $c_0 = 0$, pick $(\alpha, \beta_1, \beta_2, \beta_3)$ at random and compute $a_1, a'_2, a'_3, C'_4, C'_5$ using the same process as above, then reveal the openings. If $c_0 = 1$, pick $z_1, z_2$ at random and compute $a_1 = z_1 g_1 + z_2 h + C_1$, and let $C'_4, C'_5$ be commitments to the $x$ and $y$ coordinates of $z_1 g$. Then invoke the simulator for the point addition proof, and send over the last move. Since Pedersen commitments are unconditionally hiding and not binding if the discrete logarithm is known, the statement being proved by the point addition proof is true, so the transcript can be simulated.

*Soundness:* We demonstrate that the witnesses $(\lambda, x, y)$ may be extracted given three accepting transcripts for the same commitment: one with $c_0 = 0$ and two with $c_0 = 1$ and differing $c_1$.

Note that $\lambda = z_1 - \widehat{z_1}$ where $z_1$ is from the transcript with $c_0 = 0$ and $\widehat{z_1}$ is from one of the others. Further the prover must have $a'_2, a'_3$ on the curve as they open them when $c_0 = 0$, and $C_4, C_5$ on the curve as they open them when $c_0 = 1$. Therefore $C'_2$ and $C'_3$ are on the curve as they satisfy the addition proof.

We have openings of the commitments involved in $\pi$ by the extractability of $\pi$, and know that $a'_2$ and $a'_3$ are commitments to the $x$ and $y$ coordinates of $\alpha g+\beta_1 h$ for some $\alpha$. Furthermore we know that $z_1 g + z_2 h + C_1 = a_1 = \alpha g + \beta_1 h$, and that $C'_2, C'_3$ are commitments to the $x$ and $y$ coordinates of a point $t$ such that $z_1 g + t = \alpha g$ as the openings are extractable from $\pi$. Therefore $t = (x, y)$ can be extracted as $t = (\alpha - z_1)g$ and $C_1 = (\alpha - z_1)g + (z_2 - \beta_1)h$, which is exactly what is needed to show: $C_1$ is a commitment to the discrete logarithm of $t$. □

## 6 Proof of Knowledge of ECDSA Signature

We have a proof of scalar multiplication, but it does not immediately apply to verification of signatures under committed public keys. To apply these techniques to ECDSA we slightly recast the verification equation to make it more amenable to our techniques. Ordinarily ECDSA verification of a signature $(r, s)$ on a message $m$ takes the form of evaluating $R = u_1 g + u_2 q$ for the public key $q$, and then verifying that the truncation of the $x$ coordinate of $R$ equals $r$. In this equation $u_1 = ts^{-1} \bmod n$ and $u_2 = rs^{-1} \bmod n$ where $t$ is a function of $H(m)$.

An alternative signature scheme instead transmits $(R, z = s/r)$ as the signature, and then verifies the equation $zR - tr^{-1}g = q$ (obtained by multiplying $z$ on both sides of the previous equation). This equation is much more amenable to a zero-knowledge proof. $R$ is independent of the public key, as it is $kg$ for some random $k$, and $t$ is a function of the message alone. Therefore $R$ is independent of the key. Note that this scheme is as secure as ECDSA since the verification equations are equivalent.

It is in this form we apply the scalar multiplication proof to get our protocol. Let $Q$ be the committed public key. The prover transmits $R$, and a commitment to the scalar $z$. The prover uses the scalar multiplication proof to demonstrate correctness of a commitment to $zR$, and then use the point addition proof to demonstrate that the committed $Q$ satisfies $zR - tr^{-1}g = Q$. As $tr^{-1}g$ is a public value, the prover can simply display the opening of its commitments to $tr^{-1}g$ to verify its correctness.

More formally we let $C_{qx}, C_{qy} = \mathrm{Com}_{\mathrm{Tom}}(Q_x), \mathrm{Com}_{\mathrm{Tom}}(Q_y)$ and $C_z = zR + rh$ for a random $r$. Now, as $R$ is an adversarially chosen point, we would have a problem if $R = kh$ for some $k$ known to the adversary. To solve this we must adjust $h$ by generating it via hashing to the NIST curve based on $R$ (refer to Faz-Hernández *et al.* [18] for standard methods for hashing to curves). Then the prover generates an auxiliary commitment $C_{sx}, C_{sy} = \mathrm{Com}_{\mathrm{Tom}}(tr^{-1}g)$. The prover then generates $C_2, C_3$ commitments to the $x$ and $y$ coordinates of $zR$.

