

BRaccoon: Concurrently Secure Blind Lattice Signatures from Raccoon

Lucjan Hanzlik¹ , Mark Manulis² , Marzio Mula² , Alan Pulval-Dady² ,
Tjerand Silde³ , and Daniel Slamanig² 

¹ CISP Helmholtz Center for Information Security, Germany, hanzlik@cispa.de

² Research Institute CODE, Universität der Bundeswehr München, Germany,
firstname.lastname@unibw.de

³ Norwegian University of Science and Technology, Norway, tjerand.silde@ntnu.no

Abstract. Blind signatures are a central primitive for privacy-preserving applications such as e-cash, anonymous credentials, and e-voting. In the post-quantum setting, existing constructions typically follow one of two paradigms: either signatures are realized as non-interactive zero-knowledge (NIZK) proofs of valid underlying signatures, or they are obtained from identification schemes via the Fiat–Shamir transform. In both approaches, the resulting signatures deviate syntactically from standard signatures, incurring additional verification overhead and limiting compatibility with existing infrastructures. In contrast, classical constructions such as blind Schnorr yield signatures that are indistinguishable from ordinary ones. Achieving this property in the lattice setting has remained an open problem.

We present **BRaccoon**, the first lattice-based blind signature scheme that achieves concurrent security while producing signatures that are syntactically identical to those of a standard signature scheme. Our construction builds on the rejection-free lattice signature scheme **Raccoon**, and extends the “blind signatures from a signature assumption” paradigm of Fuchsbauer and Wolf (EUROCRYPT 2024) to lattices. At a high level, we introduce blinding at the commitment stage and enforce correct challenge and response generation via linearly homomorphic encryption combined with NIZK proofs. As a result, **BRaccoon** signatures preserve the algebraic structure of **Raccoon** signatures while remaining compact: in an optimized instantiation, signatures are 32 KB, public keys are 10 KB, and total communication is 847 KB for up to 2^{32} signatures.

A central technical challenge stems from discrete Gaussian sampling, where blinding induces a non-trivial distributional shift that precludes direct security reductions. To overcome this, we introduce a modified scheme **Raccoon*** that explicitly captures this shift. We prove that one-more unforgeability of **BRaccoon** tightly reduces to the unforgeability of **Raccoon***, which in turn reduces to that of **Raccoon**.

For a concrete instantiation, we develop a hybrid proof framework that combines lattice-based zero-knowledge arguments for linear relations with arithmetic zk-SNARKs for hash computations, linked via structured commitments. Our work demonstrates that concurrently secure blind signatures with standard-signature syntax can be achieved in the lattice setting, providing a viable path toward practical and interoperable post-quantum privacy-preserving systems.

Keywords: Blind Signature · Lattices · Concurrent Security

1 Introduction

Blind signatures, introduced by Chaum in the 1980s [Cha82], are an interactive variant of digital signatures with the key property that the signer learns nothing about the message being signed, while the transcript of the signing protocol remains unlinkable to the resulting signature. These features make blind signatures a fundamental building block for privacy-preserving applications such as e-cash [Cha82], e-voting [IKSA03], and anonymous credential systems [BL13]. In recent years, they have also seen increasing practical deployment, for example in systems such as TumbleBit [HAB⁺17] and Privacy Pass [DGS⁺18].

Blind signatures can be constructed under a variety of classical hardness assumptions, including RSA [Cha82], discrete logarithms [PS00, BL13, HKL19, CAHL⁺22], and bilinear pairings [Bol03, FHKS16, HLW23]. However, Shor’s algorithm and ongoing advances in quantum computing increasingly threaten such assumptions, creating strong demand for efficient constructions based on quantum-resistant primitives. Despite this urgency, blind signatures in the post-quantum setting remain significantly less mature and less well understood than their classical counterparts. Addressing this gap is the focus of this work.

Desirable properties. Before proceeding, we highlight properties that are particularly relevant for the practical deployment of blind signature schemes. In typical applications, multiple signing sessions may occur concurrently and interleave with other protocol executions. Consequently, it is essential for such schemes to provide *concurrent security*. Notably, this property is achieved automatically by protocols with only two moves, but is non-trivial otherwise.

While simple and efficient two-move constructions exist under classical assumptions—thanks to their favorable algebraic structure—such as those based on RSA [Cha82] or bilinear pairings [Bol03, FHKS16, HLW23], no analogous solutions are known in the post-quantum setting. The only practical approach to obtaining two-move blind signatures in this regime is to follow Fischlin’s generic transformation [Fis06], where the resulting signature consists of a non-interactive zero-knowledge (NIZK) proof attesting to the validity of a signature under some underlying signature scheme. This paradigm has proven particularly successful for lattice-based constructions [AKSY22, dPK22, BLNS23a], and has recently been extended to other families following the hash-and-sign paradigm, e.g., multivariate and code-based signatures [BFM⁺25a]. Moreover, recent work by Baum et al. [BBB⁺26] demonstrates practical instantiations using VOLE-in-the-head techniques and MAYO as an underlying multivariate signature scheme.

Although these approaches can be practically efficient—particularly in terms of interaction—their reliance on NIZK proofs leads to an important drawback: the final signature is itself a proof of knowledge of a signature, rather than a conventional signature. As a result, the computational cost and structural complexity of the proof system are inherited by every verifier and propagate to higher-level protocols, leading to increased verification overhead and more complex interfaces. In the post-quantum setting, where both signatures and zero-knowledge proofs are comparatively expensive, this overhead becomes particularly pronounced.

This observation motivates a key design goal: while auxiliary zero-knowledge mechanisms may be unavoidable during the interactive issuance phase, they should be confined to this phase, ensuring that the *final output remains a standard signature under the underlying scheme*. This preserves the native verification interface, enables reuse of existing implementations, and facilitates seamless integration into systems that already support the base signature scheme. Importantly, this goal aligns with historical practice: classical blind signature schemes – ranging from Chaum’s blind RSA to blind Schnorr and blind ECDSA – were obtained by adapting existing signatures while preserving their format and verification procedures, reflecting a strong preference for compatibility that is likely to persist in the post-quantum setting.

Constructing such schemes. Taking Schnorr signatures as the canonical example, blind signatures have traditionally been obtained by applying blinding techniques to identification schemes, which are then transformed into signature schemes via the Fiat–Shamir (FS) transform. A key drawback of this approach is that the resulting protocols are typically not round-optimal, and therefore concurrent security does not follow automatically. Indeed, it has been shown in the work of Do, Hanzlik and Paracucchi [DHP24] that standard parallel repetitions of Σ -protocols with (polynomially) small challenge spaces cannot yield blind signatures secure against polynomially many concurrent executions. While this limitation does not apply in general to exponentially large challenge spaces, it is well known that Schnorr signatures in particular are vulnerable to ROS attacks [BLL⁺21], and thus only achieve weaker—often impractical—sequential security. As a consequence, existing approaches often resort to designing specialized “blindable” identification or signature schemes that avoid such

attacks. Notable examples include recent constructions based on group actions [HLM⁺25, HLPP26] and lattice-based schemes achieving concurrent security at the cost of additional rounds and heavier machinery [KLR25].

Avenues around these problems. For Schnorr signatures, several alternative approaches have been proposed to restore concurrent security. One direction is the use of clause blind Schnorr signatures [FPS20]. Another line of work, introduced by Fuchsbauer and Wolf [FW24] and later followed by Maire and Pulval-Dady [MPD25], constructs blind signatures directly from the signature assumption. In the work of Fuchsbauer and Wolf [FW24], which constructs a blind signature scheme for Schnorr signatures, the user first commits to its randomness and message before receiving the signer’s first message, and later proves in zero knowledge that the challenge was computed correctly. This, however, requires proving knowledge of pre-images under the hash function, while the security of Schnorr signatures themselves relies on modeling the hash as a random oracle, leading to a conceptual tension. To resolve this, they instead assume that Schnorr instantiated with a concrete hash function (e.g., SHA-256) is unforgeable. Although this constitutes a non-standard assumption, it can be viewed as a minimal and practically motivated strengthening, and is implicitly adopted in several recent works on blind signatures [BBB⁺26], SNARK-friendly signatures [ZSE⁺24, ADK24, FR25] and anonymous credentials, both in the classical [Fas26, PPZ24, WOS⁺25] and post-quantum settings [PFW23, FR25].

Towards a solution. A key advantage of the “blind signatures from a signature assumption” paradigm of Fuchsbauer and Wolf [FW24], is that it simultaneously addresses the challenges identified above: if the security of the blind signature can be reduced to the EUF-CMA security of the underlying signature scheme, then concurrent security is obtained, and the resulting blind signatures retain the exact format of standard signatures. Unfortunately, adapting this paradigm to the post-quantum setting—and in particular to lattice-based signatures—is far from straightforward. Existing concurrently secure post-quantum blind signature schemes either produce signatures that are NIZK proofs of valid signatures, include NIZK components in the final signature (e.g. Kloss *et al.* [KLR25]), or require designing entirely new identification and signature schemes.

This leads us to the following research question:

Can we design a blinding technique for lattice-based signature schemes that yields blind signatures which simultaneously are (i) concurrently secure, (ii) syntactically identical to standard signatures, and (iii) practically efficient?

1.1 Our Contributions

We affirmatively answer the above question. At a high level, we show that the “blind signatures from a signature assumption” paradigm of Fuchsbauer and Wolf [FW24] can be adapted to the lattice setting, despite several fundamental obstacles.

Our starting point is to investigate whether the approach of Fuchsbauer and Wolf [FW24], which works naturally for Schnorr signatures, can be transferred to lattice-based Fiat–Shamir signatures. In the classical setting, Schnorr’s algebraic structure enables relatively simple blinding techniques. In contrast, lattice-based signatures introduce significant additional challenges. A first major obstacle is the use of rejection sampling [Lyu09], which ensures that the output distribution of signatures does not leak information about the secret key. In the blind signature setting, however, the final signature is only seen by the user, so the signer cannot verify whether rejection is required. Designing oblivious rejection sampling would in turn risk violating blindness, making this approach problematic.

To circumvent this issue, we build on Raccoon [dPKPR24], a rejection-free lattice-based signature scheme that avoids the need for rejection sampling. While this eliminates one major barrier, there are more to be solved: the Raccoon challenge generation is not a simple hash evaluation, but involves an extendable-output function (XOF) and an internal sampling procedure that is not zero-knowledge-friendly. To address this, we adapt the techniques of Policharla *et al.* [PFW23] to

modify the SampleInBall procedure so that rejections become sufficiently rare, enabling efficient zero-knowledge proofs. Furthermore, we instantiate the hash function using Poseidon [GKR⁺21], which is more amenable to efficient NIZK proofs than SHAKE-based constructions.

A second key challenge arises from the need to ensure blindness while producing standard-looking signatures. In our protocol, the user introduces a blinding shift to randomize the final signature, analogously to blind Schnorr signatures. However, unlike in the Schnorr setting, this shift leads to a non-uniform distribution in the lattice setting, potentially leaking information. To overcome this, we introduce a novel intermediate signature scheme, *Raccoon*^{*}, which internalizes this distributional shift by allowing the signing algorithm to take the shift as an explicit input. The blinding shift is sampled from a distribution wide enough to statistically drown the distribution of the signer’s initial response. This is what ensures blindness of the final signature, but it also increases the size of the resulting signature. Concretely, for our 128-bit parameter set supporting up to 2^{32} blind-signing queries, the signature size grows from about 11 KB for *Raccoon* to 32 KB for our blind-signature instantiation, while the public key size grows from about 2 KB to 10 KB. We show that the unforgeability of *Raccoon*^{*} reduces to that of *Raccoon*, enabling a tight security reduction for our blind signature.

Finally, we develop a proof framework that enforces correct behavior of both parties while preserving a practical communication profile. In particular, the user proves that the challenge derived from the blinded commitment is computed correctly and that its encryption under a linearly homomorphic scheme is consistent. To achieve this, we combine lattice-based linearly homomorphic encryption and zero-knowledge proofs for linear relations with arithmetic zk-SNARKs for hash-preimage constraints, instantiated using Poseidon. These components are linked via structured commitments. The dominant communication cost comes from the proofs of knowledge exchanged during the blind-signing protocol: our direct instantiation has total communication below 3.2 MB, while using proof-size optimization techniques reduces this to about 850 KB. These figures are based on existing lattice proof systems, and further improvements in lattice-based zero-knowledge proofs or in the concrete encoding of our statements would directly reduce the overall communication.

Overall, our construction *BRaccoon* is a concurrently secure blind signature scheme whose outputs are syntactically identical to standard *Raccoon* signatures, demonstrating that this goal can also be achieved in the lattice setting.

1.2 Related Work

Table 1 summarizes the state of the art in lattice-based blind signatures, which is the primary focus of this section. For a broader overview of blind signatures from other quantum-safe assumptions—which currently remains limited—we refer to the work of Hanzlik *et al.* [HLM⁺25, Appendix A].⁴

Fischlin-based constructions. Several lattice-based blind signatures follow Fischlin’s framework [Fis06], which yields two-move protocols where the final signature consists of a NIZK proof of knowledge of a valid pre-signature on a commitment to the message. Early lattice-based instantiations include the work of Agrawal *et al.* [AKSY22], whose security relies on the tailored *one-more-ISIS* assumption, and the work of del Pino *et al.* [dPK22], which instead relies on more standard lattice assumptions but at the cost of significantly larger signatures and transcripts.

Another round-optimal construction in the same “NIZK-in-the-signature” paradigm is the work by Lyubashevsky *et al.* [LNP22a], which requires fixing in advance an upper bound Q_s on the number of signatures per key, resulting in public key size and running time that grow linearly with Q_s . More recently, Jeudy and Sanders [JS25a] (with refinements in [JS25b]) reduces communication and signature size by reusing commitment entropy, at the cost of tagged public keys and maintaining state to avoid tag reuse.

Connections to anonymous credentials and SEPs. Several lattice-based anonymous credential constructions can be adapted to yield blind signatures. Notably, the schemes based on the ISIS-f

⁴ The currently known concurrently secure schemes from non-lattice assumptions include [HLM⁺25], based on group actions, and [BBB⁺26], based on multivariate quadratic systems.

assumption family from Bootle *et al.* [BLNS23b] and a STARK-friendly variant of Dilithium from Policharla *et al.* [PFW23], known as zkDilithium, can be viewed as further instances of the “NIZK-in-the-signature” paradigm. The latter replaces the SHAKE-based XOF with the Poseidon hash function and adapts the SampleInBall procedure accordingly (see Appendix G). However, existing implementations of SampleInBall [PFW23] focus on the proof system and do not realize the full unlinkable signing protocol, which is why we omit them from Table 1.

Similarly, post-quantum analogues of signatures with efficient protocols (SEPs) [CL03] can be used as building blocks for blind signatures [JRS23, AGJ⁺24]. In both anonymous credential systems and SEPs, unlinkable presentation of signatures typically requires either interactive verification or the generation of fresh NIZK proofs. Notably, all of the above constructions are round-optimal, and thus concurrent security follows automatically.

Schnorr-style constructions. An alternative line of work attempts to follow the template of classical Schnorr blind signatures. The first lattice-based construction in this direction is due to Rückert [Rüc10], with subsequent optimizations in [AEB20, AHJ21]. However, Hauck *et al.* [HKLN20] showed that the original security analyses of these constructions are flawed. They also provide a provably secure variant achieving sequential security, but this comes at the expense of extremely large signatures and communication, as well as a very limited number of queries per public key.

Finally, a recent four-move Fiat–Shamir-based construction aiming at concurrent security is given in [KLR25]. However, it incurs substantially larger communication costs and is based on tailored blinded identification scheme.

Discussion and comparison. In summary, in all existing post-quantum constructions, the final signature differs syntactically from a standard signature: it either consists of a NIZK proof of a valid signature or resembles a tailored blindable identification scheme. This stands in contrast to classical blind signature schemes, where the output typically retains the standard signature format.

Our scheme BRaccoon is, to the best of our knowledge, the first lattice-based blind signature scheme that produces signatures syntactically identical to those of the underlying signature scheme. While this comes at the cost of a non-optimal number of rounds (4) and higher communication (847 KB), it yields two important advantages. First, the resulting signature and public key are only 32 KB and 10 KB and therefore ranks among the most compact schemes in Table 1. Second, one-more unforgeability of BRaccoon reduces directly to the unforgeability of Raccoon, so that both schemes rely on the same underlying lattice assumptions.

For clarity, Table 1 omits auxiliary assumptions on hash functions and NIZK proof systems. For instance, [BLNS23a] relies on SHA-2/SHA-3. In contrast, our instantiation uses Poseidon [GKR⁺21], while the security of the associated NIZK proofs and homomorphic encryption components can be reduced to MLWE assumptions or statistical arguments.

1.3 Technical Overview

Our construction. The main design goal of our blind signature scheme, BRaccoon, is to produce signatures that are syntactically indistinguishable from standard lattice-based signatures. As a starting point, we use Raccoon, a rejection-free Fiat–Shamir lattice signature scheme that produces Dilithium-style signatures without relying on aborts.

At a high level, our protocol preserves the structure of a Raccoon signature while introducing a blinding transformation at the commitment stage; see Figure 1 for an overview. Recall that a Raccoon signature consists of three steps: sampling randomness to form a commitment value $\mathbf{w}$, deriving a challenge c via hashing, and computing a response $\mathbf{z}$ that binds the challenge to the secret key.

In our blind-signing protocol, the user first commits to the message $\mathbf{msg}$ together with a blinding factor $\mathbf{r}$ using a commitment scheme (which we can make extractable in our one-more unforgeability

⁵ We refer to [NS26, ALP26] for recent analyses of the hardness of ISIS-f and related SIS-with-hints assumptions.

Table 1. Comparison of concurrently secure lattice-based blind signature schemes. The column Q_s denotes the maximum number of signing queries per key supported by the security proof. We defer to Appendix F a brief discussion on how the data in this table was obtained.

Scheme	Non-standard assumptions	vk (KB)	Comm. (KB)	sig (KB)	Sec. (bits)	#Moves	Q_s
Proof of knowledge of signature (Fischlin approach)							
[AKSY22]	OM-ISIS	0.8	1.4	45.2	109	2	2^{64}
[dPK22, HSS24]	–	≈ 56	≈ 1000	≈ 100	128	2	2^{64}
[LNP22a]	–	1331	16384	150	128	2	2^{18}
[BLNS23b, LSS24]	ISIS-f ⁵	1	22	27	128	2	2^{64}
[JRS23, AGJ ⁺ 24]	–	48	45	79.6	128	2	2^{32}
[BLNS23a]	–	1	63	22	128	2	2^{20}
[JS25a, JS25b]	–	36.0	53.2	36.3	125	3	2^{32}
Blinded identification scheme (ad-hoc construction)							
[HKLN20]	–	443.7	34037.3	7915.5	128	3	7
[KLR25]	–	≈ 122750	≈ 9000	≈ 350	128	4	2^{64}
Our scheme (regular signature)							
BRaccoon _{LNP}	Raccoon	10		3110	28.00	4	2^{20}
				3150	31.78		2^{32}
				3260	43.10		2^{64}
BRaccoon _{LNPLaB}	Raccoon	10			28.00	4	2^{20}
				847	31.78		2^{32}
					43.10		2^{64}

proof). The signer then samples fresh randomness $\mathbf{r}'$ and computes the corresponding commitment $\mathbf{w}'$ as in the underlying signature scheme. The user re-randomizes this value using $\mathbf{r}$ to obtain $\mathbf{w}$ and derives the challenge c from it.

