# Morphic Accumulators and Applications: Optimal Range Proofs, Polynomial Commitments, and Ring Signatures

Dimitrios Papadopoulos[1], Qiang Tang[2], and Jiajun Xin[2⋆✉]

[1] Hong Kong University of Science and Technology
dipapado@cse.ust.hk
[2] The University of Sydney
{qiang.tang, jiajun.xin}@sydney.edu.au

**Abstract.** Cryptographic accumulators based on groups of unknown order (GUO) provide constant-size set membership proofs. For security purposes, existing works require first encoding set elements via division-intractable (DI) hash functions, typically instantiated as random oracles that destroy any algebraic structure. This confines GUO-based accumulators to a purely set-membership role, making them "incompatible" with various existing cryptographic proof techniques over committed integers in the same groups as the GUO, such as constant-size proofs of exponentiation and modular exponent relations. We introduce the notion of *morphic accumulators*, which replaces the DI hash with a discrete logarithm encoding $H_g(x) = g^x$, mapping set elements to a group before accumulation. We prove, under a variant of the subset product assumption in the generic group model, that this encoding is inherently division intractable, achieving the same security guarantee as random-oracle DI hashes, while simultaneously being a group homomorphism: accumulated elements retain their group-algebraic relationships. This resolves a fundamental tension between compact representation and algebraic structure: the accumulator serves simultaneously as a binding commitment to a set and as a substrate for homomorphic computation over its elements.
Morphic accumulators yield asymptotically optimal constructions across multiple domains: range proofs with $O(n)$ prover time, $O(1)$ proof size, $O(1)$ verification with transparent setups (the first scheme to simultaneously achieve these optimal bounds); polynomial commitments with $O(n)$ prover and $O(1)$ proof size, resolving the cubic bottleneck in prior constant-proof-size GUO-based schemes; and the first linkable ring signatures with $O(1)$ signature size, transparent setup, $O(n)$ offline signing and $O(1)$ online signing.

**Keywords:** Cryptographic accumulators · Groups of unknown order · Range proofs · Polynomial commitments · Ring signatures.

## 1 Introduction

Cryptographic accumulators [7] are a fundamental primitive that represent a large set as a single compact value, supporting efficient proofs of (non-)membership [9, 20, 55, 56]. They have become essential tools, e.g., for blockchains [23, 82, 87], digital signatures [12, 35, 51, 65, 71], and authenticated data structures [74, 83]. Among the various accumulator designs, including Merkle trees [60], bilinear accumulators [40, 65, 80, 94], and lattice-based constructions [70, 73], those based on groups of unknown order (GUO) offer distinct advantages: constant-size batched membership proofs via succinct proof-of-exponentiation techniques [9, 88], and transparent setup when instantiated over class groups [15, 42] or hyperelliptic curves [33, 81]. GUO-based accumulators have already found

⋆ Most of this work was performed while the author was affiliated with Hong Kong University of Science and Technology.

applications in stateless blockchains [9], anonymous credentials [20, 21, 90], revocation systems [55], and building authenticated data structures [84].

**Limitation of the "hash-then-accumulate" paradigm.** In a GUO-based accumulator, a set $S = \{s_1, \ldots, s_n\}$ of integers is accumulated as $Acc = g^{s_1 \cdot s_2 \cdots s_n}$, where the exponent is the integer product of the set elements. Membership of $s_i$ is proved by exhibiting a witness $w_i = g^{\prod_{j \neq i} s_j}$ satisfying $w_i^{s_i} = Acc$. The soundness of this construction requires that the accumulated integers be *mutually indivisible*: no element $s_i$ should divide the product of the remaining elements, as this would allow forging a valid membership witness for elements outside the set. The early approaches [7,20,55] ensure this by mapping each element to a distinct prime before accumulation: since no prime divides a product of other primes, the resulting encodings are trivially indivisible. This indivisible requirement was later formalized by Gennaro, Halevi, and Rabin [39] as *division intractability* (DI). A simpler, more efficient construction interprets random-oracle outputs as $L_{\mathsf{DI}}$-bit integers: here $\lambda$ is the security parameter and $L_{\mathsf{DI}}$ denotes the DI output length; classical analyses use the sufficient condition $L_{\mathsf{DI}}(\lambda) = \Theta(\lambda^2)$ [27, 39].[3] Such outputs contain large prime factors with overwhelming probability, making it intractable for any output to divide the product of others [27, 39, 56]; concretely, [50] gives $L_{\mathsf{DI}} = 1704$ bits for 128-bit DI security.

Whether using hash-to-prime or random oracle DI, both constructions share a fundamental consequence: the encoding *destroys all algebraic relationships* between the original elements. Once hashed, elements become opaque integers bearing no structural connection to their original form. This is consequential because GUOs already support powerful arithmetic proof techniques, such as PoKE [9] and PoKEModN [91], that can establish exponentiation and modular relations over committed integers with constant-sized proofs. Yet these techniques cannot operate on DI-hashed accumulator elements, because the hash severs the algebraic link between the original values and their accumulated representations. The standard paradigm thus confines accumulators to a purely set-membership role, preventing their integration with the arithmetic proof capabilities of the same underlying group.

### 1.1 Our Approach: Morphic Accumulation

Division intractability depends on the *statistical distribution* of the encoding, namely that it is uniform over a sufficiently large domain, not on the specific computational process that produces it. Traditional DI hashes (random oracles, hash-to-prime) achieve the correct distribution by destroying algebraic structure; but this destruction is an artifact of their construction, not a requirement of the property. We exploit this observation to replace the DI hash with a strictly stronger primitive: an encoding that achieves division intractability *while simultaneously preserving algebraic structure*.

We show that discrete logarithm encodings are division intractable in the generic group model [79], and that they satisfy the stronger group element division intractability (GEDI) property in the generic group model under the modular subset-product assumption (Assumption 1). Consider the encoding $H_g(x) = g^x \bmod M$, where $g$ is a generator and the output is the canonical integer representation. In the generic group model, $H_g(x)$ is uniformly random in the encoding space. When $L_{\mathsf{DI}}$ is chosen according to the DI calibration above, each output contains a large prime factor with overwhelming probability, achieving division intractability by the same mechanism as random oracle DI hashes.

Critically, this encoding also preserves algebraic structure: it is a *group homomorphism*, satisfying $H_g(x_1) \cdot H_g(x_2) = g^{x_1+x_2} = H_g(x_1 + x_2) \pmod M$. Accumulated elements remain group elements

[3] Subsequent work [69] reduces the random-oracle output length to $2\lambda$ bits under a heuristic conjecture.

with intact algebraic relationships, enabling proofs of arithmetic relations over the accumulated data. We call this *morphic accumulation*.[4]

To appreciate why this combination is significant, observe that compact representation and algebraic structure are ordinarily in tension. Combining group elements via the group operation ($g^{x_1} \cdots g^{x_n} = g^{\sum x_i}$) yields a single compact value but is irreversible: the individual inputs collapse, and no efficient proof can recover which elements were combined. DI-hash accumulation avoids this collapse by preserving each element as a separate factor in the accumulated product (extractable through membership witnesses), but the hash erases every algebraic relationship among them. Morphic accumulation resolves this tension. Each accumulated element is individually extractable (through standard membership witnesses), and the group-algebraic relationships of these group elements remain intact and provable. The accumulator thus serves simultaneously as a compact binding commitment to the set *and* as a substrate for homomorphic computation over the accumulated data.

**Homomorphic group operations over accumulated elements.** A key capability enabled by morphic accumulation is proving group operations over accumulated elements with *constant* proof size. Given group elements accumulated into a GUO-based accumulator, we can prove that a target element equals an arbitrary subset product of accumulated elements using the **PoKEModN** protocol [91], with proofs that are constant-sized independent of the subset cardinality. Informally, **PoKEModN** proves knowledge of an integer exponent $x$ committed as $C = g^x$ such that $x$ reduces to a prescribed value modulo a public modulus. A subtlety arises when proving such relations: the **PoKEModN** protocol establishes congruence modulo $M$, meaning the prover demonstrates knowledge of an integer *prod* satisfying $prod \equiv z \pmod{M}$, i.e., $prod = z + kM$ for some integer $k$. Standard DI ensures $z$ itself is unlikely to divide the accumulated product, but does not preclude $z + kM$ from doing so for $k \neq 0$. We introduce *group element division intractability* (GEDI), a strengthened notion that captures this, and prove that discrete logarithm encodings satisfy GEDI under a variant of the modular subset product assumption. We further justify this assumption by showing that even after simplifying the adversary's task, the problem reduces to modular subset-sum over an $L_{\mathsf{DI}}$-bit modulus. Moreover, by accumulating randomizers under a second generator $h$ with $\log_g h$ unknown, we can prove that a target equals the product of accumulated elements *randomized* by these randomizers. This *accumulate-and-randomize* technique yields asymptotically optimal constructions for range proofs and ring signatures, and near-optimal polynomial commitments, which have long been open.

### 1.2 Applications of Morphic Accumulators

**1) Optimal range proofs.** Range proofs are essential building blocks for confidential transactions [16, 17, 67], polynomial commitments [18, 19, 77], private voting [31, 41], and financial auditing [47, 64, 74]. Existing constructions primarily follow two approaches: binary decomposition and square decomposition.

Binary decomposition represents $x \in [0, 2^n - 1]$ as $x = \sum_{i=0}^{n-1} x_i \cdot 2^i$ with $x_i \in \{0, 1\}$, then proves each coefficient is binary and the weighted sum equals the committed value. Square decomposition leverages Lagrange's four-square theorem: to prove $x \in [0, 2^n - 1]$, one shows that both $x$ and $2^n - 1 - x$ can be expressed as sums of four squares, which implies non-negativity.

[4] From the Greek *morphē* (*μορφή*), meaning *form* or *shape*. We use *morphic* to indicate that the encoding preserves the algebraic form (group structure) of the accumulated elements, in contrast to "classic" DI hashes, which destroy it.

| Scheme | Prover | Verification | Proof Size | Aggregation |
|---|---|---|---|---|
| Bulletproof-style [17, 26, 86] | $O(n)$ | $O(n)$ | $O(\log n)$ | ✓ |
| Flashproofs [85] | $O(n^{4/3})$ | $O(n^{2/3})$ | $O(n^{2/3})$ | ✓ |
| Boudot [13] and Groth [41] | $O(n^3)$ | $O(1)$ | $O(1)$ | × |
| DewTwo [19] | $O(n \log^4 n)$ | $O(\log n)$ | $O(\log \log n)$ | × |
| Ours | $O(n)$ | $O(1)$ | $O(1)$ | ✓ |

**Table 1.** Asymptotic comparisons of various range proofs for range $[0, 2^n - 1]$ with transparent setups. Prover costs are asymptotic under the dominant operation model used by each scheme. Bulletproof-style schemes [17, 26, 85, 86] count fixed-security group scalar multiplications/MSMs; square-decomposition-based schemes [13, 19, 41] count the integer-arithmetic cost of finding the integer decomposition; our scheme counts hidden-order group exponentiations, e.g., modular exponentiations in RSA groups, by $O(L_{\mathsf{DI}})$-bit exponents.

| Scheme | Prover | Verification | Proof Size | SRS |
|---|---|---|---|---|
| Bulletproof [17] | $O(mn)$ | $O(mn)$ | $O(\log m + \log n)$ | None |
| MissileProof [38] | $O(mn \log mn)$ | $O(1)$ | $O(1)$ | $O(mn)$ |
| DekartProof [10] | $O(mn)$ | $O(\log m + \log n)$ | $O(\log m + \log n)$ | $O(m)$ |
| Ours | $O(mn)$ | $O(1)$ | $O(1)$ | None |

**Table 2.** Asymptotic comparisons of various vector range proofs for proving $m$ values are in range $[0, 2^n - 1]$. The table reports each construction's native asymptotic cost model rather than a uniform concrete operation count. Bulletproofs [17] count group scalar multiplications/MSMs; MissileProof [38] and DekartProof [10] count their native pairing-friendly group/field operations; our prover cost counts hidden-order group exponentiations by $O(L_{\mathsf{DI}})$-bit exponents. SRS denotes structured reference string size.

As shown in Table 1, existing range proofs with transparent setups exhibit various trade-offs among prover complexity, proof size, and verification time. Bulletproof-style schemes [17, 26, 86] achieve linear prover time with logarithmic proof size, but verification remains linear in the range size. Square decomposition methods [13, 19] can achieve constant or near-constant proof size and verification, but computing the decomposition requires super-linear time in the exponent bit-length. A natural question arises: *can we construct a range proof achieving $O(n)$ prover time, $O(1)$ proof size, and $O(1)$ verification simultaneously?*

We answer this affirmatively using morphic accumulation. Our key observation is that binary decomposition naturally aligns with subset membership: proving $x \in [0, 2^n - 1]$ reduces to proving that $g^x$ can be expressed as a product of elements from the set $\{g^{2^0}, g^{2^1}, \ldots, g^{2^{n-1}}\}$. In a transparent one-time setup, we accumulate these binary-power group elements into a GUO-based accumulator. To prove $x \in [0, 2^n - 1]$ for a commitment $C = g^x$, the prover identifies the subset corresponding to $x$'s binary representation and generates a batched membership proof showing their integer product reduces to $C$ modulo the group modulus. The batched PoKE techniques yield constant proof size and verification, while the prover performs only $O(n)$ hidden-order group exponentiations by $O(L_{\mathsf{DI}})$-bit exponents.[5] For range proofs, this means that the prover performs a constant amount of hidden-order group work per range bit, while proof size and verification remain independent of $n$. Our scheme therefore achieves $O(n)$ prover, $O(1)$ proof size, and $O(1)$ verification, while additionally supporting proof aggregation.

[5] The prover streams the subset factors and quotient witnesses, rather than materializing the $O(nL_{\mathsf{DI}})$-bit subset product as an integer; see Section 3.5.

The same approach also extends to vector range proofs. By encoding the binary components of all coordinates in a shared accumulator, we obtain a constant-size proof that every coordinate of a committed vector lies in its prescribed range. This yields a prover time of $O(mn)$ with $O(1)$ proof size and $O(1)$ verification time, so the vector case preserves the same optimal asymptotic profile as the scalar case. We compare our result with other vector range proofs in Table 2.

We note that there are other active lines of range proof research with different goals, e.g., minimizing concrete proof size for cryptocurrency applications [17,26,86]. Our focus is different: we target range proofs over large integers committed in groups of unknown order (e.g., 2048-bit RSA groups), where asymptotic complexity is the dominant concern. This setting arises naturally in GUO-based polynomial commitment schemes, where range proofs over large integers are the primary bottleneck, as we discuss next. We refer to [25] for a comprehensive discussion on other range proofs.

**2) Polynomial commitments.** A line of works [3, 18, 19, 37, 77] constructed polynomial commitment schemes (PCS) based on GUO. The core technique, introduced by Bünz et al. [18], encodes a polynomial $f(X) = \sum_{i=0}^{n-1} f_i X^i$ as a single integer $\hat{f}(q) = \sum_{i=0}^{n-1} f_i q^i$ for a base $q$, then commits to this integer in a GUO. This approach yields transparent PCS without trusted setup, but faces a fundamental challenge: *proving the degree bounds of the committed polynomials.*

In trusted-setup schemes like KZG [49], the structured reference string contains only powers $[g, g^s, g^{s^2}, \ldots, g^{s^n}]$ for a secret $s$, so an efficient prover cannot compute commitments to polynomials of degree larger than $n$. In contrast, the GUO-based approach uses a public base $q$; the prover can compute $q^k$ for any $k$, allowing them to encode polynomials of arbitrary degree. The standard solution is to use *range proofs* to certify that the committed encoding $\hat{f}(q)$ lies in $[0, q^n)$, which implies degree at most $n-1$. However, as shown in Table 3, computing these range proofs has been the primary bottleneck: Dew [3], Behemoth [77], and Dew++ [37] all require $O(n^3)$ prover time due to square decomposition-based range proofs.[6] Our optimal range proof directly addresses this bottleneck. Since our range proof achieves $O(n)$ prover time with constant proof size and verification, substituting it into the GUO-based PCS framework yields a PCS with $O(n)$ prover time, $O(1)$ proof size, and $O(1)$ group operation and $O(\log n)$ field operations during verification, which is close to optimal. The $O(\log n)$ operations apply to all works in this line of research [3, 37, 77], and are very fast in practice.

Another distinctive feature of GUO-based PCS is *evaluation-field flexibility* [19]: the polynomial is committed as an integer encoding, allowing evaluation over any publicly specified integer modulus. This contrasts with elliptic curve-based schemes [17, 49, 54], where the field is fixed by the curve. Evaluation-field flexibility directly enables *recursive-native SNARKs*: by aligning the PCS evaluation field with its underlying group modulus, PCS verification becomes native to the scheme itself, enabling recursive composition without field mismatch overhead. While this property applies to most GUO-based PCS, prior constructions [3,19,37,77] suffer from either cubic prover time or non-constant verification, making recursion impractical. Our construction is the first to combine evaluation-field flexibility with $O(n)$ prover time and $O(1)$ verification, enabling efficient recursive-native SNARKs in practice. Concretely, both the PCS and our range proofs require groups of unknown order; by instantiating both over hidden-order RSA groups of modulus $N$ and setting the evaluation field to $\mathbb{Z}_N$, the PCS can verify itself natively, yielding recursive-native verification. This requires a universal RSA setup (weaker than KZG [49]'s per-circuit trusted setup). Moreover, in the common paradigm

[6] The original DARK construction [18] attempted to avoid range proofs for degree bounds, but this approach was shown to have security vulnerabilities [8]. Subsequent works (Dew, Behemoth, Dew++) restored security by incorporating range proofs, at the cost of cubic prover complexity. DewTwo [19] takes a different approach, redesigning multiple components of the framework (PoKEMath, SIPA, $\log n$-square decomposition) to achieve $O(n \log^4 n)$ prover time with $O(\log \log n)$ proof size.

| Scheme | PP | Prover | Verification† | Proof Size | Transparent | PCS Field* |
|---|---|---|---|---|---|---|
| KZG [49] | $O(n)$ | $O(n)$ | $O(1)$ P | $O(1)$ | × | Prime |
| Bulletproofs [17] | $O(n)$ | $O(n)$ | $O(n)$ | $O(\log n)$ | ✓ | Prime |
| Dory [54] | $O(\sqrt{n})$ | $O(n)$ | $O(\log n)$ | $O(\log n)$ | ✓ | Prime |
| Dew [3] | $O(1)$ | $O(n^3)$ | $O(1)$ | $O(1)$ | ✓ | Flexible |
| Behemoth [77] | $O(n)$ | $O(n^3)$ | $O(1)$ | $O(1)$ | ✓ | Flexible |
| Dew++ [37] | $O(1)$ | $O(n^3)$ | $O(1)$ | $O(1)$ | ✓ | Flexible |
| DewTwo [19] | $O(n)$ | $O(n \log^4 n)$ | $O(\log n)$ | $O(\log \log n)$ | ✓ | Flexible |
| Ours (Behemoth) | $O(n)$ | $O(n)$ | $O(1)$ | $O(1)$ | × | Flexible |
| Ours (Dew) | $O(1)$ | $O(n)$ | $O(1)$ | $O(1)$ | × | Flexible |

**Table 3.** Asymptotic comparisons of the evaluation costs for various univariate polynomial commitment schemes of degree $n$. *A flexible PCS field enables recursive-native SNARKs. †Verification complexity is measured in group operations. KZG verification requires $O(1)$ pairings (P). All GUO-based schemes additionally incur $O(\log n)$ multiplications over $\lambda$-bit field operations for the verifier [3, 19]. "Ours" rows combine our range proof with the indicated evaluation protocol.

of SNARK composition where a prover-efficient inner SNARK is composed with a verifier-efficient outer SNARK [57], our construction serves as an ideal outer/buffer component: we can verify inner SNARKs over any field; verification is constant-time and requires no pairings.

**3) Optimal (linkable) ring signatures.** Ring signatures allow a signer to sign on behalf of a group without revealing identity, with applications in privacy coins [67, 75, 93] and anonymous voting. Linkable ring signatures additionally enable detecting whether two signatures were produced by the same signer, which is essential for preventing double-spending or double-voting.

As shown in Table 4, existing schemes exhibit trade-offs among efficiency, functionality, and security, with no prior scheme achieving the best of all worlds. Our morphic accumulation directly addresses this gap. In the transparent setup, we accumulate all public keys from a Schnorr group (a prime-order subgroup of $\mathbb{Z}_p^*$) together with a set of randomizers. The key insight is that proving "my public key is in the ring" reduces to proving subset membership in the accumulator, which we achieve with constant proof size. To hide which key is being used, the signer proves that a *cloaked key* (the public key multiplied by a random subset of accumulated randomizers) can be expressed as a product of accumulated elements. For linkability, we make the cloaking deterministic: the signer computes the linkable tag $g^x h^x$ by selecting randomizers corresponding to the binary decomposition of a fresh integer congruent to the private key $x$ modulo the Schnorr-group order, yielding a unique tag per signer that reveals nothing about the signer's identity. The final signature is a standard Schnorr signature [76] under the cloaked public key. This naturally separates into offline and online phases: the signer prepares a fresh offline proof packet for each signature before its message is known, then completes the signature in constant time online. The offline signing complexity $O(n)$ is optimal since any ring signature scheme must process all $n$ public keys in the ring; the online signing, signature size, and online verification are all $O(1)$. Moreover, our scheme is the first to simultaneously achieve constant-size signatures, linkability, and transparent setup (Table 4). The online signature is only 544 bytes for 112 bits of security.

### 1.3 Additional Related Work

**Relation to structure-preserving cryptography.** A cryptographic scheme is called *structure-preserving* [1, 51] if its inputs and outputs are group elements of the same group whose relations can be verified via group operations, enabling modular composition with compatible proof systems.

| Scheme | PP Size | Sign | | Size | Verify | | Transparent Setup | Linkable |
|---|---|---|---|---|---|---|---|---|
| | | Offline | Online | | Offline | Online | | |
| Nguyen [65] | $O(n)$ | $O(n)$ | $O(1)$ | $O(1)$ | $O(n)$ | $O(1)$ | × | × |
| DKNS [35] | $O(1)$ | $O(n)$ | $O(1)$ | $O(1)$ | $O(n)$ | $O(1)$ | × | × |
| BDR [12] | $O(n)$ | $O(n)$ | $O(1)$ | $O(1)$ | $O(n)$ | $O(1)$ | × | × |
| RingCT 3.0 [93] | $O(n)$ | $O(n)$ | $O(n)$ | $O(\log n)$ | $O(n)$ | $O(n)$ | ✓ | ✓ |
| DualDory [11] | $O(n)$ | - | $O(n)$ | $O(\log n)$ | $O(n)$ | $O(\log n)$ | ✓ | × |
| KMS [51] | $O(n)$ | $O(n)$ | $O(1)$ | $O(1)$ | $O(1)$ | $O(1)$ | × | ✓ |
| LLRing [45] | $O(n)$ | - | $O(n)$ | $O(\log n)$ | $O(n)$ | $O(\log n)$ | ✓ | × |
| XTAF+ [89] | $O(n)$ | - | $O(n)$ | $O(1)$ | - | $O(n)$ | × | ✓ |
| Ours | $O(1)$ | $O(n)$ | $O(1)$ | $O(1)$ | $O(n)$ | $O(1)$ | ✓ | ✓ |

**Table 4.** Asymptotic comparisons of various ring signatures. Offline costs denote work that can be performed once the ring is fixed, while online costs denote message-dependent work.

Existing work focuses on prime-order bilinear groups, where Abe et al. [1] proved that strictly structure-preserving commitments cannot be shrinking in a prime-order bilinear group: a commitment to $k$ source-group elements must itself comprise at least $k$ source-group elements. Our morphic accumulation shares the same high-level goal: accumulating group elements while preserving their algebraic structure, so that relations among accumulated elements can be proved efficiently. In this sense, morphic accumulation can be viewed as a structure-preserving accumulator for the hidden-order group setting. When the value group coincides with the accumulator group (e.g., accumulating $\mathbb{G}_N$ elements into an RSA-based $\mathbb{G}_?$), the inputs and outputs are elements of the same group, directly paralleling the structure-preserving property in the bilinear setting.

The key difference is that the hidden group order gives our construction a flexibility that the prime-order setting does not offer. In prime-order bilinear groups, the order $p$ is public, and all exponent operations are field operations in $\mathbb{Z}_p$; the algebraic setting is determined by this single field. In our GUO setting, the group order is unknown, so elements are represented as integers, not reduced modulo any known quantity. The accumulator interacts with these representations only through the division intractability property, which depends on their integer structure (large prime factors), not on the group they originated from. This makes morphic accumulation agnostic to the source group: it can accumulate elements from prime-order Schnorr groups, hidden-order RSA groups, or a mix of both, without requiring bilinear pairings or a common group order.

**Other accumulator-based ring signatures.** Several ring signature schemes are built from accumulators [12, 35, 51, 65]. The construction closest to ours is DKNS [35], which builds ring signatures from an RSA accumulator with hidden order, also achieving asymptotically optimal signing and verification overhead with $O(1)$ signature size. The key difference lies in what is accumulated: DKNS accumulates specially structured large primes, where each public key is a prime $e$ whose factorization $(e-1)/2 = e_1 \cdot e_2$ serves as the secret key, whereas we accumulate group elements $g^x$ directly. Because DKNS public keys are raw primes with no group-operation relationship to one another, their accumulator serves purely as a set-membership primitive, and the prover must rely on a dedicated RSA-based Sigma-protocol to demonstrate knowledge of the prime's factorization. This design choice has two consequences. First, the RSA modulus requires a trusted setup (safe-prime generation), ruling out transparency. Second, the lack of algebraic structure among accumulated keys precludes linkability, since there is no natural way to derive a deterministic, signer-specific tag from the accumulated data. Our morphic accumulation addresses both limitations. Because accumulated elements retain their group structure, the signer can compute a linkable tag as a provable product of

accumulated elements, and the online signature reduces to a standard Schnorr proof under $T$ (544 bytes for 112-bit security). Class groups and hyperelliptic curves are candidates for a transparent setup, subject to the stated requirements on the group and public bases.

## 2 Technical Overview

**Background: GUO-based accumulators.** The standard approach to accumulating a set $S = \{s_1, \ldots, s_n\}$ of group elements (e.g., public keys) into GUO-based accumulators follows a two-step process:

1. Encode each element via division intractable (DI) hash functions (including hash-to-prime): $t_i = \mathbf{Hash_{DI}}(s_i)$;
2. Accumulate the set of hash outputs $T = \{t_1, \ldots, t_n\}$ into the accumulator.

This paradigm, established by [7, 20, 27, 39, 56], has been widely adopted [20, 21, 61, 69, 90]. Division intractability requires that no hash output divides the product of other hash outputs. The simplest construction of DI hash uses a collision-resistant hash-to-prime function: since outputs are distinct primes, no output can divide a product of other primes. A more general approach, first observed by Gennaro, Halevi, and Rabin [39] and applied to accumulators by Lipmaa [56], uses random oracles with sufficiently large outputs. The insight is that a uniformly random integer from a large domain (e.g., $[0, 2^{2048}]$) contains a large random prime factor (e.g., around 256 bits) with overwhelming probability. This large prime factor does not divide the product of other random integers except with negligible probability (e.g., $1/2^{256}$). While these DI hash functions achieve the security requirement, they *destroy the algebraic structure of accumulated elements*: once an element is hashed, its original structure is lost, preventing efficient proofs of relations between accumulated elements. Our goal is to eliminate this preprocessing step, thereby enabling morphic accumulation.