Lastly the prover proves knowledge of openings of $C_{sx}$, $C_{sy}$, $C_{qx}$, $C_{qy}$, $C_2$, $C_3$, $C_z$ such that $C_{sx}, C_{sy}$ open to $tr^{-1}g$; $C_2, C_3$ open to the $x$ and $y$ coordinates of $zR$; and $zR - tr^{-1}g = Q$.

We can handle some variations of Schnorr signatures similarly, although not EdDSA [10]. Given a message $m$ and public key $Q = xg$, the signature is $(R, s)$ where $R = kg$, $e = H(R, m)$ and $s = k - xe$. The verification equation is then $sg + eQ = R$. We can recast this as $Q = e^{-1}R - e^{-1}sg$. Here the prover sends $R$,

and a commitment to $e^{-1}s$, then proves that the committed public key is the sum of $e^{-1}R$, known to both, and $e^{-1}sg$, computable through a scalar multiplication proof. Since EdDSA incorporates the public key of the signer into what is hashed, it is difficult to preserve signer privacy while permitting verification. SNARK based approaches would be able to handle this type of signature [19].

## 7 Applications

Having demonstrated that a message is signed with a committed key, we can then prove additional properties of the key. Depending on the properties chosen we obtain group signatures, ring signatures, and proof of non-revocation.

### 7.1 Ring Signatures

Given a list of public keys $k_1, k_2, \ldots, k_n$ in the ring, a signer takes the private key corresponding to their key $k_i$, commits to $k_i$, signs the message $m$, creates a proof of signature under committed key (Sec. 6), and then proves that the commitment is to one of the keys in the list via Groth–Kohlweiss proofs [20]. More formally:

- **Gen** generates an ECDSA keypair.
- **Sign** carries out an ECDSA signature, commits to the key, and proves the signature verifies and the key is on the list.
- **Verify** verifies the proof.

We now formally demonstrate correctness of this application. We start with unforgeability. In Section 2 we recalled a game based definition of the strongest variants unforgeability and anonymity. We now show that our construction achieves both of these, *à la* Chase and Lysyanskaya [15]. To avoid an unfortunate collision in notation we will call the ring $\Gamma$.

**Theorem 7.1.** *Our ring signature scheme achieves anonymity against full key exposure.*

*Proof.* Consider the adversary in Step 4 of Definition 2.3. They have received $R$ and a NIZK for the statement "There is a modified ECDSA signature $(R, z)$ that verifies under a public key Q that is on the list of public keys". But both $sk_{i_0}$ and $sk_{i_1}$ would produce witnesses for this statement. Now consider the adversary interacting with a game that switches which witness is used. By witness indistinguishability, the adversary would produce the same result. Therefore the adversarial advantage must be negligible. □

**Theorem 7.2.** *Our ring signature scheme achieves unforgeability against insider corruption assuming that ECDSA is unforgeable.*

*Proof.* Let **Sim** and **Ext** be the simulator and extractor for our NIZK. We take the adversary in the unforgeability game and provide it with **OSign** that works as follows: Given a private key $sk_i$, ring $\Gamma$, and associated public key $pk_i$ it computes random commitments $C$ for use in our scheme. It generates a proof that $R$ is the $R$ for a signature of a message under a public key in the ring using the simulator, with no witness. This produces a valid signature $\sigma$ for the ring signature scheme that it returns thanks to the simulator. Corruptions we handle by handing back the private key that is requested.

The adversary has no way to know the difference between this game and the original one by the NIZK properties of the Fiat-Shamir transform.

At the final step of the game the adversary returns a signature for a ring it has not queried, that does not contain any corrupted key. Assume this proof is valid. This is the proof we apply **Ext** to, obtaining an ECDSA signature and a public key $pk_i$ such that the signature is valid and $pk_i$ is in the ring. Now at no point was the adversary given a value that reveals information theoretically any private key or ECDSA signature of a value in the ring. Therefore this extracted signature violates the unforgeability of ECDSA. □

We do want to note a subtlety: our signature scheme proves knowledge of a signature on a message. Therefore an adversary who obtains an ECDSA signature of a message under a users key can create their own ring signature showing that someone in a ring containing the user signed the message. This sort of multiprotocol attack is not present in the security definition of ring signatures.