To enforce correct challenge computation, the user encrypts c under a linearly homomorphic encryption scheme, obtaining ctx_c , and proves via a NIZK proof π_1 that this encryption is consistent with the committed values. The signer then homomorphically computes an encryption of the response $\mathbf{z}'$ using the signing key $\mathbf{s}$, randomness $\mathbf{r}'$ and ctx_c , yielding $\text{ctx}_{\mathbf{z}'}$, and provides a proof π_2 that this computation was performed correctly.

Finally, the user decrypts $\text{ctx}_{\mathbf{z}'}$ to obtain $\mathbf{z}'$ and adds the blinding factor $\mathbf{r}$ to derive $\mathbf{z}$. The resulting signature $\text{sig} = (c, \mathbf{z})$ satisfies the original Raccoon verification equation (up to a slight adjustment of the acceptance bound), and is therefore syntactically identical to a standard Raccoon signature.

Security reduction. Proving one-more unforgeability of our construction assuming only the unforgeability of Raccoon naturally suggests simulating the BRaccoon signing oracle using access to a Raccoon signing oracle. However, such a direct simulation fails to perfectly reproduce the distribution of signatures generated by BRaccoon. In particular, the signatures obtained in the reduction are distinguishable from those produced by a genuine BRaccoon execution.

This issue does not arise in the setting of [FW24], where all randomness is sampled uniformly. In that case, adding a blinding shift preserves uniformity and therefore leaves the signature distribution

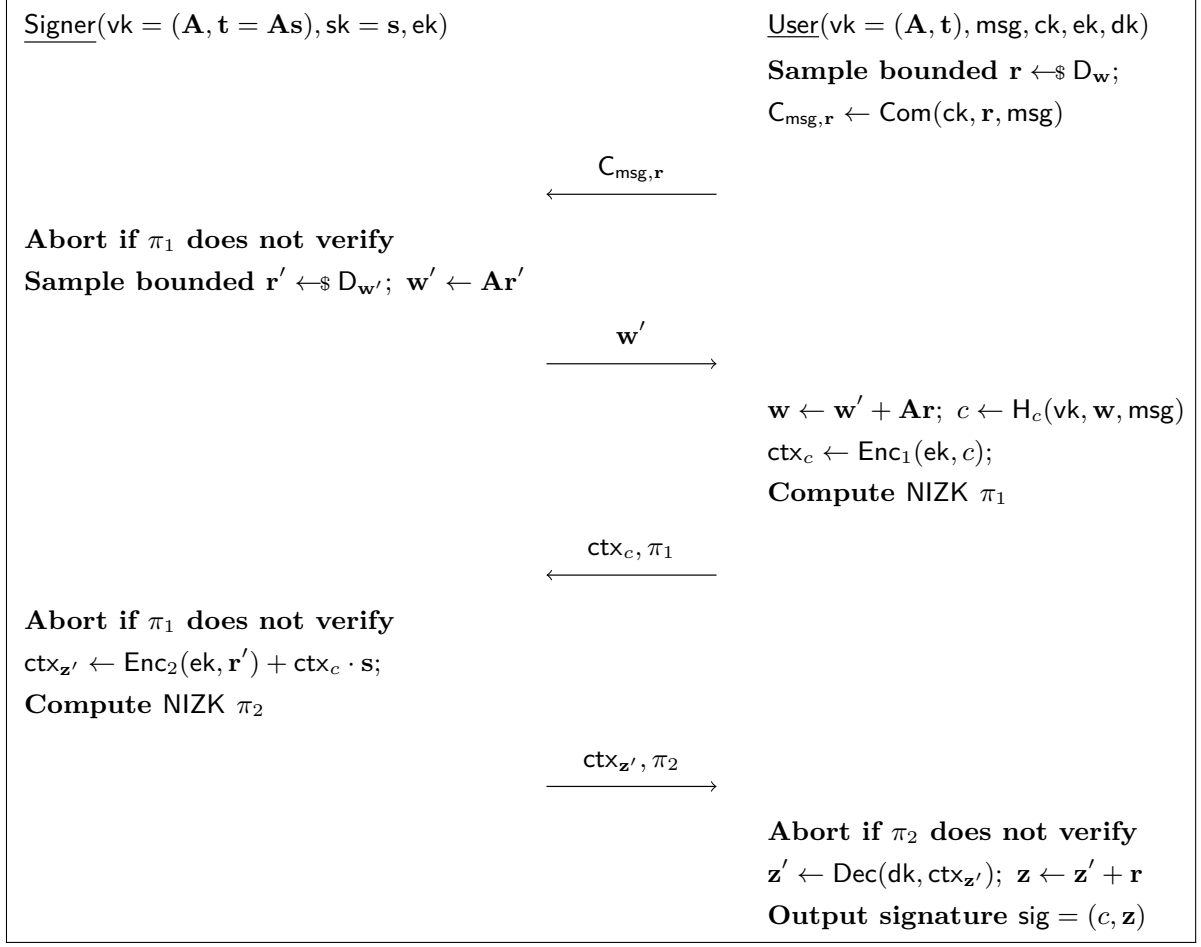


Fig. 1. High-level message flow of the blind signature protocol. Local computations, as well as the explicit computation of the error, rounding, and hint generation, are omitted. Each NIZK proves correctness of the corresponding move.

unchanged. In contrast, lattice-based schemes rely on discrete Gaussian sampling. Introducing an additive blinding shift results in a convolution of distributions, thereby altering the output distribution and preventing a straightforward simulation.

To overcome this obstacle, we introduce a modified signature scheme, denoted Raccoon^* , that captures the precise distributional effect of the blinding transformation. We first reduce the one-more unforgeability of BRaccoon to the EUF-CMA security of Raccoon^* , and then show that Raccoon^* itself is unforgeable assuming the security of Raccoon . This yields a tight and modular security argument. Furthermore, we show that strong unforgeability of Raccoon implies strong one-more unforgeability of BRaccoon . Blindness, on the other hand, follows from a standard composition of following properties: statistical drowning ($\mathbf{r}$ has to be much larger than $\mathbf{r}'$), the hiding of the commitment scheme, the IND-CPA security of the homomorphic encryption scheme, and the soundness and zero-knowledge guarantees of the underlying NIZK proofs.

Bridging lattice and hash constraints. A key challenge in instantiating our protocol concretely is the design of efficient NIZK proofs for the statements arising in the second round. Most of these statements consist of linear relations over module lattices together with explicit ℓ_2 -norm bounds. For such statements, we rely on the framework of Lyubashevsky, Nguyen, and Plançon (LNP) [LNP22b],

which provides efficient proofs for relations of the form

$$A\mathbf{s} = \mathbf{t} \pmod{q}$$

subject to norm constraints on $\mathbf{s}$. However, a complication arises when the user derives the challenge from the blinded commitment, i.e., the proof π_1 . The language L_1 to be proven using a NIZK inherently mixes two types of constraints: linear lattice relations and non-linear arithmetic operations corresponding to hash computation and sampling procedures. These belong to different proof paradigms. Lattice-based proof systems are optimized for linear relations with norm bounds, while hash evaluations are naturally expressed as arithmetic circuits. Combining both within a single framework would either incur prohibitive overhead (if expressed as a circuit) or sacrifice efficiency (if embedded in a lattice proof system).

To resolve this, we decompose the statement $L_1 = L_{1,1} \cap L_{1,2}$ into two parts $L_{1,1}$ and $L_{1,2}$ and link them via commitments. Concretely, we employ the BDLOP schemes from [BDL⁺18] to commit to all relevant witness components, including $\mathbf{msg}$, $\mathbf{w}$, and $\mathbf{r}$ and also to bind the tuple $(\mathbf{w}, \mathbf{msg}, c)$ along the different proofs. Since binding in BDLOP depends only on the short commitment randomness, we can avoid enforcing norm constraints on the committed values inside the arithmetic proof. This significantly reduces the complexity of the resulting Rank-1 Constraint System (R1CS), which only needs to capture arithmetic consistency.

On the arithmetic side, we provide dedicated R1CS encodings for the Poseidon hash function [GKR⁺21], the SampleInBall procedure [PWFW23], and the verification of BDLOP commitments. In particular, we implement SampleInBall using a truncated Fisher–Yates shuffle [Knu97], ensuring uniform support while avoiding data-dependent branching by using modular reductions as in [PWFW23].

To further reduce complexity, all polynomial operations for BDLOP commitments are carried out in the NTT domain, where polynomial multiplication reduces to component-wise multiplication. This allows us to express consistency checks using only linear constraints and coordinate-wise products, avoiding expensive convolution operations in the circuit.

In summary, the first proof, instantiated using the LNP framework, establishes correctness of the lattice relations, including commitment openings and the computation of $\mathbf{w}$ and ctx_c . The second proof, realized as a zk-SNARK, ensures that the committed values are consistent with the hash computation of the challenge. The binding property of the commitments guarantees that both proofs refer to the same underlying values. Finally, the LNP proofs can be compressed using LaBRADOR [BS23, PKN⁺25], which significantly reduces their contribution to the total proof size, as reflected in Table 1.

Instantiation. Finally, we instantiate all building blocks concretely and provide detailed parameter choices, communication complexity estimates, and signature sizes. Our evaluation demonstrates that the resulting construction is practical, while achieving the desired structural and security properties.

2 Background

2.1 Notation

We recall standard notation and definitions, following [MR04] and [LS15, §2].

Rings. Throughout this paper, let $\mathcal{R}$ denote the cyclotomic ring $\mathbb{Z}[X]/(X^n + 1)$, and let $\mathcal{R}_q$ denote its reduction modulo q , i.e., $\mathcal{R}_q = \mathcal{R}/q\mathcal{R} = \mathbb{Z}[X]/(q, X^n + 1)$.

Norms. Let $z \in \mathcal{R}_q$ be written as $z = \sum_{i=0}^{n-1} a_i X^i$ with $a_i \in (-q/2, q/2]$. We define its ∞ -norm and 2-norm as

$$\|z\|_\infty = \max_i |a_i| \quad \text{and} \quad \|z\|_2 = \sqrt{\sum_i a_i^2}.$$

For a vector $\mathbf{z} = (z_1, \dots, z_\ell) \in \mathcal{R}_q^\ell$, we extend these norms component-wise:

$$\|\mathbf{z}\|_\infty = \max_i \|z_i\|_\infty \quad \text{and} \quad \|\mathbf{z}\|_2 = \sqrt{\sum_i \|z_i\|_2^2}.$$

Modulus Rounding. Let $q, \nu \in \mathbb{N} \setminus \{0\}$ with $q > 2^\nu$. We define the *modulus rounding* function $\lfloor \cdot \rfloor_\nu : \mathbb{Z}_q \rightarrow \mathbb{Z}_{\lfloor q/2^\nu \rfloor}$ as follows.

For $x \in \mathbb{Z}_q$ represented in $\{0, \dots, q-1\}$, $\lfloor x \rfloor_\nu = x'$ is the unique integer $x' \in \{0, \dots, \lfloor q/2^\nu \rfloor - 1\}$ such that

$$x \equiv 2^\nu x' + x'' \pmod{q},$$

where $x'' \in \{-2^{\nu-1}, \dots, 2^{\nu-1} - 1\}$.

Module Lattices. Let R be a ring and let $\mathbf{A} \in R^{k \times \ell}$. The *module lattice generated by $\mathbf{A}$* is defined as

$$\mathcal{L}(\mathbf{A}) = \{\mathbf{A}\mathbf{z} \mid \mathbf{z} \in R^\ell\} \subseteq R^k.$$

The *kernel module lattice of $\mathbf{A}$* is defined as

$$\mathcal{L}^\perp(\mathbf{A}) = \{\mathbf{z} \in R^\ell \mid \mathbf{A}\mathbf{z} = \mathbf{0}\} \subseteq R^\ell.$$

In our setting, the relevant rings are $R = \mathcal{R}$ and $R = \mathcal{R}_q$. The ring $\mathcal{R}$ naturally embeds into $\mathbb{Z}^n$ by mapping a polynomial to its vector of coefficients. Consequently, when $R = \mathcal{R}$, $\mathcal{L}(\mathbf{A})$ and $\mathcal{L}^\perp(\mathbf{A})$ can be viewed as integral lattices in $\mathbb{R}^{nk}$ and $\mathbb{R}^{n\ell}$, respectively. Reducing modulo q yields the corresponding lattices over $\mathcal{R}_q$.

Additional Notation. We let λ denote the security parameter. The notation $a \leftarrow \$ \mathcal{X}$ indicates that a is sampled uniformly from a set $\mathcal{X}$. We use PPT to denote probabilistic polynomial-time algorithms. For a cryptographic scheme **Scheme** consisting of several algorithms, we write **Scheme.A** to denote a specific algorithm **A** within the scheme. $\tilde{\approx}$ denote the statistical indistinguishability.

2.2 Discrete Gaussian Distribution

We recall some standard definitions and properties regarding distributions over lattices [MR04, GPV08, LS15, Pei16]. A *distribution* D over $\mathcal{R}_q$ is a map $D : \mathcal{R}_q \rightarrow [0, 1]$ such that $\sum_{z \in \mathcal{R}_q} D(z) = 1$. Distributions over $\mathcal{R}_q$ are typically obtained by reducing modulo q a distribution – specifically, in our case, a Gaussian distribution – over $\mathcal{R}$, viewed as $\mathbb{Z}^n \subseteq \mathbb{R}^n$.

For any $s > 0$ and $\mathbf{c} \in \mathbb{R}^n$ we define the *Gaussian function* on $\mathbb{R}^n$ centered at $\mathbf{c}$ with scaling parameter s as

$$\varrho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / s^2).$$

Given a lattice $\Lambda \subseteq \mathbb{R}^n$, the *discrete Gaussian distribution* over Λ is

$$D_{\Lambda,s,\mathbf{c}}(\mathbf{x}) = \frac{\varrho_{s,\mathbf{c}}(\mathbf{x})}{\varrho_{s,\mathbf{c}}(\Lambda)},$$

where $\varrho_s(\Lambda)$ is the *Gaussian mass* of Λ ,

$$\varrho_{s,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \varrho_{s,\mathbf{c}}(\mathbf{x}).$$

When omitted, the subscripts s and $\mathbf{c}$ are taken to be 1 and $\mathbf{0}$ respectively, while Λ is taken to be $\mathcal{R} \cong \mathbb{Z}^n$.

We conclude the section with some classic results that are useful to generate secure parameters for lattice-based schemes. To see how the size of the parameter s affects the tails of the distribution

$D_{\mathbb{Z},s,c}$, following [MR04, §3] we define, for any real $\epsilon > 0$, the *smoothing parameter* of Λ , denoted by $\eta_\epsilon(\Lambda)$, as the smallest s such that $\varrho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$. Here Λ^* denotes the *dual lattice*

$$\Lambda^* = \{\mathbf{x} \mid \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z} \ \forall \mathbf{y} \in \Lambda\}.$$

The smoothing parameter, intuitively, describes the smoothness of the distribution obtained by sampling random points on Λ and adding a noise with Gaussian distribution of parameter $\eta_\epsilon(\Lambda)$. The smaller ϵ is, the closer this distribution is to the uniform distribution on $\mathbb{R}^n$.

The following result gives an upper bound for $\eta_\epsilon(\Lambda)$, relating it to the successive minima $\lambda_1, \dots, \lambda_n$ of Λ and Λ^* .

Lemma 1 ([MR04, Lemm. 3.2 and 3.3]).

$$\eta_{1/2^n}(\Lambda) \leq \min \left\{ \frac{\sqrt{n}}{\lambda_1(\Lambda^*)}, \sqrt{\frac{\ln(2n(1+2^n))}{\pi}} \cdot \lambda_n(\Lambda) \right\}.$$

For s larger than the smoothing parameter, one has the following tail inequality on $D_{\mathbb{Z},s}$.

Lemma 2 ([GPV08, Lem. 4.2]). *Let $x \leftarrow D_{\mathbb{Z},s}$. For any $\epsilon > 0$, any $s \geq \eta_\epsilon(\mathbb{Z})$, and any $t > 0$,*

$$\Pr[|x| \geq t \cdot s] \leq 2e^{-\pi t^2} \cdot \frac{1+\epsilon}{1-\epsilon}.$$

We also have, for any s , the following tail inequality for 2-norms on $D_{\mathcal{R},s}$.

Lemma 3 ([PKN⁺25, Lem. 2.1]). *Let $s > 0$. For $\mathbf{x} \leftarrow_{\$} D_{\mathcal{R},s}^k$, we have*

$$\Pr[\|\mathbf{x}\|_2 \geq e^{1/4} s \sqrt{nk}] \leq 2^{-nk/10}.$$

2.3 Lattice Problems

In this subsection we recall some standard lattice problems such as the Module-SIS and Module-LWE, which are formally described in [LS15], and the Hint-MLWE, which is a variant of MLWE described in [KLSS23]. We also introduce the SelfTargetMSIS problem from [DKL⁺18].

Definition 1 (Module-SIS (MSIS) [LS15]). *Let $\mathbf{A} \leftarrow_{\$} \mathcal{R}_q^{k \times \ell}$. The Module-SIS problem with parameters $k, \ell > 0$ and $0 < B < q$ consists of finding $\mathbf{z} \in \mathcal{R}_q^\ell$ such that $\mathbf{A}\mathbf{z} = \mathbf{0}$ over $\mathcal{R}_q$ and $0 < \|\mathbf{z}\|_2 < B$. An adversary $\mathcal{A}$ solves the MSIS $_{k,\ell,B}$ problem with advantage $\text{Adv}_{\mathcal{A}}^{\text{MSIS}}(\lambda)$ if,*

$$\Pr[0 < \|\mathbf{z}\|_2 < B \wedge \mathbf{A}\mathbf{z} = \mathbf{0} \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell} \text{ s.t. } \mathbf{z} \leftarrow \mathcal{A}(\mathbf{A})] \geq \text{Adv}_{\mathcal{A}}^{\text{MSIS}}(\lambda).$$

Definition 2 (Module-MLWE (MLWE) [LS15]). *The Module-LWE problem with parameters $k, \ell > 0$ and an error distribution D over $\mathcal{R}_q$ asks the adversary $\mathcal{A}$ to distinguish between the following two tuples:*

1. $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})$
2. $(\mathbf{A}, \mathbf{b})$,

where $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$, $\mathbf{s} \leftarrow_{\$} D^\ell$ is a secret vector, $\mathbf{e} \leftarrow_{\$} D^k$ an error vector and $\mathbf{b} \leftarrow_{\$} \mathcal{R}_q^k$. $\mathcal{A}$ is said to have advantage $\text{Adv}_{\mathcal{A}}^{\text{MLWE}}(\lambda)$ in solving MLWE $_{k,\ell,D}$ if,

$$\begin{aligned} & \left| \Pr[\text{res} = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}; \mathbf{s} \leftarrow D^\ell; \mathbf{e} \leftarrow D^k; \text{res} \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e})] \right| \\ & - \left| \Pr[\text{res} = 1 \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}; \mathbf{b} \leftarrow \mathcal{R}_q^k; \text{res} \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{b})] \right| \geq \text{Adv}_{\mathcal{A}}^{\text{MLWE}}(\lambda) \end{aligned}$$

Definition 3 (Hint-MLWE (H-MLWE) [KLSS23]). Let k, ℓ, Q be positive integers, D_1 and D_2 be probability distributions over $\mathcal{R}_q$, and C be a subset of $\mathcal{R}_q$. The Hint-MLWE problem $\text{H-MLWE}_{k,\ell,D_1,D_2,Q}$ then asks an adversary $\mathcal{A}$ to distinguish between the following two cases:

1. $(\mathbf{A}, \mathbf{As}, (c_i, \mathbf{z}_i)_{i \in [Q]})$ for $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times (\ell+k)}$
2. $(\mathbf{A}, \mathbf{b}, (c_i, \mathbf{z}_i)_{i \in [Q]})$ for $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times (\ell+k)}, \mathbf{b} \leftarrow \mathcal{R}_q^{k \times (\ell+k)},$

where $\mathbf{s} \leftarrow D_1^{\ell+k}, c_i \leftarrow C$ for $i \in [Q]$, and $\mathbf{z}_i := c_i \cdot \mathbf{s} + \mathbf{y}_i$ where $\mathbf{y}_i \leftarrow D_2^{\ell+k}$ for $i \in [Q]$.