**Morphic accumulation based on discrete logarithm hash.** We observe that *discrete logarithm encodings are inherently division intractable*. Consider the hash function $H_g(x) = g^x \bmod M$, where $g$ is a random generator and the output is the canonical integer representation in $[0, M)$. For ease of presentation, we write $g^x$ to denote both the hash function output $H_g(x)$ and its canonical integer representation, and distinguish integer multiplication ($\times$) from group operation ($\cdot$); see Section 3 for formal definitions. In the generic group model, $H_g(x)$ is uniformly random in the encoding space. When the encoding bitlength $L_{\mathsf{DI}}$ is sufficiently large, it contains a large prime factor with overwhelming probability, making it division intractable by the same reasoning that underlies random oracle-based DI hashes; see Section 3.2.

The key difference from traditional DI hashes is that this hash function *preserves algebraic structure*: the outputs are group elements, and the group operation satisfies $H_g(x_1) \cdot H_g(x_2) = g^{x_1} \cdot g^{x_2} = g^{x_1+x_2} = H_g(x_1+x_2)$. This enables *morphic accumulation*: group elements can be directly accumulated into GUO-based accumulators without preprocessing, satisfying the DI requirement for security while retaining the algebraic relationships needed for arithmetic proofs. This technique applies to any group whose operation is modular multiplication, including RSA groups and prime-order groups.

**Homomorphic group operations over accumulated elements.** Morphic accumulation enables proving group operations over accumulated elements with constant proof size. Consider group elements $s_1 = g^{x_1}, s_2 = g^{x_2}, \ldots, s_k = g^{x_k}$ accumulated into a GUO-based accumulator. To prove that a target group element $z$ equals the group product $s_1 \cdot s_2 \cdots s_k = g^{x_1+x_2+\cdots+x_k}$, the prover: (1) computes the integer product $prod = s_1 \times s_2 \times \cdots \times s_k$ of the canonical representations; (2) generates a batched membership proof showing these elements form a subset of the accumulator; and (3) proves

$prod \equiv z \pmod M$ using the **PoKEModN** protocol [91]. The integer product reduced modulo $M$ equals the group product, so step (3) establishes the desired relation. A subtlety arises in proving soundness: step (3) binds the prover to an integer $prod = z + kM$ for some $k$, not to $z$ itself. Standard DI ensures $z$ cannot divide the accumulated product $prod = \prod_{s_i \in S} s_i$, but does not preclude $z + kM$ from doing so for $k \neq 0$. This is a real threat: $prod$ can have divisors that are not subset products of the $s_i$. For instance, if $s_1 = 6$ and $s_2 = 35$, then $15 \mid 210 = s_1 \times s_2$, yet $15 \notin \{1, 6, 35, 210\}$; the value $15 = 3 \times 5$ "splits" prime factors across elements. An adversary finding $k$ with $z + kM = 15$ could falsely claim a subset relation. (Element sizes are small for illustration; for 128-bit GEDI security, encodings use $L_{\mathsf{DI}} = 1704$ bits.)

We address this by proving a stronger property called *group element division intractability* (GEDI): any divisor of $prod$ of the form $z + kM$ must be a genuine subset product. For $k = 0$, GEDI reduces to standard DI. For $k \geq 1$, we give two complementary arguments: an information-theoretic bound showing that $prod$ has too few divisors for any to match the random residue $z \bmod M$ when $n$ is moderate, and a computational reduction to the modular subset-product problem (a variant of well-studied modular subset-sum) for larger $n$.

*Concrete example.* Consider an RSA group $\mathbb{G}_N$ with generator $g$, modulus $N$ and hash function $H_g(x) = g^x \bmod N$. Let $s_1 = H_g(x_1)$ and $s_2 = H_g(x_2)$ be two hash outputs (group elements in canonical integer representation). When we accumulate $\{s_1, s_2\}$ using a GUO-based accumulator with base $g_{acc}$, we compute $Acc = g_{acc}^{e}$, where $e = s_1 \times s_2$ is the integer product of the hash outputs. We can prove group operations over accumulated hash outputs. For instance, to prove that $z = H_g(x_1 + x_2)$ equals the group product $s_1 \cdot s_2$, a prover demonstrates: (1) $g_{acc}^{z} = g_{acc}^{e \bmod N}$ using the **PoKEModN** protocol from [91], which shows $z \equiv e \pmod N$, and (2) $z \in \mathbb{G}_N$ by proving knowledge of the preimage $x_1 + x_2$. This approach extends to any number of accumulated elements, enabling efficient proofs of arbitrary group operations over hash outputs.

**Optimal range proofs from morphic accumulation.** Consider the example of an integer commitment [30] $C = g^x$ with base $g$ and hidden-order RSA modulus $N$, where we want to prove $x \in [0, 2^n - 1]$. Our approach accumulates binary powers and proves the committed value is their subset product:

Setup. Run a one-time transparent setup accumulating binary powers $\{g^1, g^2, g^4, \ldots, g^{2^{n-1}}\}$ into accumulator $Acc$.

Proof. To prove $x \in [0, 2^n - 1]$, the prover:

1. Decomposes $x$ into binaries: $x = \sum_{i \in I} 2^i$ for some index set $I \subseteq \{0, \ldots, n-1\}$;
2. Computes the subset product $prod = \prod_{i \in I} g^{2^i}$ (product of canonical representations of $g^{2^i}$);
3. Computes subset commitment $C_{sub} = g_{acc}^{prod}$ with batched membership proofs showing it corresponds to a subset of $Acc$;
4. Proves $prod \equiv C \pmod N$ using **PoKEModN**, establishing that the committed value equals the binary sum.
5. Proves knowledge of the exponent of $C$ with respect to $g$ using the **PoKE** protocol, proving that $C$ is indeed a group element.

The verifier checks the batched membership proof and the **PoKEModN**. The original batched membership proof [9] requires checking each element is indeed a DI hash output. In our protocol, it is achieved by verifying a **PoKE** protocol. The batched membership protocol, the **PoKEModN** protocol, and the **PoKE** protocol all take $O(n)$ proving time with $O(1)$ proof size and verification. This yields an optimal range proof: proof size is constant (independent of range $n$), verification is constant-time, and the prover complexity is $O(n)$, matching the information-theoretic lower bound for proving membership in a set of size $2^n$.

This protocol can be further extended to achieve zero-knowledge for Pedersen-style commitments $C = g^x h^r$, where $\log_g h$ is unknown. Intuitively, the accumulated $h$-powers cloak the subset of $g$-

powers: the prover proves membership of a mixed subset whose product encodes $g^x h^{r'}$ rather than $g^x$ alone. To achieve adaptive zero-knowledge, the prover forms $C' = g^x h^{r'}$ and proves that $C/C'$ is a power of $h$, linking the fresh commitment to the original statement.

Setup (ZK). Accumulate binary powers of both generators: $\{g^{2^0}, \ldots, g^{2^{n-1}}, h^{2^0}, \ldots, h^{2^{m-1}}\}$, where $m \geq \lceil \log_2 \text{maxord}(\mathbb{G}_{val}) \rceil + 2\lambda$.

Proof (ZK). To prove $x \in [0, 2^n - 1]$ for $C = g^x h^r$:

1. Sample $r' \xleftarrow{\$} [0, 2^m - 1]$, compute $C' = g^x h^{r'}$, and decompose $x = \sum_{i \in I} 2^i$ and $r' = \sum_{j \in J} 2^j$.
2. Compute the integer product $prod = \prod_{i \in I} g^{2^i} \times \prod_{j \in J} h^{2^j}$ and the subset commitment $C_{sub} = g_{acc}^{prod}$, together with its subset-membership proof.
3. Prove knowledge of the opening $(x, r')$ of $C'$ and of the exponent $prod$ satisfying $prod \equiv C' \pmod{M}$.
4. Prove knowledge of $\delta = r - r'$ such that $C/C' = h^\delta$.

Zero-knowledge. The protocol achieves zero-knowledge because the randomizer $r'$ serves a dual purpose: it randomizes the commitment $C' = g^x h^{r'}$, and it simultaneously randomizes the subset selection in the membership proof. A simulator can pick an arbitrary subset of $g$-powers and a random subset of $h$-powers. Thus, the simulated proof is indistinguishable from an honest prover's proof, achieving computational zero-knowledge with $2^m$ randomizer choices following the technique from [21].

**Polynomial commitments from optimal range proofs.** A polynomial commitment scheme (PCS) allows a prover to commit to a polynomial and later prove that it evaluates to a claimed value at any queried point. PCS is the central cryptographic tool for compiling polynomial IOPs [18] into succinct arguments; the resulting SNARK inherits its properties from the PCS. A line of works [3, 18, 19, 37, 77] constructs transparent PCS from groups of unknown order using two distinct approaches. Dew [3] and Dew++ [37] use a *convolution-based* inner product argument: they encode a polynomial as a large integer and extract the evaluation from a carefully designed integer multiplication. Behemoth [77] takes a different approach, directly instantiating the *KZG quotient-polynomial* strategy in a GUO without pairings. Despite these different designs, Dew, Dew++, and Behemoth share the same fundamental bottleneck: *range proofs over large integers committed in GUOs*, implemented via square decomposition (Lagrange's four-square theorem), incurring $O(n^3)$ prover time. DewTwo [19] redesigns multiple components of the Dew framework (introducing PoKE-Math, SIPA, and efficient $\log n$-square decomposition), achieving $O(n \log^4 n)$ prover time but with $O(\log \log n)$ proof size rather than $O(1)$.

Our optimal range proof directly resolves the bottleneck in Dew, Dew++, and Behemoth. Since our range proof achieves $O(n)$ prover time (in group operations) with $O(1)$ proof size and $O(1)$ group operations for verification, substituting it into the Dew/Dew++ or Behemoth evaluation protocol yields a PCS with $O(n)$ prover, $O(1)$ proof size, and verification in $O(1)$ group operations plus $O(\log n)$ field operations (multiplications over $\lambda$-bit integers for PoE verification, as in [3, 37]). The range proof is a drop-in replacement, and we can directly achieve the results in Table 3. Because the range proof must prove bounds on integers committed in the GUO itself, and morphic accumulation requires accumulating integer group elements, the GUO must be an RSA group in this case. This requires a universal trusted setup for modulus generation, forgoing the transparency of prior GUO-based PCS that use class groups [3, 19, 37, 77].

*Field flexibility.* Unlike elliptic curve-based PCS where the field is fixed by the curve order, GUO-based PCS commits to integer polynomials, allowing evaluation over any publicly specified integer modulus [19]. By aligning the evaluation field with the underlying group modulus, PCS verification becomes native to the scheme itself, enabling recursive-native SNARKs.

**Optimal linkable ring signatures.** In a linkable ring signature scheme, $n$ signers jointly commit their keys, and each signer must:

– Prove a "linkable tag" can be computed from one of the $n$ committed keys;

– Prove knowledge of the private key corresponding to this tag;

– Accomplish the above without revealing which public key is used.

Linkability allows anyone to determine whether multiple signatures were produced by the same signer without breaking ring anonymity, which is essential for preventing double-spending or double-voting.

*Construction overview.* Consider a Schnorr group [76]: a prime-order subgroup of $\mathbb{Z}_p^*$ with public order $q$ and two generators $g, h$ whose discrete logarithm relation is unknown. Private keys are sampled as $x \xleftarrow{\$} \mathbb{Z}_q$, and public keys are computed as $pk = g^x$.

Setup. For a ring of public keys $\{pk_1, \ldots, pk_n\}$, accumulate the public keys and binary powers of $h$ into accumulator $Acc$: $\{pk_1, \ldots, pk_n, h^{2^0}, \ldots, h^{2^{w-1}}\}$, where $w = \lceil \log_2 q \rceil + 2\lambda$ accommodates the fresh integer lifts used in signing.

Signing. To sign a message $m$, the signer with private key $x$ and public key $g^x$:

1. Computes the linkable tag $T = g^x h^x = (g \cdot h)^x$;
2. Proves $T$ can be computed from accumulated elements: samples $a \xleftarrow{\$} [0, 2^{2\lambda})$, sets $r' = x + qa$, decomposes $r' = \sum_{i \in I} 2^i$, computes subset product $prod = g^x \times \prod_{i \in I} h^{2^i}$, generates a batched membership proof showing this subset is in $Acc$, and proves $prod \equiv T \pmod p$ using **PoKE-ModN**;
3. Generates a standard Schnorr signature on $m$ with base $(g \cdot h)$ and public key $T$, binding the challenge to $Acc$ and $T$.

Verification. Given $(Acc, m, \sigma)$ where $\sigma = (T, \pi_{mem}, \pi_{mod}, \pi_{schnorr})$:

1. Verify the batched membership proof $\pi_{mem}$ and modular product proof $\pi_{mod}$;
2. Verify the Schnorr signature $\pi_{schnorr}$ with base $(g \cdot h)$ and public key $T$.

*Security and efficiency.* The linkable tag $T = g^x h^x$ is unique per signer but reveals nothing about the signer's identity: given only $T$, an adversary cannot determine which $pk_i = g^x$ was used. More precisely, under the DDH assumption, the tag $g^x h^x$ is computationally indistinguishable from $g^{x'} h^r$ for random $x', r$: given $(g, h, g^x, h^x)$, distinguishing this from $(g, h, g^{x'}, h^r)$ is exactly the DDH problem. The Schnorr signature with base $(g \cdot h)$ simultaneously proves knowledge of $x$ and that $T$ is well-formed (i.e., the exponents w.r.t $g$ and $h$ are equal).

The scheme naturally separates into offline and online phases. The offline phase (steps 1 and 2) can be executed as soon as the ring is formed, before any message is known: each signature uses a fresh lift $r'$ and a fresh proof packet, while the linkable tag $T$ remains fixed. This phase has $O(n)$ complexity (dominated by streaming the subset product and membership witness) but produces constant-sized proofs. The online phase (step 3) executes when the message $m$ arrives: the signer generates a standard Schnorr signature, which requires only $O(1)$ computation. This offline/online split achieves asymptotic optimality, as any ring signature must read at least $O(n)$ public keys, and our scheme confines this cost entirely to the precomputation phase.

**Summary.** Morphic accumulation provides a unified framework for constructing optimal cryptographic protocols. By eliminating the preprocessing step and preserving algebraic structure, we achieve constructions with optimal asymptotic complexity and constant-sized proofs across range proofs, ring signatures, and beyond. The technical sections that follow provide formal definitions, security proofs, and detailed efficiency analysis.

## 3 Preliminaries

Let $\lambda$ denote a security parameter. By Primes($\lambda$) we denote the set of integer primes less than $2^\lambda$. A function negl($\lambda$): $\mathbb{N} \to \mathbb{R}^+$ is negligible if for every positive polynomial $poly(\lambda)$ there exists a $\lambda_0 \in \mathbb{N}$, such that for all $\lambda > \lambda_0 : \text{negl}(\lambda) < 1/poly(\lambda)$. We denote $PPT$ as "probabilistic polynomial time" with respect to $\lambda$ throughout our paper. We denote by $x \xleftarrow{\$} F$ sampling uniformly at random from the domain $F$. Let $[i, j] = \{i, i+1, \ldots, j\}$, and $[j] = \{0, 1, \ldots, j-1\}$. We denote a vector by $\boldsymbol{a}$ and denote the size of the vector $\boldsymbol{a}$ by $|\boldsymbol{a}|$.

**Representations.** In this paper, we primarily work with three types of groups. We use $\mathbb{G}_?$ to denote a group of unknown order (GUO), where the group order is unknown. Such groups can be instantiated from hidden-order RSA groups [7], class groups [15,42], and Jacobians of hyperelliptic curves [14,34]. The notation $\mathbb{G}_p$ refers to a prime-order subgroup of the multiplicative group of integers modulo $p$; the subscript $p$ denotes the modulus (not the group order). For a group element $S$ in $\mathbb{G}_p$, we write $S \in \mathbb{G}_p$ and represent $S$ canonically as an integer in $[p]$. We use $\mathbb{G}_N$ to denote a hidden-order RSA group, which is a subgroup of the multiplicative group of integers modulo $N$, where $N$ is an RSA modulus. Similarly, if $T$ is a group element in $\mathbb{G}_N$, we write $T \in \mathbb{G}_N$ and represent $T$ canonically as an integer in $[N]$. When we write an expression such as $g^{a_1} \times g^{a_m}$, where each $a_i \in [0, M)$, we mean that each $g^{a_i}$ is first computed in its canonical group representation, and the resulting group elements are then multiplied as integers. We use $\times$ for integer multiplication and $\cdot$ for the group operation. When $s_1, s_2$ are group elements with modulus $M$, $s_1 \times s_2$ denotes the product of their integer representations, while $s_1 \cdot s_2$ denotes the group product (i.e., $s_1 \cdot s_2 = (s_1 \times s_2) \bmod M$). Throughout this paper, we use $\mathbb{G}_{val}$ to denote a value group (either $\mathbb{G}_p$ or $\mathbb{G}_N$) with generator $g_{val}$, and we denote group elements as $s_i = g_{val}^{x_i}$ where $x_i$ is the corresponding exponent (discrete logarithm).

**Smooth numbers.** An integer $a$ is called $y$-smooth if all the prime factors of $a$ are less than or equal to $y$. Let $\Psi(x, y)$ denote the number of integers no larger than $x$ that are $y$-smooth. The Dickman–de Bruijn function $\rho(\cdot)$ [43] suggests that $\Psi(x, y) \sim x\rho(u)$ where $u = \log x / \log y$.

### 3.1 Generic Group Model

The Generic Group Model (GGM), originally proposed by Shoup [79] for groups of known order and extended by Damgård and Koprowski [32] to groups of unknown order, is a heuristic framework for analyzing generic adversaries interacting with a cryptographic group. In the GGM, adversaries only interact with encodings of group elements, without access to their actual representations. The correspondence between encodings and group elements is determined by a random injective map, which is unknown to the adversary.

**Definition 1 (Generic Group Model).** *Let $\mathbb{G}$ be a cyclic group of order $n$, where $n$ may be either publicly known or hidden from the adversary. The group $\mathbb{G}$ is defined by a random injective function $\sigma : \mathbb{Z}_n \to \mathbb{F}$ such that $|\mathbb{F}| > n$. We denote the group elements as $\sigma(x_i)$, $0 \le i \le n-1$. A generic group algorithm $\mathcal{A}$ is a probabilistic algorithm, and $\mathcal{L}$ is a list initialized with the encodings given as input to $\mathcal{A}$. $\mathcal{A}$ supports two generic group queries to the oracle:*

- *Given two indices $i$ and $j$ from $\mathcal{L}$ and a sign bit, $\mathcal{O}_1$ replies with $\sigma(x_i \pm x_j \bmod n)$ and appends it to $\mathcal{L}$ if this encoding is not already present.*
- *For a random element query, $\mathcal{O}_2$ samples $r \in \mathbb{Z}_n$ uniformly at random, returns $\sigma(r)$, and appends it to $\mathcal{L}$ if this encoding is not already present.*

*The random injective function $\sigma$ ensures that the encodings are uniformly distributed over $\mathbb{F}$, revealing no information about the group structure to $\mathcal{A}$.*

### 3.2 Division Intractable Hash

Division Intractable (DI) hash is a special kind of hash function where it is hard to find any set of distinct preimages that one hash result divides the product of the rest of the hash results. It can be formally defined as follows.

**Definition 2 (Division intractable).** *A function family $\mathcal{H}$ is called division intractable (DI) if given a security parameter $\lambda$, the probability of finding $H \in \mathcal{H}$ and distinct inputs $X_1, \ldots, X_m, Y$, $m < poly(\lambda)$ such that*

$$\Pr[H(Y) \text{ divides } H(X_1) \ldots H(X_m)] < \text{negl}(\lambda).$$

Clearly, all collision-resistant hash-to-prime hash functions are DI. However, most hash-to-prime functions are time-consuming. An alternative is to use hash functions modeled as a random oracle whose outputs are large integers.

We write $L_{\mathsf{DI}} = L_{\mathsf{DI}}(\lambda)$ for the bitlength of these integer encodings. This parameter is part of the concrete encoding choice and should not be confused with the security parameter $\lambda$ used in the security definitions. The classical asymptotic sufficient condition sets $L_{\mathsf{DI}}(\lambda) = \lambda^2$. In this setting, Lemma 6 of [39] shows that random-oracle outputs are division intractable. Informally, random numbers with large bit length tend to have large prime factors (i.e., non-smooth). A large prime factor makes it unlikely that one independently sampled output divides the product of other independently sampled outputs.

**Lemma 1.** *Let $L_{\mathsf{DI}}(\lambda) = \lambda^2$, and let $x_1, \ldots, x_m$ be uniformly random $L_{\mathsf{DI}}$-bit integers chosen independently, let $j$ be any integer $j \in \{1 \ldots m\}$ and denote $E = (\prod_{i=1}^{m} x_i)/x_j$. Then, $\Pr[x_j | E] < \text{negl}(\lambda)$.*

Recent work [50] gives a tight bound showing that a uniform random $\lambda^2$-bit integer is $2^{\lambda^{3/2}}$-smooth only with negligible probability:

**Lemma 2.** *Let $\lambda$ be a security parameter and $x \xleftarrow{\$} [0, 2^{\lambda^2})$. If we denote by SMOOTH the event where the integer $x$ is $2^{\lambda^{3/2}}-$smooth, then $\Pr[SMOOTH] \leq (\frac{\lambda^{1/2} \log \lambda^{1/2}}{e})^{-\lambda^{1/2}}$.*

For concrete instantiations, we choose $L_{\mathsf{DI}}$ according to the desired concrete security level, rather than by setting $L_{\mathsf{DI}} = \lambda^2$ and solving for $\lambda$. Using the smoothness estimates above and a birthday-bound analysis, [50] derives that 128-bit DI security requires $L_{\mathsf{DI}} \geq 1704$ bits, while 192-bit DI security requires $L_{\mathsf{DI}} \geq 2896$ bits. This result is close to the numerical estimation presented in [27].

### 3.3 GUO-based Accumulators

We follow the universal accumulator definitions provided by [9, 40].

**Definition 3.** *Given security parameter $\lambda$, a universal accumulator is a tuple of four PPT algorithms* **(KeyGen, Acc, Witness, Verify)***:*

- $(ek, vk) \leftarrow$ **KeyGen**$(1^\lambda)$. *This algorithm takes as input the security parameter and outputs a public evaluation key $ek$ that will be used to generate witnesses and a public verification key $vk$ that will be used to verify witnesses.*
- $Acc \leftarrow$ **Accu**$(ek, X)$. *This algorithm takes as input a set $X \subset \mathcal{X}_\lambda$ where $\mathcal{X}_\lambda$ is the input domain for the elements to be accumulated and the evaluation key $ek$ and outputs one accumulation of the set $Acc \in \mathcal{G}_\lambda$ where $\mathcal{G}_\lambda$ is the output domain for* **Acc**.

– $(b, w_x) \leftarrow \textbf{Witness}(X, x, Acc, ek)$. *This algorithm takes as input a set* $X \subset \mathcal{X}_\lambda$*, one element* $x \in \mathcal{X}_\lambda$*, the evaluation key* $ek$ *and the accumulation of the set* $Acc$*. It outputs a boolean value* $b$ *indicating whether the element is in the set and a witness* $w_x$ *for the answer.* $b = 1$ *indicates the element is in the set, and this witness is a membership witness. Otherwise,* $b = 0$ *and this witness is a non-membership witness.*
– $0/1 \leftarrow \textbf{Verify}(Acc, x, w_x, b, vk)$. *This algorithm takes as input the accumulation value* $Acc \in \mathcal{G}_\lambda$*, one element* $x \in \mathcal{X}_\lambda$*, the witness* $w_x$*, a bit* $b$ *and the public verification key* $vk$*. It outputs* 1 *if it accepts the proof and* 0 *otherwise.*

A universal accumulator is secure with the following two properties. The first property we require from a cryptographic accumulator is completeness. Completeness states that a witness output by any sequence of invocations of the scheme algorithms, for a valid statement, is verified correctly with all but negligible probability. It can be formulated as follows.

**Definition 4.** *Completeness. A universal accumulator scheme is complete if*

$$\Pr\left[\begin{array}{c}(ek, vk) \leftarrow \textbf{KeyGen}(1^\lambda) \\ Acc \leftarrow \textbf{Accu}(ek, X, t) \\ (b, w_x) \leftarrow \textbf{Witness}(X, x, Acc, ek) : \\ 1 \leftarrow \textbf{Verify}(Acc, x, w_x, b, vk)\end{array}\right] > 1 - \text{negl}(\lambda).$$

The second property is soundness, which captures the fact that adversaries cannot provide accepting witnesses for incorrect statements. More specifically, it is computationally infeasible to find both a valid membership witness and a valid non-membership witness for any element $x$ in the defined domain $\mathcal{X}_\lambda$. It can be formulated as below.

**Definition 5.** *Soundness (Undeniability). A universal accumulator scheme is sound if, for all PPT adversaries* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}1 \leftarrow \textbf{Verify}(C, x, w_1, 1, vk) \wedge \\ 1 \leftarrow \textbf{Verify}(C, x, w_2, 0, vk) \wedge \\ x \in \mathcal{X}_\lambda\end{array}\middle|\begin{array}{c}(ek, vk) \leftarrow \textbf{KeyGen}(1^\lambda) \\ (C, x, w_1, w_2) \xleftarrow{\$} \mathcal{A}\end{array}\right] < \text{negl}(\lambda).$$

### 3.4 Proof of Knowledge of Exponentiation

Given a group of unknown order $\mathbb{G}_?$, one integer $x$ and two group elements $g, y$, a proof of knowledge of exponentiation (PoKE) protocol [9] enables a prover to prove knowledge of $x$ such that $g^x = y \in \mathbb{G}_?$ holds, with a constant-sized proof. Formally, the relation for **PoKE** can be defined as: $\mathcal{R}^{\textbf{PoKE}} = \{(g, y \in \mathbb{G}_?; x \in \mathbb{Z}) : y = g^x\}$. The protocol can be made non-interactive with the Fiat-Shamir heuristic [9, 36]. We denote the non-interactive version of **PoKE** as follows:
– $\pi \leftarrow \textbf{NI-PoKE}(g, y; x)$: algorithm to generate a proof;
– $1/0 \leftarrow \textbf{NI-PoKE.Verify}(g, y, \pi)$: algorithm to verify a proof.
We also use the notation $\textbf{NI-PoKExp}_{\mathbb{G}}(g, y; x)$ for the generic task of proving knowledge of an exponent $x$ such that $y = g^x$ in a specified group $\mathbb{G}$. When $\mathbb{G}$ is a group of unknown order, this interface is instantiated by **NI-PoKE**. When $\mathbb{G}$ is a prime-order subgroup of $\mathbb{Z}_p^*$, it is instantiated by a standard Schnorr protocol [76].