### 7.2 Group Signatures

Once again Chase and Lysynaskaya anticipate us, as do Bellare *et al.* [5,15]. Both papers combine a public-key encryption scheme, a signature scheme, and a zero-knowledge proof system into a group signature scheme. A signer signs with their private key and encrypts the public key to the group manager, and the signature is a zero-knowledge proof that the private key was used to sign the message and that the public key is encrypted to the group manager and is in an accumulator.

Our zero-knowledge proof can be used to show a committed public key was used to sign, and if encryption to the group manager is done via ElGamal encryption [17] the same proof techniques can show the committed public key is correctly encrypted. We can also hash public keys via a Pedersen hash: after chopping the bit expansion of the $x$-coordinate of the key up into small pieces $x_0, x_1, x_2$ we can take $x_0g_1 + x_2g_2 + x_3g_3 + tg_4$ as the hash using our proof of scalar multiplication, where $t$ is a parameter used by the prover to make the output have a prime $x$-coordinate. This enables our scheme to be used together with an RSA accumulator, using proofs for membership of Benarroch *et al.* [7].

### 7.3 Non-revocation

Another application which we shall not treat formally is demonstrating non-membership on a list. In applications such as WebAuthn, attestations are sig-

natures under a key built into the device. Revealing the key reveals the issuing device, harming user privacy. At the same time revocation is necessary to ensure that compromised devices are distrusted. By using a Bayer–Groth proof of non-membership [4], we obtain a proof of signature guaranteeing the use of an unrevoked key; moreover, this proof does not reveal which key was used.

We also discuss a non-application. It is tempting to apply our technique to the case of anonymous credentials [13], where the issuer would sign a message with ECDSA, and a zero-knowledge proof of the credential is then used to anonymize the credential itself. However our scheme hides the identity of the key, not the signature itself, and therefore repeat proofs with the same signature are linkable. Our scheme reveals the commitment to randomness used in the signature, but this alone cannot reveal the key as it is independent of the key.

## 8 Implementation

ZKAttest originated as an effort to improve the privacy of WebAuthn attestation, to enable it to be used as an alternative to CAPTCHAs [2]. WebAuthn attestation takes the form of signing an attestation message with a device attestation key that chains up to a manufacturer's key via the PKI. However, each attestation does reveal a hard-coded certificate associated with the device class. If the certificate were unique, it could be used to track a user's attestation across multiple challenges and make inferences about that user's browsing patterns. The FIDO standard [3, Section 4.1.2.1.1] that specifies certificates should be batched and shared across at least 100,000 devices, so there is a moderately sized list of all valid attestation keys. By using our ring signature to sign the attestation message instead of presenting the ECDSA signature and chain, the user's privacy is further protected as all that is learned is that they have a WebAuthn device that is trusted, rather than which one they have. Currently we have deployed a proof of concept implementation that dynamically adds the attestation key to a much smaller list of randomly generated keys, and then verifies the ring signature server side.

Because our demonstration runs in the browser we wrote our implementation in TypeScript, and implemented our proof for ECDSA with the P-256 curve. No other language would allow us to run in the browser without a complex transpilation step. Internally, we use JavaScript's native big integer library [24] and built on top of it elliptic curve arithmetic for the NIST and Tom curves. The source code is available at https://github.com/cloudflare/zkp-ecdsa .

Note that JavaScript's runtime does not guarantee that operations on big integers run in constant-time; moreover, some operations can be performed much faster when targeting a specific computer architecture. These impact performance negatively, but we still get adequate performance for occasional interactive use by users. Our implementation is capable of proving membership on a list within ten seconds, and can prove membership on lists of up to several thousand with no appreciable slowdown. Verification time is half a second, albeit with an acceptable soundness loss through verifying only some parallel compositions.

We trade soundness for performance by checking a randomly chosen subset of the cases for the scalar multiplication proof. This ensures sufficient security for our application while accelerating verification time. Each case has the same 1/2 failure probability, so after checking 20 cases the probability of forgery is now $2^{-20}$. As the relations to be checked are not known to the prover ahead of time, they must interact with the verifier to attempt cheating. In our application, requiring a million interactions for a single successful forgery is acceptable, because we have other means for rate limiting malicious actors and can afford a small false positive rate. As we continue to optimize our implementation, we can reduce this rate even more. The cost of these checks is linear in the number performed.