Definition 4 (SelfTargetMSIS [DKL⁺18]). Let k, ℓ be positive integers and $0 < \eta \ll q$. Let $H: \mathcal{R}_q^k \times \{0, 1\}^* \rightarrow \mathcal{C}$ be a cryptographically secure hash function modeled as a random oracle. Then, given $(\mathbf{A}, \mathbf{t}) \in \mathcal{R}_q^{k \times \ell} \times \mathcal{R}_q^k$, the Self-Target MSIS problem asks an adversary $\mathcal{A}$ to find $(\mathbf{z}, \mu) \in \mathcal{R}_q^\ell \times \{0, 1\}^*$ such that $0 < \|\mathbf{z}\|_2 \leq \eta$ and $H([\mathbf{A} \mid \mathbf{t}] \cdot \mathbf{r}, \mu) = c$, where $\mathbf{r} = [\mathbf{z} \ c]^\top$. $\mathcal{A}$ is said to have advantage $\text{Adv}_{\mathcal{A}}^{\text{SelfTargetMSIS}}(\lambda)$ in solving $\text{SelfTargetMSIS}_{k,\ell,\eta}$ if

$$\Pr \left[\begin{array}{l} H([\mathbf{A} \mid \mathbf{t}] \cdot \mathbf{r}, \mu) = c \\ \wedge 0 < \|\mathbf{z}\|_2 \leq \eta \end{array} \mid \mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}; \left(\mathbf{r} = \begin{bmatrix} \mathbf{z} \\ c \end{bmatrix}, \mu \right) \leftarrow \mathcal{A}^H(\mathbf{A}, \mathbf{t}) \right] \geq \text{Adv}_{\mathcal{A}}^{\text{SelfTargetMSIS}}(\lambda).$$

2.4 Blind Signatures

A blind signature (BS) scheme, introduced in [Cha82] and later formalized in [PS96] can be defined as a 4-tuple of PPT algorithms $(\text{Setup}, \text{KGen}, \Pi := (\text{User}, \text{Signer}), \text{Verify})$, where $\Pi := (\text{User}, \text{Signer})$ is an interactive protocol between a user, who holds a message msg , and a signer, who holds a signing key sk defined as below,

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: Takes the security parameter λ as input and outputs public parameters pp .
- $(\text{sk}, \text{vk}) \leftarrow \text{KGen}(\text{pp})$: Take parameters pp as input and output a secret key sk and a verification key vk .
- $\Pi := (\text{User}, \text{Signer})$: This is the interactive protocol between a user and a signer, each entity has a set of subroutines that we describe below,
 - $(\rho_1, \text{state}_{\mathcal{U}}) \leftarrow \text{User}_1(\text{pp}, \text{vk}, \text{msg})$: The user uses a verification key vk and a message $\text{msg} \in \{0, 1\}^*$ to compute a state $\text{state}_{\mathcal{U}}$ and a first message ρ_1 . The message ρ_1 is sent to **Signer**.
 - $(\rho_2, \text{state}_{\mathcal{S}}) \leftarrow \text{Signer}_1(\text{pp}, \text{sk}, \rho_1)$: The signer uses a secret key sk and the first message ρ_1 to compute a state $\text{state}_{\mathcal{S}}$ and a message ρ_2 . The message ρ_2 is sent to **User**.
 - $(\rho_3, \text{state}_{\mathcal{U}}) \leftarrow \text{User}_2(\text{pp}, \text{vk}, \text{state}_{\mathcal{U}}, \rho_2)$: The user uses a public key vk and the message ρ_2 to compute a state $\text{state}_{\mathcal{U}}$ and a message ρ_3 . The message ρ_3 is sent to **Signer**.
 - $\rho_4 \leftarrow \text{Signer}_2(\text{pp}, \text{sk}, \text{state}_{\mathcal{S}}, \rho_3)$: The signer uses a secret key sk and the message ρ_3 to compute a message ρ_4 . The message ρ_4 is sent to **User**.
 - $\sigma \leftarrow \text{User}_3(\text{pp}, \text{vk}, \text{state}_{\mathcal{U}}, \rho_4)$: The last part of the interaction, where the user uses ρ_4 to derive a signature σ or $\perp$ if the signing protocol fails.
- $\text{Verify}(\text{pp}, \text{vk}, \text{msg}, \text{sig})$: A verification algorithm that takes as input a verification key vk , a message msg and a signature sig and outputs a bit to indicate whether the signature is deemed valid (1) or invalid (0).

At the end of the protocol, the user obtains a valid signature sig on msg without learning anything about the signing key, while the signer learns nothing about the message. More precisely, we require the following three properties to hold (as in [Cha82, PS00, SU12]),

Definition 5 (Correctness). A blind signature $\text{BS} = (\text{Setup}, \text{KGen}, (\text{User}, \text{Signer}), \text{Verify})$ is correct if there exists a negligible function $\text{negl}(\lambda)$ such that for any message msg ,

we have that

$$\Pr \left[\text{Verify}(\text{pp}, \text{vk}, \text{msg}, \text{sig}) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ (\text{sk}, \text{vk}) \leftarrow \$ \text{KGen}(\text{pp}), \\ (\rho_1, \text{state}_{\mathcal{U}}) \leftarrow \$ \text{User}_1(\text{pp}, \text{vk}, \text{msg}), \\ (\rho_2, \text{state}_{\mathcal{S}}) \leftarrow \$ \text{Signer}_1(\text{pp}, \text{sk}, \rho_1), \\ (\rho_3, \text{state}_{\mathcal{U}}) \leftarrow \$ \text{User}_2(\text{pp}, \text{vk}, \text{state}_{\mathcal{U}}, \rho_2), \\ \rho_4 \leftarrow \$ \text{Signer}_2(\text{pp}, \text{sk}, \text{state}_{\mathcal{S}}, \rho_3), \\ \text{sig} \leftarrow \$ \text{User}_3(\text{pp}, \text{vk}, \text{state}_{\mathcal{U}}, \rho_4) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

Definition 6 (Blindness). A blind signature scheme BS , consisting of $(\text{Setup}, \text{KGen}, \Pi := (\text{User}, \text{Signer}), \text{Verify})$, is blind if for all polynomial-time adversaries $\mathcal{A}$, there exists a negligible function $\text{Adv}_{\text{BS}}^{\text{BLD}}(\lambda)$ such that for any two message $\text{msg}_0, \text{msg}_1$, we have :

$$\Pr \left[b^* = b \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ (\text{vk}, \text{msg}_0, \text{msg}_1, \text{state}_{\mathcal{S}}) \leftarrow \$ \mathcal{A}(\text{pp}), \\ b \leftarrow \$ \{0, 1\}, \\ \forall b \in \{0, 1\}, (\rho_{1,b}, \text{state}_{\mathcal{U},b}) \leftarrow \$ \text{User}_1(\text{pp}, \text{vk}, \text{msg}_b), \\ (\rho_{2,b}, \rho_{2,1-b}, \text{state}_{\mathcal{S}}) \leftarrow \$ \mathcal{A}(\text{pp}, \text{state}_{\mathcal{S}}, \rho_{1,b}, \rho_{1,1-b}), \\ \forall b \in \{0, 1\}, (\rho_{3,b}, \text{state}_{\mathcal{U},b}) \leftarrow \$ \text{User}_2(\text{pp}, \text{vk}, \text{state}_{\mathcal{U},b}, \rho_{2,b}), \\ (\rho_{4,b}, \rho_{4,1-b}, \text{state}_{\mathcal{S}}) \leftarrow \$ \mathcal{A}(\text{pp}, \text{state}_{\mathcal{S}}, \rho_{3,b}, \rho_{3,1-b}), \\ \forall b \in \{0, 1\}, \text{sig}_b \leftarrow \$ \text{User}_3(\text{pp}, \text{vk}, \text{state}_{\mathcal{U},b}, \rho_{4,b}), \\ \text{if } \perp \in \{\text{sig}_b, \text{sig}_{b-1}\} \text{ then } \{\text{sig}_b, \text{sig}_{b-1}\} = \{\perp, \perp\}, \\ b^* \leftarrow \$ \mathcal{A}(\text{pp}, \text{sig}_b, \text{sig}_{1-b}, \text{state}_{\mathcal{S}}) \end{array} \right] \leq \frac{1}{2} + \text{Adv}_{\text{BS}}^{\text{BLD}}(\lambda).$$

Definition 7 (One-More Unforgeability). Let $Q_s: \mathbb{N} \rightarrow \mathbb{N}$. A blind signature $(\text{Setup}, \text{KGen}, \Pi := (\text{User}, \text{Signer}), \text{Verify})$ is one-more unforgeable if for every polynomial-time adversary $\mathcal{A}$ that makes at most $Q_s(\lambda)$ **concurrent queries** there exists a negligible function $\text{Adv}_{\text{BS}}^{\text{OMUF}}(\lambda)$ such that,

$$\Pr \left[\begin{array}{l} \forall i \in [\ell], \text{Verify}(\text{pp}, \text{vk}, m_i, \text{sig}) = 1 \quad \wedge \\ \forall 1 \leq i \leq j \leq \ell, m_i \neq m_j \end{array} \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ \text{KGen}(\text{pp}) \leftarrow \$ (\text{sk}, \text{vk}), \\ \{(m_i, \text{sig}_i)\}_{i \in [\ell]} \leftarrow \$ \mathcal{A}^{\text{Signer}(\text{pp}, \text{sk}, \cdot)}(\text{pp}) \end{array} \right] \leq \text{Adv}_{\text{BS}}^{\text{OMUF}}(\lambda),$$

where $\ell = Q_s(\lambda) + 1$ and $\text{Signer}(\text{pp}, \text{sk}, \cdot)$ denotes the algorithms $\text{Signer}_1(\text{pp}, \text{sk}, \cdot)$ and $\text{Signer}_2(\text{pp}, \text{sk}, \cdot)$ given as oracle to the adversary, given their respective inputs they output honestly computed ρ_2 and ρ_4 . The adversary plays as the user in the blind signature.

Remark 1 (One-more Strong Unforgeability). In this work, we also consider the notion of one-more strong unforgeability, in which the adversary wins the one-more unforgeability experiment if it produces a fresh signature on an already signed message. One-more strong unforgeability is defined similarly to standard one-more unforgeability, with a slight change in the winning condition of the adversary from

$$\begin{array}{l} \forall 1 \leq i \leq j \leq Q_s(\lambda) + 1, m_i \neq m_j, \text{ to} \\ \forall 1 \leq i \leq j \leq Q_s(\lambda) + 1, (m_i, \text{sig}_i) \neq (m_j, \text{sig}_j). \end{array}$$

3 The Raccoon Signature and Its Modified Variant

In this section, we introduce Raccoon^* , a modified version of Raccoon used as an intermediate step in the security proof of our blind signature. Raccoon^* keeps the same verification equation as Raccoon ,

but modifies the signing procedure so as to capture the distributional shift in the signature induced by the blinding process of our blind signature protocol. We first recall **Raccoon** and then describe **Raccoon***. Finally, we show that the EUF-CMA security of **Raccoon*** reduces to that of **Raccoon**, which will allow us to prove the security of our blind signature.

3.1 Raccoon Signature Scheme

Raccoon [dPPRS23, dPKPR24, dEK⁺23, DKM⁺24] is a rejection-free lattice-based digital signature scheme based on the H-MLWE [KLSS23] and MSIS [LS15] assumptions. In particular, the **Raccoon** variant that we now describe is the one by del Pino *et al.* [DKM⁺24] where the noise distribution is a discrete Gaussian rather than a sum of uniform.

Table 2. Parameters description for **Raccoon**, **Raccoon*** and **BRaccoon**

Parameter	Description
$\mathcal{R}_q$	Polynomial ring $\mathbb{Z}[X]/(q, X^n + 1)$
ℓ	Secret dimension
k	Public key dimension
B_2^*	Norm bound on $(\mathbf{z}, \mathbf{h})$
$B_{\mathbf{w}'}$	Norm bound on $(\mathbf{r}', \mathbf{e}')$
$B_{\mathbf{w}}$	Norm bound on $(\mathbf{r}, \mathbf{e})$
ν_t	Public key rounding parameter
ν_w	Commitment rounding parameter
D_t	Key generation distribution
$D_{\mathbf{w}'}$	Signing randomness distribution
$D_{\mathbf{w}}$	Blinding factor distribution
$\sigma_{\mathbf{w}'}$	Gaussian width of the randomness in $\mathbf{w}'$
$\sigma_{\mathbf{w}}$	Gaussian width of the randomness in $\mathbf{w}$
σ_t	Gaussian width of the user signing key secrets in $\mathbf{t}$

Construction Overview. Let $q_t = \lfloor q \rfloor_{\nu_t}$ and $q_w = \lfloor q \rfloor_{\nu_w}$. The public parameters $\mathbf{pp}$ include the security parameter λ , the ring $\mathcal{R}_q$, and two hash functions H and H_c . The first hash compresses the verification key and the message into a fixed-length digest $\mu = H(\mathbf{vk}, \mathbf{msg}) \in \mathcal{S}$, where in our concrete instantiation $\mathcal{S} = \{0, 1\}^{2\lambda}$. The challenge hash is modeled as

$$H_c : \mathcal{S} \times \mathcal{R}_{q_w}^k \rightarrow \mathcal{C},$$

and maps $(\mu, \mathbf{w})$ to the challenge space $\mathcal{C} = \{c \in \mathcal{R}_q \mid \|c\|_\infty = 1, \|c\|_1 = \omega\}$. In the security proof of **Raccoon**, H_c is modeled as a random oracle. The scheme, given implicitly generated public parameters $\mathbf{pp}$ by a setup algorithm, works as follows:

KGen Samples $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$ and $(\mathbf{s}, \mathbf{e}_{\text{kg}}) \leftarrow D_t^\ell \times D_t^k$, computes $\mathbf{t} = \lfloor \mathbf{A} \cdot \mathbf{s} + \mathbf{e}_{\text{kg}} \rfloor_{\nu_t}$, and outputs $\mathbf{vk} = (\mathbf{A}, \mathbf{t})$ and $\mathbf{sk} = \mathbf{s}$.

Sign On input $(\mathbf{vk}, \mathbf{sk}, \mathbf{msg})$, where $\mathbf{msg}$ is an arbitrary-length string, computes $\mu = H(\mathbf{vk}, \mathbf{msg})$, samples $(\mathbf{r}', \mathbf{e}') \leftarrow D_{\mathbf{w}'}^\ell \times D_{\mathbf{w}'}^k$, and computes $\mathbf{w}' = \lfloor \mathbf{A} \cdot \mathbf{r}' + \mathbf{e}' \rfloor_{\nu_w}$. It then sets $c = H_c(\mu, \mathbf{w}')$ and $\mathbf{z}' = c \cdot \mathbf{s} + \mathbf{r}' \in \mathcal{R}_q^\ell$. Finally, it computes

$$\mathbf{h} = \mathbf{w}' - \lfloor \mathbf{A} \cdot \mathbf{z}' - 2^{\nu_t} \cdot c \cdot \mathbf{t} \rfloor_{\nu_w} \in \mathcal{R}_{q_w}^k$$

and outputs $\mathbf{sig} = (c, \mathbf{z}', \mathbf{h})$.

Verify On input $(\mathbf{vk}, \mathbf{msg}, \mathbf{sig})$, where $\mathbf{sig} = (c, \mathbf{z}', \mathbf{h})$, recomputes $\mu = H(\mathbf{vk}, \mathbf{msg})$ and $\mathbf{w} = \lfloor \mathbf{A} \cdot \mathbf{z}' - 2^{\nu_t} \cdot c \cdot \mathbf{t} \rfloor_{\nu_w} + \mathbf{h}$. It then computes $c' = H_c(\mu, \mathbf{w})$ and accepts if $c = c'$ and $\|(\mathbf{z}', 2^{\nu_w} \cdot \mathbf{h})\|_2 \leq B_2$. Otherwise, it rejects.