A similar protocol [9] can prove the knowledge of the preimage of a group representation with respect to public generators.[7] Formally, it can be defined as $\mathcal{R}^{\mathbf{PoKRep}} = \{(\boldsymbol{g}, y \in \mathbb{G}_?; \boldsymbol{x} \in \mathbb{Z}) : y = \prod g_i^{x_i}\}$. We denote the non-interactive version of **PoKRep** as follows:
– $\pi \leftarrow$ **NI-PoKRep**$(\boldsymbol{g}, y; \boldsymbol{x})$: algorithm to generate a proof;
– $1/0 \leftarrow$ **NI-PoKRep.Verify**$(\boldsymbol{g}, y, \pi)$: algorithm to verify a proof.

Xin and Papadopoulos [91] further extended $\mathcal{R}^{\mathbf{PoKE}}$ to support exponentiation modulo an integer $n$, resulting in the relation $\mathcal{R}_{\mathbf{PoKEModN}}$:

$$\mathcal{R}_{\mathbf{PoKEModN}} = \{(C \in \mathbb{G}_?, n \in \mathbb{Z}, \hat{x} \in [n]; x \in \mathbb{Z}) : C = g^x,\ x \bmod n = \hat{x}\}.$$

They provided an argument of knowledge protocol for the relation $\mathcal{R}_{\mathbf{PoKEModN}}$, denoted as **PoKE-ModN**. We denote the non-interactive version as:
– $\pi \leftarrow$ **NI-PoKEModN**$(C, n, \hat{x}; x)$: algorithm to generate a proof;
– $1/0 \leftarrow$ **NI-PoKEModN.Verify**$(C, n, \hat{x}, \pi)$: algorithm to verify a proof.

### 3.5 Streaming PoKE Witnesses for Product Exponents

In our applications, the witness exponent in a **PoKE** or **PoKEModN** instance is often a product $x = \prod_{s_i \in S} s_i$ over a set $S = \{s_1, \ldots, s_n\}$ of canonical integer encodings. Although such integer products are usually fast in practice, materializing $x$ is not a linear-time operation in a formal bit-complexity analysis: the product has $O(nL_{\mathsf{DI}})$ bits, and product-tree multiplication incurs logarithmic overhead, e.g., $O\left(nL_{\mathsf{DI}} \log^2(nL_{\mathsf{DI}}) \log\log(nL_{\mathsf{DI}})\right)$ bit operations using standard fast-multiplications. But we can avoid this computation entirely. The prover treats $x$ only as a mathematical witness and streams the factors $s_1, \ldots, s_n$, maintaining only group elements and residues whose bitlength is independent of $n$. Throughout our accumulator-based prover analyses, our prover complexity counts group exponentiations by $O(L_{\mathsf{DI}})$-bit exponents; the accompanying integer divisions and reductions are over fixed-size operands and do not grow with the number of accumulated elements. For notational simplicity, later protocols may still write $x = \prod_{s_i \in S} s_i$ as a mathematical witness exponent; operationally, the prover streams the factors as described here and never materializes this full integer product.

First, when the statement element $D = g^x$ must be constructed, it can be computed by the sequential updates $D_0 = g, D_i = D_{i-1}^{s_i} (i = 1, \ldots, n)$. By induction, $D_i = g^{\prod_{t=1}^{i} s_t}$, and hence $D_n = g^x$, after $n$ group exponentiations and without ever forming $x$ as an integer.

It remains to generate the **PoKE** quotient after the challenge $\ell$ is fixed. The prover must output $(Q = g^q, r)$ such that $x = q\ell + r, 0 \le r < \ell$. Let $X_i = \prod_{t=1}^{i} s_t$. The prover maintains $Q_i = g^{q_i}$ and $r_i$ satisfying $X_i = q_i\ell + r_i, 0 \le r_i < \ell$. Initialize $Q_0 = 1_{\mathbb{G}_?}$ and $r_0 = 1$. On input $s_i$, compute the Euclidean division $r_{i-1}s_i = \delta_i\ell + r_i, 0 \le r_i < \ell$, and update $Q_i = Q_{i-1}^{s_i} \cdot g^{\delta_i}$. The invariant follows since $q_i = q_{i-1}s_i + \delta_i$. Moreover, $\delta_i < s_i$, so both exponents used in the update have $O(L_{\mathsf{DI}})$ bits. After processing all $n$ factors, $(Q_n, r_n) = (g^q, r)$ is the required **PoKE** response.

The same streaming procedure applies to **PoKEModN**. Writing $M$ for the residue modulus, the prover decomposes with respect to $\ell M$ instead of $\ell$: it maintains $X_i = q_i(\ell M) + r_i, 0 \le r_i < \ell M$, computes $r_{i-1}s_i = \delta_i(\ell M) + r_i$, and uses the same group update $Q_i = Q_{i-1}^{s_i} \cdot g^{\delta_i}$. The final residue $r_n \bmod M$ is then checked against the claimed value. Thus, when both $g^x$ and the quotient witness are generated from streamed factors, the prover uses at most three $O(L_{\mathsf{DI}})$-bit group exponentiations

[7] As in prior GUO-based constructions [3, 9, 77], we analyze these arguments in the generic group model, which does not capture that square roots are efficiently computable in class groups, so that the strong RSA assumption fails there [19]; knowledge extraction in class groups then binds the prover only to dyadic rationals rather than integers [29].

per factor; if $g^x$ is already part of the statement, only the two-exponentiation quotient pass is needed. In either case, the prover time is $O(n)$ in our group-exponentiation cost model, with no $O(nL_{\mathsf{DI}})$-bit integer product ever formed.

For constructions needing zero-knowledge (Section 6, Sections B.1), we use their NIZK variants from [9,91] and listed their concrete proof sizes (in group elements and $2\lambda$-bit fields): **NIZK-PoKE** ($5\mathbb{G}_? + 2\mathbb{Z}_{2\lambda}$), **NIZK-PoKRep** ($2\mathbb{G}_? + |\boldsymbol{x}|\mathbb{Z}_{2\lambda}$), and **NIZK-PoKEModN** ($3\mathbb{G}_? + \mathbb{Z}_{2\lambda} + \mathbb{Z}_{|N|+2\lambda}$). These prove the same relations as their non-ZK counterparts and admit efficient simulators. Extended preliminaries on the Order assumption, discrete logarithm assumption, Decisional Diffie–Hellman (DDH) assumption, and DDH-II assumption can be found in Appendix A.

# 4 Morphic Accumulation

In this section we construct a morphic universal accumulator, and demonstrate how to efficiently prove group operations over accumulated subsets.

## 4.1 Division Intractability of Discrete Logarithm Encodings

The key insight enabling morphic accumulation is that discrete logarithm encodings function as division intractable hash functions in the generic group Model. Consider the hash function $H_g : \mathbb{Z} \to [0, M)$ defined by $H_g(x) = g^x \bmod M$, where $g$ is a generator of a multiplicative group modulo $M$ and the output is the canonical integer representation. Unlike traditional DI hash functions that destroy algebraic structure, $H_g$ preserves homomorphic relationships:

$$H_g(x_1) \cdot H_g(x_2) = g^{x_1} \cdot g^{x_2} = g^{x_1+x_2} = H_g(x_1 + x_2) \pmod{M}.$$

For the DI statement below, exponent inputs are considered modulo the equivalence relation $x = x'$ if and only if $g^x = g^{x'}$, with the identity elements excluded. Thus, distinct DI inputs correspond to distinct non-identity group elements, rather than merely distinct integer representatives. We now prove the corresponding DI statement in the GGM.

**Theorem 1 (Division intractability of discrete logarithm encodings).** *In the generic group model, let $\mathbb{G}$ be a group with $\geq \lambda^2$-bit encodings and a uniformly sampled generator $g$. For any PPT adversary $\mathcal{A}$ and any $n = \mathrm{poly}(\lambda)$, the probability of finding distinct exponents $x_1, \ldots, x_n, y$ s.t. the canonical integer representation of $g^y$ divides the integer product $\prod_{i=1}^n g^{x_i}$ is negligible.*

*Proof.* In the GGM, the group $\mathbb{G}$ is defined by a random injective mapping $\sigma : \mathbb{Z}_{|\mathbb{G}|} \to \mathbb{F}$ where $|\mathbb{F}| \geq 2^{\lambda^2}$ (Definition 1). For any set of distinct exponents $\{x_1, \ldots, x_n, y\}$, the corresponding encodings $\sigma(x_1), \ldots, \sigma(x_n), \sigma(y)$ are uniformly distributed over $\mathbb{F}$ without replacement. Since $n + 1 = \mathrm{poly}(\lambda)$ encodings are sampled without replacement from a space of size $|\mathbb{F}| \geq 2^{\lambda^2}$, the statistical distance from $n + 1$ independent uniform samples is at most $\binom{n+1}{2}/|\mathbb{F}| = \mathrm{negl}(\lambda)$. Thus, we may treat the canonical integer representations of $(g^{x_1}, \ldots, g^{x_n}, g^y)$ as independent, uniformly random integers in $[0, 2^{\lambda^2})$.

By Lemma 1, for independent, uniformly random $\lambda^2$-bit integers, the probability that one divides the product of the others is negligible. Therefore, by a union bound over the negligible statistical distance and the division probability, finding a division relation among canonical integer representations of generic group elements occurs with negligible probability. □

This theorem establishes that group elements from integer groups with sufficiently large encodings can be directly accumulated into GUO-based accumulators without preprocessing, while preserving their algebraic structure for efficient proofs of group operations. Our approach applies to integer groups commonly used in cryptography:

– **Prime-order groups:** Subgroups of $\mathbb{Z}_p^*$ with large safe prime $p$
– **Groups of unknown order:** Subgroups of $\mathbb{Z}_N^*$ with RSA modulus $N$

**Concrete parameters.** The bit-length requirement applies to the *group encoding size*, not the group order itself (though they are often related). Based on the analysis in [50] and Lemma 1, 128-bit security against division attacks requires group-element encodings of length $L_{\mathsf{DI}} \geq 1704$ bits. For 192-bit security, this increases to $L_{\mathsf{DI}} \geq 2896$ bits. This condition is naturally met by standard parameters for RSA groups (e.g., 2048-bit modulus) and prime-order subgroups of $\mathbb{Z}_p^*$ with large safe primes.

**Remark on hidden-order RSA groups.** For hidden-order RSA groups $\mathbb{G}_N$, the input space $\mathbb{Z}$ in the definition of $H_g$ is natural: since the group order is hidden, the map $x \mapsto g^x$ is computationally binding over all integers (message space). This aligns with integer commitments [30], where $C = g^x$ commits to an arbitrary integer $x \in \mathbb{Z}$.

**Remark on prime-order groups.** For prime-order groups $\mathbb{G}_p$, the map $x \mapsto g^x$ is only binding for $x \in \mathbb{Z}_p$; different exponents $x$ and $x + kp$ yield the same group element. However, the DI property in Theorem 1 relies on the distribution of canonical integer representations in the encoding space, not on whether the encoding function is injective on its input domain. Since the canonical representation of any group element is uniformly random in the GGM regardless of which exponent produced it, the DI property holds for prime-order groups.

**Extension to multiple generators.** Theorem 1 extends naturally to encodings with multiple generators. Consider $H_{g,h}(x, y) = g^x h^y \bmod M$ where $g$ and $h$ are generators with unknown discrete logarithm relation. In the GGM, the encoding of $g^x h^y$ is uniformly distributed over the encoding space $\mathbb{F}$: since each generator maps to independent uniformly random encodings, their product (computed via the group operation oracle) remains uniformly distributed. Thus, the DI property holds for multi-generator encodings by the same argument as Theorem 1.

**Morphic Accumulator Construction.** Since discrete logarithm encodings satisfy DI (Theorem 1), they can be directly used in standard GUO-based accumulators [9, 20, 55]. The setup, accumulation, and non-membership proofs follow exactly the standard paradigm [9, 20, 55].

Recall from Definition 5 that accumulator soundness requires the element $s$ to be in the accumulation domain $\mathcal{X}_\lambda$. For DI hash-based accumulators, $\mathcal{X}_\lambda$ is the set of valid DI hash outputs, and the verifier must check $s \in \mathcal{X}_\lambda$ to prevent an adversary from claiming membership for an arbitrary integer that happens to divide the accumulated product. For discrete logarithm encodings, $\mathcal{X}_\lambda = \mathbb{G}_p \setminus \{1\}$. Unlike random oracle-based DI hashes where the verifier must recompute the hash from the revealed preimage [56], we can verify $s \in \mathbb{G}_p$ by proving knowledge of its exponent via $\textbf{NI-PoKExp}_{\mathbb{G}_p}$. Let $Acc = g_{acc}^{\prod_{s_i \in S} s_i}$ be an accumulator over $S = \{s_1, \ldots, s_n\} \subset \mathbb{G}_p$, where each $s_i = g_p^{x_i}$ for some exponent $x_i$. To prove membership of $s_t = g_p^{x_t} \in S$, the prover:

1. Computes $C = g_{acc}^{\prod_{s_i \in S \setminus \{s_t\}} s_i}$ (standard membership witness).
2. Generates $\pi \leftarrow \textbf{NI-PoKExp}_{\mathbb{G}_p}(g_p, s_t; x_t)$ to prove $s_t \in \mathcal{X}_\lambda$.
3. Sends $(C, \pi)$ to the verifier.

The verifier checks $C^{s_t} \stackrel{?}{=} Acc$ and $\textbf{NI-PoKExp}_{\mathbb{G}_p}.\textbf{Verify}(g_p, s_t, \pi) \stackrel{?}{=} 1$. Security follows from standard GUO accumulator [20, 55] combined with Theorem 1.

### 4.2 Homomorphic Group Operations within Accumulators

Since accumulated elements preserve their algebraic structure, we can efficiently prove group operations over any subset. Combined with the **PoKEModN** protocol [91], a prover can demonstrate that a target group element $z$ equals the product of a subset of accumulated elements modulo the group order. Let $\mathbb{G}_{val}$ denote the value group (either a prime-order group $\mathbb{G}_p$ with modulus $p$, or a hidden-order RSA group $\mathbb{G}_N$ with modulus $N$) with generator $g_{val}$. Given an accumulator $Acc = g_{acc}^{\prod_{s_i \in S} s_i}$ over elements $S = \{s_1, \ldots, s_n\} \subset \mathbb{G}_{val}$, where each $s_i = g_{val}^{x_i}$ for some exponent $x_i$, we define the relation:

$$\mathcal{R}_{\mathbf{SPAcc}} = \{(Acc \in \mathbb{G}_?, z \in \mathbb{G}_{val}, S \subset \mathbb{G}_{val}; S' \subseteq S, x_z) : \\ Acc = g_{acc}^{\prod_{s_i \in S} s_i}, \prod_{s_i \in S'} s_i \equiv z \pmod{M},\ z = g_{val}^{x_z}\},$$

where $M$ is the modulus of $\mathbb{G}_{val}$ (either $p$ or $N$), and products are over canonical integer representations.

However, designing a sound protocol for relation $\mathcal{R}_{\mathbf{SPAcc}}$ under the standard DI definition is not straightforward. The subtlety arises from the modular arithmetic in the relation: we require $\prod_{s_i \in S'} s_i \equiv z \pmod{M}$, meaning the integer product $prod = \prod_{s_i \in S'} s_i$ satisfies $prod = z + kM$ for some integer $k$. When the prover uses **PoKEModN** to demonstrate this congruence, the extractor recovers the integer $prod$, not the group element $z$ directly.

The security concern is as follows. Suppose a malicious prover claims that a group element $z$ is the product of a subset modulo $M$. The standard DI property (Theorem 1) guarantees that $z$ itself is unlikely to divide the accumulated product $\prod_{s_i \in S} s_i$ unless $z$ is indeed a subset product. However, the prover actually works with the integer $prod = z + kM$, and standard DI does not preclude the possibility that $z + kM$ divides the product for some cleverly chosen $k \neq 0$. If an adversary can find such an integer $k$, soundness would be violated: the adversary could claim a false subset relation by exploiting modular wraparound.

**The splitting attack.** To illustrate the threat concretely, consider two group elements with encodings $g^{x_1} = 6$ and $g^{x_2} = 35$, so the accumulated integer product is $P = 6 \times 35 = 210$. The legitimate subset products of $\{g^{x_1}, g^{x_2}\}$ are $\{1, 6, 35, 210\}$. Now consider the value 15: since $15 \mid 210$ but $15 \notin \{1, 6, 35, 210\}$, it is a *non-subset-product divisor* of $P$. It arises by "splitting" elements: $15 = 3 \times 5$, where the factor 3 is borrowed from $g^{x_1} = 2 \times 3$ and the factor 5 from $g^{x_2} = 5 \times 7$. A split divisor is not by itself a false proof: the adversary must also find an exponent $y$ such that $g^y + kM = 15$ for some integer $k$; only then could it falsely claim subset membership by exploiting this non-subset-product divisor. (In practice, encodings have length $L_{\mathsf{DI}} \geq 1704$ bits.)

To address this gap, we introduce a strengthened notion called *group element division intractability* (GEDI), which formalizes that such splitting attacks are computationally infeasible: any divisor of the accumulated product of the form $H(Y) + kM$ must be a genuine subset product of the hash outputs.

**Definition 6 (Group element division intractable).** *A function family $\mathcal{H}$ is called group element division intractable (GEDI) with respect to modulus $M$ if, for all PPT adversaries $\mathcal{A}$, given a security parameter $\lambda$, the probability that $\mathcal{A}$ outputs $H \in \mathcal{H}$, an integer $k$, and distinct inputs $X_1, \ldots, X_m, Y$ with $m < \mathrm{poly}(\lambda)$ such that $(H(Y) + kM)$ divides $\prod_{i=1}^m H(X_i)$ but is not equal to $\prod_{j \in S} H(X_j)$ for any subset $S \subseteq [m]$, is negligible:*

$$\Pr\left[\begin{array}{c} (H(Y) + kM) \text{ divides } \prod_{i=1}^m H(X_i) \\ \wedge\ (H(Y) + kM) \notin \left\{ \prod_{j \in S} H(X_j) : S \subseteq [m] \right\} \end{array}\right] < \mathrm{negl}(\lambda).$$

*Equivalently, any divisor of $\prod_{i=1}^{m} H(X_i)$ of the form $H(Y) + kM$ must be a product of some subset of $\{H(X_1), \ldots, H(X_m)\}$, except with negligible probability.*

**Analysis of GEDI for discrete logarithm encodings.** For concrete parameter estimates below, we substitute the concrete encoding length $L_{\mathsf{DI}}$ for the asymptotic quantity $\lambda^2$. In the GGM, the encodings $g^{x_1}, \ldots, g^{x_n}$ and $z = g^y$ are independent uniform integers in $[0, 2^{L_{\mathsf{DI}}})$. Let $P = \prod_{i=1}^{n} g^{x_i}$. We analyze the cases $k = 0$ and $k \geq 1$ separately.

**Case $k = 0$: reduction to standard DI.** The candidate divisor is $z$ itself. Since $z$ is independent of $g^{x_1}, \ldots, g^{x_n}$, the event $z \mid P$ is exactly the standard DI condition, and $\Pr[\text{win with } k = 0] \leq \Pr[z \mid P] < \mathrm{negl}(\lambda)$ by Lemma 1.

**Case $k \geq 1$: information-theoretic bound.** For $k \geq 1$, the candidate divisor satisfies $z + kM \geq M \geq 2^{L_{\mathsf{DI}}}$. Condition on $g^{x_1}, \ldots, g^{x_n}$, fixing $P$; the encoding $z$ remains uniform in $[0, 2^{L_{\mathsf{DI}}})$. The adversary wins iff $z$ is congruent modulo $M$ to some non-subset-product divisor of $P$. Denote $d(P)$ as the number of positive divisors of $P$. Since each divisor determines a unique residue class modulo $M$, while $z$ is uniform over $2^{L_{\mathsf{DI}}}$ values, $\Pr[\text{win with } k \geq 1 \mid P] \leq d(P)/2^{L_{\mathsf{DI}}}$. It remains to bound $\mathbb{E}[d(P)]$. By the Dirichlet divisor estimate [2], a uniform random integer in $[1, 2^{L_{\mathsf{DI}}}]$ has on average at most $L_{\mathsf{DI}} \ln 2$ divisors. Since $d(ab) \leq d(a)d(b)$ for all positive integers, $d(P) \leq \prod_{i=1}^{n} d(g^{x_i})$. By independence of the $g^{x_i}$, we obtain $\mathbb{E}[d(P)] \leq \prod_{i=1}^{n} \mathbb{E}[d(g^{x_i})] \leq (L_{\mathsf{DI}} \ln 2 + 1)^n$. Therefore:

$$\Pr[\text{win with } k \geq 1] \leq \frac{(L_{\mathsf{DI}} \ln 2 + 1)^n}{2^{L_{\mathsf{DI}}}}. \tag{1}$$

Thus the information-theoretic bound is meaningful when $n < n_0 := \frac{L_{\mathsf{DI}}}{\log_2(L_{\mathsf{DI}} \ln 2 + 1)}$. Asymptotically, when $L_{\mathsf{DI}}(\lambda) = \lambda^2$, this condition is $n = o(\lambda^2 / \log \lambda)$. For $L_{\mathsf{DI}} = 1704$ using the 128-bit DI calibration of [50], this gives $n_0 \approx 166.94$. Below this threshold, a non-subset-product divisor with the correct residue is unlikely to *exist*.

**Case $k \geq 1$: computational bound.** Above the information-theoretic threshold, a matching divisor may exist among the $d(P)$ divisors of $P$. However, *existence does not imply efficient computability.* We now show that finding such a divisor reduces to well-studied hard problems. Define $\gamma := P \cdot z^{-1} \bmod M$ (the complementary residue, computable from the group structure when $z$ is invertible modulo $M$, which holds with overwhelming probability). A GEDI violation with $k \geq 1$ requires solving the constrained factoring equation

$$(z + k_1 M)(\gamma + k_2 M) = P,$$

where $z$, $\gamma$, $M$, $P$ are known and only $k_1 \geq 1$, $k_2 \geq 0$ are unknown. Since $k_2$ is determined by $k_1$, this reduces to finding a factor of the $nL_{\mathsf{DI}}$-bit integer $P$ in the residue class $z \bmod M$. We formalize this as a special *modular subset-product problem.* The intuition is that no efficient adversary can find a divisor of $P$ that is congruent to $z$ modulo $M$.

**Assumption 1** (Modular Subset-Product)**.** *Let $\lambda$ be a security parameter, $M \geq 2^{\lambda^2}$, and $n = \mathrm{poly}(\lambda)$. Sample $a_1, \ldots, a_n \xleftarrow{\$} [0, 2^{\lambda^2})$ independently and uniformly at random, let $P = \prod_{i=1}^{n} a_i$, and sample $z \xleftarrow{\$} [0, 2^{\lambda^2})$ independently. For every PPT adversary $\mathcal{A}$:*

$$\Pr\left[d \leftarrow \mathcal{A}(a_1, \ldots, a_n, z, M) : d \text{ divides } P \wedge d \equiv z \pmod{M}\right] < \mathrm{negl}(\lambda).$$

**Hardness of the modular subset-product assumption.** We analyze the hardness of Assumption 1 by progressively simplifying the adversary's task. Let $P = p_1^{e_1} \cdots p_t^{e_t}$ be the prime decomposition of $P$. Revealing the factorization to the adversary (making the problem strictly easier) reduces

Assumption 1 to finding exponents $0 \leq f_i \leq e_i$ such that $\prod p_i^{f_i} \equiv z \pmod{M}$ with $\prod p_i^{f_i} \geq M$, a classical *modular subset-product instance* over the prime-power factors [24, 68]. The hardness of this instance depends on the modulus $M$. For prime modulus $M = p$, the group $\mathbb{Z}_p^*$ has known order $p - 1$, so the only barrier to linearizing the problem is computing discrete logarithms, which is sub-exponential via the number field sieve [59]. For RSA modulus $M = N$, the group $\mathbb{Z}_N^*$ has hidden structure: the group order $\phi(N)$ is unknown (requiring factoring $N$ to learn), and computing discrete logarithms in the subgroups poses an additional barrier [59]. The RSA case is therefore strictly harder [68], as the adversary faces two barriers (factoring + discrete logarithm) rather than one, making the modular subset-product problem potentially harder than either factoring or the discrete logarithm problem individually.

In either case, the modular subset-product problem is at least as hard as the modular subset-sum problem, which is NP-hard in the worst case and hard on average for random instances [46]: for any prime $M$, exponentiation by a primitive root reduces subset-sum over $\mathbb{Z}_{M-1}$ to subset-product over $\mathbb{Z}_M^*$ [24]. Lattice-based attacks on multiplicative knapsack instances succeed only when the prime factors are small and revealed [68]; without a discrete logarithm oracle, the multiplicative structure admits no known lattice formulation, and the best known generic attack is meet-in-the-middle in time $O(2^{t/2})$.

Simplifying further by granting the adversary a discrete logarithm oracle (infeasible for RSA modulus, sub-exponential for prime modulus) transforms the subset-product instance into a modular subset-sum instance with $t$ variables and $L_{\mathsf{DI}}$-bit modulus. For an $L_{\mathsf{DI}}$-bit random integer, the expected number of prime factors is about $\ln L_{\mathsf{DI}}$. Thus $t \approx n \ln L_{\mathsf{DI}}$ and $\delta \approx n \ln L_{\mathsf{DI}} / L_{\mathsf{DI}}$. The known attacks on this instance are:

- **Lattice methods** (LLL/BKZ): solve random instances in polynomial time when $\delta < 0.9408$ [28, 48, 52], i.e., when $n \lesssim 0.9408 L_{\mathsf{DI}} / \ln L_{\mathsf{DI}}$ (e.g., $n \leq 215$ for $L_{\mathsf{DI}} = 1704$ at 128-bit security per [27, 50]). For $\delta \geq 0.9408$, lattice methods become ineffective.
- **Meet-in-the-middle** [44]: runs in time $O(2^{t/2}) \approx O(2^{n \ln L_{\mathsf{DI}}/2})$ for any density. For the applications in this paper ($n \geq 256$), this exceeds $2^{950}$ operations.
- **General-purpose algorithms**: the best known algorithms for modular subset sum run in $\tilde{O}(M)$ time [5, 22], which for $M \geq 2^{L_{\mathsf{DI}}}$ is super-exponential in $\lambda$. No algorithm is known to avoid this dependence on $M$ while also bypassing the density-dependent lattice methods above.

In summary, Assumption 1 is supported by a chain of reductions: even after factoring $P$ and computing discrete logarithms, the adversary faces a modular subset-sum instance whose hardness depends on the density $\delta \approx n \ln L_{\mathsf{DI}} / L_{\mathsf{DI}}$.