Several high-level optimizations were required to achieve this level of performance. The major one is batching verification equations. In our protocol we have a set of commitments, some from the theorems that are to be proven and some from the proofs, $C_1, \ldots C_n$, that are then hashed to form a challenge $e$; and then a set of revealed scalars $s_{ij}$ such that $\sum s_{ij} C_i = 0$ for all $j$. Verifying each of these equations separately is expensive. Our implementation instead selects a random vector of scalars $r_i$, and verifies that $\sum s_{ij} C_i r_i = 0$. If any of the $\sum s_{ij} C_i$ are not the identity, the probability that $r_i$ lies in the kernel of the resulting linear operator is at most $1/|G|$. This technique was also used by Bernstein *et al.* [10] to accelerate verification of multiple signatures.

Some limitations of JavaScript prevent from performing well-known low-level optimizations. For instance, JavaScript's multiprecision arithmetic operators [24] are limited in expressiveness so every field operation involves a full division by the modulus. Montgomery arithmetic is hard to express. Inversion has to be accomplished through either Fermat's little theorem or extended Euclidean algorithm: there is no way to approximate the quotient efficiently for use in asymptotically faster gcd algorithms to make up for the slow division. Each step of the extended Euclidean algorithm is an expensive division even though the quotients are usually small. We have not investigated the use of binary algorithms to accelerate the Euclidean algorithm to potentially accelerate it.

Access to machine word operations is central for obtaining better performance for field operations. An alternative is moving computations to either WebAssembly or AssemblyScript, but we did not explore that path due to time constraints.

## 9 Related Work

There is a rich literature on anonymous credentials, ring signatures, and group signatures, only some of which we have consulted. Agrawal *et al.* [1] presented a proof of exponentiation relation, that our scalar multiplication proof is related to. Their proof had a verification equation that did not include $C_1$. Therefore an attacker could generate any $C_1$ value and the verifier would accept it, contradicting the security claim of that paper. We have verified the correction we present with the authors of that paper. Agrawal *et al.* consider the challenges of

proving statements with both algebraic relations and "arithmetic" ones, while our techniques avoid "arithmetic" relations. In addition Agrawal *et al.* say that the complex multiplication method is quite inefficient and makes protocols impractical, choosing to use a fixed, much larger group instead. The opposite is true: a short one time calculation reduces the number of commitments and the size of the group.

The general approach of constructing ring signatures via proofs on signatures is found in Chase and Lysyanskaya [15], although they approach the question with an eye toward delegatable anonymous credentials, and do not provide a proof of the intuitive claim that their ring signature scheme is secure. Our work can be seen as instantiating their scheme for an ECDSA signature, but in settings where the message is public.

SNARKs provide a general method to create a succinct argument of knowledge for any circuit [19]. One could apply such a technique to the ECDSA verification circuit and thus obtain a proof of signature generically, just as we have applied the composition of $\Sigma$-protocols to obtain one by hand. SNARKs are deployed in Zcash [22] where proofs of signatures are used and efficient verification is a requirement. However SNARK design and compilation is tricky, with very few tools targeting JavaScript and SNARKs depend on unfalsifiable assumptions even in the CRS model, and the common reference strings are large. In the future these barriers are likely to improve with further research.

Groth and Kohlweiss considered applying their one-out-of-many scheme to produce ring signatures that use a preexisting PKI [20]. A user would reveal that they know how to open their public key to zero, and use this for a ring signature scheme. Our work can be seen as a natural extension to use preexisting infrastructure such as smart cards or other forms of hardware with a limited interface for the use of private keys.

## 10 Conclusion

We have presented a practical scheme with minimal assumptions for proving knowledge of an ECDSA signature under a committed key. Our approach does not use expensive pairing computations, is practical and efficient, and widely applicable. Unlike SNARK based approaches we do not have a large common reference string and our security does not need knowledge assumptions [19]. By decoupling the proof of the signature from proving information about the key it enables a great array of applications, such as improving the privacy of remote attestation and demonstrating that keys have not been revoked. Our proof-of-concept implementation demonstrates that our techniques are practical even in a limited computing environment, such as a web browser. We believe ZKAttest has numerous additional applications.