KGen(pp)	PreSign(pp)
1 : $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$	1 : $(\mathbf{r}, \mathbf{e}) \leftarrow \mathcal{D}_{\mathbf{w}}^\ell \times \mathcal{D}_{\mathbf{w}}^k$
2 : $(\mathbf{s}, \mathbf{e}_{\text{kg}}) \leftarrow \mathcal{D}_{\mathbf{t}}^\ell \times \mathcal{D}_{\mathbf{t}}^k$	2 : return $(\mathbf{r}, \mathbf{e})$
3 : $\mathbf{t} := \lfloor \mathbf{A}\mathbf{s} + \mathbf{e}_{\text{kg}} \rfloor_{\nu_{\mathbf{t}}}$	
4 : $\text{vk} := (\mathbf{A}, \mathbf{t})$	
5 : $\text{sk} := \mathbf{s}$	
6 : return (sk, vk)	
Sign(pp, state, sk, msg, $\mathbf{r}, \mathbf{e}$)	Verify(pp, vk, sig, msg)
1 : if $(\mathbf{r}, \mathbf{e}) > B_{\mathbf{w}}$ Abort	1 : $(c, \mathbf{z}, \mathbf{h}) \leftarrow \text{parse}(\text{sig})$
2 : $\mu := \text{H}(\text{vk}, \text{msg})$	2 : $\mu := \text{H}(\text{vk}, \text{msg})$
3 : $(\mathbf{r}', \mathbf{e}') \leftarrow \mathcal{D}_{\mathbf{w}'}^\ell \times \mathcal{D}_{\mathbf{w}'}^k$	3 : $c' := \text{H}_c(\mu, \lfloor \mathbf{A}\mathbf{z} - 2^{\nu_{\mathbf{t}}} \mathbf{c}\mathbf{t} \rfloor_{\nu_{\mathbf{w}}} + \mathbf{h})$
4 : $\mathbf{w}' := \mathbf{A}\mathbf{r}' + \mathbf{e}'$	4 : if $c = c'$ and $\ (\mathbf{z}, 2^{\nu_{\mathbf{w}}} \mathbf{h})\ _2 \leq B_2^*$
5 : $\mathbf{w} := \lfloor \mathbf{w}' + \mathbf{A}\mathbf{r} + \mathbf{e} \rfloor_{\nu_{\mathbf{w}}}$	5 : return 1
6 : $c := \text{H}_c(\mu, \mathbf{w})$	6 : return 0
7 : $\mathbf{z} := c\mathbf{s} + \mathbf{r}' + \mathbf{r}$	
8 : $\mathbf{y} := \lfloor \mathbf{A}\mathbf{z} - 2^{\nu_{\mathbf{t}}} \cdot \mathbf{c}\mathbf{t} \rfloor_{\nu_{\mathbf{w}}}$	
9 : $\mathbf{h} := \mathbf{w} - \mathbf{y}$	
10 : $\text{sig} := (c, \mathbf{z}, \mathbf{h})$	
11 : if $\ (\mathbf{z}, 2^{\nu_{\mathbf{w}}} \mathbf{h})\ _2 > B_2^*$ Restart	
12 : return sig	

Fig. 2. Protocol description of Raccoon^* . The protocol is given implicitly generated public parameters pp by a setup algorithm. The values $(\mathbf{r}, \mathbf{e}) \leftarrow \mathcal{D}_{\mathbf{w}}^\ell \times \mathcal{D}_{\mathbf{w}}^k$ are auxiliary inputs that model the blinding process of our blind signature scheme. Note that $\mathcal{D}_{\mathbf{w}}$ is chosen such that $\mathcal{D}_{\mathbf{w}+\mathbf{w}'} \stackrel{s}{\approx} \mathcal{D}_{\mathbf{w}}$ which implies that one cannot link $\mathbf{w}'$ to $\mathbf{w}$ (resp. $c\mathbf{s} + \mathbf{r}'$ to $c\mathbf{s} + \mathbf{r}' + \mathbf{r}$)

Lemma 4 (Theorem 5.1 [DKM⁺24]). *Let Q_s be the number of signing queries. The signature scheme described above is correct and EUF-CMA secure under the $\text{H-MLWE}_{q,\ell,k,Q_s,\sigma_{\mathbf{t}},\sigma_{\mathbf{w}'},\mathcal{C}}$ and $\text{SelfTargetMSIS}_{q,\ell+1,k,\mathcal{C},B_{\text{MSIS}}}$ assumptions.*

3.2 Raccoon^* : A Modified Version of Raccoon

We now provide a description of Raccoon^* . This signature scheme models the behavior of our blind signature construction introduced in Section 4. To this end, we first mimic the behavior of Raccoon , and then apply a shift to the signature in order to capture the blindness mechanism of our blind signature. This transformation is crucial for the security proof, as without it the views of the adversary in consecutive games would no longer be indistinguishable. A complete description of the key generation, signing, and verification algorithms is given in Figure 2.

Security of Raccoon^* . We will now show that the security of Raccoon^* reduces to the security of Raccoon . Note that since the schemes are really close to each other, there will be redundancy with the proof of the correctness of the underlying signature scheme in [DKM⁺24].

Theorem 1 (Correctness). *The Raccoon^{*} signature scheme in Fig. 2 is correct, where ω is the 1-norm of c , and, if $\nu_{\mathbf{w}} \geq 4$, signatures are bounded by*

$$B_2^* = e^{\frac{1}{4}}(\omega\sigma_{\mathbf{t}} + \sigma_{\mathbf{w}'} + \sigma_{\mathbf{w}})\sqrt{n(k+\ell)} + (\omega 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1})\sqrt{nk}.$$

Proof. See Appendix C.1 for detailed calculations.

Theorem 2 (Unforgeability). *Consider an adversary $\mathcal{A}$ breaking the EUF-CMA of Raccoon^{*} in Figure 2 making at most Q_{H_c} and Q_s queries to the random oracle H_c and the signing oracle, respectively, then there exist an adversary $\mathcal{B}$ breaking the EUF-CMA of Raccoon such that*

$$\text{Adv}_{\text{Raccoon}^*, \mathcal{A}}^{\text{EUF-CMA}}(\lambda) \leq \text{Adv}_{\text{Raccoon}, \mathcal{B}}^{\text{EUF-CMA}}(\lambda)$$

Proof (sketch). The proof proceeds through a sequence of games. The first game corresponds to the EUF-CMA experiment instantiated with Raccoon^{*}. In the second game, we simulate the hash oracle via lazy sampling. Finally, in the last game, we simulate the signing oracle of Raccoon^{*} using the signing oracle of Raccoon, by adding a shift to the Raccoon signature and programming the oracle on c , which concludes the argument. See Appendix C.2 for the full proof.

4 Concurrently Secure Blind Signature from Raccoon

We now introduce BRaccoon, a concurrently secure blind signature scheme whose security reduces to the Raccoon^{*} signature scheme, whose security, in turn, reduces to the Raccoon signature scheme. The protocol consists of four moves and produces signatures that are syntactically identical to standard Raccoon signatures, up to a suitable parameter adjustment that allows blinding.

4.1 Protocol Overview

Our blind signature protocol follows a design paradigm inspired by the “blind signatures from a signature assumption” paradigm of Fuchsbauer–Wolf [FW24] in combination with subsequent threshold lattice signatures [GKS24, GHK⁺25], combining commitment schemes, public-key encryption, and NIZK proofs. At a high level, the user computes a commitment $C_{\text{msg}, \mathbf{r}}$ to the message together with a blinding factor (using a commitment scheme that in the security proof allows extraction). The signer responds with a commitment $\mathbf{w}'$ following the underlying signature scheme. The user then re-randomizes this commitment using randomness $\mathbf{r}$, computes the signature challenge, and encrypts it (ctx_c) under a linearly homomorphic encryption scheme, proving correctness in zero-knowledge (using language L_1). The signer completes the signature computation in the encrypted domain and returns an encrypted response $\text{ctx}_{\mathbf{z}'}$ along with a NIZK proof of correct computation (using language L_2), which the user decrypts and unblinds to obtain the final signature $(c, \mathbf{z})$.

By construction, the output is a standard signature of the underlying scheme, while blindness and one-more unforgeability are enforced through commitments, encryption, and NIZK proofs. Figures 3 and 4 detail the different algorithms that define our blind signature scheme. We denote by $\mathbf{pp}$ the collection of public parameters required to instantiate all the cryptographic building blocks used in our blind signature scheme. In the real protocol, these parameters are generated transparently from public seeds (or, equivalently, from the root of a PRG tree): all public matrices and auxiliary public values used by the commitment schemes, the homomorphic encryption scheme, and the NIZK proof systems are deterministically expanded from these seeds. Hence, no trusted setup is required, and the explicit public parameters may be generated on demand. In the security reductions, we model this seed expansion as a random oracle.

Language descriptions. To achieve blindness and one-more unforgeability, our blind signature protocol relies on several non-interactive zero-knowledge proofs for carefully designed languages. These languages enforce consistency between protocol rounds and enable the security reductions. For readability, we omit explicit domain specifications whenever they are clear from context.

$$L_1 := \left\{ \begin{array}{l} (C_{\text{msg}, \mathbf{r}, \mathbf{e}}, \text{ctx}_c, \mathbf{w}', \\ \text{vk} = (\mathbf{A}, \mathbf{t}), \text{ek}, \text{pp}) \end{array} \middle| \begin{array}{l} \exists (\mathbf{r}, \mathbf{e}, \mathbf{w}, \text{msg}) \text{ s.t.}, \\ \text{CS.Verify}(\text{pp}, (\text{msg}, \mathbf{r}, \mathbf{e}), C_{\text{msg}, \mathbf{r}, \mathbf{e}}) = 1, \\ \mathbf{w} = \lceil \mathbf{w}' + \mathbf{A}\mathbf{r} + \mathbf{e} \rceil_{\nu_{\mathbf{w}}}, \\ \mu = H(\text{vk}, \text{msg}) \\ c = H_c(\mu, \mathbf{w}), \\ \text{ctx}_c = \text{HE.Enc}(\text{pp}, \text{ek}, c) \end{array} \right\}.$$

The language L_1 enforces honest computation of the challenge c and its encryption. In particular, it guarantees that the challenge is derived consistently from the committed message msg and the corresponding commitment $\mathbf{w}$, and that the ciphertext ctx_c encrypts exactly this value. Moreover, L_1 links the randomness used across protocol rounds, preventing the user from adapting the challenge to the signer's behavior.

$$L_2 := \left\{ \begin{array}{l} (\text{ctx}_{\mathbf{z}'}, \text{ctx}_c, \text{ctx}_{\mathbf{r}'}, \mathbf{w}', \\ \text{vk} = (\mathbf{A}, \mathbf{t}), \text{ek}, B_{\mathbf{w}'}, \text{pp}) \end{array} \middle| \begin{array}{l} \exists (\mathbf{r}', \mathbf{e}', \mathbf{s}, \mathbf{e}_{\text{kg}}) \text{ s.t.}, \\ \mathbf{w}' = \mathbf{A}\mathbf{r}' + \mathbf{e}', \\ \mathbf{t} = \lceil \mathbf{A}\mathbf{s} + \mathbf{e}_{\text{kg}} \rceil_{\nu_{\mathbf{t}}}, \\ \text{ctx}_{\mathbf{r}'} = \text{HE.Enc}(\text{pp}, \text{ek}, \mathbf{r}'), \\ \text{ctx}_{\mathbf{z}'} = \text{HE.Eval}(\text{pp}, \text{ek}, f_{\mathbf{s}}, (\text{ctx}_c, \text{ctx}_{\mathbf{r}'})), \\ \|\mathbf{r}', \mathbf{e}'\|_2 \leq B_{\mathbf{w}'} \end{array} \right\}.$$

The language L_2 ensures that the signer computes the encrypted response honestly. In particular, it enforces that $\text{ctx}_{\mathbf{z}'}$ is obtained by homomorphically combining a fresh encryption of $\mathbf{r}'$ with the encrypted challenge and the signing key. This prevents a malicious signer from embedding subliminal information or deviating from the prescribed computation, which is essential for blindness. Here and in the protocol below, f is defined as the function $f_a(b, c) = a \cdot b + c$.

KGen(pp)	Verify(pp, vk, sig, msg)
1 : $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times \ell}$	1 : $(c, \mathbf{z}, \mathbf{h}) := \text{parse}(\text{sig})$
2 : $(\mathbf{s}, \mathbf{e}_{\text{kg}}) \leftarrow \mathcal{D}_{\mathbf{t}}^{\ell} \times \mathcal{D}_{\mathbf{t}}^k$	2 : $\mu := H(\text{vk}, \text{msg})$
3 : $\mathbf{t} := \lceil \mathbf{A}\mathbf{s} + \mathbf{e}_{\text{kg}} \rceil_{\nu_{\mathbf{t}}}$	3 : $c' := H_c(\mu, \lceil \mathbf{A}\mathbf{z} - 2^{\nu_{\mathbf{t}}} \mathbf{c} \rceil_{\nu_{\mathbf{w}}} + \mathbf{h})$
4 : $\text{vk} := (\mathbf{A}, \mathbf{t})$	4 : if $c = c'$ And $\ (\mathbf{z}, 2^{\nu_{\mathbf{w}}} \mathbf{h})\ _2 \leq B_2^*$ then
5 : $\text{sk} := \mathbf{s}$	5 : return 1
6 : return (sk, vk)	6 : return 0

Fig. 3. KeyGen and Verify algorithms of BRaccoon.

4.2 Security of the Protocol

Theorem 3 (Correctness). *Let $B_2^* = e^{\frac{1}{4}}(\omega\sigma_{\mathbf{t}} + \sigma_{\mathbf{w}'} + \sigma_{\mathbf{w}}) \sqrt{n(k+\ell)} + (\omega 2^{\nu_{\mathbf{t}}} + 2^{\nu_{\mathbf{w}}+1})\sqrt{nk}$ where ω is the 1-norm of c . Let $\text{sig} = (c, \mathbf{z}, \mathbf{h})$ be a blind signature generated by honest participants using our protocol BRaccoon. We claim that the signature sig is valid with overwhelming probability.*

Proof. Since Raccoon* faithfully models the signing procedure of our blind signature protocol in the honest-participant setting, the correctness of our blind signature scheme follows directly from the correctness of Raccoon* and the correctness of the underlying primitives, including the non-interactive zero-knowledge proofs NIZK, the homomorphic encryption HE, and the commitment scheme CS.

User₁ (pp, vk, msg, ek, dk)	Signer₁ (pp, ek, sk, vk, C')
1 : $(\mathbf{r}, \mathbf{e}) \leftarrow D_{\mathbf{w}}^{\ell} \times D_{\mathbf{w}}^k$ 2 : $C_{\text{msg}, \mathbf{r}, \mathbf{e}} := \text{CS.Com}(\text{pp}, (\mathbf{r}, \mathbf{e}, \text{msg}))$ 3 : $\text{state}_U := (\mathbf{r}, \mathbf{e}, \text{msg}, \text{vk}, \text{ek}, \text{dk})$ 4 : return $(C_{\text{msg}, \mathbf{r}, \mathbf{e}}, \text{state}_U)$	1 : $(\mathbf{r}', \mathbf{e}') \leftarrow D_{\mathbf{w}'}^{\ell} \times D_{\mathbf{w}'}^k$ 2 : $\mathbf{w}' := \mathbf{A}\mathbf{r}' + \mathbf{e}'$ 3 : $\text{state}_S := (\text{ek}, \text{sk}, \text{vk}, \mathbf{r}', \mathbf{e}')$ 4 : return $(\mathbf{w}', \text{state}_S)$
User₂ (pp, state _U , w')	Signer₂ (pp, state _S , π ₁ , ctx _c)
1 : $\mathbf{w} := \lfloor \mathbf{w}' + \mathbf{A}\mathbf{r} + \mathbf{e} \rfloor_{\nu_{\mathbf{w}}}$ 2 : $\mu := H(\text{vk}, \text{msg})$ 3 : $c := H_c(\mu, \mathbf{w})$ 4 : $\text{ctx}_c := \text{HE.Enc}(\text{pp}, \text{ek}, c)$ 5 : $\pi_1 := \text{NIZK}_{L_1}.\text{Prove}(\text{pp}, \text{msg}, \mathbf{r}, \mathbf{e}, \mathbf{w}, \mu)$ 6 : $\text{state}_U := \text{state}_U (c, \mathbf{w})$ 7 : return $(\text{state}_U, \pi_1, \text{ctx}_c)$	1 : if $\text{NIZK}_{L_1}.\text{Verify}(\pi_1) = 0$ then 2 : abort 3 : $\text{ctx}_{\mathbf{r}'} := \text{HE.Enc}(\text{pp}, \text{ek}, \mathbf{r}')$ 4 : $\text{ctx}_{\mathbf{z}'} := \text{HE.Eval}(\text{pp}, \text{ek}, f, (\text{ctx}_c, \mathbf{s}, \text{ctx}_{\mathbf{r}'}))$ 5 : $\pi_2 := \text{NIZK}_{L_2}.\text{Prove}(\text{pp}, \mathbf{r}', \mathbf{e}', \mathbf{s}, \text{e}_{\text{kg}})$ 6 : return $(\pi_2, \text{ctx}_{\mathbf{z}'})$
User₃ (pp, state _U , π ₂ , ctx' _z)	
1 : if $\text{NIZK}_{L_2}.\text{Verify}(\pi_2) = 0$ then 2 : abort 3 : $\mathbf{z}' := \text{HE.Dec}(\text{pp}, \text{dk}, \text{ctx}_{\mathbf{z}'})$ 4 : $\mathbf{z} := \mathbf{z}' + \mathbf{r}$ 5 : $\mathbf{y} := \lfloor \mathbf{A}\mathbf{z} - 2^{\nu_t} \text{ct} \rfloor_{\nu_{\mathbf{w}}}$ 6 : $\mathbf{h} := \mathbf{w} - \mathbf{y}$ 7 : return $\text{sig} = (c, \mathbf{z}, \mathbf{h})$	

Fig. 4. User and Signer subroutines for BRaccoon.

We now show that the blind signature scheme BRaccoon is one-more (strongly) unforgeable assuming the underlying signature scheme Raccoon^{*} is (strongly) unforgeable. Concretely, any adversary that wins the one-more (strong) unforgeability game against BRaccoon can be converted into an adversary that forges Raccoon^{*}. Thus, the security of Raccoon^{*} (and of the base Raccoon scheme) implies the security of BRaccoon. What is more, this directly implies that one-more (strong) unforgeability of BRaccoon holds concurrently. We formalize this below.

Theorem 4 (One-More Unforgeability). *Consider an adversary $\mathcal{A}$ breaking the one-more unforgeability of our protocol BRaccoon in Figure 1. Then there exist PPT adversaries $\mathcal{R}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{V}$ such that $\mathcal{R}$ breaks the EUF-CMA security of Raccoon^{*} with $Q_s(\lambda)$ queries to the signing oracle $\mathcal{O}^{\text{Raccoon}^*}$, $\mathcal{B}$ breaks the setup indistinguishability of the commitment scheme CS, $\mathcal{C}$ breaks the extractability of CS, $\mathcal{D}$ breaks the zero-knowledge property of π_2 , $\mathcal{E}$ breaks the soundness of π_1 , and $\mathcal{V}$ breaks the circuit privacy of HE. In particular, for any security parameter λ , we have:*

$$\begin{aligned} \mathcal{A}_{\text{OMUF}}^{\text{BRaccoon}}(\mathcal{A}) &\leq \text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q,n,k_{\text{ext}},l_{\text{ext}},D_{\text{ext}}}}(\lambda) + \frac{Q_{\text{setup}}}{2^\kappa} \\ &\quad + Q_s \cdot \text{Adv}_{\mathcal{C},\mathcal{C}}^{\text{Ext}}(\lambda) \\ &\quad + Q_s \cdot \text{Adv}_{\pi_2,\mathcal{D}}^{\text{ZK}}(\lambda) \\ &\quad + Q_s \cdot \text{Adv}_{\pi_1,\mathcal{E}}^{\text{SND}}(\lambda) \\ &\quad + Q_s \cdot \text{Adv}_{\text{HE},\mathcal{V}}^{\text{CP}}(\lambda) \\ &\quad + \text{Adv}_{\text{Raccoon},\mathcal{R}}^{\text{EUF-CMA}}(\lambda). \end{aligned}$$

Proof. We prove one-more unforgeability via a sequence of hybrid games, each introducing a controlled modification, until we obtain a reduction to the EUF-CMA security of Raccoon^{*}.

G₀: We start with the initial game **G₀**, in which the protocol is executed honestly and the adversary $\mathcal{A}$ interacts with the OMUF game instantiated with our protocol shown in Figure 3 and Figure 4. By definition of the OMUF game, the adversary $\mathcal{A}$ issues at most Q_s concurrent signing queries. The reduction $\mathcal{R}$ forwards all messages from the signer to the adversary without any modification. Let S_0 denote the event that the adversary wins in game **G₀**.