**Security regimes as a function of $n$.** Combining the information-theoretic bound (Eq. 1) with the above, the analysis decomposes into three regimes. The first regime is information-theoretic, and the latter two describe the resulting computational attack landscape, using factoring and discrete-logarithm oracles to obtain a conservative upper bound on adversarial power:

1. **Below the information-theoretic threshold** ($n < n_0$; $n < 167$ for $L_{\mathsf{DI}} = 1704$): Non-subset-product divisors in the correct residue class are unlikely to exist (Eq. 1). Security is information-theoretic; no computational assumption is needed.
2. **Transition regime** ($n_0 \leq n \lesssim 0.9408 L_{\mathsf{DI}} / \ln L_{\mathsf{DI}}$, i.e., $\delta < 0.9408$; $167 \leq n \leq 215$ for $L_{\mathsf{DI}} = 1704$): Divisors in the correct residue class may exist. An adversary with factoring and DL oracles could mount a polynomial-time lattice attack [28, 48, 52]. Security in this regime rests on the sub-exponential hardness of factoring and discrete logarithms at $L_{\mathsf{DI}}$ bits. Without these oracles, the multiplicative structure admits no known lattice formulation.
3. **High-density regime** ($n \geq 0.9408 L_{\mathsf{DI}} / \ln L_{\mathsf{DI}}$, i.e., $\delta \geq 0.9408$; $n \geq 216$ for $L_{\mathsf{DI}} = 1704$): Even with factoring and DL oracles, lattice methods become ineffective. The best known attack is

meet-in-the-middle over the $t$ prime-power factors, running in time $O(2^{t/2})$, analogous to the Coron–Naccache sub-exponential attack on standard DI [27].

**Remark on the hybrid model.** The GEDI theorem below combines the GGM (which provides independent uniform encodings) with the modular subset-product assumption (Assumption 1). This hybrid is necessary because the GGM alone gives an information-theoretic bound for $k \geq 1$ that is meaningful only when $n < n_0$ (e.g., $n < 167$ for $L_{\mathsf{DI}} = 1704$), while the modular subset-product assumption extends the guarantee to all $n = \mathrm{poly}(\lambda)$. For the applications in this paper (e.g., $n$-bit range proofs with $n \geq 256$), $n$ exceeds the lattice threshold ($n \geq 216$ for $L_{\mathsf{DI}} = 1704$), placing the problem squarely in the high-density regime where even an adversary with factoring and DL oracles faces a meet-in-the-middle barrier of $O(2^{t/2})$.

**Theorem 2 (GEDI of discrete logarithm encodings).** *Let $\mathbb{G}$ be a group with encoding size at least $\lambda^2$ bits and let $M \geq 2^{\lambda^2}$ denote the group modulus. In the generic group model, under the modular subset-product assumption (Assumption 1), for any PPT adversary $\mathcal{A}$ and any $n = \mathrm{poly}(\lambda)$, the probability of finding distinct exponents $x_1, \ldots, x_n, y$ and any integer $k$ such that $g^y + kM$ divides $\prod_{i=1}^{n} g^{x_i}$ but $g^y + kM \neq \prod_{j \in S} g^{x_j}$ for any $S \subseteq \{1, \ldots, n\}$, is negligible.*

*Proof.* In the GGM, the group is defined by a random injective mapping from $\mathbb{Z}_{|\mathbb{G}|}$ to an encoding space of size at least $2^{\lambda^2}$. For any set of $n+1$ distinct exponents, the corresponding encodings are sampled without replacement from this space. Since $n + 1 = \mathrm{poly}(\lambda)$ values are drawn from a space of size at least $2^{\lambda^2}$, the statistical distance from $n+1$ independent uniform samples in $[0, 2^{\lambda^2})$ is at most $\binom{n+1}{2}/2^{\lambda^2} = \mathrm{negl}(\lambda)$ (as in the proof of Theorem 1). Hence we may treat $g^{x_1}, \ldots, g^{x_n}, z = g^y$ as independent uniform integers in $[0, 2^{\lambda^2})$. Let $P = \prod_{i=1}^{n} g^{x_i}$. Since $M \geq 2^{\lambda^2}$ and $z < 2^{\lambda^2}$, only $k \geq 0$ is relevant.

*Case $k = 0$.* The event $z \mid P$ is exactly the standard DI condition. By Lemma 1, $\Pr[z \mid P] < \mathrm{negl}(\lambda)$.

*Case $k \geq 1$.* The winning event—that $(z + kM)$ divides $P$ and is not a subset product—depends only on the integer values of the encodings and the adversary's output $k$. Since the integer encodings are independent uniform in $[0, 2^{\lambda^2})$, any GGM adversary $\mathcal{A}$ can be simulated by a PPT algorithm $\mathcal{B}$ that receives only the integer encodings $(a_1, \ldots, a_n, z)$ as input: the GGM oracle responses for newly computed group elements are fresh random encodings, independent of the initial integer encodings, and hence provide no additional information useful for the divisibility condition. $\mathcal{B}$ samples these oracle responses as fresh random values and runs $\mathcal{A}$ internally, achieving the same winning probability. Therefore, the winning probability is at most the probability that a PPT algorithm, given $(a_1, \ldots, a_n, z, M)$ with $a_i, z$ independent uniform in $[0, 2^{\lambda^2})$, finds a divisor $d$ of $P = \prod a_i$ with $d \equiv z \pmod{M}$, $d \geq M$, and $d \notin \mathrm{SubProd}$. The exclusion constraint $d \notin \mathrm{SubProd}$ can be dropped when upper-bounding the winning probability, since removing this constraint only enlarges the success event. The remaining problem, which is finding $d$ such that $d \mid P$, $d \equiv z \pmod{M}$, and $d \geq M$, is exactly Assumption 1. Therefore $\Pr[\text{win with } k \geq 1] < \mathrm{negl}(\lambda)$ under Assumption 1.

Combining both cases by a union bound: $\Pr[\text{GEDI violation}] \leq \Pr[\text{win with } k = 0] + \Pr[\text{win with } k \geq 1] < \mathrm{negl}(\lambda)$. □

**Protocol SPAcc (Figure 1).** By combining **PoKE** and **PoKEModN** [9, 91], we construct a protocol for $\mathcal{R}_{\mathbf{SPAcc}}$. The protocol requires three components:

1. The prover computes the intermediate commitment $C = g_{acc}^{prod}$ where $prod = \prod_{s_i \in S'} s_i$ is the integer product of the claimed subset.
2. **Subset membership** ($\pi_1$): Using **NI-PoKE** on $(C, Acc)$, the prover demonstrates knowledge of an exponent $y$ such that $C^y = Acc$. Since $Acc = g_{acc}^{\prod_{s_i \in S} s_i}$ and $C = g_{acc}^{prod}$, this proves that $prod$

Protocol **SPAcc** for Relation $\mathcal{R}_{\mathbf{SPAcc}}$
Params: $g_{acc} \in \mathbb{G}_?$, $g_{val} \in \mathbb{G}_{val}$, modulus $M$;
Inputs: $Acc \in \mathbb{G}_?$, $z \in \mathbb{G}_{val}$, $S \subset \mathbb{G}_{val}$; Witness: $S' \subseteq S$, $x_z$;
Claim: $z = g_{val}^{x_z}, \prod_{s_i \in S'} s_i \equiv z \pmod M$.

1. Prover computes $prod = \prod_{s_i \in S'} s_i$, $C = g_{acc}^{prod}$.
   Prover computes $\pi_1 \leftarrow$ **NI-PoKE**$(C, Acc; \prod_{s_i \in S \setminus S'} s_i)$,
   $\pi_2 \leftarrow$ **NI-PoKExp**$_{\mathbb{G}_{val}}(g_{val}, z; x_z)$, and
   $\pi_3 \leftarrow$ **NI-PoKEModN**$(C, M, z; prod)$. Prover sends $(C, \pi_1, \pi_2, \pi_3)$.
2. Verifier checks:
   **NI-PoKE.Verify**$(C, Acc, \pi_1) = 1$, **NI-PoKExp**$_{\mathbb{G}_{val}}$**.Verify**$(g_{val}, z, \pi_2) = 1$, and
   **NI-PoKEModN.Verify**$(C, M, z, \pi_3) = 1$.

**Fig. 1.** Protocol **SPAcc**: morphic accumulation with homomorphic group operations.

divides $\prod_{s_i \in S} s_i$ as integers. Combined with the domain membership and modular equivalence proofs, the knowledge soundness argument (Theorem 3) shows that *prod* must in fact be an integer product of a subset of $S$.

3. **Domain membership** ($\pi_2$): Using **NI-PoKExp**$_{\mathbb{G}_{val}}$ on $(g_{val}, z)$, the prover demonstrates that $z$ is a valid group element in $\mathbb{G}_{val}$ by proving knowledge of $x_z$ such that $z = g_{val}^{x_z}$. This prevents the adversary from claiming membership for an arbitrary integer that happens to divide the accumulated product; the element must genuinely be in the accumulation domain $\mathcal{X}_\lambda = \mathbb{G}_{val}$.
4. **Modular equivalence** ($\pi_3$): Using **NI-PoKEModN** on $(C, M, z)$, the prover shows that the exponent of $C$ (i.e., *prod*) satisfies $prod \equiv z \pmod M$. This binds the integer product to the claimed group element $z$.

**Theorem 3.** *The protocol in Figure 1 is an argument of knowledge for* $\mathcal{R}_{\mathbf{SPAcc}}$ *in GGM, assuming* **NI-PoKE**, **NI-PoKExp**$_{\mathbb{G}_{val}}$ *and* **NI-PoKEModN** *are arguments of knowledge, the order assumption holds for* $\mathbb{G}_?$*, and the GEDI property (Theorem 2) holds for discrete logarithm encodings in* $\mathbb{G}_{val}$ *(which requires the modular subset-product assumption, Assumption 1).*

*Proof.* **Completeness.** For a valid witness $(S', x_z)$, let $prod = \prod_{s_i \in S'} s_i$ and $y = \prod_{s_i \in S \setminus S'} s_i$. Then $C = g_{acc}^{prod}$, $Acc = C^y$, and $prod \equiv z \pmod M$.

**Knowledge soundness.** *Proof overview:* we extract integer witnesses from each subproof, then show the extracted integer $\bar{prod}$ is a genuine subset product of $S$ by running a greedy trial-division procedure. The key step establishes that the trial-division remainder must equal 1; if it exceeds 1, we derive a contradiction to the GEDI property (Theorem 2) via case analysis.

We invoke the knowledge extractors on each subproof separately: from $\pi_1$ (applying the **NI-PoKE** extractor to $(C, Acc)$), we obtain an integer $y$ such that $Acc = C^y$; from $\pi_3$ (applying the **NI-PoKEModN** extractor to $(C, M, z)$), we obtain an integer $\bar{prod}$ such that $C = g_{acc}^{\bar{prod}}$ and $\bar{prod} \bmod M = z$; from $\pi_2$ (applying the **NI-PoKExp**$_{\mathbb{G}_{val}}$ extractor to $(g_{val}, z)$), we obtain an integer $x_z$ such that $z = g_{val}^{x_z}$. From the first two extractions, $Acc = C^y = (g_{acc}^{\bar{prod}})^y = g_{acc}^{\bar{prod} \cdot y}$. Since $Acc = g_{acc}^{\prod_{s_i \in S} s_i}$, we have $g_{acc}^{\bar{prod} \cdot y} = g_{acc}^{\prod_{s_i \in S} s_i}$. If $\bar{prod} \cdot y \neq \prod_{s_i \in S} s_i$, then $|\bar{prod} \cdot y - \prod_{s_i \in S} s_i|$ is a nonzero multiple of the order of $g_{acc}$, violating the order assumption for $\mathbb{G}_?$. Therefore $prod \cdot y = \prod_{s_i \in S} s_i$, and $\bar{prod} \mid \prod_{s_i \in S} s_i$.

We now show that $\bar{prod}$ is a subset product of $S$. We run a greedy trial-division algorithm on $\bar{prod}$: initialize a running quotient $E \leftarrow \bar{prod}$; for each $s_i \in S$ (processed in an arbitrary fixed order), if $s_i \mid E$, include $s_i$ and update $E \leftarrow E/s_i$. Let $T \subseteq S$ be the set of elements extracted, and let

$R = \bar{prod}/\prod_{s_i \in T} s_i$ be the final remainder. If $R = 1$, then $\bar{prod} = \prod_{s_i \in T} s_i$ is a valid subset product, and extraction succeeds.

Note that $R > 1$ can occur even for a valid subset product $\bar{prod}$ if accumulated elements share prime factors (i.e., $\gcd(s_i, s_j) > 1$ for some $i \neq j$, which may cause the greedy algorithm to "consume" a shared factor from the wrong element). We stress that the correctness of the extraction does not require pairwise coprimality of the $s_i$; instead, the GEDI-based case analysis below shows that $R > 1$ leads to a contradiction regardless of the cause, establishing that this event occurs with negligible probability. Since $\bar{prod} \mid \prod_{s_i \in S} s_i$ and $\prod_{s_i \in T} s_i \mid \bar{prod}$, we have $R \mid \prod_{s_i \in S \setminus T} s_i$. Now, define the group element $\beta = z \cdot (\prod_{s_i \in T} s_i)^{-1} \bmod M \in \mathbb{G}_{val}$. (The modular inverse exists with overwhelming probability: for prime $M = p$ any nonzero element is invertible, and for RSA modulus $M = N$ the probability that $\prod_{s_i \in T} s_i$ shares a factor with $N$ is at most $|T| \cdot (1/p + 1/q) = \mathrm{negl}(\lambda)$ where $N = pq$.) Since $\bar{prod} = \prod_{s_i \in T} s_i \cdot R$ and $\bar{prod} \bmod M = z$, we have $\beta = R \bmod M$, so $R = \beta + k'M$ for some $k' \geq 0$. The exponent of $\beta$ is computable from the extraction: $\beta = g_{val}^{x_z - \sum_{s_i \in T} x_i}$, so $\beta$ is a valid group element with a known exponent. We consider two cases based on whether $\beta$ coincides with an element of $S \setminus T$. If $\beta$'s exponent is distinct from all exponents of elements in $S \setminus T$ (which holds with overwhelming probability in the GGM, since equality would constitute a non-trivial algebraic relation among independent random encodings), then $(\beta + k'M) \mid \prod_{s_i \in S \setminus T} s_i$ contradicts Theorem 2: $R$ divides $\prod_{s_i \in S \setminus T} s_i$ but is not a subset product. Next, we consider the case $\beta = s_j$ for some $s_j \in S \setminus T$.

*Sub-case $k' = 0$:* Then $R = s_j$. Since $R$ divides the running quotient at the step when $s_j$ was tested, $s_j$ must also divide that running quotient, so the greedy algorithm would have included $s_j$ in $T$—contradicting $s_j \notin T$.

*Sub-case $k' > 0$:* Define $\alpha = \prod_{s_i \in S \setminus T} s_i / R$. Since $R \mid \prod_{s_i \in S \setminus T} s_i$, $\alpha$ is a positive integer. Since $R \bmod M = \beta = s_j$, we have $\alpha \bmod M = s_j^{-1} \cdot \prod_{s_i \in S \setminus T} s_i \bmod M = g_{val}^{\sum_{s_i \in (S \setminus T) \setminus \{s_j\}} x_i}$. Write $\alpha = (\alpha \bmod M) + k''M$ for some $k'' \geq 0$. We exhaust the remaining sub-cases on $|(S \setminus T) \setminus \{s_j\}|$:

- If $|(S \setminus T) \setminus \{s_j\}| \geq 2$: the exponent of $(\alpha \bmod M)$ is a sum of at least two independent random exponents, distinct from any individual exponent with overwhelming probability in the GGM (equality would be a non-trivial algebraic relation). Thus $(\alpha \bmod M + k''M) \mid \prod_{s_i \in S \setminus T} s_i$ contradicts Theorem 2.
- If $|(S \setminus T) \setminus \{s_j\}| = 1$, say $\{s_l\}$: then $\prod_{s_i \in S \setminus T} s_i = s_j \cdot s_l$ and $R \cdot \alpha = s_j \cdot s_l$. If $k'' > 0$, then $R \cdot \alpha = (s_j + k'M)(s_l + k''M) > s_j \cdot s_l$, a contradiction. If $k'' = 0$, then $\alpha = s_l$, so $R \cdot s_l = s_j \cdot s_l$ gives $R = s_j$, contradicting $k' > 0$.
- If $(S \setminus T) \setminus \{s_j\} = \emptyset$: then $R \mid s_j$, but $R = s_j + k'M > s_j$, a contradiction.

Thus $R = 1$, and the extractor outputs $S' = T$ with $\prod_{s_i \in S'} s_i = \bar{prod}$ and $\bar{prod} \bmod M = z = g_{val}^{x_z}$. □

**Efficiency.** The prover computes $C = g_{acc}^{prod}$ using the streaming algorithm in Section 3.5, which requires $O(|S|)$ group exponentiations by $O(L_{\mathsf{DI}})$-bit exponents. The prover then generates one **NI-PoKE** proof for subset membership, one **NI-PoKExp**$_{\mathbb{G}_{val}}$ proof for domain membership, and one **NI-PoKEModN** proof. They have $O(1)$ proof size and $O(1)$ verification time. Thus, the total prover overhead is $O(|S|)$ (dominated by streaming the subset factors), with $O(1)$ verifier overhead.

**Lemma 3 (Unique extraction).** *Under the conditions of Theorem 3, the extracted subset $S'$ is unique: no two distinct subsets of $S$ have the same integer product, except with negligible probability.*

*Proof.* We prove by contradiction. Suppose distinct subsets $U \neq V$ of $S$ satisfy $\prod_{s_i \in U} s_i = \prod_{s_j \in V} s_j$ over the integers. Let $U' = U \setminus V$ and $V' = V \setminus U$; both are non-empty (since $U \neq V$), disjoint,

and satisfy $\prod_{s_i \in U'} s_i = \prod_{s_j \in V'} s_j$ because the common factors cancel. Pick any $s \in U'$. Then $s \mid \prod_{s_j \in V'} s_j$, and hence $s \mid \prod_{t \in S \setminus \{s\}} t$, since $V' \subseteq S \setminus \{s\}$. By Theorem 1, each event $s_i \mid \prod_{j \neq i} s_j$ has probability at most $\epsilon_{\mathrm{DI}}(\lambda)$, for a common negligible function $\epsilon_{\mathrm{DI}}$. Since any subset-product collision implies at least one of these $n$ events, a union bound gives

$$\Pr[\text{a subset-product collision exists}] \leq n \cdot \epsilon_{\mathrm{DI}}(\lambda) = \mathrm{negl}(\lambda),$$

since $n = \mathrm{poly}(\lambda)$. □

## 5 Optimal Range Proofs

A range proof demonstrates that a committed value $x$ lies within a specified interval $[0, 2^n - 1]$. We present a unified construction that works for any value group $\mathbb{G}_{val}$ whose elements can be accumulated into a morphic accumulator (Section 5.1), then extend it to achieve zero-knowledge for Pedersen-style commitments (Section B.1).

### 5.1 Basic Range Proof

Let $\mathbb{G}_{val}$ denote a value group (either a prime-order group $\mathbb{G}_p$ with modulus $p$, or a hidden-order RSA group $\mathbb{G}_N$ with modulus $N$) with generator $g_{val}$. The basic range proof relation is:

$$\mathcal{R}_{\mathbf{RangeProof}} = \{(g_{val} \in \mathbb{G}_{val}, C \in \mathbb{G}_{val}; x \in \mathbb{Z}) : C = g_{val}^x \wedge x \in [0, 2^n - 1]\}.$$

**Core idea.** The key observation is that binary decomposition naturally aligns with subset membership in morphic accumulators. Any integer $x \in [0, 2^n - 1]$ has a unique binary representation $x = \sum_{i=0}^{n-1} b_i 2^i$ with $b_i \in \{0, 1\}$. We precompute the group elements $u_i = g_{val}^{2^i}$ for $i \in \{0, \ldots, n-1\}$ and accumulate them into $Acc$. To prove $C = g_{val}^x$ for some $x \in [0, 2^n - 1]$, the prover identifies the subset $X' = \{u_i \mid b_i = 1\}$ corresponding to the binary decomposition and uses the **SPAcc** protocol (Figure 1) to prove that the integer product of $X'$ reduces to $C$ modulo $M$. Since $(\prod_{u_i \in X'} u_i) \bmod M = \prod_{b_i=1} g_{val}^{2^i} = g_{val}^{\sum b_i 2^i} = g_{val}^x = C$, the **SPAcc** proof establishes that $C$ equals a subset product of accumulated binary powers, which implies $x \in [0, 2^n - 1]$.

**Theorem 4.** *The protocol in Figure 2 is an argument of knowledge for relation $\mathcal{R}_{\mathbf{RangeProof}}$ in the generic group model, assuming the* **SPAcc** *protocol (Theorem 3) is secure.*

*Proof.* **Completeness.** If $x \in [0, 2^n - 1]$, then $x$ has a unique binary decomposition $x = \sum_{i \in I} 2^i$ for some index set $I \subseteq \{0, \ldots, n-1\}$. The corresponding subset $X' = \{u_i \mid i \in I\}$ has integer product $prod = \prod_{i \in I} u_i$. Reducing modulo $M$: $prod \bmod M = \prod_{i \in I} g_{val}^{2^i} = g_{val}^{\sum_{i \in I} 2^i} = g_{val}^x = C$. Thus the **SPAcc** relation $prod \equiv C \pmod M$ holds, and completeness follows from Theorem 3.
**Knowledge soundness.** By the knowledge soundness of **SPAcc** (Theorem 3), from any accepting proof, we can extract a subset $X' \subseteq \{u_0, \ldots, u_{n-1}\}$ and an integer $\bar{prod}$ such that $\bar{prod} = \prod_{u_i \in X'} u_i$ and $\bar{prod} \equiv C \pmod M$. Let $I \subseteq \{0, \ldots, n-1\}$ be the indices of this subset. Then $\bar{prod} \bmod M = \prod_{i \in I} g_{val}^{2^i} = g_{val}^{\sum_{i \in I} 2^i}$. Since $\bar{prod} \equiv C \pmod M$, we have $C = g_{val}^{x'}$ where $x' = \sum_{i \in I} 2^i$. Since $I$ is a set of distinct indices from $\{0, \ldots, n-1\}$, the sum $x' = \sum_{i \in I} 2^i$ is a valid binary expansion satisfying $0 \leq x' \leq \sum_{i=0}^{n-1} 2^i = 2^n - 1$. Therefore, the extractor outputs $x' \in [0, 2^n - 1]$ with $C = g_{val}^{x'}$. □

Range Proof Protocol for Relation $\mathcal{R}_{\mathbf{RangeProof}}$

Params $pp$: security parameter $\lambda$, accumulator group $\mathbb{G}_?$ with base $g_{acc}$, value group $\mathbb{G}_{val}$ with generator $g_{val}$, modulus $M$, range parameter $n$.

**Setup**$(pp)$:

1. Compute binary-power bases: $u_i = g_{val}^{2^i}$ for $i \in \{0, \ldots, n-1\}$.
2. Compute accumulated product: $e = \prod_{i=0}^{n-1} u_i$ (integer product of canonicals).
3. Compute accumulator: $Acc = g_{acc}^{e} \in \mathbb{G}_?$.
4. Output: $(Acc, \{u_i\}_{i=0}^{n-1})$.

**Prove**$(pp, Acc, \{u_i\}, C, x)$ where $C = g_{val}^{x}$:

1. Decompose $x = \sum_{i=0}^{n-1} b_i 2^i$ with $b_i \in \{0, 1\}$.
2. Identify subset $X' = \{u_i \mid b_i = 1\}$.
3. Run **SPAcc** prover with inputs $(Acc, C, \{u_i\}_{i=0}^{n-1})$ and witness $(X', x)$.
4. Output: $\pi \leftarrow \pi_{\mathbf{SPAcc}}$.

**Verify**$(pp, Acc, C, \pi)$:

1. Run **SPAcc** verifier on $(Acc, C, \{u_i\}_{i=0}^{n-1}, \pi)$.
2. Output: 1 if verification succeeds, 0 otherwise.

**Fig. 2.** Range Proof Protocol based on morphic accumulation.

**Complexity analysis.** The setup phase computes $n$ group exponentiations and one accumulator, which can be performed once and reused for all range proofs with the same parameters. The prover decomposes $x$ into $n$ bits in $O(n)$ time, computes the subset product $prod = \prod_{i \in I} u_i$ using $O(n)$ multiplications, and generates the **SPAcc** proof. By the efficiency of the **SPAcc** protocol (see the analysis following Theorem 3), the proof has $O(1)$ size, and verification requires $O(1)$ time. The total prover complexity is $O(n)$, dominated by streaming the subset factors, quotient witnesses, and membership witnesses. The proof size and verification time are both $O(1)$, independent of the range parameter $n$. We refer to Appendix B for extensions, including zero-knowledge range proofs and vector range proofs.

**Concrete estimate.** The above cost model also gives a useful concrete estimate. For the basic range proof, the prover performs at most $3n$ hidden-order group exponentiations by $O(L_{\mathsf{DI}})$-bit exponents. Using the Gryphes [92] measurement of 8.5 seconds to accumulate $2^{15}$ elements with 2048-bit hidden-order exponentiations on four cores, this corresponds to approximately 0.26 ms per exponentiation on four cores, or about 1.04 ms after a conservative single-thread normalization. Thus a 64-bit basic range proof requires roughly $3 \cdot 64 \cdot 1.04$ ms $\approx 200$ ms. For comparison, SwiftRange and Bulletproofs report roughly 550 ms and 750 ms, respectively, for 64-bit range proofs in the single-thread setting of [86], though this comparison should be interpreted cautiously because the groups, implementations, and security parameters differ. Nevertheless, it suggests that the hidden-order exponentiation cost is not prohibitive in the parameter range relevant to 64-bit range proofs. Moreover, recent vectorized hidden-order arithmetic techniques such as VROOM [53] can further reduce this cost.

**Discussion on instantiations.** While the protocol is unified, the choice of $\mathbb{G}_{val}$ has implications for security and applicability.

*Prime-order groups ($\mathbb{G}_p$).* For a prime-order subgroup of $\mathbb{Z}_p^*$ with order q, the elements $u_i = g_{val}^{2^i}$ are group elements with canonical integer representations in $[0, p)$. Security relies on the discrete logarithm assumption in $\mathbb{G}_p$ and the division intractability of discrete logarithm encodings (Theorem 1). The range parameter $n$ must satisfy $2^n < q$ to prevent modular wrap-around in the exponent, ensuring the binary decomposition is unique over the integers.

*Hidden-order RSA groups ($\mathbb{G}_N$).* For an RSA group $\mathbb{G}_N$ with hidden modulus $N$, the exponents are integers in $\mathbb{Z}$ rather than $\mathbb{Z}_p$. The range parameter $n$ can be arbitrarily large, not limited by $N$, since the order is unknown and binding holds over the integers [30]. Security relies on the strong RSA assumption and the order assumption. This setting is essential for polynomial commitment schemes (Section 5.2), where degree-bound proofs require range proofs over large integers (e.g., $O(d \log q)$ bits for degree-$d$ polynomials over $\mathbb{Z}_q$).

### 5.2 Application to Polynomial Commitments

As established in the introduction, range proofs over integers committed in groups of unknown order are the dominant bottleneck in GUO-based polynomial commitment schemes [3, 18, 37, 77], causing $O(n^3)$ prover time due to square decomposition. The range proof is a cleanly separable component in both existing constant-proof-size paradigms: the rest of each evaluation protocol already runs in $O(n)$ group operations. Our range proof therefore serves as a drop-in replacement, yielding the complexities in Table 3.