## References


1. Agrawal, S., Ganesh, C., Mohassel, P.: Non-interactive zero-knowledge proofs for composite statements. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. p. 643–673. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96878-0_22
2. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using Hard AI Problems for Security. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. p. 294–311. Springer, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_18
3. Balfanz, D., Czeskis, A., Lundberg, E., Jones, J., Hodges, J., Jones, M., Lindemann, R., Kumar, A., Liao, H.: FIDO UAF Protocol Specification v1.0. FIDO Alliance Standard, FIDO (Dec 2014), https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html
4. Bayer, S., Groth, J.: Zero-knowledge argument for polynomial evaluation with application to blacklists. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. p. 646–663. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_38
5. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656, pp. 614–629. Springer (2003). https://doi.org/10.1007/3-540-39200-9_38
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993. pp. 62–73. ACM (1993). https://doi.org/10.1145/168588.168596
7. Benarroch, D., Campanelli, M., Fiore, D., Kolonelos, D.: Zero-knowledge proofs for set membership: Efficient, succinct, modular. Cryptology ePrint Archive, Report 2019/1255 (Oct 2019), https://eprint.iacr.org/2019/1255
8. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. J. Cryptol. **22**(1), 114–138 (2009). https://doi.org/10.1007/s00145-007-9011-9
9. Bernhard, D., Pereira, O., Warinschi, B.: How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. p. 626–643. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_38
10. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of Cryptographic Engineering **2**(2), 77–89 (2012). https://doi.org/10.1007/s13389-012-0027-1
11. Bröker, R.: Constructing Elliptic Curves of Prescribed Order. Ph.D. thesis, Leiden (2006)
12. Camenisch, J.: Group signature schemes and payment systems based on the discrete logarithm problem. Ph.D. thesis, ETH Zurich (1998)
13. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001. p. 93–118. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_7

14. Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: Stern, J. (ed.) Advances in Cryptology — EUROCRYPT '99. p. 107–122. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_8
15. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006. p. 78–96. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11818175_5
16. Damgård, I.: On $\Sigma$-protocols (2010), https://www.cs.au.dk/~ivan/Sigma.pdf
17. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer (1984). https://doi.org/10.1007/3-540-39568-7_2
18. Faz-Hernández, A., Scott, S., Sullivan, N., Wahby, R.S., Wood, C.A.: Hashing to Elliptic Curves. Internet-draft, Internet Engineering Task Force (Apr 2021), https://datatracker.ietf.org/doc/draft-irtf-cfrg-hash-to-curve/, (work in progress)
19. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 305–326. Springer (2016). https://doi.org/10.1007/978-3-662-49896-5_11
20. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 253–280. Springer (2015). https://doi.org/10.1007/978-3-662-46803-6_9
21. Hodges, J., Jones, J., Jones, M.B., Kumar, A., Lundberg, E.: Web Authentication: An API for accessing Public Key Credentials - Level 2. W3C recommendation, W3C (Apr 2021), https://www.w3.org/TR/webauthn-2
22. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification (Aug 2021), https://zips.z.cash/protocol/protocol.pdf
23. Johnson, D., Menezes, A., Vanstone, S.: The Elliptic Curve Digital Signature Algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (Aug 2001). https://doi.org/10.1007/s102070100002
24. MDN contributors: Bigint (2021), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/BigInt
25. National Institute of Standards and Technology: FIPS 186-2: Digital Signature Standard (DSS). Federal Information Processing Standards Publication (Jan 2000), https://csrc.nist.gov/CSRC/media/Publications/fips/186/2/archive/2000-01-27/documents/fips186-2.pdf
26. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. p. 129–140. Springer, Berlin, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
27. Renes, J., Costello, C., Batina, L.: Complete addition formulas for prime order elliptic curves. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings,

Part I. Lecture Notes in Computer Science, vol. 9665, pp. 403–428. Springer (2016). https://doi.org/10.1007/978-3-662-49890-3_16
28. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2248, pp. 552–565. Springer (2001). https://doi.org/10.1007/3-540-45682-1_32
29. Silverman, J.H.: The geometry of elliptic curves. In: The Arithmetic of Elliptic Curves, chap. 3, p. 41–114. Springer, New York, NY (2009). https://doi.org/10.1007/978-0-387-09494-6_3
30. Solinas, J.A.: Generalized Mersenne Numbers. Tech. rep., Centre for Applied Cryptographic Research, University of Waterloo (Jun 1999), https://cacr.uwaterloo.ca/techreports/1999/corr99-39.pdf
31. Standards for Efficient Cryptography Group: SEC 2: Recommended Elliptic Curve Domain Parameters. Standards for Efficient Cryptography (SEC) (Sep 2000), https://www.secg.org/sec2-v1.pdf
32. The PARI Group, Univ. Bordeaux: PARI/GP version `2.13.0` (2019), available from http://pari.math.u-bordeaux.fr/