G₁: In game **G₁**, we modify the way the public parameters of the commitment scheme are generated. Recall that, in the real protocol, the public matrices used by CS are not sampled explicitly by a trusted party, but are deterministically expanded from public seeds. We model this seed expansion as a random oracle $\mathcal{O}_{\text{setup}}$. Instead of answering the setup-expansion queries honestly, the challenger first samples the public seeds that will define the parameters of CS and then programs $\mathcal{O}_{\text{setup}}$ on these seeds so that the resulting matrices are distributed as in the simulated setup

$$(\text{pp}_{\text{CS}}, \mathcal{T}) \leftarrow \text{Setup}_{\text{Ext}}(1^\lambda).$$

All subsequent queries to $\mathcal{O}_{\text{setup}}$ are answered consistently with this programming, while all other oracle queries are answered as in the real game. The only way in which this programming can be detected before the public seeds are revealed is if the adversary queries $\mathcal{O}_{\text{setup}}$ on one of these seeds in advance. Let $\text{Bad}_{\text{setup}}$ denote this event. If the seeds have length κ and the adversary makes at most Q_{setup} setup-oracle queries, then

$$\Pr[\text{Bad}_{\text{setup}}] \leq \frac{Q_{\text{setup}}}{2^\kappa}.$$

Conditioned on $\neg \text{Bad}_{\text{setup}}$, the adversary's view differs from the real setup only in that the public matrices are generated according to the simulated setup distribution. Hence, by setup indistinguishability of the commitment parameters, there exists a PPT adversary $\mathcal{B}$ such that

$$|\Pr[S_0] - \Pr[S_1]| \leq \text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q,n,k_{\text{ext}},l_{\text{ext}},D_{\text{ext}}}}(\lambda) + \frac{Q_{\text{setup}}}{2^\kappa}.$$

For our BDLOP-based instantiation, setup indistinguishability follows from the MLWE assumption, since the simulated matrices have the form $\mathbf{B} = \mathbf{S}\mathbf{A} + p\mathbf{E}$ for short $\mathbf{S}, \mathbf{E}$, which is computationally indistinguishable from uniform.

G₂: In game **G₂**, we modify the subroutine Signer_1 so that, upon receiving a commitment $C_{\text{msg}, \mathbf{r}, \mathbf{e}}$ from the adversary, it uses the trapdoor $\mathcal{T}$ to extract the values $(\text{msg}, \mathbf{r}, \mathbf{e})$. If the extraction fails, the subroutine outputs $\perp$ and aborts. Let S_2 be the event that the adversary wins in game **G₂**, and let F_2 denote the event that extraction fails. The two games **G₁** and **G₂** are identical unless the event F_2 occurs, that is,

$$S_1 \wedge \neg F_2 \iff S_2 \wedge \neg F_2.$$

Therefore, by the difference lemma, we have

$$|\Pr[S_1] - \Pr[S_2]| \leq \Pr[F_2].$$

By the extractability of the commitment scheme, there exists a PPT adversary $\mathcal{B}$ such that

$$\Pr[F_2] \leq Q_s \cdot \text{Adv}_{\text{CS}, \mathcal{C}}^{\text{Ext}}(\lambda), \quad (1)$$

where Q_s is the number of signing queries. Combining the above bounds, we obtain,

$$|\Pr[S_1] - \Pr[S_2]| \leq Q_s \cdot \text{Adv}_{\text{CS}, \mathcal{C}}^{\text{Ext}}(\lambda).$$

G₃: In game **G₃**, the proof π_2 forwarded by the subroutine Signer_2 is replaced by its simulated proof. Let S_3 denote the event that the adversary wins in game **G₃**. By the zero-knowledge property and a union bound over the at most Q_s signing queries, we obtain

$$|\Pr[S_2] - \Pr[S_3]| \leq Q_s \cdot \text{Adv}_{\pi_2, \mathcal{D}}^{\text{ZK}}(\lambda).$$

G₄: In game **G₄**, we modify the subroutine Signer_2 so that the challenge c is recomputed from the values $(\text{msg}, \mathbf{r}, \mathbf{e})$ extracted from the commitment $C_{\text{msg}, \mathbf{r}, \mathbf{e}}$.

More precisely, the reduction computes

$$\mu = H(\text{vk}, \text{msg}) \quad \text{and} \quad c = H_c(\mu, \mathbf{w}),$$

where

$$\mathbf{w} = \lfloor \mathbf{w}' + \mathbf{A}\mathbf{r} + \mathbf{e} \rfloor_{\nu_w}.$$

The ciphertext $\text{ctx}_{z'}$ and all remaining components are computed exactly as in **G₃**. Let S_4 denote the event that the adversary wins in game **G₄**.

We now show that any adversary distinguishing **G₃** from **G₄** yields an adversary against the soundness of the proof system π_1 .

Suppose there exists a distinguisher $\mathcal{A}$ that achieves a non-negligible advantage between the two games. We construct an adversary $\mathcal{B}$ breaking soundness of π_1 as follows. Adversary $\mathcal{B}$ receives a statement $(\text{ctx}_c, C_{\text{msg}, \mathbf{r}, \mathbf{e}}, \mathbf{w}', \dots)$ and a purported proof π_1 , and simulates the execution of the protocol for $\mathcal{A}$. Upon receiving a commitment $C_{\text{msg}, \mathbf{r}, \mathbf{e}}$ from $\mathcal{A}$, $\mathcal{B}$ uses the extractor (from Game **G₂**) to recover $(\text{msg}, \mathbf{r}, \mathbf{e})$ and recomputes

$$\mu = H(\text{vk}, \text{msg}), \quad c' = H_c(\mu, \mathbf{w}).$$

At this point, two cases may occur:

- If $c' = c$ (where c is the value encoded in the statement of π_1 , i.e., the value such that π_1 proves correctness of $c = H_c(\mu, \mathbf{w})$), then the view of $\mathcal{A}$ is distributed exactly as in **G₃**.
- If $c' \neq c$, then $\mathcal{B}$ outputs the statement together with π_1 as a violation of soundness of π_1 , since π_1 is accepted while corresponding to inconsistent values of the committed data and the derived challenge.

Therefore, distinguishing between $\mathbf{G}_3$ and $\mathbf{G}_4$ implies that $\mathcal{A}$ produces, with non-negligible probability, a valid proof π_1 for a statement where the challenge encoded in the proof is inconsistent with the recomputed one. This directly violates the soundness property of π_1 .

By a union bound over the at most Q_s signing sessions, there exists a PPT adversary $\mathcal{B}$ such that

$$|\Pr[S_3] - \Pr[S_4]| \leq Q_s \cdot \text{Adv}_{\pi_1, \mathcal{E}}^{\text{SND}}(\lambda).$$

G₅: In game $\mathbf{G}_5$, we modify the subroutine Signer_2 by changing the way the ciphertext $\text{ctx}_{z'}$ is computed. Instead of computing

$$\text{ctx}_{z'} \leftarrow \text{HE.Eval}(\text{pp}, \text{ek}, f, (\text{ctx}_c, \mathbf{s}, \text{ctx}_{\mathbf{r}'})),$$

we compute

$$\widetilde{\text{ctx}}_{z'} \leftarrow \text{HE.Enc}(\text{pp}, \text{ek}, f(c, \mathbf{s}, \mathbf{r}')).$$

All other components of the protocol remain unchanged.

Let S_5 denote the event that the adversary wins in game $\mathbf{G}_5$.

We first observe that in both $\mathbf{G}_4$ and $\mathbf{G}_5$, the resulting ciphertext encrypts the same plaintext value, namely

$$f(c, \mathbf{s}, \mathbf{r}') = c\mathbf{s} + \mathbf{r}'.$$

The only difference between the two games lies in the distribution of the ciphertext: in $\mathbf{G}_4$, it is obtained by homomorphic evaluation using HE.Eval , whereas in $\mathbf{G}_5$, it is produced as a fresh encryption using HE.Enc .

We assume circuit privacy in the sense that the output of HE.Eval is computationally indistinguishable from a fresh encryption of the evaluated plaintext. Then, by the circuit privacy property of the homomorphic encryption scheme HE , the output of the evaluation algorithm on ciphertexts encrypting $(c, \mathbf{r}')$ is computationally indistinguishable from a fresh encryption of the evaluated plaintext $f(c, \mathbf{s}, \mathbf{r}')$. More precisely, for any efficient distinguisher $\mathcal{D}$, we have

$$\text{HE.Eval}(\text{pp}, \text{ek}, f, (\text{ctx}_c, \mathbf{s}, \text{ctx}_{\mathbf{r}'})) \approx_c \text{HE.Enc}(\text{pp}, \text{ek}, f(c, \mathbf{s}, \mathbf{r}')).$$

Therefore, there exists a PPT adversary $\mathcal{V}$ such that

$$|\Pr[S_4] - \Pr[S_5]| \leq Q_s \cdot \text{Adv}_{\text{HE}, \mathcal{V}}^{\text{CP}}(\lambda),$$

where the factor Q_s accounts for a union bound over all signing queries.

G₆: In game $\mathbf{G}_6$, we complete the reduction to the EUF-CMA security of Raccoon^* . Recall that, from Game $\mathbf{G}_2$ onward, the reduction extracts $(\text{msg}, \mathbf{r}, \mathbf{e})$ from each commitment $C_{\text{msg}, \mathbf{r}, \mathbf{e}}$. By the consistency argument given earlier, these values correspond to those used throughout the execution of the protocol.

For each signing session, the reduction proceeds as follows. Upon receiving a signing request on $(\text{msg}, \mathbf{r}, \mathbf{e})$, it queries the Raccoon^* signing oracle and obtains a signature

$$(\mathbf{z}^*, c^*, \mathbf{h}^*), \quad \text{where } \mathbf{z}^* = c^*\mathbf{s} + \mathbf{r}^* + \mathbf{r}.$$

The reduction then simulates the protocol transcript by setting

$$\mathbf{w}' := \mathbf{A}\mathbf{z}^* + c^*\mathbf{t} - \mathbf{A}\mathbf{r} + \mathbf{e}.$$

By construction of Raccoon^* , this yields

$$\mathbf{w}' = \mathbf{A}\mathbf{r}^* + \mathbf{e}^*,$$

which matches the distribution of $\mathbf{w}'$ in an honest execution.

Furthermore, instead of computing the ciphertext via homomorphic evaluation, the reduction sets

$$\widetilde{\text{ctx}}_{z'} \leftarrow \text{HE.Enc}(\text{pp}, \text{ek}, \mathbf{z}^* - \mathbf{r}),$$

which encrypts the same plaintext value used in previous games.

It follows that, for each signing session, the simulated transcript is identically distributed to an honestly generated execution of the protocol producing a valid Raccoon^* signature on the extracted message msg .

Let $\mathcal{L}$ denote the set of messages corresponding to completed signing sessions. By construction, we have $|\mathcal{L}| \leq Q_s$.

At the end of the experiment, the adversary $\mathcal{A}$ outputs a set of pairs

$$\{(\text{msg}_j^*, \sigma_j^*)\}_{j \in [n]}.$$

If the event S_6 occurs, then all output signatures are valid and pairwise distinct. Since the adversary interacts in at most Q_s signing sessions but outputs $n \geq Q_s + 1$ valid signatures, there must exist an index j^* such that $\text{msg}_{j^*}^* \notin \mathcal{L}$.

Therefore, $(\text{msg}_{j^*}^*, \sigma_{j^*}^*)$ constitutes a valid EUF-CMA forgery for Raccoon^* , i.e., a valid signature on a message that was never submitted to the signing oracle. Thus, there exists a PPT adversary $\mathcal{R}$ such that

$$\Pr[S_6] \leq \text{Adv}_{\text{Raccoon}^*, \mathcal{R}}^{\text{EUF-CMA}}(\lambda).$$

Finally, since the unforgeability of Raccoon^* reduces to that of Raccoon , we obtain

$$\text{Adv}_{\text{Raccoon}^*, \mathcal{R}}^{\text{EUF-CMA}}(\lambda) \leq \text{Adv}_{\text{Raccoon}, \mathcal{R}}^{\text{EUF-CMA}}(\lambda),$$

which concludes the reduction. Combining the bounds from all hybrid games, we obtain that for any PPT adversary $\mathcal{A}$,

$$\begin{aligned} \mathcal{A}^{\text{BRaccoon}}_{\text{OMUF}}(\mathcal{A}) &\leq \text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q, n, k_{\text{ext}}, l_{\text{ext}}, D_{\text{ext}}}}(\lambda) + \frac{Q_{\text{setup}}}{2^\kappa} \\ &\quad + Q_s \cdot \text{Adv}_{\text{CS}, \mathcal{C}}^{\text{Ext}}(\lambda) \\ &\quad + Q_s \cdot \text{Adv}_{\pi_2, \mathcal{D}}^{\text{ZK}}(\lambda) \\ &\quad + Q_s \cdot \text{Adv}_{\pi_1, \mathcal{E}}^{\text{SND}}(\lambda) \\ &\quad + Q_s \cdot \text{Adv}_{\text{HE}, \mathcal{V}}^{\text{CP}}(\lambda) \\ &\quad + \text{Adv}_{\text{Raccoon}, \mathcal{R}}^{\text{EUF-CMA}}(\lambda). \end{aligned}$$

□

Lemma 5 (One-More Strong Unforgeability). *Assume that Raccoon^* is SUF-CMA secure. Then the same reduction as in Theorem 4 gives one-more strong unforgeability of BRaccoon . More precisely, for every PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{R}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{V}$ such that*

$$\begin{aligned} \text{Adv}_{\text{BRaccoon}, \mathcal{A}}^{\text{OMSUF}}(\lambda) &\leq \text{Adv}_{\text{Raccoon}, \mathcal{R}}^{\text{SUF-CMA}}(\lambda) \\ &\quad + \text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q, n, k_{\text{ext}}, l_{\text{ext}}, D_{\text{ext}}}}(\lambda) \\ &\quad + Q_s \cdot \left(\text{Adv}_{\text{CS}, \mathcal{C}}^{\text{Ext}}(\lambda) + \text{Adv}_{\pi_1, \mathcal{E}}^{\text{SND}}(\lambda) + \text{Adv}_{\pi_2, \mathcal{D}}^{\text{ZK}}(\lambda) + \text{Adv}_{\text{HE}, \mathcal{V}}^{\text{CP}}(\lambda) \right). \end{aligned}$$

Proof (Sketch). The proof follows the same hybrid sequence as the proof of one-more unforgeability. The only difference is in the final reduction. After the hybrids, each completed signing session corresponds to one signing query to Raccoon^* on the extracted message. If $\mathcal{A}$ wins the one-more strong unforgeability game, then it outputs either a signature on a fresh message or a new valid signature on a previously queried message. This is exactly a SUF-CMA forgery against Raccoon^* , instead of merely an EUF-CMA forgery. Hence the same hybrid losses apply, with the final term replaced by $\text{Adv}_{\text{Raccoon}^*, \mathcal{R}}^{\text{SUF-CMA}}(\lambda)$. Finally,

$$\text{Adv}_{\text{Raccoon}^*, \mathcal{R}}^{\text{SUF-CMA}}(\lambda) \leq \text{Adv}_{\text{Raccoon}, \mathcal{R}}^{\text{SUF-CMA}}(\lambda).$$

Theorem 5 (Blindness). *Let $\mathcal{A}$ be any PPT adversary that wins the blindness experiment instantiated with our blind signature in Figure 1 with advantage $\text{Adv}_{\text{BRaccoon}, \mathcal{A}}^{\text{BLD}}(\lambda)$. Then there exist PPT adversaries $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{K}$ and $\mathcal{S}$ such that, for the security parameter λ , we have*

$$\begin{aligned} \text{Adv}_{\text{BRaccoon}, \mathcal{A}}^{\text{BLD}}(\lambda) &\leq \text{Adv}_{\pi_1, \mathcal{B}}^{\text{ZK}}(\lambda) + \text{Adv}_{\text{CS}, \mathcal{C}}^{\text{HID}}(\lambda) \\ &\quad + \text{Adv}_{\pi_2, \mathcal{K}}^{\text{KS}}(\lambda) + \text{Adv}_{\text{HE}, \mathcal{S}}^{\text{IND-CPA}}(\lambda) + \epsilon_{\text{Stat}} \end{aligned}$$

Proof (Sketch). We prove blindness through a sequence of hybrid games. We start from the real blindness experiment. The first hybrid replaces the user proof π_1 by a simulated proof, which is indistinguishable by the zero-knowledge property of π_1 .

We then replace the commitment to $(\text{msg}, \mathbf{r}, \mathbf{e})$ by a commitment to an independently sampled message. By the hiding property of the commitment scheme, this hides the branch message used by the challenger. Next, after verifying the signer proof π_2 , the challenger extracts the corresponding witness. Any difference from the previous game is bounded by the knowledge error of π_2 .

Using the extracted signing key and first-round randomness, the challenger recomputes the response as

$$\mathbf{z} = \text{cs} + \mathbf{r}' + \mathbf{r},$$

which has the same distribution as in the real protocol. We then replace the ciphertext ctx_c by an encryption of an independently sampled challenge value, relying on the IND-CPA security of the homomorphic encryption scheme.

In the final hybrid, all values visible to the adversary are independent of the hidden bit b : the proof is simulated, the commitment hides the message, the ciphertext encrypts an independent value, and the response is computed from extracted signer values together with fresh user blinding randomness. The only remaining dependence on signer randomness is statistically hidden by the larger blinding noise $\mathbf{r}$, giving statistical distance at most ϵ_{Stat} .

Thus, the adversary's blindness advantage is bounded by

$$\mathcal{A}_{\text{BLD}}^{\text{BRaccoon}}(\mathcal{A}) \leq \text{Adv}_{\pi_1}^{\text{ZK}}(\lambda) + \text{Adv}_{\text{CS}}^{\text{HID}}(\lambda) + \text{Adv}_{\pi_2}^{\text{KS}}(\lambda) + \text{Adv}_{\text{HE}}^{\text{IND-CPA}}(\lambda) + \epsilon_{\text{Stat}}.$$

The full proof appears in Appendix D.1.

5 Concrete Instantiation and Efficiency

We now give a concrete instantiation of the blind signature protocol from Section 4. We specify the linearly homomorphic encryption scheme, the hash functions, the commitments used to link proof components, and the concrete proof systems used for the resulting statements. We then give parameters targeting 128 bits of security and estimate the communication cost of the interactive protocol.

5.1 Building Blocks and Proof Architecture

The main technical difficulty is to instantiate the zero-knowledge proofs for the languages $\mathcal{L}_1$ and $\mathcal{L}_2$. These statements contain two different types of constraints. The module-lattice relations, including commitments, encryptions, linear evaluations, and norm bounds, are naturally handled by lattice-based proof systems. In contrast, the challenge generation procedure requires non-linear arithmetic: a hash evaluation followed by the deterministic procedure `SampleInBall`, which maps a hash value to a sparse challenge polynomial

$$c \in \mathcal{C} := \{c \in \mathcal{R}_q : \|c\|_\infty = 1, \|c\|_1 = \omega\}.$$

These operations are more naturally expressed as arithmetic circuits.