We highlight a structural difference in the range proof requirements between the two paradigms, which affects how our construction applies. In the quotient-polynomial paradigm (Behemoth [77]), the evaluation protocol reduces to proving that a single integer evaluation $\hat{s}$ lies in $[0, \mathcal{B}]$, where $\mathcal{B} = O(q^n)$ is the universal evaluation bound. This is a plain range proof on one $O(n \log q)$-bit integer, and our construction (Section 5.1) applies directly. In the convolution-based paradigm (Dew [3], Dew++ [37]), the evaluation protocol requires certifying that the committed encoding and certain cross-term integers have base-$q$ digits bounded in $[0, p-1]$, amounting to 6 separate range proofs via square decomposition in the original constructions. Our range proof can replace each of these. We note that DewTwo [19] integrates range proofs more tightly into a unified framework with PoKEMath and SIPA, so our range proof does not substitute in the same modular fashion; nevertheless, our approach could replace DewTwo's $\log n$-square decomposition components, potentially yielding $O(1)$ proof size (versus $O(\log \log n)$) with $O(n)$ prover time.

## 6 Optimal Ring Signatures

Ring signatures allow a member of a group to sign a message on behalf of the group without revealing their identity. In a ring signature scheme with $n$ signers, each signer must prove: (1) a "cloaked key" is one valid key among the $n$ committed keys, (2) they know the private key of the corresponding "cloaked key", and (3) these properties hold without revealing which public key is cloaked. Using our morphic accumulator, we construct an optimal ring signature scheme where the "ring" is a set of public keys $S = \{pk_1, \ldots, pk_n\}$ accumulated together with randomizers in a GUO-based accumulator $Acc$. The key innovation is using morphic accumulation to achieve constant-sized signatures with optimal prover complexity. We refer to Appendix C.1 for the definitions and security model of the ring signatures.

### 6.1 Our Construction

We instantiate the linkable ring signature from a Schnorr group and our morphic accumulator. The construction uses the following building blocks:

– A group of unknown order $\mathbb{G}_?$ with an accumulator base $g_{acc}$ and auxiliary proof bases sampled independently according to the chosen instantiation, with zero-knowledge proof protocols **NIZK-PoKE** and **NIZK-PoKEModN** for subset membership and modular equivalence.

– A Schnorr group $\mathbb{G}_p$ of prime order $q$ inside $\mathbb{Z}_p^*$, where $p = 2q+1$ is a safe prime, with two generators $g, h$ whose discrete logarithm relation is unknown. The safe-prime choice ensures $\mathbb{G}_p$ has no small subgroups, so the discrete logarithm problem in $\mathbb{G}_p$ is maximally hard for the given bit-length of $p$. The ambient group $\mathbb{Z}_p^* = \mathbb{G}_p \times \{\pm 1\}$ still contains $-1$, so the verifier checks $T \in \mathbb{G}_p$ (a Legendre-symbol computation); otherwise the same key could also produce valid signatures under the tag $-T \bmod p$. The DDH assumption (Assumption 4) holds in $\mathbb{G}_p$ [79].
– Cryptographic hash function $H : \{0,1\}^* \to [0, 2^{2\lambda})$ (modeled as a random oracle).

We define the randomizer powers $v_j = h^{2^j}$ for $j \in \{0, \ldots, w-1\}$, where $w = \lceil \log_2 q \rceil + 2\lambda$. For each signature, the signer samples $a \xleftarrow{\$} [0, 2^{2\lambda})$ and sets $r' = x + qa$, with $x$ represented in $[0, q)$. Then $0 \le r' < q2^{2\lambda} \le 2^w$, so $r'$ has a binary decomposition using these powers, while $h^{r'} = h^x$ preserves the tag. The Schnorr parameters and honest keys are generated independently of the accumulator setup, with $g, h$ sampled independently subject to $gh \neq 1$. As in the zero-knowledge range proof, the security analysis assumes that, except with negligible probability, the randomizer entries are distinct and non-identity, and distinct subsets of this table have distinct integer products. The formal construction is given in Figure 3. The **NIZK-PoKE** instantiation and the security proof are given in Appendix C.2.

**Efficiency.** The scheme separates into offline and online phases. The offline phase (steps 1–7) uses the ring $R$, the signer's key, and fresh randomness to prepare one proof packet per signature, before its message is known. It requires $O(n+w)$ time to compute the accumulator, subset product, and membership witness, producing constant-sized proofs $(D, \pi_{mem}, \pi_{mod})$. The online phase (steps 8–9) executes when the message $m$ is known and requires only $O(1)$ computation for a standard Schnorr signature. Verification requires $O(n+w)$ time to recompute $Acc$ from $R$, plus $O(1)$ for the proof checks; when $Acc$ is precomputed, verification is $O(1)$. The total signature size is $O(1)$: one element in $\mathbb{G}_p$ (the tag $T$), one element in $\mathbb{G}_?$ (the subset commitment $D$), two constant-sized proofs ($\pi_{mem}$, $\pi_{mod}$), and one Schnorr response $s \in \mathbb{Z}_q$ and one $2\lambda$-bit challenge $c$. It costs $9\mathbb{G}_? + 3\mathbb{Z}_{2\lambda} + \mathbb{Z}_{|p|+2\lambda}$ during offline phase and $1\mathbb{G}_p + 1\mathbb{Z}_q + 1\mathbb{Z}_{2\lambda}$ during online phase. Based on the NIST standard [66], to achieve 112-bit security, we need $p$ to have 2048 bits. For these concrete sizes, we use $\lambda = 128$, giving 256-bit challenges; the online tuple $(T, c, s)$ occupies $256 + 32 + 256 = 544$ bytes, with the overall security limited to 112 bits by the Schnorr group. If we implement the accumulator group with 2048-bit elements of the signed quadratic-residue group $QR_N^+$, the offline proof size is 2688 bytes; for a class-group instantiation satisfying the stated setup requirements, the offline proof size is around 2256 bytes [33], using the 1664-bit class-group encoding estimated for 128-bit security.

## Acknowledgments

We would like to thank the anonymous reviewers for their constructive feedback. Dimitrios Papadopoulos was supported by the Hong Kong RGC under GRF 16213625. Qiang Tang and Jiajun Xin were supported by the ARC Discovery Project(DP250101739), research awards from the Stellar Development Foundation, and Input Output as part of the Cardano Vision 2026 research initiative.

Linkable Ring Signature Protocol

Params $pp$: security parameter $\lambda$, accumulator group $\mathbb{G}_?$ with base $g_{acc}$, Schnorr group $\mathbb{G}_p$ of order $q$ with generators $g, h$ satisfying $gh \neq 1$, randomizer powers $v_j = h^{2^j}$ for $j \in \{0, \ldots, w-1\}$ where $w = \lceil \log_2 q \rceil + 2\lambda$, hash function $H$. Both **Sign** and **Verify** reject rings containing unregistered keys or elements of $\{1_{\mathbb{G}_p}, v_0, \ldots, v_{w-1}\}$; key registration requires a Schnorr proof of knowledge of $x$ such that $VK = g^x$.

**KeyGen**$(1^\lambda)$:

1. Sample $x \xleftarrow{\$} \mathbb{Z}_q$ and compute $VK = g^x$.
2. Output $(VK, SK)$ where $SK = x$.

**Sign**$(SK = x,\ m,\ R = (VK_1, \ldots, VK_n))$:

1. Let $S = \{VK_1, \ldots, VK_n, v_0, \ldots, v_{w-1}\}$, compute $Acc = g_{acc}^{\prod_{s \in S} s}$.
2. Compute $VK = g^x$ and the linkable tag: $T = (g \cdot h)^x$.
3. Sample $a \xleftarrow{\$} [0, 2^{2\lambda})$, set $r' = x + qa$, and decompose $r' = \sum_{i \in I} 2^i$ for $I \subseteq \{0, \ldots, w-1\}$.
4. Identify subset $S' = \{VK, v_{i_1}, \ldots, v_{i_k}\}_{i_j \in I}$ and compute integer product $prod = VK \times \prod_{i \in I} v_i$.
5. Compute subset commitment: $D = g_{acc}^{prod}$.
6. Membership proof: $\pi_{mem} \leftarrow$ **NIZK-PoKE**$(D, Acc;\ \prod_{s \in S \setminus S'} s)$.
7. Modular proof: $\pi_{mod} \leftarrow$ **NIZK-PoKEModN**$(D, p, T;\ prod)$.

**8.** Sample $k \xleftarrow{\$} \mathbb{Z}_q$, compute $R_{sch} = (g \cdot h)^k$.

**9.** Compute $c = H(Acc||T||D||\pi_{mem}||\pi_{mod}||R_{sch}||m)$ and $s = k - c \cdot x \bmod q$.

Output: $\sigma = (D, \pi_{mem}, \pi_{mod}, T, c, s)$.

**Verify**$(R = (VK_1, \ldots, VK_n),\ m,\ \sigma = (D, \pi_{mem}, \pi_{mod}, T, c, s))$:

1. Recompute $S$ and $Acc$ from $R$ as in step 1 of **Sign**.
2. Check $T \in \mathbb{G}_p \setminus \{1_{\mathbb{G}_p}\}$.
3. Check **NIZK-PoKE.Verify**$(D, Acc, \pi_{mem}) = 1$.
4. Check **NIZK-PoKEModN.Verify**$(D, p, T, \pi_{mod}) = 1$.

**5.** Compute $R_{sch} = (g \cdot h)^s \cdot T^c$ and check $c = H(Acc||T||D||\pi_{mem}||\pi_{mod}||R_{sch}||m)$.

Output 1 if all checks pass, 0 otherwise.

**Link**$(\sigma_1,\ \sigma_2,\ m_1,\ m_2)$:

1. Parse $T_1$ from $\sigma_1$ and $T_2$ from $\sigma_2$.

Output 1 if $T_1 = T_2$, and 0 otherwise.

**Fig. 3.** Linkable ring signature from morphic accumulation. Shaded steps are the online phase, requiring only $O(1)$ computation once the message $m$ is known; all other steps can be precomputed when the ring is formed.

## References


1. Abe, M., Haralambiev, K., Ohkubo, M.: Group to group commitments do not shrink. In: EUROCRYPT (2012)
2. Apostol, T.M.: Introduction to analytic number theory. Springer Science & Business Media (2013)
3. Arun, A., Ganesh, C., Lokam, S., Mopuri, T., Sridhar, S.: Dew: transparent constant-sized zksnarks. In: Public Key Cryptography (PKC) (2023)
4. Attema, T., Cramer, R.: Compressed $\sigma$-protocol theory and practical application to plug & play secure algorithmics. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO (2020)
5. Axiotis, K., Backurs, A., Bringmann, K., Jin, C., Nakos, V., Tzamos, C., Wu, H.: Fast and simple modular subset sum. In: Symposium on Simplicity in Algorithms (SOSA) (2021)

6. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: logarithmic-size, no setup—from standard assumptions. In: EUROCRYPT (2019)
7. Benaloh, J., De Mare, M.: One-way accumulators: A decentralized alternative to digital signatures. In: EUROCRYPT (1993)
8. Block, A.R., Holmgren, J., Rosen, A., Rothblum, R.D., Soni, P.: Time-and space-efficient arguments from groups of unknown order. In: CRYPTO (2021)
9. Boneh, D., Bünz, B., Fisch, B.: Batching techniques for accumulators with applications to iops and stateless blockchains. In: CRYPTO (2019)
10. Boneh, D., Datta, T., Fernando, R., Nazirkhanova, K., Tomescu, A.: DekartProof: Efficient vector range proofs and their applications. Cryptology ePrint Archive, Paper 2025/1159 (2025), https://eprint.iacr.org/2025/1159
11. Bootle, J., Elkhiyaoui, K., Hesse, J., Manevich, Y.: Dualdory: logarithmic-verifier linkable ring signatures through preprocessing. In: ESORICS (2022)
12. Bose, P., Das, D., Rangan, C.P.: Constant size ring signature without random oracle. In: ACISP (2015)
13. Boudot, F.: Efficient proofs that a committed number lies in an interval. In: EUROCRYPT (2000)
14. Brent, R.P.: Public key cryptography with a group of unknown order. Tech. rep., Technical Report, Oxford University (2000)
15. Buchmann, J., Williams, H.C.: A key-exchange system based on imaginary quadratic fields. Journal of Cryptology (1988)
16. Bünz, B., Agrawal, S., Zamani, M., Boneh, D.: Zether: Towards privacy in a smart contract world. In: International Conference on Financial Cryptography and Data Security (FC) (2020)
17. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: IEEE Symposium on Security and Privacy, SP (2018)
18. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from dark compilers. In: EUROCRYPT (2020)
19. Bünz, B., Mopuri, T., Shirzad, A., Sridhar, S.: Dewtwo: a transparent pcs with quasi-linear prover, logarithmic verifier and 4.5 kb proofs from falsifiable assumptions. In: CRYPTO (2025)
20. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: CRYPTO (2002)
21. Campanelli, M., Fiore, D., Han, S., Kim, J., Kolonelos, D., Oh, H.: Succinct zero-knowledge batch proofs for set accumulators. In: Proceedings of the ACM CCS (2022)
22. Cardinal, J., Iacono, J.: Modular subset sum, dynamic strings, and zero-sum sets. In: SIAM Symposium on Simplicity in Algorithms (SOSA) (2021)
23. Chaliasos, S., Reif, I., Torralba-Agell, A., Ernstberger, J., Kattis, A., Livshits, B.: Analyzing and benchmarking zk-rollups. In: Advances in Financial Technologies (AFT) (2024)
24. Chor, B., Rivest, R.L.: A knapsack-type public key cryptosystem based on arithmetic in finite fields. IEEE Transactions on Information Theory (1988)
25. Christ, M., Baldimtsi, F., Chalkias, K.K., Maram, D., Roy, A., Wang, J.: SoK: Zero-knowledge range proofs. Cryptology ePrint Archive, Paper 2024/430 (2024), https://eprint.iacr.org/2024/430
26. Chung, H., Han, K., Ju, C., Kim, M., Seo, J.H.: Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. IEEE Access (2022)
27. Coron, J.S., Naccache, D.: Security analysis of the gennaro-halevi-rabin signature scheme. In: EUROCRYPT (2000)
28. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: Improved low-density subset sum algorithms. Computational complexity (1992)
29. Couteau, G., Goudarzi, D., Klooß, M., Reichle, M.: Sharp: Short relaxed range proofs. In: ACM CCS (2022)
30. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: ASIACRYPT (2002)
31. Damgård, I., Jurik, M., Nielsen, J.B.: A generalization of paillier's public-key system with applications to electronic voting. International Journal of Information Security (2010)
32. Damgård, I., Koprowski, M.: Generic lower bounds for root extraction and signature schemes in general groups. In: EUROCRYPT (2002)

33. Dobson, S., Galbraith, S., Smith, B.: Trustless unknown-order groups. Transactions on Mathematical Cryptology (2022)
34. Dobson, S., Galbraith, S.D.: Trustless groups of unknown order with hyperelliptic curves. IACR Cryptol. ePrint Arch. (2020), https://eprint.iacr.org/2020/196
35. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: EUROCRYPT (2004)
36. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO (1986)
37. Fu, S.: Improved constant-sized polynomial commitment schemes without trusted setup. IACR Communications in Cryptology (2025)
38. Gao, R., Wan, Z., Hu, Y., Wang, H.: A succinct range proof for polynomial-based vector commitment. In: Proceedings of the ACM CCS (2024)
39. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: EUROCRYPT (1999)
40. Ghosh, E., Ohrimenko, O., Papadopoulos, D., Tamassia, R., Triandopoulos, N.: Zero-knowledge accumulators and set algebra. In: ASIACRYPT (2016)
41. Groth, J.: Non-interactive zero-knowledge arguments for voting. In: ACNS (2005)
42. Hamdy, S., Möller, B.: Security of cryptosystems based on class groups of imaginary quadratic orders. In: ASIACRYPT (2000)
43. Hildebrand, A., Tenenbaum, G.: Integers without large prime factors. Journal de théorie des nombres de Bordeaux (1993)
44. Horowitz, E., Sahni, S.: Computing partitions with applications to the knapsack problem. J. ACM (1974)
45. Hui, X., Chau, S.C.K.: Llring: logarithmic linkable ring signatures with transparent setup. In: ESORICS (2024)
46. Impagliazzo, R., Naor, M.: Efficient cryptographic schemes provably as secure as subset sum. Journal of cryptology **9**(4), 199–216 (1996)
47. Ji, Y., Chalkias, K.: Generalized proof of liabilities. In: Proceedings of the ACM CCS (2021)
48. Joux, A., Węgrzycki, K.: Improving lagarias-odlyzko algorithm for average-case subset sum: Modular arithmetic approach. arXiv preprint arXiv:2408.16108 (2024)
49. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT (2010)
50. Kemmoe, V.Y., Lysyanskaya, A.: Rsa-based dynamic accumulator without hashing into primes. In: ACM CCS (2024)
51. Krenn, S., Mir, O., Slamanig, D.: Structure-preserving compressing primitives: Vector commitments and accumulators and applications. Cryptology ePrint Archive (2024)
52. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. Journal of the ACM (JACM) (1985)
53. Langowski, S., He, K., Devadas, S.: Vroom: Accelerating (almost all) number-theoretic cryptography using vectorization and the residue number system. In: USENIX Security Symposium (2026)
54. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. In: Theory of Cryptography Conference (TCC) (2021)
55. Li, J., Li, N., Xue, R.: Universal accumulators with efficient nonmembership proofs. In: Applied Cryptography and Network Security ACNS (2007)
56. Lipmaa, H.: Secure accumulators from euclidean rings without trusted setup. In: Applied Cryptography and Network Security ACNS (2012)
57. Lipmaa, H.: Polymath: Groth16 is not the limit. In: CRYPTO (2024)
58. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. IEEE Transactions on Knowledge and Data Engineering (2013)
59. McCurley, K.S.: The discrete logarithm problem. In: Proc. of Symp. in Applied Math. vol. 42, pp. 49–74. USA (1990)
60. Merkle, R.C.: A digital signature based on a conventional encryption function. In: CRYPTO (1987)
61. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed e-cash from bitcoin. In: IEEE Symposium on Security and Privacy, SP (2013)

62. Miller, G.L.: Riemann's hypothesis and tests for primality. In: Proceedings of the seventh annual ACM symposium on Theory of computing (1975)
63. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: CRYPTO (2009)
64. Narula, N., Vasquez, W., Virza, M.: {zkLedger}:{Privacy-Preserving} auditing for distributed ledgers. In: NSDI (2018)
65. Nguyen, L.: Accumulators from bilinear pairings and applications. In: CTRSA (2005)
66. NIST: Sp 800-52 rev. 1. nist special publication (2013). https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-52r1.pdf/ (2013), accessed: 2023-01-02
67. Noether, S.: Ring signature confidential transactions for monero. Cryptology ePrint Archive, Paper 2015/1098 (2015), https://eprint.iacr.org/2015/1098
68. Odlyzko, A.: Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on shamir's fast signature scheme. IEEE Transactions on Information Theory **30**(4), 594–601 (2003)
69. Ozdemir, A., Wahby, R., Whitehat, B., Boneh, D.: Scaling verifiable computation using efficient set accumulators. In: USENIX Security Symposium (2020)
70. Papamanthou, C., Shi, E., Tamassia, R., Yi, K.: Streaming authenticated data structures. In: Eurocrypt (2013)
71. Papamanthou, C., Srinivasan, S., Gailly, N., Hishon-Rezaizadeh, I., Salumets, A., Golemac, S.: Reckle trees: Updatable merkle batch proofs with applications. In: CCS (2024)
72. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology (2000)
73. Qian, Y., Zhang, Y., Chen, X., Papamanthou, C.: Streaming authenticated data structures: Abstraction and implementation. In: Proceedings of CCSW (2014)
74. Reijsbergen, D., Maw, A., Yang, Z., Dinh, T.T.A., Zhou, J.: TAP: transparent and privacy-preserving data services. In: USENIX Security Symposium (2023)
75. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: IEEE Symposium on Security and Privacy, SP (2014)
76. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: CRYPTO (1990)
77. Seres, I.A., Burcsi, P.: Behemoth: transparent polynomial commitment scheme with constant opening proof size and verifier time. In: AFRICACRYPT (2025)
78. Seres, I.A., Burcsi, P., Kutas, P.: How (not) to hash into class groups of imaginary quadratic fields? In: Patra, A. (ed.) CT-RSA (2025)
79. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: EUROCRYPT (1997)
80. Srinivasan, S., Karantaidou, I., Baldimtsi, F., Papamanthou, C.: Batching, aggregation, and zero-knowledge proofs in bilinear accumulators. In: Proceedings of the ACM CCS (2022)
81. Thakur, S.: Constructing hidden order groups using genus three jacobians. Cryptology ePrint Archive, Paper 2020/348 (2020)
82. Thibault, L.T., Sarry, T., Hafid, A.S.: Blockchain scaling using rollups: A comprehensive survey. IEEE Access (2022)
83. Tomescu, A., Bhupatiraju, V., Papadopoulos, D., Papamanthou, C., Triandopoulos, N., Devadas, S.: Transparency logs via append-only authenticated dictionaries. In: Proceedings of the ACM CCS (2019)
84. Tomescu, A., Xia, Y., Newman, Z.: Authenticated dictionaries with cross-incremental proof (dis) aggregation. Cryptology ePrint Archive (2020)
85. Wang, N., Chau, S.C.K.: Flashproofs: Efficient zero-knowledge arguments of range and polynomial evaluation with transparent setup. In: ASIACRYPT (2022)
86. Wang, N., Chau, S.C.K., Liu, D.: Swiftrange: a short and efficient zero-knowledge range argument for confidential transactions and more. In: IEEE Symposium on Security and Privacy, SP (2024)
87. Wang, W., Zhou, L., Yaish, A., Zhang, F., Fisch, B., Livshits, B.: Mechanism design for zk-rollup prover markets. arXiv preprint arXiv:2404.06495 (2024)
88. Wesolowski, B.: Efficient verifiable delay functions. In: EUROCRYPT (2019)
89. Xie, M., Tu, Z., Au, M.H., Fang, J., Wang, X., Jiang, Z.L.: Efficient constant-size linkable ring signatures for ad-hoc rings via pairing-based set membership arguments. In: Proceedings of the ACM CCS (2025)
90. Xin, J., Haghighi, A., Tian, X., Papadopoulos, D.: Notus: Dynamic proofs of liabilities from zero-knowledge RSA accumulators. Cryptology ePrint Archive, Paper 2024/395 (2024)

91. Xin, J., Papadopoulos, D.: "check-before-you-solve"': Verifiable time-lock puzzles. In: IEEE Symposium on Security and Privacy, SP (2025)
92. Xin, J., Tin, S.C.O., Pappas, C., Huang, Y., Papadopoulos, D.: Gryphes: Hybrid proofs for modular snarks with applications to zkrollups. In: PoPETs (2026)
93. Yuen, T.H., Sun, S.F., Liu, J.K., Au, M.H., Esgin, M.F., Zhang, Q., Gu, D.: Ringct 3.0 for blockchain confidential transaction: Shorter size and stronger security. In: International Conference on Financial Cryptography and Data Security (FC) (2020)
94. Zhang, Y., Katz, J., Papamanthou, C.: An expressive (zero-knowledge) set accumulator. In: IEEE European Symposium on Security and Privacy, EuroS&P (2017)

# A Additional Preliminaries

**Order assumption.** Given a group of unknown order $\mathbb{G}_?$, and a random group element $g \in \mathbb{G}_?$, it is hard to find any multiples of its order [62]. It can be formally stated as:

**Assumption 2.** *Order assumption. For every PPT algorithm $\mathcal{A}$ and $GGen(1^\lambda)$ is an algorithm that generates a group $\mathbb{G}$ of unknown order.*

$$\Pr\left[\begin{array}{c} g^e = 1_{\mathbb{G}_?} \\ \wedge e \neq 0 \end{array} \middle| \begin{array}{c} \mathbb{G}_? \leftarrow GGen(1^\lambda) \\ g \xleftarrow{\$} \mathbb{G}_? \\ e \leftarrow \mathcal{A}(\mathbb{G}_?, g) \end{array}\right] < \text{negl}(\lambda).$$

**DL assumption.** The Discrete Logarithm (DL) assumption states that given a cyclic group $\mathbb{G}$ of prime order $q$ and uniformly random elements $g, h \in \mathbb{G}$, it is hard to compute the discrete logarithm $\log_g h$.

**Assumption 3.** *DL assumption. Let $\mathsf{GGen}(1^\lambda)$ be a group generation algorithm that outputs a cyclic group $\mathbb{G}$ of prime order $q$. We say that the DL assumption holds with respect to $\mathbb{G}$ if, for every PPT algorithm $\mathcal{A}$, with $g \xleftarrow{\$} \mathbb{G}$, $\alpha \xleftarrow{\$} \mathbb{Z}_q$ and $h = g^\alpha$,*

$$\Pr[\mathcal{A}(\mathbb{G}, g, h) = \alpha] < \text{negl}(\lambda).$$

**DDH assumption.** The Decisional Diffie–Hellman (DDH) assumption [63] states that the ensembles $\{(\mathbb{G}, g_1, g_2, g_1^r, g_2^r)\}$ and $\{(\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2})\}$ are computationally indistinguishable.

**Assumption 4.** *DDH assumption. Let $\mathsf{GGen}(1^\lambda)$ be a group generation algorithm that outputs a cyclic group $\mathbb{G}$ of prime order $q$. We say that the DDH assumption holds with respect to $\mathsf{GGen}$ if, for every PPT algorithm $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(\mathbb{G}, g_1, g_2, g_1^r, g_2^r) = 0] - \Pr[\mathcal{A}(\mathbb{G}, g_1, g_2, g_1^{r_1}, g_2^{r_2}) = 0]| < \text{negl}(\lambda),$$

*where the elements $g_1, g_2 \in \mathbb{G}$ and $r, r_1, r_2 \in \mathbb{Z}_q$ are chosen independently and uniformly at random.*

**DDH-II assumption.** DDH-II assumption for hidden order groups was first introduced in [21] for a well-spread distribution. A distribution $\mathcal{WS}_{2\lambda}$ with domain $\mathcal{X}_{2\lambda}$ is called well-spread if $\Pr[X = x | X \xleftarrow{\$} \mathcal{WS}_{2\lambda}] \leq 2^{-2\lambda}$ for each $x \in \mathcal{X}_{2\lambda}$. In other words, elements sampled from this distribution are sufficiently random. Because the order of the group is unknown, we denote the upper bound of the group order as maxord($\mathbb{G}$) and the lower bound of the group as minord($\mathbb{G}$). It can be formally stated as follows:

**Assumption 5.** *DDH-II assumption. For every PPT algorithm $\mathcal{A}$, and $GGen(1^\lambda)$ is an algorithm that generates a group $\mathbb{G}$ with hidden order, $g \xleftarrow{\$} \mathbb{G}$. Let $\mathcal{WS}_{2\lambda}$ be a well-spread distribution with domain $\mathcal{X}_{2\lambda} \subseteq [1, \text{minord}(\mathbb{G})]$. Then for any PPT adversary $\mathcal{A}$*

$|\Pr[\mathcal{A}(g^x, g^y, g^{xy}) = 0] - \Pr[\mathcal{A}(g^x, g^y, g^r) = 0]| < \text{negl}(\lambda)$ *where* $x \xleftarrow{\$} \mathcal{WS}_{2\lambda}$, $y, r \xleftarrow{\$} [1, \text{maxord}(\mathbb{G})2^\lambda]$.