We therefore use a hybrid proof architecture. Linear module-lattice relations are proved using the LNP framework of Lyubashevsky, Nguyen, and Plançon [LNP22b], instantiated via the LaZer⁶ library [LSS24]. The arithmetic part is encoded as an R1CS instance and proved using a plausibly post-quantum hash-based proof system for arithmetic circuits. The two proof systems are linked by a BDLOP commitment [BDL⁺18] to the values shared between the lattice and arithmetic statements.

We use the linearly homomorphic MLWE encryption scheme described in Section A.2. Namely, from an encryption of the challenge representation c and an encryption of the first-round randomness $\mathbf{r}'$, the signer homomorphically obtains an encryption of

$$\mathbf{z}' = c \cdot \mathbf{s} + \mathbf{r}',$$

where $\mathbf{s}$ is the signing key. To obtain circuit privacy, the encryption of $\mathbf{r}'$ uses fresh randomness large enough to flood the noise already present in the evaluated ciphertext.

For hashing, we use Poseidon [GKR⁺21]. The user first computes

$$\mu = H(\text{msg}, \text{vk}),$$

where msg is the message to be signed and vk is the verification key. The arithmetic proof then checks that the committed values satisfy,

$$\rho = \text{Poseidon}(\mu, \mathbf{w}),$$

and then verifies that

$$c = \text{SampleInBall}(\rho).$$

In the final signature verification algorithm, the verifier knows msg and vk , and recomputes $\mu = H(\text{msg}, \text{vk})$ directly.

The protocol requires two proofs. The user-side language $\mathbf{L}_1$ contains both linear lattice relations and challenge-generation constraints, so we decompose it as

$$\mathbf{L}_1 = \mathbf{L}_{1,1} \cap \mathbf{L}_{1,2}.$$

The sub-language $\mathbf{L}_{1,1}$ contains the lattice-linear relations, norm bounds, and encryption-consistency checks. The sub-language $\mathbf{L}_{1,2}$ contains the arithmetic verification of Poseidon and SampleInBall; see Appendix G for the algorithm description. The two sub-statements are linked by a BDLOP commitment to the shared values $(\mathbf{w}, \text{msg}, c)$. The signer-side language $\mathbf{L}_2$ contains only linear module-lattice relations and norm bounds, and is therefore proved directly with the lattice proof system.

We consider two instantiations of the lattice proofs. The baseline instantiation uses LNP directly via LaZer. A compact instantiation replaces the direct LNP proofs by their LaBRADOR-compressed variant [PKN⁺25], reducing communication at the cost of additional prover and verifier work. In both cases, the arithmetic proof for $\mathbf{L}_{1,2}$ is unchanged.

5.2 Decomposing and Instantiating $\mathbf{L}_1$

The language $\mathbf{L}_1$ proves correctness of the user's computation after receiving the signer's first-round value $\mathbf{w}'$. The user must prove that the blinded value $\mathbf{w}$ is correctly derived from $\mathbf{w}'$, that the challenge representation c is obtained from the committed values, and that c is correctly encrypted.

We write $\text{BDLOP.Verify}(\text{pp}, m, C) = 1$ to denote the relation stating that C is a valid BDLOP commitment to m . The commitment opening is part of the witness and is left implicit. Similarly, HE.Enc and HE.Eval denote the corresponding MLWE encryption and evaluation relations, including the randomness and noise witnesses. The fully expanded relations are given in Appendix H.

⁶ LaZer Library repository: <https://github.com/lazer-crypto/lazer>

The lattice part of the user-side proof is:

$$\mathbf{L}_{1,1} := \left\{ \begin{array}{l} ((\mathbf{t}_A, \mathbf{t}_B), (\mathbf{t}_A', \mathbf{t}_B'), \text{ctx}_c, \mathbf{w}', \\ \text{vk} = (\mathbf{A}, \mathbf{t}), \text{ek}, \text{pp}, B_{\mathbf{w}}) \end{array} \middle| \begin{array}{l} \exists(\text{msg}, c, \mathbf{r}, \mathbf{e}, \mathbf{w}, \mathbf{e}_w) \text{ s.t.} \\ \text{BDLOP.Verify}(\text{pp}, (\text{msg}, \mathbf{r}, \mathbf{e}), (\mathbf{t}_A, \mathbf{t}_B)) = 1, \\ \text{BDLOP.Verify}(\text{pp}, (\mathbf{w}, \text{msg}, c), (\mathbf{t}_A', \mathbf{t}_B')) = 1, \\ \mathbf{w}' = 2^{\nu_w} \mathbf{w} + \mathbf{e}_w - \mathbf{A}\mathbf{r} - \mathbf{e} \pmod{q}, \\ \text{ctx}_c = \text{HE.Enc}(\text{pp}, \text{ek}, c), \\ \|\mathbf{r}, \mathbf{e}\|_2 \leq B_{\mathbf{w}}, \\ \|\mathbf{e}_w\|_\infty < 2^{\nu_w-1}. \end{array} \right\}.$$

The error term $\mathbf{e}_w$ makes the rounding explicit. Equivalently, the proof shows that

$$\mathbf{w} = \lfloor \mathbf{w}' + \mathbf{A}\mathbf{r} + \mathbf{e} \rfloor_{\nu_w},$$

but expressed as a linear equation plus a small coefficient-wise rounding bound.

The arithmetic part checks that the same committed values lead to the challenge:

$$\mathbf{L}_{1,2} := \left\{ \begin{array}{l} ((\mathbf{t}_A', \mathbf{t}_B'), \text{pp}, \text{vk}) \end{array} \middle| \begin{array}{l} \exists(\mathbf{w}, \text{msg}, c) \text{ s.t.} \\ \text{BDLOP.Verify}(\text{pp}, (\mathbf{w}, \text{msg}, c), (\mathbf{t}_A', \mathbf{t}_B')) = 1, \\ \mu = \text{H}(\text{vk}, \text{msg}) \\ \rho = \text{Poseidon}(\mu, \mathbf{w}), \\ c = \text{SampleInBall}(\rho). \end{array} \right\}.$$

Thus, the circuit contains two Poseidon hash absorbing msg and vk then μ and $\mathbf{w}$, followed by the SampleInBall checks. The binding property of the BDLOP commitment ensures that the values used in $\mathbf{L}_{1,1}$ and $\mathbf{L}_{1,2}$ are the same.

5.3 Instantiating $\mathbf{L}_2$

The signer-side proof enforces honest computation of the encrypted response. After receiving ctx_c , the signer proves that the returned ciphertext encrypts

$$\mathbf{z}' = c \cdot \mathbf{s} + \mathbf{r}',$$

and that the first-round value $\mathbf{w}'$ was generated from short randomness. The statement is purely linear over module lattices once the encryption randomness and noise terms are explicit:

$$\mathbf{L}_2 := \left\{ \begin{array}{l} (\text{ctx}_{\mathbf{z}'}, \text{ctx}_c, \mathbf{w}', \text{vk} = (\mathbf{A}, \mathbf{t}), \text{ek}, \text{pp}, B_{\mathbf{w}'}) \end{array} \middle| \begin{array}{l} \exists(\mathbf{r}', \mathbf{e}', \mathbf{s}, \mathbf{e}_{\text{kg}}, \mathbf{e}_t, \text{ctx}_{\mathbf{r}'}) \text{ s.t.} \\ \mathbf{w}' = \mathbf{A}\mathbf{r}' + \mathbf{e}' \pmod{q}, \\ \mathbf{A}\mathbf{s} + \mathbf{e}_{\text{kg}} = 2^{\nu_t} \mathbf{t} + \mathbf{e}_t \pmod{q}, \\ \text{ctx}_{\mathbf{r}'} = \text{HE.Enc}(\text{pp}, \text{ek}, \mathbf{r}'; D_{\text{fl}}), \\ \text{ctx}_{\mathbf{z}'} = \text{HE.Eval}(\text{pp}, \text{ek}, f_{\mathbf{s}}, (\text{ctx}_c, \text{ctx}_{\mathbf{r}'})), \\ \|\mathbf{r}', \mathbf{e}'\|_2 \leq B_{\mathbf{w}'}, \\ \|\mathbf{e}_t\|_\infty < 2^{\nu_t-1}. \end{array} \right\},$$

where $f_{\mathbf{s}}(c, \mathbf{r}') = c \cdot \mathbf{s} + \mathbf{r}'$. The second equation is the linearized verification-key rounding relation. The flooding distribution D_{fl} is chosen so that the ciphertext of $\mathbf{r}'$ statistically hides the evaluation noise while preserving decryption correctness.

5.4 Concrete Parameters and Communication Cost

Table 3 gives concrete BRaccoon parameters⁷ targeting 128 bits of security for several bounds Q_s on the number of signing queries. The dimensions are chosen so that key recovery and signing forgery security reduces to concretely hard instances of MLWE and MSIS instance at the target security level.

Table 3. Concrete BRaccoon parameters targeting 128 bits of security.

Signing queries	Algebraic params.		Gaussian params.		Rounding params.		Sizes							
$Q_s = 2^{20}$	n	256	$\sigma_{\mathbf{t}}$	2^{10}	$\nu_{\mathbf{t}}$	52	vk size	10 KB						
	$\log_2 q$	78							$\sigma_{\mathbf{w}'}$	2^{17}	$\nu_{\mathbf{w}}$	55	sig size	28.00 KB
	k	12												
	ℓ	14												
	ω	23												
$Q_s = 2^{32}$	n	256	$\sigma_{\mathbf{t}}$	2^{10}	$\nu_{\mathbf{t}}$	52	vk size	10 KB						
	$\log_2 q$	78							$\sigma_{\mathbf{w}'}$	2^{17}	$\nu_{\mathbf{w}}$	55	sig size	31.78 KB
	k	12												
	ℓ	16												
	ω	23												
$Q_s = 2^{64}$	n	256	$\sigma_{\mathbf{t}}$	2^{10}	$\nu_{\mathbf{t}}$	52	vk size	10 KB						
	$\log_2 q$	78							$\sigma_{\mathbf{w}'}$	2^{17}	$\nu_{\mathbf{w}}$	55	sig size	43.10 KB
	k	12												
	ℓ	22												
	ω	23												

For the baseline instantiation, lattice proof sizes are measured with LaZer. The arithmetic proof for $L_{1,2}$ consists of a BDLOP opening check, two Poseidon hash $\mu = H(\text{vk}, \text{msg})$ and $\rho = H(\mu, \mathbf{w})$, and the SampleInBall circuit. For the above parameters, the resulting R1CS contains approximately 2^{18} constraints. The signer proof NIZK_{L_2} is implemented as four independent LNP proofs, and its communication cost is reported as the sum of the four parts.

Table 4. Estimated communication cost of the baseline instantiation with LNP proofs.

Signing queries	NIZKs		Ciphertexts		First-round commitment	Total
$Q_s = 2^{20}$	$\text{NIZK}_{L_{1,1}}$	399.67 KB	ctx_c	27.46 KB	$\mathbf{w}'$: 30 KB	≤ 3.11 MB
	$\text{NIZK}_{L_{1,2}}$	≤ 130 KB	$\text{ctx}_{\mathbf{z}'}$	329.52 KB		
	NIZK_{L_2}	2186.24 KB				
$Q_s = 2^{32}$	$\text{NIZK}_{L_{1,1}}$	405.35 KB	ctx_c	27.46 KB	$\mathbf{w}'$: 30 KB	≤ 3.15 MB
	$\text{NIZK}_{L_{1,2}}$	≤ 130 KB	$\text{ctx}_{\mathbf{z}'}$	329.52 KB		
	NIZK_{L_2}	2220.17 KB				
$Q_s = 2^{64}$	$\text{NIZK}_{L_{1,1}}$	419.83 KB	ctx_c	27.46 KB	$\mathbf{w}'$: 30 KB	≤ 3.26 MB
	$\text{NIZK}_{L_{1,2}}$	≤ 130 KB	$\text{ctx}_{\mathbf{z}'}$	329.52 KB		
	NIZK_{L_2}	2316.84 KB				

⁷ We provide a Sage script that verifies the security for our parameters; both for the signature scheme and the other lattice-based building blocks: <https://github.com/Heltorfx/BRaccoon-Parameters>.

Benchmark methodology. We also benchmarked the lattice proofs generated by LaZer [LSS24]. Proving times include the empirical rejection rate of the LNP proofs: if N attempts produce A accepting proofs, we report the average time to obtain one accepting proof, i.e. the average proving time per attempt multiplied by N/A . Verification time is averaged over all verification attempts, independently of whether the proof accepts or rejects.

The benchmarks were run under WSL2 on an Intel(R) Core(TM) Ultra 7 165U machine with 7 cores and 14 hardware threads. The WSL environment was Ubuntu 24.04.3 LTS with Linux kernel 6.6.114.1-microsoft-standard-WSL2, running on Windows build 10.0.26200.8246. WSL was configured with 24 GB of memory and 16 GB of swap.

Table 5. Benchmarks for the lattice proofs. Proving times account for the empirical rejection rate. The signer proof π_2 is implemented as four independent proofs; the parallel estimate assumes that the four parts are generated concurrently.

Signing queries	User proof π_1		Signer proof π_2		
	Prove	Verify	Prove, sequential	Prove, parallel	Verify, sequential
$Q_s = 2^{20}$	8 s	2 s	1 min 04 s	16 s	22 s
$Q_s = 2^{32}$	8 s	2 s	2 min	30 s	24 s
$Q_s = 2^{64}$	11.5 s	2.2 s	2 min	30 s	24 s

Remark 2. The reported timings should be viewed as measurements of our current prototype rather than optimized performance figures. Our implementation is not optimized for proving time. We therefore expect further implementation optimizations to substantially reduce our benchmarks.

5.5 Alternative Proof Instantiations

The protocol is independent of the concrete proof system used for the lattice statements $L_{1,1}$ and L_2 . Replacing the direct LNP proofs by LaBRADOR-compressed LNP proofs [PKN⁺25] significantly reduces communication, while increasing prover and verifier work. The arithmetic proof for $L_{1,2}$ is unchanged.

Since our implementation of L_2 is split into four independent lattice proofs, Table 6 conservatively reports the compressed L_2 cost as four compressed LNP proofs. A more integrated LaBRADOR instantiation may be able to compress the full statement more tightly, but we do not rely on this optimization in the estimate.

Table 6. Estimated communication cost using LaBRADOR-compressed LNP proofs.

Signing queries	NIZKs		Ciphertexts		First-round commitment	Total
$Q_s = 2^{20}, 2^{32}, \text{ or } 2^{64}$	NIZK $_{L_{1,1}}$	66 KB	ctx $_c$	27.46 KB	$\mathbf{w}'$: 30 KB	≤ 847 KB
	NIZK $_{L_{1,2}}$	≤ 130 KB	ctx $_{z'}$	329.52 KB		
	NIZK $_{L_2}$	$4 \cdot 66 = 264$ KB				

Thus, the baseline profile gives a direct implementation using LaZer, while the compact profile optimizes communication by compressing the lattice proofs. In both profiles, all zero-knowledge machinery is confined to the interactive rounds; the final output remains a standard Raccoon-shaped signature.

Acknowledgments. We want to thank Patrick Hough for help with the Sage scripts and Guilhem Niot for his helpful suggestions regarding Raccoon*. This work was supported in part by the EU Digital Europe Programme under Grant Agreement No. 101190366 (PiQASO), the project LIONS within dtec.bw, funded by the European Commission under the NextGenerationEU program.

References

- ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Berlin, Heidelberg, August 2016.
- ADK24. Shahla Atapoor, Cyprien Delpech de Saint Guilhem, and Al Kindi. STARK-based signatures from the RPO permutation. Cryptology ePrint Archive, Report 2024/1553, 2024.
- AEB20. Nabil Alkeilani Alkadri, Rachid El Bansarkhani, and Johannes Buchmann. BLAZE: Practical lattice-based blind signatures for privacy-preserving applications. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 484–502. Springer, Cham, February 2020.
- AGJ⁺24. Sven Argo, Tim Güneysu, Corentin Jeudy, Georg Land, Adeline Roux-Langlois, and Olivier Sanders. Practical post-quantum signatures for privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 1523–1537. ACM Press, October 2024.
- AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligerio: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.
- AHJ21. Nabil Alkeilani Alkadri, Patrick Harasser, and Christian Janson. BlindOR: an efficient lattice-based blind signature scheme from OR-proofs. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *CANS 21*, volume 13099 of *LNCS*, pages 95–115. Springer, Cham, December 2021.
- AKSY22. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53. ACM Press, November 2022.
- ALP26. Martin R. Albrecht, Russell W. F. Lai, and Eamonn W. Postlethwaite. Hardness of hinted ISIS from the space-time hardness of lattice problems. Cryptology ePrint Archive, Paper 2026/187, 2026.
- BBB⁺26. Carsten Baum, Marvin Beckmann, Ward Beullens, Shibam Mukherjee, and Christian Rechberger. Concretely efficient blind signatures based on VOLE-in-the-head proofs and the MAYO trapdoor. Cryptology ePrint Archive, Paper 2026/109, 2026.
- BCC⁺17. Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinfeld, and Eran Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, October 2017.
- BCOS20. Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. Efficient post-quantum SNARKs for RSIS and RLWE and their applications to privacy. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 247–267. Springer, Cham, 2020.
- BCR⁺19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019.
- BDL⁺18. Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385. Springer, Cham, September 2018.
- BFM⁺25a. Charles Bouillaguet, Thibault Feneuil, Jules Maire, Matthieu Rivain, Julia Sauvage, and Damien Vergnaud. Blinding post-quantum hash-and-sign signatures. Cryptology ePrint Archive, Paper 2025/895, 2025.
- BFM⁺25b. Charles Bouillaguet, Thibault Feneuil, Jules Maire, Matthieu Rivain, Julia Sauvage, and Damien Vergnaud. Blinding post-quantum hash-and-sign signatures. Cryptology ePrint Archive, Report 2025/895, 2025.

- BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.
- BL13. Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098. ACM Press, November 2013.
- BLL⁺21. Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53. Springer, Cham, October 2021.
- BLNS23a. Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 16–29. ACM Press, November 2023.
- BLNS23b. Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 384–417. Springer, Cham, August 2023.
- Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Berlin, Heidelberg, January 2003.
- BS23. Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548. Springer, Cham, August 2023.
- BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011.
- CAHL⁺22. Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 3–31. Springer, Cham, August 2022.
- Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO’82*, pages 199–203. Plenum Press, New York, USA, 1982.
- CJL16. Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016.
- CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Berlin, Heidelberg, September 2003.
- dEK⁺23. Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani Saarinen. Raccoon. Technical report, National Institute of Standards and Technology, 2023. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures>.
- DGS⁺18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.
- DHP24. Khue Do, Lucjan Hanzlik, and Eugenio Paracucchi. M&M’S: Mix and match attacks on Schnorr-type blind signatures with repetition. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 363–387. Springer, Cham, May 2024.
- DKL⁺18. Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR TCHES*, 2018(1):238–268, 2018.
- DKM⁺24. Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.
- dPK22. Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336. Springer, Cham, August 2022.

- dPKPR24. Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 409–444. Springer, Cham, August 2024.
- dPPRS23. Rafaël del Pino, Thomas Prest, Mélissa Rossi, and Markku-Juhani O. Saarinen. High-order masking of lattice signatures in quasilinear time. In *2023 IEEE Symposium on Security and Privacy*, pages 1168–1185. IEEE Computer Society Press, May 2023.
- Fas26. Matteo Frigo and abhi shelat. Anonymous credentials from ECDSA. *IACR Communications in Cryptology*, 3(1), 2026.
- FHKS16. Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Cham, August / September 2016.
- Fis06. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Berlin, Heidelberg, August 2006.
- FPS20. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 63–95. Springer, Cham, May 2020.
- FR25. Thibault Feneuil and Matthieu Rivain. CAPSS: A framework for SNARK-friendly post-quantum signatures. Cryptology ePrint Archive, Report 2025/061, 2025.
- FW24. Georg Fuchsbauer and Mathias Wolf. Concurrently secure blind Schnorr signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 124–160. Springer, Cham, May 2024.
- Gen09. Craig Gentry. *A Fully Homomorphic Encryption Scheme*. Ph.d. dissertation, Stanford University, Stanford, CA, USA, September 2009.
- GHK⁺25. Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. Olingo: Threshold lattice signatures with DKG and identifiable abort. Cryptology ePrint Archive, Report 2025/1789, 2025.
- GKR⁺21. Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 519–535. USENIX Association, August 2021.
- GKS24. Kamil Doruk Gür, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *PQCrypto 2024, Part II*, pages 266–300. Springer, Cham, June 2024.
- GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
- HAB⁺17. Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub. In *NDSS 2017*. The Internet Society, February / March 2017.
- HKL19. Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, Cham, May 2019.
- HKLN20. Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529. Springer, Cham, August 2020.
- HLM⁺25. Lucjan Hanzlik, Yi-Fu Lai, Marzio Mula, Eugenio Paracucchi, Daniel Slamanig, and Gang Tang. Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum group actions. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 35–69. Springer, Singapore, December 2025.
- HLPP26. Lucjan Hanzlik, Yi-Fu Lai, Eugenio Paracucchi, and Edoardo Persichetti. Wombat: Post-quantum blind signature from standard group action assumptions and more. Cryptology ePrint Archive, Paper 2026/928, 2026.
- HLW23. Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Rai-choo! Evolving blind signatures to the next level. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 753–783. Springer, Cham, April 2023.
- HSS24. Patrick Hough, Caroline Sandsbråten, and Tjerand Silde. More efficient lattice-based electronic voting from NTRU. *CiC*, 1(4):10, 2024.

- IKSA03. S. Ibrahim, M. Kamat, M. Salleh, and S.R.A. Aziz. Secure e-voting with blind signature. In *4th National Conference of Telecommunication Technology, 2003. NCTT 2003 Proceedings.*, pages 193–197, 2003.
- JRS23. Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. Lattice signature with efficient protocols, application to anonymous credentials. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part II*, volume 14082 of *LNCS*, pages 351–383. Springer, Cham, August 2023.
- JS25a. Corentin Jeudy and Olivier Sanders. Improved lattice blind signatures from recycled entropy. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part I*, volume 16000 of *LNCS*, pages 477–513. Springer, Cham, August 2025.
- JS25b. Corentin Jeudy and Olivier Sanders. Worst-case lattice sampler with truncated gadgets and applications. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part III*, volume 16247 of *LNCS*, pages 200–232. Springer, Singapore, December 2025.
- KLR25. Michael Kloof, Russell W. F. Lai, and Michael Reichle. Blind signatures from arguments of inequality. Cryptology ePrint Archive, Paper 2025/1886, 2025.
- KLSS23. Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. Toward practical lattice-based proof of knowledge from hint-MLWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 549–580. Springer, Cham, August 2023.
- Knu97. Donald E. Knuth. *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.
- LNP22a. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 498–527. Springer, Cham, March 2022.
- LNP22b. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101. Springer, Cham, August 2022.
- LS15. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *DCC*, 75(3):565–599, 2015.
- LSS24. Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The LaZer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3125–3137. ACM Press, October 2024.
- Lyu09. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Berlin, Heidelberg, December 2009.
- MPD25. Jules Maire and Alan Pulval-Dady. Blind ECDSA from the ECDSA assumption. *CiC*, 2(3):38, 2025.
- MR04. D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 372–381, 2004.
- NS26. Ngoc Khanh Nguyen and Jan Niklas Siemer. Tight reductions for SIS-with-hints assumptions with applications to anonymous credentials. Cryptology ePrint Archive, Paper 2026/291, 2026. To appear at PKC 2026.
- Pei16. Chris Peikert. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, 10(4):283–424, 2016.
- PKN⁺25. Rafaël Del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. Unmasking TRaccoon: A lattice-based threshold signature with an efficient identifiable abort protocol. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VI*, volume 16005 of *LNCS*, pages 423–456. Springer, Cham, August 2025.
- PPZ24. Christian Paquin, Guru-Vamsi Policharla, and Greg Zaverucha. Crescent: Stronger privacy for existing credentials. Cryptology ePrint Archive, Report 2024/2013, 2024.
- PS96. David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT’96*, volume 1163 of *LNCS*, pages 252–265. Springer, Berlin, Heidelberg, November 1996.
- PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

- PWFW23. Guru-Vamsi Policharla, Bas Westerbaan, Armando Faz-Hernández, and Christopher A Wood. Post-quantum privacy pass via post-quantum anonymous credentials. Cryptology ePrint Archive, Report 2023/414, 2023.
- Rüc10. Markus Rückert. Lattice-based blind signatures. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 413–430. Springer, Berlin, Heidelberg, December 2010.
- SU12. Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 662–679. Springer, Berlin, Heidelberg, May 2012.
- WOS⁺25. Anna P. Y. Woo, Alex Ozdemir, Chad Sharp, Thomas Pornin, and Paul Grubbs. Efficient proofs of possession for legacy signatures. Cryptology ePrint Archive, Report 2025/538, 2025.
- XZS22. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 299–328. Springer, Cham, August 2022.
- ZSE⁺24. Xinyu Zhang, Ron Steinfeld, Muhammed F. Esgin, Joseph K. Liu, Dongxi Liu, and Sushmita Ruj. Loquat: A SNARK-friendly post-quantum signature based on the Legendre PRF with applications in ring and aggregate signatures. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part I*, volume 14920 of *LNCS*, pages 3–38. Springer, Cham, August 2024.

A Extended Background

A.1 Homomorphic Encryption Scheme

Homomorphic Encryption Scheme. A homomorphic (public key) encryption scheme HE is a 5-tuple of PPT algorithms $(\text{Setup}, \text{KGen}, \text{Enc}, \text{Dec}, \text{Eval})$ defined as follows. On input the security parameter $\lambda \in \mathbb{N}$, the setup algorithm $\text{HE.Setup}(1^\lambda)$ outputs public parameters pp , which include a description of the message space $\mathcal{M}$. The key generation algorithm $\text{HE.KGen}(\text{pp})$ outputs a public/secret key pair (ek, dk) . Given pp , a public key ek , and a message $\text{msg} \in \mathcal{M}$, the encryption algorithm $\text{HE.Enc}(\text{pp}, \text{ek}, \text{msg})$ outputs a ciphertext ctx_{msg} . Given pp , ek , a function $f : \mathcal{M}^n \rightarrow \mathcal{M}$ from an admissible function class $\mathcal{F}$, ciphertexts $(\text{ctx}_1, \dots, \text{ctx}_n)$, the evaluation algorithm $\text{HE.Eval}(\text{pp}, \text{ek}, f, (\text{ctx}_1, \dots, \text{ctx}_n))$ outputs a ciphertext ctx_f . Finally, the decryption algorithm $\text{HE.Dec}(\text{pp}, \text{dk}, \text{ctx}_{\text{msg}})$ outputs the message $\text{msg} \in \mathcal{M}$ or $\perp$ in case of decryption failure.

Correctness. A homomorphic encryption scheme is *correct* if for all $\lambda \in \mathbb{N}$,

$$\Pr \left[\text{Dec}(\text{pp}, \text{dk}, \text{Enc}(\text{pp}, \text{ek}, \text{msg})) = \text{msg} \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ (\text{ek}, \text{dk}) \leftarrow \$ \text{KGen}(\text{pp}), \\ \text{msg} \leftarrow \$ \mathcal{M} \end{array} \right] = 1$$

and for any function $f \in \mathcal{F}$,

$$\Pr \left[\begin{array}{l} \text{Dec}(\text{Eval}(\text{pp}, \text{ek}, f, (\text{ctx}_1, \dots, \text{ctx}_n))) \\ = f(\text{msg}_1, \dots, \text{msg}_n) \end{array} \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ (\text{ek}, \text{dk}) \leftarrow \$ \text{KGen}(\text{pp}), \\ (\text{msg}_1, \dots, \text{msg}_n) \leftarrow \$ \mathcal{M}^n, \\ \forall i \in [n], \text{ctx}_i \leftarrow \$ \text{Enc}(\text{pp}, \text{ek}, \text{msg}_i) \end{array} \right] = 1 - \text{negl}(\lambda).$$

IND-CPA Security. We recall the notion of IND-CPA (Indistinguishability under Chosen-Plaintext Attack), which is a standard security definition for encryption schemes. Let $t : \mathbb{N} \rightarrow \mathbb{N}$ and $\epsilon : \mathbb{N} \rightarrow [0, 1]$. The encryption scheme is said to be (t, ϵ) -IND-CPA secure if for all two-phase adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ running in time at most $t(\lambda)$, we have:

$$\Pr \left[b = b' \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), (\text{ek}, \text{dk}) \leftarrow \$ \text{KGen}(\text{pp}), \\ (\text{msg}_0, \text{msg}_1, \text{state}) \leftarrow \$ \mathcal{A}_1(\text{pp}, \text{ek}), b \leftarrow \$ \{0, 1\}, \\ c \leftarrow \$ \text{Enc}(\text{pp}, \text{ek}, \text{msg}_b), b' \leftarrow \$ \mathcal{A}_2(\text{pp}, c, \text{state}) \end{array} \right] \leq \frac{1}{2} + \epsilon(\lambda).$$

Circuit Privacy. A further security notion for homomorphic encryption schemes is *circuit privacy* [Gen09]. Loosely speaking, it means that the output of Evaluate does not leak any information on the input circuit/function.

Formally, a homomorphic encryption scheme $\text{HE} := (\text{Setup}, \text{KGen}, \text{Enc}, \text{Dec}, \text{Eval})$ is said to be circuit private if the following property holds:

$$\text{HE.Enc}(\text{pp}, \text{ek}, f(\text{msg}_1, \dots, \text{msg}_n)) \stackrel{s}{\approx} \text{HE.Eval}(\text{pp}, \text{ek}, f, (\text{ctx}_1, \dots, \text{ctx}_n)),$$

For any function $f \in \mathcal{F}$, any message $\text{msg}_i \in \mathcal{M}, i \in [n]$ s.t $\text{ctx}_i = \text{HE.Enc}(\text{pp}, \text{ek}, \text{msg}_i)$. While the definitions above hold for general homomorphic evaluations, we desire only the circuit privacy to hold for a restricted family of function.

A.2 MLWE-Based Linearly Homomorphic Encryption

This subsection specifies the linear homomorphic encryption primitive used in our construction. We present a lightweight MLWE-based encryption scheme, inspired by the BGV construction [BGV12] and its module variant from [GHK+25], but intentionally restricted to the evaluation of linear functions in the following family:

$$\mathcal{F} = \{f_{\alpha} : \mathcal{M}^2 \rightarrow \mathcal{M}, \quad (C, M) \mapsto \alpha \odot C + M \mid \alpha \in \mathcal{R}_p^{\ell}\}. \quad (2)$$

Here

$$\mathcal{M} := \mathcal{R}_p^{\ell}$$

is the message space, while $\odot$ denotes component-wise multiplication. As in the original BGV construction, a key ingredient is the *modulus reduction* introduced in [BV11]. More precisely, ciphertexts are defined over the larger ring $\mathcal{R}_q$, while plaintexts are recovered modulo the smaller plaintext modulus p .

The scheme is defined in two layers. We first describe a scalar encryption algorithm encrypting messages in $\mathcal{R}_p$. We then extend it coordinate-wise to the message space $\mathcal{M} = \mathcal{R}_p^{\ell}$. To avoid ambiguity, we denote the scalar encryption and decryption algorithms by $\text{BGV.Enc}_{\text{sc}}$ and $\text{BGV.Dec}_{\text{sc}}$, respectively, while BGV.Enc and BGV.Dec denote their coordinate-wise vector extensions.

By default, encryption uses the noise distribution D specified in the public parameters. When an alternative noise distribution D' is used, we write

$$\text{BGV.Enc}_{\text{sc}}(\text{pp}, \text{ek}, m; D') \quad \text{or} \quad \text{BGV.Enc}(\text{pp}, \text{ek}, M; D')$$

for scalar and vector encryption, respectively. The distribution appears after a semicolon to emphasize that it is an additional sampling parameter, rather than part of the plaintext input.

- $\text{pp} \leftarrow_{\$} \text{BGV.Setup}(1^{\lambda})$: The setup algorithm outputs public parameters

$$\text{pp} = (\mathcal{R}_q, p, k, \ell, D),$$

where $p \ll q$ is the plaintext modulus and D is a discrete Gaussian over $\mathcal{R}_q$. Parameters are chosen such that $\text{MLWE}_{k, \ell, D}$ is hard.

- $(\text{dk}, \text{ek}) \leftarrow \text{BGV.KGen}(\text{pp})$: On input the public parameters pp , sample a uniform matrix $\mathbf{A} \leftarrow_{\$} \mathcal{R}_q^{k \times k}$, a secret vector $\mathbf{s} \leftarrow_{\$} D^k$, and an error vector $\mathbf{e} \leftarrow_{\$} D^k$. Compute

$$\mathbf{b} := \mathbf{A}\mathbf{s} + p\mathbf{e} \in \mathcal{R}_q^k.$$

The encryption key is $\text{ek} = (\mathbf{A}, \mathbf{b})$ and the decryption key is $\text{dk} = \mathbf{s}$.

- $\text{ctx} \leftarrow \text{BGV.Enc}_{\text{sc}}(\text{pp}, \text{ek}, m)$: On input the encryption key $\text{ek} = (\mathbf{A}, \mathbf{b})$ and a scalar message $m \in \mathcal{R}_p$, sample encryption randomness $\mathbf{r} \leftarrow_{\$} D^k$ and noise terms $\mathbf{e}_1 \leftarrow_{\$} D^k$ and $e_2 \leftarrow_{\$} D$, and compute

$$\mathbf{u} := \mathbf{A}^T \mathbf{r} + p\mathbf{e}_1 \in \mathcal{R}_q^k, \quad v := \mathbf{b}^T \mathbf{r} + pe_2 + m \in \mathcal{R}_q.$$

It outputs the scalar ciphertext $\text{ctx} = (\mathbf{u}, v)$. More generally, when an alternative distribution D' is specified, the notation

$$\text{ctx} \leftarrow \text{BGV.Enc}_{\text{sc}}(\text{pp}, \text{ek}, m; D')$$

denotes the same algorithm, except that $\mathbf{r}$, $\mathbf{e}_1$, and e_2 are sampled from D' instead of D .

- $\mathbf{ctx} \leftarrow \text{BGV.Enc}(\text{pp}, \text{ek}, M)$: For a vector message $M = (M_1, \dots, M_\ell) \in \mathcal{M}$, define the vector encryption algorithm by applying the scalar encryption algorithm coordinate-wise:

$$\mathbf{ctx} := (\text{ctx}^{(1)}, \dots, \text{ctx}^{(\ell)}), \quad \text{ctx}^{(j)} \leftarrow \text{BGV.Enc}_{\text{sc}}(\text{pp}, \text{ek}, M_j) \quad \text{for } j \in [\ell].$$

Similarly, the notation

$$\mathbf{ctx} \leftarrow \text{BGV.Enc}(\text{pp}, \text{ek}, M; D')$$

denotes coordinate-wise encryption using $\text{BGV.Enc}_{\text{sc}}(\text{pp}, \text{ek}, \cdot; D')$ in every coordinate. For a scalar $c \in \mathcal{R}_p$, we write

$$\bar{c} := (c, \dots, c) \in \mathcal{M}.$$

An encryption of $\bar{c}$ may either be represented as a vector ciphertext

$$\text{BGV.Enc}(\text{pp}, \text{ek}, \bar{c}),$$

or compactly by a single scalar ciphertext

$$\text{ctx}_c \leftarrow \text{BGV.Enc}_{\text{sc}}(\text{pp}, \text{ek}, c).$$

In the compact representation, the same scalar ciphertext ctx_c is reused in every coordinate during evaluation.

- $\mathbf{ctx}' \leftarrow \text{BGV.Eval}(\text{pp}, \text{ek}, f_\alpha, (\mathbf{ctx}_C, \mathbf{ctx}_M))$: On input $f_\alpha \in \mathcal{F}$, parse

$$\mathbf{ctx}_C = (\text{ctx}_C^{(1)}, \dots, \text{ctx}_C^{(\ell)}), \quad \mathbf{ctx}_M = (\text{ctx}_M^{(1)}, \dots, \text{ctx}_M^{(\ell)}),$$

where

$$\text{ctx}_C^{(j)} = (\mathbf{u}_C^{(j)}, v_C^{(j)}), \quad \text{ctx}_M^{(j)} = (\mathbf{u}_M^{(j)}, v_M^{(j)}).$$

If the first input is given in compact form by a scalar ciphertext $\text{ctx}_c = (\mathbf{u}_c, v_c)$ encrypting $c \in \mathcal{R}_p$, we set

$$\text{ctx}_C^{(j)} := \text{ctx}_c \quad \text{for every } j \in [\ell].$$

Let

$$\alpha = (\alpha_1, \dots, \alpha_\ell) \in \mathcal{R}_p^\ell.$$

The evaluation algorithm outputs,

$$\mathbf{ctx}' = (\text{ctx}'^{(1)}, \dots, \text{ctx}'^{(\ell)}),$$

where, for every $j \in [\ell]$,

$$\text{ctx}'^{(j)} = (\mathbf{u}'^{(j)}, v'^{(j)}) := (\alpha_j \mathbf{u}_C^{(j)} + \mathbf{u}_M^{(j)}, \alpha_j v_C^{(j)} + v_M^{(j)}) \bmod q.$$

- $\mathbf{ctx}^{\text{fl}} \leftarrow \text{BGV.Eval}^{\text{fl}}(\text{pp}, \text{ek}, f_\alpha, (\mathbf{ctx}_C, \mathbf{ctx}_M))$: The flooded evaluation algorithm first computes

$$\mathbf{ctx}^{\text{ev}} \leftarrow \text{BGV.Eval}(\text{pp}, \text{ek}, f_\alpha, (\mathbf{ctx}_C, \mathbf{ctx}_M)).$$

It then samples an independent encryption of the zero vector using a flooding distribution D_{fl} :

$$\mathbf{ctx}_0 \leftarrow \text{BGV.Enc}(\text{pp}, \text{ek}, 0^\ell; D_{\text{fl}}).$$

Finally, it outputs

$$\mathbf{ctx}^{\text{fl}} := \mathbf{ctx}^{\text{ev}} + \mathbf{ctx}_0,$$

where addition is performed coordinate-wise over $\mathcal{R}_q$. The distribution D_{fl} is chosen so that the noise of $\mathbf{ctx}_0$ statistically dominates the noise already present in $\mathbf{ctx}^{\text{ev}}$.