# B Range Proof Extensions

## B.1 Zero-Knowledge Range Proof

To achieve zero-knowledge, we extend the protocol to Pedersen-style commitments with a second generator. Let $g_{val}, h_{val} \in \mathbb{G}_{val}$ be two generators with an unknown discrete logarithm relation (i.e., $\log_{g_{val}} h_{val}$ is unknown). The zero-knowledge range proof relation is:

$$\mathcal{R}_{\textbf{ZKRangeProof}} = \{(g_{val}, h_{val}, C \in \mathbb{G}_{val}; x, r \in \mathbb{Z}) : C = g_{val}^x h_{val}^r \wedge x \in [0, 2^n - 1]\}.$$

**Core idea.** We accumulate binary powers of both generators into a single set $\{g_{val}^{2^0}, \ldots, g_{val}^{2^{n-1}}, h_{val}^{2^0}, \ldots, h_{val}^{2^{m-1}}\}$, where $m$ provides sufficient randomness. For an arbitrary input commitment $C = g_{val}^x h_{val}^r$, we apply this idea to $C' = g_{val}^x h_{val}^{r'}$, using a freshly sampled $r' \xleftarrow{\$} [0, 2^m)$. The prover decomposes both $x = \sum_{i \in I} 2^i$ and $r' = \sum_{j \in J} 2^j$, then proves that $C'$ equals the product of the selected $g_{val}$-powers and $h_{val}$-powers. The key insight is that the randomizer $r'$ serves a dual purpose: it randomizes the commitment $C' = g_{val}^x h_{val}^{r'}$, and it simultaneously randomizes the subset visible in the membership proof. For a fixed value $x$, choosing $m$ sufficiently large leaves many possible randomizer subsets even after $C'$ is fixed, providing the randomness used to hide the selected value powers. A proof of knowledge of $\delta$ satisfying $C/C' = h_{val}^\delta$, using $\delta = r - r'$, links the subset proof to the original commitment $C$.

**Randomizer constraint.** The binary decomposition applies to the fresh randomizer $r' \in [0, 2^m)$; the original opening randomizer $r \in \mathbb{Z}$ need not satisfy this bound. Choose $m \geq \lceil \log_2 \text{maxord}(\mathbb{G}_{val}) \rceil + 2\lambda$. For any fixed $x$, each possible commitment $C' = g_{val}^x h_{val}^{r'}$ is consistent with at least $2^{2\lambda}$ choices of $r' \in [0, 2^m)$. Thus, fixing $C'$ still leaves many possible randomizer subsets to randomize the subset selections.

**Modification to domain membership.** The basic **SPAcc** protocol proves $C = g_{val}^{x_z}$ by using **NI-PoKExp**$_{\mathbb{G}_{val}}(g_{val}, C; x_z)$. For the fresh commitments $C'$, we instead prove knowledge of the representation $(x, r')$ using the zero-knowledge variant **NIZK-PoKRep**$((g_{val}, h_{val}), C'; x, r')$. Similarly, all subprotocols in the zero-knowledge range proof use their NIZK variants to enable simulation. This modification ensures the prover knows a valid opening of the commitment without revealing either component.

**Entropic exponent hiding.** We isolate the following hiding property of exponentiation: an integer drawn from a sufficiently spread distribution produces a pseudorandom group element. The distribution need not be uniform, and its exponents may be much larger than the group order. We first state this property as an assumption and then justify it in the generic group model under the Order assumption.

Let $\mathsf{GGen}(1^\lambda)$ output public parameters $pp$ specifying a finite abelian group $\mathbb{G}$ of hidden order $\nu$, a base $g$ sampled according to the chosen instantiation, and a bound $\nu \leq \text{maxord}(\mathbb{G}) \leq 2^{\kappa(\lambda)}$ for a polynomial $\kappa$. We use the well-spread distributions $\mathcal{WS}_{2\lambda}$ defined above: each integer value is sampled with probability at most $2^{-2\lambda}$. For example, the uniform distribution on any set of at least $2^{2\lambda}$ distinct integers is well-spread.

Zero-Knowledge Range Proof Protocol for Relation $\mathcal{R}_{\mathbf{ZKRangeProof}}$

Params $pp$: security parameter $\lambda$, accumulator group $\mathbb{G}_?$ with base $g_{acc}$, value group $\mathbb{G}_{val}$ with generators $g_{val}, h_{val}$, modulus $M$, range parameter $n$, randomizer bits $m$.

**Setup**($pp$):

1. Compute value bases: $u_i = g_{val}^{2^i}$ for $i \in \{0, \ldots, n-1\}$.
2. Compute randomizer bases: $v_j = h_{val}^{2^j}$ for $j \in \{0, \ldots, m-1\}$.
3. Compute accumulated product: $e = \prod_{i=0}^{n-1} u_i \times \prod_{j=0}^{m-1} v_j$ (integer product).
4. Compute accumulator: $Acc = g_{acc}^{e} \in \mathbb{G}_?$.
5. Output: $(Acc, \{u_i\}_{i=0}^{n-1}, \{v_j\}_{j=0}^{m-1})$.

**Prove**($pp, Acc, \{u_i\}, \{v_j\}, C, x, r$) where $C = g_{val}^{x} h_{val}^{r}$:

1. Sample $r' \xleftarrow{\$} [0, 2^m)$ and compute $C' = g_{val}^{x} h_{val}^{r'}$ and $\delta = r - r'$.
2. Decompose $x = \sum_{i \in I} 2^i$ and $r' = \sum_{j \in J} 2^j$.
3. Let $S = \{u_0, \ldots, u_{n-1}, v_0, \ldots, v_{m-1}\}$ and identify subset $S' \subseteq S$ containing $\{u_i\}_{i \in I}$ and $\{v_j\}_{j \in J}$.
4. Compute subset product $prod = \prod_{s \in S'} s$ (integer product of canonicals).
5. Compute intermediate commitment $D = g_{acc}^{prod}$.
6. Generate $\pi_1 \leftarrow \mathbf{NIZK\text{-}PoKE}(D, Acc; \prod_{s \in S \setminus S'} s)$.
7. Generate $\pi_2 \leftarrow \mathbf{NIZK\text{-}PoKRep}((g_{val}, h_{val}), C'; (x, r'))$.
8. Generate $\pi_3 \leftarrow \mathbf{NIZK\text{-}PoKEModN}(D, M, C'; prod)$.
9. Generate $\pi_4 \leftarrow \mathbf{NIZK\text{-}PoKRep}((h_{val}), C/C'; \delta)$.
10. Output: $\pi = (C', D, \pi_1, \pi_2, \pi_3, \pi_4)$.

**Verify**($pp, Acc, C, \pi$):

1. Parse $\pi = (C', D, \pi_1, \pi_2, \pi_3, \pi_4)$.
2. Check $\mathbf{NIZK\text{-}PoKE.Verify}(D, Acc, \pi_1) = 1$.
3. Check $\mathbf{NIZK\text{-}PoKRep.Verify}((g_{val}, h_{val}), C', \pi_2) = 1$.
4. Check $\mathbf{NIZK\text{-}PoKEModN.Verify}(D, M, C', \pi_3) = 1$.
5. Check $\mathbf{NIZK\text{-}PoKRep.Verify}((h_{val}), C/C', \pi_4) = 1$.
6. Output: 1 if all checks pass, 0 otherwise.

**Fig. 4.** Zero-Knowledge Range Proof Protocol.

**Assumption 6** (Entropic exponent hiding (EEH))**.** *Let a PPT sampler output $(z, X)$ independently of $pp \leftarrow \mathsf{GGen}(1^\lambda)$, where $X$ is a nonnegative integer of polynomial bitlength. Suppose that, except with negligible probability over $z$, the conditional distribution of $X$ given $z$ belongs to $\mathcal{WS}_{2\lambda}$. Then $(pp, z, g^X) \approx_c (pp, z, g^R)$, where $R \xleftarrow{\$} [1, 2^\lambda \mathrm{maxord}(\mathbb{G})]$ uses fresh randomness.*

Here $z$ records information retained alongside the exponent, and $\approx_c$ denotes computational indistinguishability against PPT distinguishers. The long interval makes $g^R$ statistically close to a uniform element of $\langle g \rangle$, with distance at most $2^{-\lambda}$.

*Remark 1.* For class groups, we generate the accumulator and auxiliary bases independently using the prime-form sampler of [78], with the sampling randomness included in the public parameters. For the membership proof, we assume subgroup indistinguishability with respect to this sampler, as formulated in Sharp [29]: given $h_0$ and its sampling randomness, a uniform element of $\langle h_0 \rangle$ is computationally indistinguishable from a uniform element of $\mathbb{G}_?$. The membership proof in [9] then provides computational zero-knowledge with the simulator used below.

**Lemma 4 (Generic justification).** *For cyclic groups with a uniform generator and public parameters $pp = (\mathbb{G}, g, \mathrm{maxord}(\mathbb{G}))$, EEH follows from the Order assumption in the model of Definition 1,*

*when the joint sample $(z, X)$ is independent of the random encoding map. More precisely, for every PPT generic distinguisher whose view contains at most $Q$ group-element records, there is a PPT generic algorithm attacking the Order assumption, with polynomial overhead, such that the distinguishing advantage is at most*

$$\binom{Q}{2}\left(2^{-2\lambda} + 2^{-\lambda}\right) + 2(\kappa+2)\varepsilon_{\mathrm{Order}} + 2(\kappa+1)Q\,2^{-\lambda} + \mathrm{negl}(\lambda).$$

*Here $\varepsilon_{\mathrm{Order}}$ is its success probability in the experiment for the Order assumption on the same group family: given a uniform $h \xleftarrow{\$} \mathbb{G}$ with the same public group description and group-oracle access, output an integer $e \neq 0$ such that $h^e = 1$.*

*Proof.* We compare the two experiments through a game that assigns random labels to formal expressions in the challenge exponent. The game differs from a real execution only when distinct expressions evaluate to the same element. Such an equality either determines the exponent as an integer or gives a nonzero multiple of the hidden order.

Let $Z$ denote the challenge exponent and put $T = 2^{\lambda}\mathrm{maxord}(\mathbb{G})$. Count the identity, initial elements, and all oracle replies, including repetitions, toward $Q$. For a fixed group with $\nu > Q$, the game maintains a private table of integer pairs $(a, b)$ representing $g^{a+bZ}$. Equal pairs receive the same label; each new pair receives a uniformly random unused label. The game gives $z$ and the public parameters to the distinguisher and answers queries as follows.

1. Initialize the identity, base, and challenge with $(0, 0)$, $(1, 0)$, and $(0, 1)$, respectively.
2. For products and quotients, add and subtract the corresponding pairs.
3. For a random-element query, sample $s \xleftarrow{\$} [1, T_\nu]$, where $T_\nu = \nu\lfloor T/\nu \rfloor$, and return the label for $(s, 0)$, keeping $s$ private.

The last rule gives an exactly uniform group element. The order-dependent interval is used only to describe this game; the reduction below uses the public interval $[1, T]$. All coefficients have polynomial bitlength, and, conditional on $z$ and the group, the game is independent of $Z$. Its transcript therefore has the same distribution for $Z = X$ and $Z = R$.

Couple each real experiment with this game until two distinct records $(a, b)$ and $(a', b')$ satisfy

$$g^{\Delta} = 1, \qquad \Delta = (a - a') + (b - b')Z.$$

Formal identities already have equal records and cause no discrepancy. If $\Delta = 0$, the distinct records determine the single value $Z = (a' - a)/(b - b')$. Fixing $z$, the group, and the game transcript fixes these records independently of $Z$. Except for the negligible exceptional probability over $z$, this equality has probability at most $2^{-2\lambda}$ for $X$ and at most $1/T \leq 2^{-\lambda}$ for $R$. A union bound over the record pairs gives

$$\binom{Q}{2}\left(2^{-2\lambda} + 2^{-\lambda}\right) + \mathrm{negl}(\lambda).$$

If $\Delta \neq 0$, a collision gives an Order solution. For each choice of $Z$, a reduction receives $h$, sets $g := h$, and runs the joint sampler to obtain $(z, X)$. It uses $Z = X$ in the first case and samples a fresh $Z = R$ in the second. It gives $(pp, z, g^Z)$ to the distinguisher, uses the actual group oracle while tracking the coefficient pairs, and answers random-element queries with $g^s$ for fresh $s \xleftarrow{\$} [1, T]$. At the first collision between distinct records, it outputs $|\Delta|$ if $\Delta \neq 0$, and otherwise aborts. The reduction knows $Z$ and the coefficients, so its output satisfies $h^{|\Delta|} = 1$ without knowing the group order.

For each fixed group, $h$ is a generator with probability

$$\theta = \frac{\varphi(\nu)}{\nu} \geq \frac{1}{1+\log_2 \nu} \geq \frac{1}{1+\kappa}.$$

Indeed, if $\nu$ has $t$ distinct prime factors, the $i$-th smallest is at least $i+1$, so $\varphi(\nu)/\nu \geq \prod_{i=1}^{t} i/(i+1) = 1/(t+1)$ and $t \leq \log_2 \nu$. Conditional on $h$ being a generator, using $[1, T_\nu]$ would give the corresponding real experiment exactly. Replacing each such sample by one from $[1, T]$ changes the execution by at most $Q2^{-\lambda}$. Thus, if the first collision has $\Delta \neq 0$ with probability $p_Z$ for this group, the reduction succeeds with probability at least

$$\frac{p_Z}{1+\kappa} - Q2^{-\lambda}.$$

One PPT Order algorithm runs both collision reductions and also tests the candidate $Q!$, returning any candidate $e \neq 0$ for which $h^e = 1$. Its success probability $\varepsilon_{\mathrm{Order}}$ is at least that of either reduction, and it succeeds whenever $\nu \leq Q$. Averaging the preceding bound over groups, with $p_Z = 0$ for $\nu \leq Q$, bounds each nonzero-collision contribution by $(\kappa+1)(\varepsilon_{\mathrm{Order}} + Q2^{-\lambda})$. Adding the integer-equality bound and at most $\varepsilon_{\mathrm{Order}}$ for small groups in each experiment gives the stated bound. □

*Remark 2 (Relation to DDH-II).* DDH-II also gives an exponent-hiding comparison: projecting the tuple in Assumption 5 onto its last two entries yields $(g^y, (g^y)^x) \approx_c (g^y, g^r)$ for the distributions specified there. That assumption restricts $x$ to $[1, \mathrm{minord}(\mathbb{G})]$ and uses the randomized base $g^y$. Here we use the sampled base $g$ and allow exponents of polynomial bitlength, which can exceed that interval. EEH states this property directly, and Lemma 4 justifies it for cyclic groups with a uniform generator.

*Remark 3 (Group representations).* EEH requires small-index subgroups to be hard to recognize. If a fixed small prime $\ell$ divides the order of $\langle g \rangle$ with non-negligible probability, the well-spread exponent $X = \ell Y$, with $Y \xleftarrow{\$} [1, 2^{2\lambda}]$, always gives $g^X \in \langle g^\ell \rangle$. An efficient membership test for this subgroup would distinguish these powers from a uniform group element of $\langle g \rangle$. For example, on a cyclic subgroup generated by $g \in \mathbb{Z}_N^*$ with Jacobi symbol $-1$, the Jacobi symbol recognizes the even powers of $g$. Such a test uses the concrete representation and is outside the generic group model. For RSA accumulators using EEH, we use the signed quadratic residues $QR_N^+ = \{a \in \mathbb{Z}_N^* : \mathrm{Jacobi}(a, N) = 1\}/\{\pm 1\}$, where $N$ is a product of distinct large safe primes. Its order has no small prime factors, so the small-prime-index consequence above is vacuous.

**Theorem 5.** *The protocol in Figure 4 is a zero-knowledge argument of knowledge for relation* $\mathcal{R}_{\mathbf{ZKRangeProof}}$*, assuming* **NIZK-PoKE**, **NIZK-PoKRep**, *and* **NIZK-PoKEModN** *are zero-knowledge arguments of knowledge and the Order assumption holds for* $\mathbb{G}_?$*, and computing* $\log_{g_{val}} h_{val}$ *is intractable (ensuring the binding of Pedersen commitments in* $\mathbb{G}_{val}$*) in the generic group model and random oracle model. For zero-knowledge, additionally assume that EEH (Assumption 6) holds for* $\mathbb{G}_?$*.*

*Proof.* **Completeness.** We first consider setups in which the $n+m$ table entries are distinct and nonidentity. If $x \in [0, 2^n - 1]$ and $r' \in [0, 2^m - 1]$, both have valid binary decompositions $x = \sum_{i \in I} 2^i$ and $r' = \sum_{j \in J} 2^j$. The subset product satisfies:

$$prod = \prod_{i \in I} u_i \times \prod_{j \in J} v_j, \quad \text{and} \quad prod \bmod M = g_{val}^x \cdot h_{val}^{r'} = C'.$$

Thus $\pi_1$ verifies because $D^{\prod_{s\in S\setminus S'} s} = Acc$, $\pi_2$ verifies because $(x, r')$ is a valid opening of $C'$, and $\pi_3$ verifies because $prod \equiv C' \pmod M$. The fourth proof has the valid witness $\delta = r - r'$ because $C/C' = h_{val}^{\delta}$.

**Knowledge soundness.** By the extractors of the subprotocols:

- From $\pi_1$: extract $y$ such that $D^y = Acc$.
- From $\pi_2$: extract $(\bar{x}, \bar{r})$ such that $C' = g_{val}^{\bar{x}} h_{val}^{\bar{r}}$.
- From $\pi_3$: extract $\bar{prod}$ such that $D = g_{acc}^{\bar{prod}}$ and $\bar{prod} \equiv C' \pmod M$.
- From $\pi_4$: extract $\widehat{\delta}$ such that $C/C' = h_{val}^{\widehat{\delta}}$.

We first establish integer divisibility. The extracted witnesses satisfy $g_{acc}^{\bar{prod}\, y - e} = 1$. A nonzero difference $\bar{prod}\, y - e$ would violate the Order assumption, so $\bar{prod}\, y = e$ and $\bar{prod} \mid e$, except with negligible probability. By the same extraction argument as in Theorem 3 (trial division on $\bar{prod}$ with GEDI ensuring complete decomposition), we can extract subsets $U' \subseteq \{u_i\}$ and $V' \subseteq \{v_j\}$ such that $\bar{prod} = \prod_{u\in U'} u \times \prod_{v\in V'} v$. Let $I$ and $J$ be the index sets of $U'$ and $V'$ respectively. The recovered subsets give $C' = g_{val}^{\sum_{i\in I} 2^i} h_{val}^{\sum_{j\in J} 2^j}$. Combining this equality with $C/C' = h_{val}^{\widehat{\delta}}$, the extractor outputs the opening $\left(\sum_{i\in I} 2^i,\ \sum_{j\in J} 2^j + \widehat{\delta}\right)$ of $C$. Since $I \subseteq \{0, \ldots, n-1\}$ consists of distinct indices, the extracted value lies in $[0, 2^n - 1]$.

**Zero-knowledge.** We prove zero knowledge by simulating the four proofs $\pi_1, \pi_2, \pi_3, \pi_4$ and comparing the jointly generated commitments, following the randomized accumulator technique of [21]. The commitment $C'$ is statistically hiding, and EEH provides the exponent-hiding property needed for $D$. In both executions, the same integer subset product determines the two commitments through $D = g_{acc}^{prod}$ and $C' = prod \bmod M$.

The value parameters and their integer tables are generated independently of the accumulator setup. We assume that, except with negligible probability, the combined table has distinct nonidentity entries and distinct subsets of the randomizer table have distinct integer products. These four proof protocols admit joint zero-knowledge simulation from public inputs with consistent programming of a shared random oracle that preserves previously answered queries.

*Simulator.* Without knowing an opening of $C$, the simulator $\mathcal{S}$ honestly proves that a fresh commitment to zero lies in the range and simulates only its link to $C$. Given $(pp, Acc, C)$ and the public tables, it samples $\tilde{r}' \xleftarrow{\$} [0, 2^m)$ and lets $\tilde{J}$ be its set of nonzero binary positions. It computes $\tilde{C}' = h_{val}^{\tilde{r}'}$, $\tilde{prod} = \prod_{j\in\tilde{J}} v_j$, and $\tilde{D} = g_{acc}^{\tilde{prod}}$. It generates $\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_3$ using the witnesses $e/\tilde{prod}$, $(0, \tilde{r}')$, and $\tilde{prod}$, respectively. It simulates $\tilde{\pi}_4$ for $C/\tilde{C}'$ with base $h_{val}$ and returns $(\tilde{C}', \tilde{D}, \tilde{\pi}_1, \tilde{\pi}_2, \tilde{\pi}_3, \tilde{\pi}_4)$.

We compare the honest execution with $\mathcal{S}$ through three hybrids. A PPT selector may choose the valid statement and witness after setup, before the fresh proof randomizer is sampled. Every hybrid retains the public setup, tables, $C$, and the selector's state.

- $\mathsf{H}_0$ is the honest execution.
- $\mathsf{H}_1$ retains the honestly generated $(C', D)$ and simulates all four proofs.
- $\mathsf{H}_2$ is the output of $\mathcal{S}$: it generates $(\tilde{C}', \tilde{D})$, proves the first three statements honestly, and simulates the fourth.

From $\mathsf{H}_0$ to $\mathsf{H}_1$. We remove the witness-dependent proof messages while retaining their public statements. All four statements are true, so joint zero knowledge of these four proof protocols gives $\mathsf{H}_0 \approx_c \mathsf{H}_1$.

From $\mathsf{H}_1$ to $\mathsf{H}_2$. Following the randomized accumulator technique of [21], we replace $D$ while retaining $C'$, shift $C'$ to a commitment to zero, and restore the simulator's joint distribution. The simulator knows witnesses for its first three statements, and its fourth statement is true because

$h_{val}$ generates $\mathbb{G}_{val}$. We may therefore temporarily simulate its first three proofs, changing the distribution only negligibly. Both experiments then apply the same proof simulators to their public inputs, so it remains to establish the joint indistinguishability of the commitments.

Sample the value parameters and tables as above, choose $r' \xleftarrow{\$} [0, 2^m)$, and set $C_0 = h_{val}^{r'}$ and $X = \prod_{j \in J(r')} v_j$. Let $z$ consist of the value parameters, tables, and $C_0$. For each $z$ with an admissible table, at least $\lfloor 2^m / |\mathbb{G}_{val}| \rfloor \geq 2^{2\lambda}$ equally likely randomizers produce $C_0$, and their integer products are distinct. Thus $X$ has polynomial bitlength and is well-spread conditioned on $z$.

We now construct the matching $C'$ from the EEH challenge without knowing the sampled $r'$. Given $(pp, z, Y)$, the reduction computes $Acc = g_{acc}^e$ and runs the selector on the public setup without revealing $C_0$ or $Y$. For the selected value $x$ and its index set $I$, it outputs $(g_{val}^x C_0, Y^{\prod_{i \in I} u_i})$. When $Y = g_{acc}^X$, this is the honest joint distribution of $(C', D)$. Consequently, a distinguisher for these joint outputs would distinguish the EEH challenge, and replacing $Y$ by $g_{acc}^R$ changes the joint distribution only negligibly.

We use this reduction to replace the randomizer product by an independent exponent. Let $W \xleftarrow{\$} [1, 2^{2\lambda}]$ be independent of the setup and proof randomizer. Applying EEH to $(z, W)$ through the same reduction compares its output with the same $g_{acc}^R$ experiment. Hence $D$ can be replaced by $g_{acc}^{W \prod_{i \in I} u_i}$ while retaining $C'$.

We next remove the selected value factors while keeping $C'$ unchanged. For a uniformly sampled value-table index $i \in \{0, \ldots, n-1\}$ retained in $z$, both $u_i W$ and $W$ are well-spread conditioned on $z$, so EEH gives $(pp, z, g_{acc}^{u_i W}) \approx_c (pp, z, g_{acc}^W)$ through the same random-exponent distribution. The factors are removed one at a time because the selected set $I$ may depend on the accumulator setup, whereas an EEH source must be independent of it. For each index, the reduction first runs the selector; if the index is selected, it raises the challenge to the product of the other selected factors not yet removed. Otherwise, it ignores the challenge and samples the identical distribution on both sides. These polynomially many comparisons establish $(C', D) \approx_c (C', g_{acc}^W)$, with the common public inputs and selector state retained.

We now use the independence of $W$ to shift $C'$ to a commitment to zero. For each fixed setup and selector output, the statistical distance between $g_{val}^x h_{val}^{r'}$ and $h_{val}^{r'}$ is at most $|\mathbb{G}_{val}|/2^m \leq 2^{-2\lambda}$. Since $W$ is independent of $r'$, the same bound holds when $g_{acc}^W$ is included.

We recover the simulator's commitments by restoring their shared randomizer. Reverse the joint EEH comparison using the reduction output $(C_0, Y)$. This replaces $g_{acc}^W$ by $g_{acc}^{\prod_{j \in J} v_j}$ while retaining the matching $h_{val}^{r'}$. The resulting joint distribution is exactly that of $(\tilde{C}', \tilde{D})$.

We finish by transferring this comparison to the complete proof. Applying the same simulators and restoring the first three honest proofs at the simulator endpoint gives $\mathsf{H}_1 \approx_c \mathsf{H}_2$. Together with $\mathsf{H}_0 \approx_c \mathsf{H}_1$, this establishes zero knowledge.

□

### B.2 Vector Range Proofs

The range proof extends to vectors of committed values with heterogeneous ranges. Given generators $g_1, \ldots, g_\ell, h \in \mathbb{G}_{val}$ sampled independently at random and fixed during setup, with unknown discrete-logarithm relations between them, the vector range proof relation is:

$$\mathcal{R}_{\mathbf{VecRangeProof}} = \{(g_1, \ldots, g_\ell, h, C \in \mathbb{G}_{val}; x_1, \ldots, x_\ell, r \in \mathbb{Z}) : \\ C = \left( \prod_{k=1}^{\ell} g_k^{x_k} \right) h^r \wedge x_k \in [0, 2^{n_k} - 1] \text{ for all } k\}.$$

Here $r$ is an unrestricted randomizer. In the prime-order setting, $\mathbb{G}_{val}$ is a subgroup of $\mathbb{Z}_p^*$ of odd prime order, and the verifier checks $C, Z \in \mathbb{G}_{val}$; since $-1 \notin \mathbb{G}_{val}$, negative extracted divisors are ruled out. In the RSA setting, we take $\mathbb{G}_{val} = \mathbb{Z}_N^*/\{\pm 1\}$, the standard choice for **PoKE**-type arguments in RSA groups [9]: integers in $[N]$ represent their classes, with $a$ and $N-a$ representing the same element, so equalities in $\mathbb{G}_{val}$ hold up to sign. The construction accumulates prime encodings, defined below, of the binary powers $\{g_k^{2^i}\}_{i=0}^{n_k-1}$ of all $\ell$ generators into a single accumulator. The prover decomposes each $x_k$ into its binary representation and links the selected encodings to $C$ through a scalar exponent-knowledge proof under $h$.

**The representation-proof bottleneck.** A direct extension of Figure 4 would use **NI-PoKRep** to prove knowledge of the entire vector opening. Such representation proofs exist for different bases in prime-order groups, for example through compressed sigma protocols [4]. However, their discrete-logarithm-based instantiation has logarithmic communication and linear verification costs in $\ell$; we are not aware of an instantiation in the transparent setting that simultaneously achieves linear prover time, constant proof size, and constant verification time as $\ell$ grows. Thus, simply substituting a vector **NI-PoKRep** does not establish the desired efficiency.

We first revisit why **NI-PoKRep** is needed. As the extraction argument in Theorem 3 shows, the membership and modular-equivalence proofs establish that an extracted integer divides the accumulated product and reduces to the claimed group element modulo $M$. However, divisibility alone does not imply subset membership: when the accumulated integers are composite, an adversary may select their factors rather than whole elements. The domain-knowledge proof $\pi_2$, together with GEDI, is used to exclude this possibility. This suggests an alternative approach: eliminate the ambiguity during transparent setup by ensuring that all accumulated integers are distinct primes. Every positive divisor of their product then corresponds to a unique subset of accumulated elements.

**Prime encoding during transparent preprocessing.** We use the fixed public base $h$ to randomize the binary powers before accumulation. For each bit position $(k, i)$, find an integer offset $\rho_{k,i}$ such that $p_{k,i} = g_k^{2^i} h^{\rho_{k,i}}$ is a prime distinct from all previously selected primes. The offsets $\rho_{k,i}$ are public, and we accumulate these encodings (canonical representations of each $p_{k,i}$) into an accumulator:

$$Acc = g_{acc}^{P}, \qquad P = \prod_{k=1}^{\ell} \prod_{i=0}^{n_k-1} p_{k,i}.$$

The encoding relations, primality, and distinctness are publicly checkable. Efficient preprocessing assumes that the chosen group representation and offset-sampling procedure yield suitable primes. The key observation is that these encodings retain a known relationship to the original binary powers, while their primality removes the ambiguity between selecting elements and selecting their factors.

Let $T$ be the subset selected by the binary decompositions of $x_1, \ldots, x_\ell$, and define

$$e = \prod_{(k,i)\in T} p_{k,i}, \qquad \rho_T = \sum_{(k,i)\in T} \rho_{k,i}, \qquad Z = e \bmod M.$$

For a commitment $C = (\prod_{k=1}^{\ell} g_k^{x_k}) h^r$, the selected product satisfies

$$Z = \left( \prod_{k=1}^{\ell} g_k^{x_k} \right) h^{\rho_T} = C\, h^{\rho_T - r}.$$

Consequently, the contributions from all value bases cancel in $Z/C$, leaving a single exponent under $h$. The prover links $Z$ to the selected product using the membership and modular-equivalence arguments, and supplies

$$\pi_2 \leftarrow \textbf{NI-PoKExp}_{\mathbb{G}_{val}}(h, Z/C; \rho_T - r).$$

This scalar exponent-knowledge proof is instantiated by Schnorr in the prime-order setting and by **NI-PoKE** in the hidden-order setting.

**Extraction intuition.** Let $D$ denote the intermediate accumulator-group commitment. The subprotocol extractors recover integers $\bar{e}, y, \delta$ satisfying

$$D = g_{acc}^{\bar{e}}, \qquad D^y = Acc, \qquad \bar{e} \bmod M = Z, \qquad Z/C = h^{\delta}.$$

Since $Acc = g_{acc}^{P}$, we have $g_{acc}^{\bar{e}y-P} = 1$. The Order assumption implies $\bar{e}y = P$ except with negligible probability. Since $P$ is a product of distinct primes, trial division of $|\bar{e}|$ by the public primes recovers a unique subset $T$ with $|\bar{e}| = d = \prod_{(k,i)\in T} p_{k,i}$. In the prime-order setting, $\bar{e} = -d$ would give $Z(d \bmod M)^{-1} = -1$, contradicting $Z \in \mathbb{G}_{val}$ and $-1 \notin \mathbb{G}_{val}$; in the RSA setting, $\bar{e} = \pm d$ give the same element of $\mathbb{G}_{val}$. In both cases, $Z = d \bmod M$ holds in $\mathbb{G}_{val}$, so the extracted exponent determines a genuine subset product. For this subset $T$, define

$$\widehat{x}_k = \sum_{i:(k,i)\in T} 2^i, \qquad \widehat{r} = \sum_{(k,i)\in T} \rho_{k,i} - \delta.$$

The selected primes encode the values $\widehat{x}_k$ together with the public offsets associated with $T$. Combining their encoding relations with the extracted equation $Z/C = h^{\delta}$ gives

$$C = Zh^{-\delta} = \left(\prod_{k=1}^{\ell} g_k^{\widehat{x}_k}\right) h^{\sum_{(k,i)\in T} \rho_{k,i} - \delta} = \left(\prod_{k=1}^{\ell} g_k^{\widehat{x}_k}\right) h^{\widehat{r}}.$$

Since $T$ selects each bit position at most once, every $\widehat{x}_k$ lies in $[0, 2^{n_k} - 1]$. Thus, the extractor obtains a bounded vector opening together with the randomizer $\widehat{r} = \sum_{(k,i)\in T} \rho_{k,i} - \delta$. This replaces the vector **NI-PoKRep** with a single scalar exponent-knowledge proof. The extraction uses only integer exponent arithmetic and therefore does not require knowledge of the value group's order.

**Proof Aggregation.** Our construction allows $k$ separate commitments $C_i = g_i^{x_i} h^{r_i}$, under independently generated bases $g_1, \ldots, g_k, h$, to share one range-proof component. Each $C_i$ requires a proof of knowledge of its opening under $(g_i, h)$. A prover knowing all openings applies the vector range proof to $C = \prod_i C_i$; assuming computational binding of the vector commitment, these proofs jointly establish the prescribed range for every original commitment. After validating the shared preprocessing, verifying the opening proofs, and computing $C$, the remaining range-proof component has $O(1)$ proof size and verification cost, with $O(\sum_i n_i)$ prover work in the preceding cost model. Including the constant-size, constant-verification opening proofs, the total proof size and verification cost are $O(k)$. The aggregation entry in Table 1 refers to this shared range-proof component with per-commitment opening-proof overhead.

# C Linkable Ring Signature Definitions and Security Analysis

## C.1 Definitions and Security Model

We formalize the syntax and security requirements below, following the security model of [6,58].

**Definition 7 (Linkable Ring Signature).** *A linkable ring signature scheme LRS consists of four PPT algorithms* (**KeyGen**, **Sign**, **Verify**, **Link**)*:*

– $(VK, SK) \leftarrow$ **KeyGen**$(1^{\lambda})$*: Takes as input the security parameter* $1^{\lambda}$ *and outputs a pair* $(VK, SK)$ *of verification and signing keys.*

– $\sigma \leftarrow \mathbf{Sign}(SK, m, R)$: *Takes as input a signing key* $SK$*, a message* $m \in \mathcal{M}_\lambda$*, and a ring* $R = (VK_1, \ldots, VK_\ell)$*, and outputs a signature* $\sigma$*.*
– $0/1 \leftarrow \mathbf{Verify}(R, m, \sigma)$: *Takes as input a ring* $R = (VK_1, \ldots, VK_\ell)$*, a message* $m \in \mathcal{M}_\lambda$*, and a signature* $\sigma$*, and outputs either* 0 *or* 1*.*
– $0/1 \leftarrow \mathbf{Link}(\sigma_1, \sigma_2, m_1, m_2)$: *Takes as input two signatures and two messages, and outputs either* 0 *or* 1*.*

A ring is a finite set of distinct verification keys; tuple notation lists its elements, and ring equality in all security experiments is set equality. We require five properties: *correctness*, *linkability*, *linkable anonymity*, *non-slanderability*, and standard *unforgeability*.

**Definition 8 (Correctness).** *We say that a linkable ring signature scheme* $LRS = (\mathbf{KeyGen}, \mathbf{Sign}, \mathbf{Verify}, \mathbf{Link})$ *is* correct *if for all* $\lambda \in \mathbb{N}$*, all* $\ell = \mathrm{poly}(\lambda)$*, all* $i^* \in [\ell]$*, and all messages* $m \in \mathcal{M}_\lambda$*, given that* $(VK_i, SK_i) \leftarrow \mathbf{KeyGen}(1^\lambda)$ *for* $i \in [\ell]$ *and* $\sigma \leftarrow \mathbf{Sign}(SK_{i^*}, m, R)$ *where* $R = (VK_1, \ldots, VK_\ell)$*, it holds that*

$$\Pr[\mathbf{Verify}(R, m, \sigma) = 1] = 1 - \mathrm{negl}(\lambda).$$

Linkability requires that an adversary who corrupts at most one key cannot produce two valid signatures that do not link with each other. This notion is known as *weak linkability* [58], achieved by [11, 45, 58], in contrast to *strong linkability* [6], which allows the adversary to control $\ell$ keys and requires that it cannot produce $\ell + 1$ pairwise unlinkable signatures.

**Definition 9 (Linkability).** *We say that* $LRS$ *satisfies* linkability *if for every* $\ell = \mathrm{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, the advantage* $\mathrm{Adv}_{LRS\text{-}Link}(\mathcal{A}) = \Pr[\mathcal{A}\ wins]$ *is negligible in the following game. The experiment generates* $(VK_i, SK_i) \leftarrow \mathbf{KeyGen}(1^\lambda)$ *for* $i = 1, \ldots, \ell$*, sets* $\mathcal{VK} = \{VK_1, \ldots, VK_\ell\}$*, and provides* $\mathcal{VK}$ *to* $\mathcal{A}$*.* $\mathcal{A}$ *may adaptively query a signing oracle* $(\mathbf{sign}, VK_i, m, R)$ *and may corrupt a single key* $(\mathbf{corrupt}, VK_i)$ *to learn* $SK_i$*.* $\mathcal{A}$ *outputs* $(R_1, m_1, \sigma_1, R_2, m_2, \sigma_2)$ *and wins if:*

– $R_1, R_2 \subseteq \mathcal{VK}$ *and both* $\mathbf{Verify}(R_j, m_j, \sigma_j) = 1$*;*
– *the corruption oracle was queried at most once;*
– *neither* $\sigma_1$ *nor* $\sigma_2$ *was obtained from the signing oracle; and*
– $\mathbf{Link}(\sigma_1, \sigma_2, m_1, m_2) = 0$*.*

Linkable anonymity requires that even though signatures by the same signer are publicly linkable, it remains infeasible to determine which ring member produced them. The experiment permutes the roles of two challenge keys according to a hidden bit $b$; the adversary, who may corrupt all other keys and observe arbitrarily many signatures, must detect the swap. The condition that both challenge keys must appear in every queried ring (or neither) prevents trivial wins where the adversary requests a signature from a ring containing only one challenge key.

**Definition 10 (Linkable Anonymity).** *We say that* $LRS$ *is* linkably anonymous *if for every* $\ell = \mathrm{poly}(\lambda)$ *and every PPT adversary* $\mathcal{A}$*, the advantage* $\mathrm{Adv}_{LRS\text{-}Anon}(\mathcal{A}) = \left|\Pr[\mathrm{Exp}_{LRS\text{-}Anon}(\mathcal{A}) = 1] - \frac{1}{2}\right|$ *is negligible, where the experiment* $\mathrm{Exp}_{LRS\text{-}Anon}(\mathcal{A})$ *proceeds as follows.*

1. *For* $i = 1, \ldots, \ell$*, the experiment generates* $(VK_i, SK_i) \leftarrow \mathbf{KeyGen}(1^\lambda)$ *and samples a bit* $b \xleftarrow{\$} \{0, 1\}$*.*
2. *The experiment provides* $\mathcal{VK} = \{VK_1, \ldots, VK_\ell\}$ *to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *selects a set of verification keys* $\mathcal{U} \subset \mathcal{VK}$ *and two challenge keys* $VK_0^*, VK_1^* \in \mathcal{VK} \setminus \mathcal{U}$*. Let* $SK_0^*, SK_1^*$ *be the corresponding secret keys. The experiment returns* $SK_i$ *for all* $VK_i \in \mathcal{U}$ *(i.e., the adversary corrupts keys in* $\mathcal{U}$*, but not the two challenge keys).*

4. *$\mathcal{A}$ may adaptively query for signatures by specifying a ring $R$ containing some $VK \in \mathcal{VK} \setminus \mathcal{U}$ and a message $m$:*
   - *If exactly one of $VK_0^*, VK_1^*$ is in $R$ (but not both), the experiment returns a uniformly random bit and aborts.*
   - *If $VK \notin \{VK_0^*, VK_1^*\}$, the experiment returns $\sigma \leftarrow \mathbf{Sign}(SK, m, R)$ where $SK$ corresponds to $VK$.*
   - *If $VK = VK_0^*$, the experiment returns $\sigma \leftarrow \mathbf{Sign}(SK_b^*, m, R)$.*
   - *If $VK = VK_1^*$, the experiment returns $\sigma \leftarrow \mathbf{Sign}(SK_{1-b}^*, m, R)$.*
5. *$\mathcal{A}$ outputs a guess $\hat{b} \in \{0,1\}$. The experiment outputs 1 if $\hat{b} = b$, and 0 otherwise.*

Non-slanderability (also called non-frameability [6]) guarantees that no adversary can forge a signature that links with an honest user's signature. The two-stage structure strengthens this guarantee: the adversary must commit its forged signature $\sigma^*$ in Stage 1 *before* all secret keys are revealed in Stage 2, then produce a second signature $\sigma^\dagger$ that links with $\sigma^*$. The adversary wins only if both signatures verify, the ring of $\sigma^*$ consists of uncorrupted keys, and $\sigma^*$ was not obtained from the signing oracle.

**Definition 11 (Non-Slanderability).** *We say that LRS is* non-slanderable *if for every $\ell = \mathrm{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{\text{LRS-NS}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\text{LRS-NS}}(\mathcal{A}) = 1]$ is negligible, where the experiment $\mathrm{Exp}_{\text{LRS-NS}}(\mathcal{A})$ proceeds as follows.*
Stage 1 *(adversary does not know honest secret keys):*

1. *For $i = 1, \ldots, \ell$, the experiment generates $(VK_i, SK_i) \leftarrow \mathbf{KeyGen}(1^\lambda)$. It sets $\mathcal{VK} = \{VK_1, \ldots, VK_\ell\}$ and initializes $\mathcal{C} = \emptyset$.*
2. *The experiment provides $VK_1, \ldots, VK_\ell$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ may adaptively make the following queries:*
   - *$(\mathbf{sign}, VK_i, m, R)$: The experiment checks that $VK_i \in R$, and if so computes $\sigma \leftarrow \mathbf{Sign}(SK_i, m, R)$ and returns $\sigma$ to $\mathcal{A}$. The ring $R$ may contain adversarially generated verification keys.*
   - *$(\mathbf{corrupt}, VK_i)$: The experiment adds $VK_i$ to $\mathcal{C}$ and returns $SK_i$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ commits to $(R^*, m^*, \sigma^*)$.*

Stage 2 *(all secret keys revealed after $\sigma^*$ is committed):*

5. *The experiment reveals all secret keys $SK_1, \ldots, SK_\ell$ to $\mathcal{A}$.*
6. *$\mathcal{A}$ outputs $(R^\dagger, m^\dagger, \sigma^\dagger)$. The experiment returns 1 if all of the following hold:*
   - *$\mathbf{Verify}(R^*, m^*, \sigma^*) = 1$ and $\mathbf{Verify}(R^\dagger, m^\dagger, \sigma^\dagger) = 1$.*
   - *$R^* \subseteq \mathcal{VK}$ and for all $VK_i \in R^*$, we have $VK_i \notin \mathcal{C}$ (all keys in $R^*$ are not corrupted at the time $\sigma^*$ was committed).*
   - *$\sigma^*$ was not obtained from the signing oracle.*
   - *$\mathbf{Link}(\sigma^*, \sigma^\dagger, m^*, m^\dagger) = 1$.*

   *Otherwise, the experiment returns 0.*

Unforgeability requires that no adversary, even one who can corrupt some ring members and obtain signatures from others, can produce a valid signature on a new message-ring pair for a ring of entirely honest keys. The freshness condition (the adversary never queried the signing oracle with the same message and ring) ensures the forgery is genuinely new.

**Definition 12 (Unforgeability).** *We say that LRS is* unforgeable with respect to insider corruption *if for every $\ell = \mathrm{poly}(\lambda)$ and every PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_{\text{LRS-Unf}}(\mathcal{A}) = \Pr[\mathrm{Exp}_{\text{LRS-Unf}}(\mathcal{A}) = 1]$ is negligible, where the experiment $\mathrm{Exp}_{\text{LRS-Unf}}(\mathcal{A})$ proceeds as follows.*

1. *For $i = 1, \ldots, \ell$, the experiment generates $(VK_i, SK_i) \leftarrow \mathbf{KeyGen}(1^\lambda)$. It sets $\mathcal{VK} = \{VK_1, \ldots, VK_\ell\}$ and initializes $\mathcal{C} = \emptyset$.*
2. *The experiment provides $VK_1, \ldots, VK_\ell$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ may adaptively make the following queries:*
   - *$(\mathbf{sign}, VK_i, m, R)$: The experiment checks that $VK_i \in R$, and if so computes $\sigma \leftarrow \mathbf{Sign}(SK_i, m, R)$ and returns $\sigma$ to $\mathcal{A}$.*
   - *$(\mathbf{corrupt}, VK_i)$: The experiment adds $VK_i$ to $\mathcal{C}$ and returns $SK_i$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $(R^*, m^*, \sigma^*)$. The experiment outputs 1 if all of the following hold:*
   - *$R^* \subseteq \mathcal{VK} \setminus \mathcal{C}$ (none of the keys in $R^*$ were corrupted).*
   - *$\mathcal{A}$ never made a signing query of the form $(\mathbf{sign}, \cdot, m^*, R^*)$.*
   - *$\mathbf{Verify}(R^*, m^*, \sigma^*) = 1$.*

   *Otherwise, the experiment outputs 0.*

### C.2 Security Analysis

Throughout the paper, **NIZK-PoKE** uses auxiliary bases $g_0, h_0$ sampled independently of each other and of the accumulator base. In the cyclic-group setup, $g_{acc}, g_0, h_0$ are independently sampled within negligible statistical distance of uniform generators. The public parameters $pp$ include these bases and an integer $B_{\mathrm{mem}} \geq 2^{2\lambda} L \,\mathrm{maxord}(\mathbb{G}_?)$, where $L$ bounds the prime challenge. The honest prover samples $k, \rho_y, \rho_k$ independently and uniformly from $[-B_{\mathrm{mem}}, B_{\mathrm{mem}}]$. For an invocation $\mathbf{NIZK\text{-}PoKE}(D, Acc; y)$, the proof contains the complete first message $(z, A_g, A_u)$ of the underlying interactive protocol [9]. The prover and verifier compute

$$(c_{\mathrm{mem}}, \ell_{\mathrm{mem}}) = H_{\mathrm{mem}}\big(\mathrm{enc}(pp, D, Acc, z, A_g, A_u)\big).$$

Here $H_{\mathrm{mem}}$ is a random oracle independent of all other random oracles, with uniform output in $[0, 2^{2\lambda}) \times \mathrm{Primes}(2\lambda)$, and enc is an injective encoding.

**Relation to SPAcc.** The ring signature construction uses the proofs of **SPAcc** (Figure 1) but replaces the domain membership proof $\pi_2$ with the Schnorr signature. Recall that **SPAcc** requires three proofs: $\pi_1$ (subset membership via **NIZK-PoKE**), $\pi_2$ (domain membership via **NI-PoKExp** proving $z = g_{val}^{x_z}$), and $\pi_3$ (modular equivalence via **NIZK-PoKEModN**). In our construction, $\pi_{mem}$ corresponds to $\pi_1$ and $\pi_{mod}$ corresponds to $\pi_3$. The role of $\pi_2$ is fulfilled by the Schnorr signature, which is a zero-knowledge proof of knowledge of $x$ such that $T = (g \cdot h)^x$ [76], providing the same extraction guarantee as $\pi_2$ with $g_{val} = g \cdot h$. The security analysis below uses the knowledge soundness of each proof and assumes that the two accumulator proofs and the Schnorr signature admit joint zero-knowledge simulation from public inputs over polynomially many adaptive requests. The simulators run in polynomial time on every well-formed input used below, preserve previously answered random-oracle queries, and abort with negligible probability.

**Correctness.** For an honest signer with secret key $x$ and public key $VK = g^x$, the tag $T = (g \cdot h)^x \neq 1_{\mathbb{G}_p}$ with overwhelming probability (since $x \xleftarrow{\$} \mathbb{Z}_q$ and $T = 1$ iff $x = 0$, which occurs with probability $1/q = \mathrm{negl}(\lambda)$), so the non-triviality check passes. The subset $S' = \{VK, v_{i_1}, \ldots, v_{i_k}\}_{i_j \in I}$ consists of accumulated elements, so the membership proof $\pi_{mem}$ verifies. The integer product satisfies $prod \bmod p = g^x \cdot \prod_{i \in I} h^{2^i} = g^x h^{r'} = g^x h^x = (g \cdot h)^x = T$, since $r' \equiv x \pmod q$, so $prod \equiv T \pmod p$ and $\pi_{mod}$ verifies. The Schnorr verification reconstructs $R_{sch} = (g \cdot h)^s \cdot T^c = (g \cdot h)^{k - cx} \cdot (g \cdot h)^{cx} = (g \cdot h)^k$, matching the original commitment. For linking, two signatures by the same signer produce the same tag $T = (g \cdot h)^x$, so **Link** outputs 1. Conversely, two different signers with different secret keys produce distinct tags, so **Link** outputs 0.

**Theorem 6 (Unforgeability and Non-Slanderability).** *The construction in Figure 3 is unforgeable (Definition 12) and non-slanderable (Definition 11) in the generic group model for $\mathbb{G}_?$ and the random oracle model, assuming the DL assumption (Assumption 3) and sEUF-CMA security of Schnorr signatures [72, 76] hold in $\mathbb{G}_p$, the argument-of-knowledge property of the underlying interactive PoKE, the knowledge soundness of* **NIZK-PoKEModN***, EEH (Assumption 6) and the Order assumption (Assumption 2) for $\mathbb{G}_?$, and the GEDI property (Definition 6) holds for discrete logarithm encodings in $\mathbb{G}_p$.*

*Proof.* **Unforgeability.** *Intuition.* Recall that the sEUF-CMA experiment for Schnorr signatures provides the adversary with a generator $g'$ of $\mathbb{G}_p$, a public key $T^* = (g')^{x^*}$ for an unknown $x^*$, and a signing oracle $\mathcal{O}_{Sch}$ that produces signatures under $T^*$ with base $g'$. The adversary wins if it outputs a valid message-signature pair that was not returned by $\mathcal{O}_{Sch}$. Our reduction embeds $T^*$ as one signer's linkable tag. The membership and modular proofs ensure that, except with negligible probability, the forgery uses a genuine ring member's tag. When the guessed index $i^*$ identifies that member, the forgery yields a Schnorr forgery under $T^*$.
*Reduction setup.* The outside adversary $\mathcal{B}$ receives a Schnorr sEUF-CMA challenge: a generator $g'$, a public key $T^* = (g')^{x^*}$ (for unknown $x^*$), and a signing oracle $\mathcal{O}_{Sch}$. $\mathcal{B}$ samples $\alpha \xleftarrow{\$} \mathbb{Z}_q \setminus \{0, -1\}$ and sets $g = (g')^{(1+\alpha)^{-1}}$ and $h = g^{\alpha}$, so that $g \cdot h = g^{1+\alpha} = g'$. This gives $\mathcal{B}$ the ring signature public parameters $(g, h)$ with $\log_g h = \alpha$ known to $\mathcal{B}$ but hidden from $\mathcal{A}$ behind the discrete logarithm problem. $\mathcal{B}$ computes the corresponding verification key $VK_{i^*} = (T^*)^{(1+\alpha)^{-1}} = g^{x^*}$. It guesses $i^* \xleftarrow{\$} [\ell]$ uniformly among the $\ell$ honest keys and sets $VK_{i^*}$ accordingly. For all $i \neq i^*$, $\mathcal{B}$ generates $(VK_i, SK_i)$ honestly.
*Oracle simulation.*

- *Corruption oracle* $(\mathbf{corrupt}, VK_i)$: Return $SK_i$ if $i \neq i^*$, and abort otherwise.
- *Signing oracle* $(\mathbf{sign}, VK_i, m, R)$: $\mathcal{B}$ computes $Acc$ from $R$ and uses the genuine tag $T = (gh)^{SK_i}$ if $i \neq i^*$, or $T = T^*$ otherwise. It samples $a \xleftarrow{\$} [0, 2^{2\lambda})$, sets $r = T + pa$ and $D = g_{acc}^{r}$, and generates $\pi_{mod}$ with witness $r$. It records $(D, Acc, T, r)$, the signer and the modular prover's state. Without a membership witness, it simulates $\pi_{mem}$ by sampling $(c_{\text{mem}}, \ell_{\text{mem}})$ from its prescribed distribution and, independently, $\eta, q_x, q_\rho \leftarrow [-B_{\text{mem}}, B_{\text{mem}}]$ and $r_x, r_\rho \leftarrow [0, \ell_{\text{mem}})$, then setting $z = h_0^{\eta}, Q_g = g_0^{q_x} h_0^{q_\rho}, Q_u = D^{q_x}, A_g = Q_g^{\ell_{\text{mem}}} g_0^{r_x} h_0^{r_\rho} z^{-c_{\text{mem}}}, A_u = Q_u^{\ell_{\text{mem}}} D^{r_x} Acc^{-c_{\text{mem}}}$. It programs $H_{\text{mem}}(\text{enc}(pp, D, Acc, z, A_g, A_u)) = (c_{\text{mem}}, \ell_{\text{mem}})$, aborting if that input is already defined.
  For Schnorr, it signs $\mu = Acc||T||D||\pi_{mem}||\pi_{mod}||m$ using $\mathcal{O}_{Sch}$ if $i = i^*$ and $SK_i$ otherwise.

*Simulation analysis.* The membership simulator produces an accepting transcript for every well-formed statement. For a true statement $D^y = Acc$ and fixed $(c_{\text{mem}}, \ell_{\text{mem}})$, let $\nu = |\mathbb{G}_?|$. The honest prover's compressed transcript depends only on $(k + c_{\text{mem}} y) \bmod (\ell_{\text{mem}}\nu)$, $(\rho_k + c_{\text{mem}}\rho_y) \bmod (\ell_{\text{mem}}\nu)$ and $\rho_y \bmod \nu$. Before programming, any PPT distinguisher has advantage at most $(2\ell_{\text{mem}} + 4)\nu/(2B_{\text{mem}}+1) + \text{negl}(\lambda)$ in distinguishing the honest and simulated transcripts.[8] The programming-conflict probability is at most $Q_H/\nu + \text{negl}(\lambda)$, where $Q_H$ counts earlier membership-oracle inputs. Both bounds are negligible under the chosen $B_{\text{mem}}$ and the stated assumptions.

For the signing view, apply Games 0–2 of Theorem 7 to all honest keys. With the proofs still simulated, the same signing-request and key-index hybrids apply EEH to replace $D$ by $g_{acc}^{T+pa}$: the Schnorr parameters and key data are sampled independently of the accumulator setup, and, conditional on these data and the designated key index, the $2^{2\lambda}$ equally likely values of $a$ give distinct

[8] For class groups, we apply the subgroup-indistinguishability argument of [29] to $z$ after the response comparison, obtaining computational zero knowledge. The same uniform-$z$ hybrid also bounds programming conflicts.

polynomial-bitlength exponents. Zero knowledge restores honest generation of $\pi_{mod}$ with witness $r$ and the prescribed Schnorr signatures. The auxiliary DL reductions use the same accumulator simulation.

*Forgery extraction.* $\mathcal{A}$ outputs a forgery $(R^*, m^*, \sigma^*)$ with $\sigma^* = (D^*, \pi^*_{mem}, \pi^*_{mod}, T^{*\prime}, c^*, s^*)$. If $VK_{i^*} \notin R^*$, abort. The core of the proof is showing $T^{*\prime} = T^*$; once this is established, $(c^*, s^*)$ is directly a Schnorr forgery under the challenge key. We proceed in two steps.

*Step 1: Subset extraction (what subset did the adversary use?).* Recall from the construction (Figure 3) that the accumulated set is $S^* = \{VK_1, \ldots, VK_n\} \cup \{v_0, \ldots, v_{w-1}\}$, where each $v_j = h^{2^j}$ is a *randomizer power*. The randomizer powers serve a specific role: an honest signer with secret key $x$ samples a fresh lift $r' = x + qa$, decomposes $r' = \sum_{i \in I} 2^i$ in binary, and selects the subset $\{VK\} \cup \{v_i : i \in I\}$. The integer product of this subset, reduced modulo $p$, gives $g^x \cdot \prod_{i \in I} h^{2^i} = g^x h^{r'} = g^x h^x = (g \cdot h)^x = T$, the linkable tag. In other words, the randomizer powers under $h$ provide the "$h$-component" of the tag, while the public key provides the "$g$-component," and together they form the tag $T = (g \cdot h)^x$.

We show that the forgery's tag either identifies a signer recorded by the simulation or admits a subset representation to which Step 2 applies. The witness extraction below is used only in the auxiliary reductions. The main reduction $\mathcal{B}$ simply checks whether $T^{*\prime} = T^*$ and, if so, outputs the Schnorr signature.

The extracted witnesses must refer to the statements in the original forgery, although they may be recovered in different continuations. The auxiliary reductions return separately to the original native challenge queries of the proofs being extracted. After guessing their indices, let $\delta$ be the probability of the successful-forgery event in question with each such proof using its selected query. There are at most four such queries, so guessing their indices costs only a polynomial factor. For $k \leq 4$ required checkpoints, the successful runs having conditional continuation probability below $\delta/(2k)$ at any checkpoint have total probability at most $\delta/2$. On the remaining runs, each extractor has inverse-polynomial continuation success. Run the extractors separately from these original prefixes, requiring each continuation to use the fixed challenge input for that extraction. Polynomial truncation and repetition retain non-negligible extraction probability whenever $\delta$ is non-negligible. The continuations need not reproduce the other parts of the original forgery.

We next recover the modular witness, which is needed in both cases. Every signing response generates its modular proof honestly, so no modular challenge is programmed and the reduction retains the witnesses of honestly supplied inner proofs. Let $B$ be the masking element in the original modular proof. The inner extractor recovers $\rho$ such that $B = g_{acc}^{\rho}$; if $B$ matches a recorded masking element, the reduction uses its retained exponent instead. The outer extractor recovers $\beta$ such that

$$D^* B^p = g_{acc}^{\beta}, \qquad \beta \bmod p = T^{*\prime}.$$

The inner query fixes $B$, while the outer query fixes $D^*, T^{*\prime}, B$ and the inner proof. Thus their witnesses combine for the original statement, even if the extraction continuations differ:

$$\bar{prod} = \beta - p\rho, \qquad D^* = g_{acc}^{\bar{prod}}, \qquad \bar{prod} \bmod p = T^{*\prime}.$$

If $(D^*, Acc^*)$ was recorded, we identify its signer directly from the modular witness. Let $r_i$ and $T_i$ be its recorded exponent and tag. Then $g_{acc}^{\bar{prod} - r_i} = 1$, so the Order assumption gives $\bar{prod} = r_i$ and hence $T^{*\prime} = T_i$, except with negligible probability. The Order assumption and Lemma 3 bind the common accumulator to the same ring set, so the recorded signer belongs to $R^*$. This case requires no extraction from the simulated membership proof.

If $(D^*, Acc^*)$ was not recorded, we recover a divisor of the accumulated product and a Schnorr preimage for its residue. For membership extraction, the native checkpoint is immediately before the

first answer to the complete input $\mathrm{enc}(pp, D^*, Acc^*, z^*, A_g^*, A_u^*)$. Queries made by final verification are included when selecting this checkpoint. The wrapper reserves the entire membership-oracle domain for $(D^*, Acc^*)$, covering both challenge coordinates and all first messages for that statement. It aborts if a later signing simulation would program this reserved domain; otherwise, it continues the prescribed simulation on other statements. No successful continuation in the fresh-statement case is removed: any such programming would itself make $(D^*, Acc^*)$ recorded. Freshness also excludes earlier simulated programming anywhere in this domain. Ordinary answers already present at other inputs are retained.

The prefix up to the selected query, including the earlier simulations, forms the first stage of an interactive PoKE prover with fixed statement and first message. Given an external challenge pair $(c_{\mathrm{mem}}, \ell_{\mathrm{mem}})$, its second stage places that pair at the unanswered selected input and resumes the execution. It returns a response only if the continuation produces a successful fresh-statement forgery using exactly that input. The preceding checkpoint argument supplies inverse-polynomial success for this prover. All simulations are internal, and their unknown-order group operations are public generic computations; each replay restores the adversary, simulator, random-tape and random-oracle prefix state, with one consistent generic-group encoding and the group-operation records needed by the PoKE extractor. The resulting two-stage prover is generated within the original knowledge experiment and accepts whenever the selected fresh-forgery continuation succeeds. The underlying interactive PoKE extractor therefore applies under its original setup, generic-group, and runtime conditions. Together with the preceding checkpoint and truncation argument, this yields an integer $y$ satisfying $(D^*)^y = Acc^*$ with non-negligible probability whenever the successful fresh-forgery event has non-negligible probability. The Schnorr input has not been programmed by a signing simulation either, since it contains $(Acc^*, T^{*\prime}, D^*)$. The membership extractor and the separate Schnorr fork thus recover witnesses for the original statements, giving

$$(D^*)^y = Acc^*, \qquad T^{*\prime} = (gh)^{x^{*\prime}}.$$

Combining the membership and modular witnesses, the Order assumption gives

$$\bar{prod}y = \prod_{s \in S^*} s,$$

except with negligible probability.

It remains to turn this integer divisibility into a subset representation of the tag. If $T^{*\prime} \in S^*$, take $S' = \{T^{*\prime}\}$. Otherwise, analyze each auxiliary reduction with all setup exponents retained solely for applying GEDI. Writing $h = g^\alpha$, the table preimages relative to $gh$ are $x_i/(1+\alpha)$ and $\alpha 2^j/(1+\alpha)$, whereas the extracted target preimage is $x^{*\prime}$. Since $\bar{T^{*\prime}} \notin S^*$, the target preimage is distinct from the table preimages. Theorem 2 therefore gives that $\bar{prod}$ is a subset product, and the DI trial-division argument of Theorem 3 recovers $S'$. The extra retained exponents are not used by the adversary or extraction algorithm. Retaining them changes no signing response, corruption-abort decision, checkpoint selection, or rewind, so the same subset conclusion holds in the corresponding DL reduction. In either case,

$$\prod_{s \in S'} s \bmod p = T^{*\prime}.$$

Step 2 applies to this fresh-membership-statement case.

*Step 2: The forgery must target exactly one key.* We show that $S'_{VK}$ contains exactly one public key, with overwhelming probability. Partition $S' = S'_{VK} \cup S'_v$, where $S'_{VK} \subseteq \{VK_1, \ldots, VK_n\}$ collects the public keys and $S'_v \subseteq \{v_0, \ldots, v_{w-1}\}$ collects the randomizer powers. Define $a = \sum_{VK_j \in S'_{VK}} x_j \bmod q$

(the $g$-exponent) and $b = \sum_{v_l \in S'_v} 2^l \bmod q$ (the $h$-exponent). The tag satisfies $T^{*\prime} = g^a \cdot h^b = g^{a+\alpha b}$ (since $h = g^\alpha$), and also $T^{*\prime} = (g \cdot h)^{x^{*\prime}} = g^{(1+\alpha)x^{*\prime}}$ for a unique $x^{*\prime} \in \mathbb{Z}_q$. Equating:

$$a + \alpha \cdot b \;=\; (1+\alpha) \cdot x^{*\prime} \pmod q. \tag{2}$$

We rule out three bad events.

- $S'_{VK} = \emptyset$ *and* $S'_v = \emptyset$*:* Then $a = b = 0$ and $T^{*\prime} = 1_{\mathbb{G}_p}$. This case is excluded by the verification algorithm, which rejects $T = 1_{\mathbb{G}_p}$. (If $S'_{VK} = \emptyset$ but $S'_v \neq \emptyset$, then $a = 0$; $b = 0$ again gives $T^{*\prime} = 1$, which is rejected, while $b \neq 0$ falls into the next case.)
- $a \neq b$ (under the DL assumption)*:* Rearranging Equation (2) gives $\alpha = (x^{*\prime} - a)/(b - x^{*\prime})$. We build a separate DL reduction $\mathcal{B}'$: given a DL challenge $(\hat{g}, \hat{h} = \hat{g}^{\alpha_0})$ in $\mathbb{G}_p$, $\mathcal{B}'$ sets $g = \hat{g}$ and $h = \hat{h}$, generates all keys honestly (knowing every $SK_i$), and runs $\mathcal{A}$. If $\mathcal{A}$ outputs a valid forgery, $\mathcal{B}'$ extracts the subset $S'$ and exponents $a, b$ via the NIZK extractors, and extracts $x^{*\prime}$ from the Schnorr signature via the forking lemma [72]. Whenever $a \neq b$, $\mathcal{B}'$ computes $\alpha_0 = (x^{*\prime} - a)/(b - x^{*\prime}) = \log_g h$, breaking the DL assumption.
- $|S'_{VK}| \geq 2$ *(with* $a = b$*)* (under the DL assumption)*:* We build another DL reduction $\mathcal{B}''$: given a DL challenge $(\hat{g}, \hat{h} = \hat{g}^{\alpha_0})$, it guesses $j_1 \xleftarrow{\$} [\ell]$, sets $g = \hat{g}$ and $VK_{j_1} = \hat{h}$, and generates all other keys honestly. It samples $\beta \xleftarrow{\$} \mathbb{Z}_q \setminus \{0, -1\}$ and sets $h = \hat{g}^\beta$. All signing queries use the accumulator simulation above, retaining the genuine tags; for $VK_{j_1}$, the tag is $(VK_{j_1})^{1+\beta}$ and the Schnorr signature is simulated by programming the random oracle. $\mathcal{B}''$ aborts if $VK_{j_1}$ is corrupted. If $|S'_{VK}| \geq 2$ and $VK_{j_1} \in S'_{VK}$, then $a = b$ gives

$$x_{j_1} = b - \sum_{\substack{VK_j \in S'_{VK} \\ j \neq j_1}} x_j \pmod q,$$

  breaking DL. The guess $VK_{j_1} \in S'_{VK}$ succeeds with probability at least $1/\ell$.

The only remaining case is $|S'_{VK}| = 1$ with $a = b$. If $S'_{VK} = \{VK_j\}$ for $j \neq i^*$, the adversary targeted a different key; this does not yield a forgery under $T^*$ and is accounted for by the $1/\ell$ guessing loss. If $S'_{VK} = \{VK_{i^*}\}$, then $a = x^*$, hence $T^{*\prime} = (g \cdot h)^{x^*} = T^*$, and $(c^*, s^*)$ is a valid Schnorr signature under the challenge key $T^*$.

*Freshness.* It remains to show that $(\mu^*, c^*, s^*)$ was not returned by $\mathcal{O}_{Sch}$, so that it constitutes a genuine sEUF-CMA forgery. The game condition (Definition 12) guarantees that $\mathcal{A}$ never queried the signing oracle with the pair $(m^*, R^*)$. In particular, $\mathcal{B}$ never forwarded a query with ring $R^*$ and message $m^*$ to $\mathcal{O}_{Sch}$. By the Order assumption and Lemma 3, distinct ring sets yield distinct accumulators except with negligible probability. Hence the forgery's effective Schnorr message $\mu^* = Acc^*||T^*||D^*||\pi^*_{mem}||\pi^*_{mod}||m^*$ differs from every message previously submitted to $\mathcal{O}_{Sch}$. Therefore $(c^*, s^*)$ is a valid sEUF-CMA forgery on a fresh message.

*Success probability.* Write $\epsilon = \mathrm{Adv}_{\text{LRS-Unf}}(\mathcal{A})$. We now convert the auxiliary tag argument into the success probability of the main reduction. For a recorded membership statement, the tag already belongs to a member of $R^*$. For a fresh membership statement, the auxiliary extraction and the two DL reductions above show that an accepting tag cannot differ from every genuine tag of the members of $R^*$, except with negligible probability. All polynomial losses from query guessing, extraction, and forking occur only in these auxiliary reductions. The bad-tag event is efficiently testable given the genuine tags, so its negligible probability transfers through the signing hybrids to the main reduction.

Consequently, except with negligible probability, the forgery has the genuine tag of an uncorrupted member of $R^*$. The main reduction succeeds when its independently guessed index $i^*$ is that member; it does not run the auxiliary extractors. Hence

$$\mathrm{Adv}_{\text{sEUF-CMA}}(\mathcal{B}) \;\geq\; \frac{\epsilon}{\ell} - \mathrm{negl}(\lambda).$$

The $1/\ell$ guessing loss is standard in ring signature unforgeability proofs [6,58] and is inherent to the "guess-then-embed" paradigm.

**Non-slanderability.** The non-slanderability experiment (Definition 11) differs from unforgeability in two ways: (1) the adversary must produce a *linking pair* $(\sigma^*, \sigma^\dagger)$ rather than a single forgery, and (2) the freshness condition is weaker. We show that the same reduction as above applies, with one modification in the freshness argument.

*Reduction and simulation.* $\mathcal{B}$ sets up the sEUF-CMA challenge and simulates oracles exactly as in the unforgeability reduction. In Stage 1, $\mathcal{A}$ interacts with the signing and corruption oracles. At the end of Stage 1, $\mathcal{A}$ commits to $(R^*, m^*, \sigma^*)$. Since $R^*$ consists of uncorrupted keys at the time $\sigma^*$ is committed, and $\mathcal{B}$ has not revealed $SK_{i^*}$, the forgery extraction (Steps 1–2) applies identically: $\mathcal{B}$ extracts the Schnorr forgery from $\sigma^*$ at the end of Stage 1.

*Freshness (the key difference from unforgeability).* In the unforgeability game, the condition "$\mathcal{A}$ never queried $(\mathbf{sign}, \cdot, m^*, R^*)$" directly implies the Schnorr message was never submitted to $\mathcal{O}_{Sch}$. The non-slanderability game has a weaker condition: it only requires that $\sigma^*$ itself was not obtained from the signing oracle, but $\mathcal{A}$ *may* have queried $(\mathbf{sign}, VK_{i^*}, m^*, R^*)$ and received a different signature $\sigma'$. In that case, $\mathcal{B}$ submitted $\mu' = Acc^*||T^*||D'||\pi'_{mem}||\pi'_{mod}||m^*$ to $\mathcal{O}_{Sch}$ and received $(c', s')$.

Since the challenge binds the full transcript, two accepting signatures with the same $(c, s)$ coincide in all fields, except with negligible random-oracle collision probability. Thus, a fresh ring signature yields a new effective-message/Schnorr-signature pair and constitutes a valid sEUF-CMA forgery.

*Stage 2 does not affect the reduction.* Stage 2 reveals all secret keys to $\mathcal{A}$, but $\sigma^*$ is already committed, so the key disclosure cannot retroactively alter it. $\mathcal{A}$ then outputs $\sigma^\dagger$ that links with $\sigma^*$ (i.e., $T^\dagger = T^*$); producing $\sigma^\dagger$ is trivial once $\mathcal{A}$ knows all keys and does not affect the reduction. $\mathcal{B}$ extracts its Schnorr forgery from $\sigma^*$ (committed in Stage 1), not from $\sigma^\dagger$. □

**Theorem 7 (Linkable Anonymity).** *The construction in Figure 3 is linkably anonymous (Definition 10) in the random oracle model under the DDH assumption (Assumption 4) in $\mathbb{G}_p$ and EEH (Assumption 6) in $\mathbb{G}_?$, with the independent setup and randomizer-table conditions stated above.*

*Proof.* We use a sequence of games. Write $\mathrm{Adv}_i$ for $\mathcal{A}$'s advantage in Game $i$. We account once for the negligible exceptional table event and for honest keys that are identities, repeat another key, or coincide with a randomizer entry.

**Game 0** (Real). The linkable anonymity experiment (Definition 10) runs as specified. $\mathrm{Adv}_0 = \mathrm{Adv}_{\text{LRS-Anon}}(\mathcal{A})$.

**Game 1** (Simulate NIZK proofs). Same as Game 0, but in every signature that involves a challenge key ($VK_0^*$ or $VK_1^*$), both accumulator proofs and the Schnorr signature are simulated from their public inputs. $T$ and $D$ are unchanged. For Schnorr, sample $s \xleftarrow{\$} \mathbb{Z}_q$ and $c \xleftarrow{\$} [0, 2^{2\lambda})$, set $R_{sch} = (g \cdot h)^s T^c$, and program $H(Acc||T||D||\pi_{mem}||\pi_{mod}||R_{sch}||m) = c$ without changing any previous answer.

*Transition* $0 \to 1$. Joint zero knowledge over the polynomially many signing requests makes $|\mathrm{Adv}_0 - \mathrm{Adv}_1|$ negligible, including the assumed programming-abort error. In subsequent games, the same public-input simulators are applied to the updated statements.

**Game 2** (Randomize $D$). Same as Game 1, but at each signing request involving a challenge key, sample a fresh independent $r \xleftarrow{\$} [1, 2^\lambda \mathrm{maxord}(\mathbb{G}_?)]$ and set $D = g_{acc}^r$.

*Transition* $1 \to 2$. The fresh lift randomizes the integer subset product while leaving the tag fixed. Sample the Schnorr parameters, randomizer table, and honest key list, along with their secret exponents, independently of the accumulator setup; choose an honest-key index $i$ uniformly and retain these data as $z$. Sample $a \xleftarrow{\$} [0, 2^{2\lambda})$, let $I$ be the binary support of $x_i + qa$, and define the integer source

$$X_i = VK_i \prod_{j \in I} v_j.$$

The $2^{2\lambda}$ choices of $a$ give distinct lifts in $[0, 2^w)$ and hence distinct randomizer subsets. Their integer products are distinct, and multiplication by the fixed positive integer $VK_i$ preserves distinctness. Thus, conditioned on any $z$ with a valid table, each value of $X_i$ has probability at most $2^{-2\lambda}$, and $X_i$ has polynomial bitlength. EEH therefore gives $(pp, z, g_{acc}^{X_i}) \approx_c (pp, z, g_{acc}^{r})$ with the independent exponent distribution of Game 2.[9]

Apply this comparison through hybrids indexed by the signing request and honest-key index, choosing the request position uniformly from a fixed polynomial bound. The reduction generates the prefix before revealing its EEH challenge and uses that challenge at the designated request only if the actual signer has the designated index; otherwise it generates identical distributions on both sides. It computes *Acc* publicly from the requested ring, retains the genuine tag, and simulates $\pi_{mem}$, $\pi_{mod}$, and the Schnorr signature. After this simulation, the target lift is used only through $D$, so the reduction reproduces the entire view and its subsequent adaptive continuation. Averaging the adjacent hybrid differences over the request position and key index incurs only a polynomial loss; unused request positions contribute zero. Following the randomized accumulator technique of [21], this hides the selected subset while preserving its public modular image, here the tag $T$. Hence $|\mathrm{Adv}_1 - \mathrm{Adv}_2|$ is negligible under EEH and the stated table and simulation conditions.

**Game 3** (Randomize tags $T$). Same as Game 2, but for each challenge key, replace $T = (g \cdot h)^x$ by an independent uniform element of $\mathbb{G}_p \setminus \{1_{\mathbb{G}_p}\}$, sampled once and reused on every signing request for that key. Continue to simulate both accumulator proofs and the Schnorr signature from their updated public inputs.

*Transition* $2 \to 3$. We replace the two challenge tags one at a time using DDH (Assumption 4). For one replacement, the reduction receives $(g, h, A, B)$, where $A = g^a$ and either $B = h^a$ or $B = h^r$ for independent uniform $r$. Before presenting the honest key list, it guesses which of the $\ell$ key indices will be selected for the challenge role being replaced, embeds $VK^* = A$, and generates the other keys normally. It aborts with a fixed output if that index is not selected for this role or is corrupted. The prefix before this selection does not use $B$, so this guess costs a factor of $\ell$. For the embedded key, set $T = A \cdot B$.

- *DDH tuple.* $T = g^a h^a = (g \cdot h)^a$, giving the genuine tag.
- *Random tuple.* $T = g^a h^r$ is uniform independently of $VK^*$, giving the random-tag distribution up to the negligible identity-exclusion error.

Conditioning the public bases on $g, h \neq 1$ and $gh \neq 1$ likewise changes the distributions only negligibly. By Game 2, sampling $D$ requires no secret exponent, and we can simulate both accumulator proofs and the Schnorr signature as in Game 1. Thus the reduction answers every permitted signing

[9] For class groups, the reduction samples the auxiliary bases independently as specified above. In the cyclic setup, it samples each base as $g_{acc}^s$, choosing $s$ uniformly from a sufficiently long integer interval of polynomial bitlength, conditioned to be coprime to every prime below a suitable polynomial bound. Since the group order has polynomially many prime factors, this approximates independent uniform generators to any desired inverse-polynomial accuracy.

request without knowing $a$ or the discrete logarithm of the replacement tag. The two DDH transitions, with their polynomial guessing losses and negligible conditioning errors, make $|\mathrm{Adv}_2 - \mathrm{Adv}_3|$ negligible.

**Analysis of Game 3.** The two challenge tags are independent and identically distributed, fixed per key, while every challenge signature uses a fresh independent $D$ and proofs simulated from the public inputs. Exchanging the two tag assignments exchanges the two challenge oracle roles without changing the joint distribution. Every permitted challenge ring contains both challenge keys, so this exchange also preserves the ring-validation outcome. Therefore, the view is independent of $b$, and $\mathrm{Adv}_3 = 0$.

**Conclusion.** All three transitions have negligible distinguishing advantage under the stated premises, so $\mathrm{Adv}_{\text{LRS-Anon}}(\mathcal{A}) = \mathrm{negl}(\lambda)$. □

**Theorem 8 (Linkability).** *The construction in Figure 3 satisfies linkability (Definition 9) in the generic group model for $\mathbb{G}_?$ and the random oracle model, assuming the DL assumption (Assumption 3) and sEUF-CMA security of Schnorr signatures [72,76] hold in $\mathbb{G}_p$, the argument-of-knowledge property of the underlying interactive PoKE, the knowledge soundness of* **NIZK-PoKEModN***, EEH (Assumption 6) and the Order assumption (Assumption 2) for $\mathbb{G}_?$, and the GEDI property (Definition 6) holds for discrete logarithm encodings in $\mathbb{G}_p$.*

*Proof.* Each secret key $x$ yields a unique tag $T = (g \cdot h)^x$. With at most one corrupted key, the adversary can produce a valid Schnorr signature under at most one tag value, making two unlinkable signatures infeasible.

*Setup.* Suppose $\mathcal{A}$ corrupts at most one key ($VK_c$ with secret $x_c$, or none) and outputs two valid unlinkable signatures $(\sigma_1, \sigma_2)$ for rings $R_1, R_2 \subseteq \mathcal{VK}$ with tags $T_1 \neq T_2$. Write $T_i = (g \cdot h)^{x_i}$; since $(g \cdot h)$ has prime order $q$, we have $x_1 \neq x_2$. At most one of $x_1, x_2$ equals $x_c$, so at least one signature, say $\sigma_2$, carries a Schnorr signature under a tag whose secret $\mathcal{A}$ does not know.

*Reduction.* The outside adversary $\mathcal{B}$ receives a Schnorr sEUF-CMA challenge: a generator $g'$, a public key $T^* = (g')^{x^*}$, and a signing oracle $\mathcal{O}_{Sch}$. $\mathcal{B}$ samples $\alpha \xleftarrow{\$} \mathbb{Z}_q \setminus \{0, -1\}$, sets $g = (g')^{(1+\alpha)^{-1}}$ and $h = g^\alpha$ so that $g \cdot h = g'$, and computes $VK_{i^*} = (T^*)^{(1+\alpha)^{-1}} = g^{x^*}$. $\mathcal{B}$ guesses $i^* \xleftarrow{\$} [\ell]$ and independently chooses one of the two output signatures uniformly as $\sigma^*$. Signing and corruption queries are answered exactly as in the oracle simulation of Theorem 6. We show that at least one output carries the genuine tag of an uncorrupted ring member. The auxiliary argument of Theorem 6 also applies with at most one corrupted key: any extracted subset containing at least two public keys contains an uncorrupted key at which the second DL challenge can be embedded. Thus, except with negligible probability, each output tag is the genuine tag of a member of its ring. Since the two tags differ, at least one such member is uncorrupted.

To obtain a Schnorr forgery, $\mathcal{B}$ needs only to guess that member and its output signature. It checks that the guessed tag equals $T^*$ and outputs the Schnorr signature, without running the auxiliary extractors. The Schnorr signature $(c^*, s^*)$ is therefore a forgery under $T^*$.

*Freshness.* Neither $\sigma_1$ nor $\sigma_2$ was obtained from the signing oracle. Unlike the unforgeability game, this condition does not preclude $\mathcal{A}$ from having queried $(\mathbf{sign}, VK_{i^*}, m^*, R^*)$; the effective Schnorr message $\mu^* = Acc^*||T^*||D^*||\pi^*_{mem}||\pi^*_{mod}||m^*$ may therefore have been submitted to $\mathcal{O}_{Sch}$. By the full-transcript binding argument in the non-slanderability proof, freshness of $\sigma^*$ implies that $(\mu^*, (c^*, s^*))$ is a new effective-message/signature pair, except with negligible probability. Therefore, this pair is a valid sEUF-CMA forgery under $T^*$.

*Success probability.* The guess of $i^*$ succeeds with probability $1/\ell$; the guess of which signature is the forgery succeeds with probability $1/2$. Overall, $\mathcal{B}$ breaks Schnorr sEUF-CMA with advantage at least $\epsilon/(2\ell) - \mathrm{negl}(\lambda)$, where $\epsilon = \mathrm{Adv}_{\text{LRS-Link}}(\mathcal{A})$. □