- $m \leftarrow \text{BGV.Dec}_{\text{sc}}(\text{pp}, \text{dk}, \text{ctx})$: On input the decryption key $\text{dk} = \mathbf{s}$ and a scalar ciphertext $\text{ctx} = (\mathbf{u}, v)$, output

$$m := v - \mathbf{s}^T \mathbf{u} \bmod q \bmod p.$$

- $M \leftarrow \text{BGV.Dec}(\text{pp}, \text{dk}, \mathbf{ctx})$: For a vector ciphertext $\mathbf{ctx} = (\text{ctx}^{(1)}, \dots, \text{ctx}^{(\ell)})$, output

$$M := (\text{BGV.Dec}_{\text{sc}}(\text{pp}, \text{dk}, \text{ctx}^{(1)}), \dots, \text{BGV.Dec}_{\text{sc}}(\text{pp}, \text{dk}, \text{ctx}^{(\ell)})) \in \mathcal{M}.$$

Definition 8 (Decryption residual). For a scalar ciphertext $\text{ctx} = (\mathbf{u}, v)$ encrypting $m \in \mathcal{R}_p$, we define its decryption residual as

$$\delta(\text{ctx}) := v - \mathbf{s}^T \mathbf{u} = m + pz \pmod{q},$$

where z is the noise term induced by the encryption randomness. For a vector ciphertext $\mathbf{ctx} = (\text{ctx}^{(1)}, \dots, \text{ctx}^{(\ell)})$, we define,

$$\delta(\mathbf{ctx}) := (\delta(\text{ctx}^{(1)}), \dots, \delta(\text{ctx}^{(\ell)})).$$

Lemma 6 (Correctness). Let

$$\mathbf{ctx}' \leftarrow \text{BGV.Eval}(\text{pp}, \text{ek}, f_{\alpha}, (\mathbf{ctx}_C, \mathbf{ctx}_M)),$$

where $\mathbf{ctx}_C$ encrypts $C \in \mathcal{M}$ and $\mathbf{ctx}_M$ encrypts $M \in \mathcal{M}$. Suppose that, except with negligible probability, for every coordinate $j \in [\ell]$, the decryption residual of $\text{ctx}'^{(j)}$ can be written as

$$\delta(\text{ctx}'^{(j)}) = \alpha_j C_j + M_j + pz_{\text{ev}}^{(j)} \pmod{q}$$

with

$$\|z_{\text{ev}}^{(j)}\|_{\infty} \leq B. \quad (3)$$

Assume further that

$$pB < \frac{q}{2}.$$

Then the linear homomorphic encryption scheme BGV is correct with respect to the family of transformations $f_{\alpha} \in \mathcal{F}$. Namely,

$$\text{BGV.Dec}(\text{pp}, \text{dk}, \mathbf{ctx}') = f_{\alpha}(C, M) = \alpha \odot C + M.$$

The same statement holds for the flooded evaluation $\text{BGV.Eval}^{\text{fl}}$, with B replaced by a bound B_{fl} on the total decryption noise after flooding.

Proof. It suffices to prove correctness coordinate-wise. Fix $j \in [\ell]$ and write

$$\text{ctx}_C^{(j)} = (\mathbf{u}_C^{(j)}, v_C^{(j)}), \quad \text{ctx}_M^{(j)} = (\mathbf{u}_M^{(j)}, v_M^{(j)}).$$

By definition of the decryption residual, there exist noise terms $z_C^{(j)}$ and $z_M^{(j)}$ such that

$$\delta(\text{ctx}_C^{(j)}) = v_C^{(j)} - \mathbf{s}^T \mathbf{u}_C^{(j)} = C_j + pz_C^{(j)} \pmod{q}$$

and

$$\delta(\text{ctx}_M^{(j)}) = v_M^{(j)} - \mathbf{s}^T \mathbf{u}_M^{(j)} = M_j + pz_M^{(j)} \pmod{q}.$$

The evaluated ciphertext in coordinate j is

$$\text{ctx}'^{(j)} = (\alpha_j \mathbf{u}_C^{(j)} + \mathbf{u}_M^{(j)}, \alpha_j v_C^{(j)} + v_M^{(j)}) \bmod q.$$

Therefore,

$$\begin{aligned}\delta(\text{ctx}'^{(j)}) &= \alpha_j v_C^{(j)} + v_M^{(j)} - \mathbf{s}^T (\alpha_j \mathbf{u}_C^{(j)} + \mathbf{u}_M^{(j)}) \pmod{q} \\ &= \alpha_j (v_C^{(j)} - \mathbf{s}^T \mathbf{u}_C^{(j)}) + (v_M^{(j)} - \mathbf{s}^T \mathbf{u}_M^{(j)}) \pmod{q} \\ &= \alpha_j C_j + M_j + p(\alpha_j z_C^{(j)} + z_M^{(j)}) \pmod{q}.\end{aligned}$$

Hence the evaluated noise is

$$z_{\text{ev}}^{(j)} = \alpha_j z_C^{(j)} + z_M^{(j)}.$$

By assumption, $\|z_{\text{ev}}^{(j)}\|_\infty \leq B$ and $pB < q/2$. Therefore the noise term does not wrap around modulo q . Reducing the decryption residual modulo p removes the noise term and yields

$$\text{BGV.Dec}_{\text{sc}}(\text{pp}, \text{dk}, \text{ctx}'^{(j)}) = \alpha_j C_j + M_j \pmod{p}.$$

Applying this argument to every coordinate gives

$$\text{BGV.Dec}(\text{pp}, \text{dk}, \text{ctx}') = \alpha \odot C + M.$$

For the flooded evaluation, write

$$\text{ctx}^{\text{fl}} = \text{ctx}^{\text{ev}} + \text{ctx}_0,$$

where ctx_0 encrypts 0^ℓ . In each coordinate, adding $\text{ctx}_0^{(j)}$ preserves the plaintext and only adds an additional multiple of p to the decryption residual. Thus the same argument applies with the evaluated noise bound B replaced by a bound B_{fl} on the total noise after flooding.

Lemma 7 (IND-CPA Security). *Assuming the decisional $\text{MLWE}_{k,\ell,\mathbb{D}}$ problem is hard for the parameters generated by Setup, the scalar encryption scheme $\text{BGV.Enc}_{\text{sc}}$ is IND-CPA secure. Consequently, its coordinate-wise vector extension BGV.Enc is also IND-CPA secure.*

More precisely, for any PPT adversary $\mathcal{A}$, there exist PPT distinguishers $\mathcal{B}$ and $\mathcal{C}$ such that

$$\text{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q,n,k_{\text{ek}},\ell_{\text{ek}},\text{D}_{\text{bgv}}}}(\lambda) + \text{Adv}_{\mathcal{C}}^{\text{MLWE}_{q,n,\ell_{\text{ek}}+1,k_{\text{ek}},\text{D}_{\text{bgv}}}}(\lambda).$$

Proof. We first argue security of the scalar encryption scheme.

The public key is of the form

$$\text{ek} = (\mathbf{A}, \mathbf{b}), \quad \mathbf{b} = \mathbf{A}\mathbf{s} + p\mathbf{e}.$$

By the decisional MLWE assumption, the distribution of $(\mathbf{A}, \mathbf{b})$ is computationally indistinguishable from $(\mathbf{A}, \mathbf{b}^*)$, where $\mathbf{b}^*$ is uniformly random in $\mathcal{R}_q^k$. Thus, replacing the public key component $\mathbf{b}$ by a uniform vector changes the adversary's advantage by at most $\text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q,n,k_{\text{ek}},\ell_{\text{ek}},\text{D}_{\text{bgv}}}}(\lambda)$ for some PPT distinguisher $\mathcal{B}$.

Now consider a challenge ciphertext under a public key $(\mathbf{A}, \mathbf{b}^*)$ with uniform $\mathbf{b}^*$. An honest scalar encryption of m_b has the form

$$\mathbf{u}^* = \mathbf{A}^T \mathbf{r} + p\mathbf{e}_1, \quad v^* = (\mathbf{b}^*)^T \mathbf{r} + pe_2 + m_b.$$

Again, by the decisional MLWE assumption, the pair $(\mathbf{u}^*, v^*)$ is computationally indistinguishable from a uniformly random element of $\mathcal{R}_q^k \times \mathcal{R}_q$. Since adding the plaintext m_b to a uniform ring element does not change its distribution, the resulting challenge ciphertext is independent of the challenge bit b . This transition costs at most $\text{Adv}_{\mathcal{C}}^{\text{MLWE}_{q,n,\ell_{\text{ek}}+1,k_{\text{ek}},\text{D}_{\text{bgv}}}}(\lambda)$ for some PPT distinguisher $\mathcal{C}$.

In the final game, the challenge ciphertext is independent of b , and hence the adversary wins with probability exactly $1/2$. Therefore,

$$\text{Adv}_{\mathcal{A}}^{\text{IND-CPA}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{MLWE}_{q,n,k_{\text{ek}},\ell_{\text{ek}},\text{D}_{\text{bgv}}}}(\lambda) + \text{Adv}_{\mathcal{C}}^{\text{MLWE}_{q,n,\ell_{\text{ek}}+1,k_{\text{ek}},\text{D}_{\text{bgv}}}}(\lambda).$$

The vector encryption algorithm BGV.Enc encrypts each coordinate using an independent invocation of the scalar encryption algorithm. Security of the vector scheme follows from a standard hybrid argument over the ℓ coordinates.

Lemma 8 (Circuit privacy via noise flooding). *Let*

$$\mathbf{ctx}^{\text{ev}} \leftarrow \text{BGV.Eval}(\text{pp}, \text{ek}, f_{\alpha}, (\mathbf{ctx}_C, \mathbf{ctx}_M))$$

be the evaluated ciphertext, where $\mathbf{ctx}_C$ encrypts $C \in \mathcal{M}$ and $\mathbf{ctx}_M$ encrypts $M \in \mathcal{M}$. Let

$$\mathbf{ctx}_0 \leftarrow \text{BGV.Enc}(\text{pp}, \text{ek}, 0^\ell; \mathcal{D}_{\text{fl}})$$

be an independent encryption of the zero vector using a flooding distribution $\mathcal{D}_{\text{fl}}$, and define

$$\mathbf{ctx}^{\text{fl}} := \mathbf{ctx}^{\text{ev}} + \mathbf{ctx}_0.$$

Then $\mathbf{ctx}^{\text{fl}}$ decrypts to

$$f_{\alpha}(C, M) = \alpha \odot C + M.$$

Moreover, for every coordinate $j \in [\ell]$, write

$$\delta(\mathbf{ctx}_C^{(j)}) = C_j + pz_C^{(j)}, \quad \delta(\mathbf{ctx}_M^{(j)}) = M_j + pz_M^{(j)},$$

and

$$\delta(\mathbf{ctx}_0^{(j)}) = pz_0^{(j)}.$$

Then the flooded ciphertext satisfies

$$\delta(\mathbf{ctx}_{\text{fl}}^{(j)}) = \alpha_j C_j + M_j + p \left(\underbrace{\alpha_j z_C^{(j)} + z_M^{(j)}}_{\text{evaluation noise}} + \underbrace{z_0^{(j)}}_{\text{flooding noise}} \right) \pmod{q}.$$

Assume that the flooding distribution is chosen so that, with overwhelming probability,

$$\|z_0^{(j)}\|_{\infty} \gg \|\alpha_j z_C^{(j)} + z_M^{(j)}\|_{\infty} \quad \text{for every } j \in [\ell],$$

and, more precisely, so that

$$z_0^{(j)} + \alpha_j z_C^{(j)} + z_M^{(j)} \approx_{\text{stat}} z_0^{(j)} \quad \text{for every } j \in [\ell].$$

Then the distribution of $\mathbf{ctx}^{\text{fl}}$, conditioned on the plaintext $f_{\alpha}(C, M)$, is statistically close to that of a fresh encryption of $f_{\alpha}(C, M)$ generated with noise distribution $\mathcal{D}_{\text{fl}}$. In particular, the flooded evaluation leaks no additional information about the evaluated linear function beyond the output plaintext.

Proof (Proof sketch). The proof follows from the linearity of the decryption residual. For every coordinate $j \in [\ell]$, evaluation gives

$$\begin{aligned} \delta(\mathbf{ctx}_{\text{ev}}^{(j)}) &= \alpha_j \delta(\mathbf{ctx}_C^{(j)}) + \delta(\mathbf{ctx}_M^{(j)}) \\ &= \alpha_j C_j + M_j + p(\alpha_j z_C^{(j)} + z_M^{(j)}) \pmod{q}. \end{aligned}$$

Adding an independent encryption of zero with flooding noise $z_0^{(j)}$ preserves the plaintext and only modifies the noise:

$$\delta(\mathbf{ctx}_{\text{fl}}^{(j)}) = \alpha_j C_j + M_j + p(\alpha_j z_C^{(j)} + z_M^{(j)} + z_0^{(j)}) \pmod{q}.$$

By the domination assumption on $\mathcal{D}_{\text{fl}}$, the distribution of the final noise term

$$z_0^{(j)} + \alpha_j z_C^{(j)} + z_M^{(j)}$$

is statistically close to the flooding noise $z_0^{(j)}$ alone. Hence, conditioned on the plaintext $\alpha_j C_j + M_j$, the flooded ciphertext in coordinate j is statistically close to a fresh scalar encryption generated with the flooding distribution. Applying this argument coordinate-wise proves the claim for the full vector ciphertext.

Parameter selection. The parameters are chosen so that the decryption noise remains below the modulus-reduction threshold after evaluation. This bound depends on the coefficients of the evaluated function, on the randomness and error distributions used to generate the input ciphertexts, and, for flooded evaluation, on the flooding distribution. Namely, if the input ciphertext noises are $z_C^{(j)}$ and $z_M^{(j)}$, then evaluating $f_\alpha(C, M) = \alpha \odot C + M$ produces noise

$$z_{\text{ev}}^{(j)} = \alpha_j z_C^{(j)} + z_M^{(j)}.$$

For flooded evaluation, adding an encryption of zero with noise $z_0^{(j)}$ produces total noise

$$z_{\text{fl}}^{(j)} = \alpha_j z_C^{(j)} + z_M^{(j)} + z_0^{(j)}.$$

Accordingly, the modulus q is chosen so that, except with negligible probability,

$$p \|z_{\text{ev}}^{(j)}\|_\infty < \frac{q}{2} \quad \text{and, in the flooded case,} \quad p \|z_{\text{fl}}^{(j)}\|_\infty < \frac{q}{2}$$

for every $j \in [\ell]$. The flooding distribution is chosen wide enough so that $z_0^{(j)}$ statistically dominates the evaluation noise $\alpha_j z_C^{(j)} + z_M^{(j)}$. Concretely, we set the standard deviation of the flooding distribution to be larger than the standard deviation of the evaluation noise by a statistical security factor 2^{40} . This ensures that the residual contribution of the evaluation noise is statistically hidden by the flooding noise, while correctness is preserved by choosing q large enough to accommodate the resulting total noise.

A.3 Commitment Schemes

A commitment scheme CS is a triple of PPT algorithms $\text{CS} := (\text{Setup}, \text{Com}, \text{Verify})$ such that:

- $\text{Setup}(1^\lambda) \rightarrow \text{pp}$. On input λ , outputs public parameters pp defining the message space aux .
- $\text{Com}(\text{pp}, \text{msg}) \rightarrow (C, \text{aux})$. On input pp and $\text{msg} \in \text{aux}$, outputs a commitment C together with opening information aux .
- $\text{Verify}(\text{pp}, \text{msg}, (C, \text{aux})) \rightarrow b \in \{0, 1\}$. Outputs 1 if (C, aux) is a valid opening of msg , and 0 otherwise.

Correctness. For all $\lambda \in \mathbb{N}$, we have

$$\Pr [\text{Verify}(\text{pp}, \text{msg}, (C, \text{aux})) = 1 \mid \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \text{msg} \leftarrow \$ \text{aux}, (C, \text{aux}) \leftarrow \$ \text{Com}(\text{pp}, \text{msg})] = 1.$$

Hiding. The scheme is computationally hiding if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$$\Pr \left[b = b' \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ (M_0, M_1, \text{state}) \leftarrow \$ \mathcal{A}_1(\text{pp}), \\ b \leftarrow \$ \{0, 1\}, \\ (C, \text{aux}) \leftarrow \$ \text{Com}(\text{pp}, M_b), \\ b' \leftarrow \$ \mathcal{A}_2(C, \text{state}) \end{array} \right] \leq \frac{1}{2} + \text{Adv}_{\text{CS}}^{\text{HID}}(\lambda).$$

Binding. The scheme is computationally binding if for all PPT adversaries $\mathcal{A}$:

$$\Pr \left[\begin{array}{l} M_1 \neq M_2, \\ \text{Verify}(\text{pp}, M_1, (C, \text{aux}_1)) = 1, \\ \text{Verify}(\text{pp}, M_2, (C, \text{aux}_2)) = 1 \end{array} \mid \begin{array}{l} \text{pp} \leftarrow \$ \text{Setup}(1^\lambda), \\ (M_1, M_2, \text{aux}_1, \text{aux}_2, C) \leftarrow \$ \mathcal{A}(\text{pp}) \end{array} \right] \leq \text{negl}(\lambda).$$

Extractable commitment schemes. A commitment scheme is extractable if there exists an alternative setup algorithm $\text{Setup}_{\text{Ext}}$ that outputs $(\text{pp}, \mathcal{T})$ such that:

- **Setup indistinguishability.** For any PPT adversary $\mathcal{D}$, there exists a PPT algorithm $\mathcal{B}$ such that

$$\left| \Pr[\mathcal{D}(\text{pp}) = 1 \mid \text{pp} \leftarrow \$ \text{Setup}(1^\lambda)] - \Pr[\mathcal{D}(\text{pp}) = 1 \mid (\text{pp}, \mathcal{T}) \leftarrow \$ \text{Setup}_{\text{Ext}}(1^\lambda)] \right| \leq \text{Adv}_{\text{CS}, \mathcal{B}}^{\text{SetupInd}}(\lambda).$$

- **Extractability.** There exists a PPT extractor Ext such that for any PPT adversary $\mathcal{A}$:

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{pp}, m, (C, \text{aux})) = 1, \\ (m, \text{aux}) \neq \text{Ext}(\mathcal{T}, C) \end{array} \mid \begin{array}{l} (\text{pp}, \mathcal{T}) \leftarrow \$ \text{Setup}_{\text{Ext}}(1^\lambda), \\ (C, \text{aux}) \leftarrow \$ \mathcal{A}(\text{pp}) \end{array} \right] \leq \text{Adv}_{\text{CS}, \mathcal{B}'}^{\text{Ext}}(\lambda).$$

A.4 BDLOP Commitments

We now describe the BDLOP commitment scheme following the above formalism.

- $\text{BDLOP.Setup}(1^\lambda) \rightarrow \text{pp}$. Sample parameters (k, n_r, ℓ, q) as a function of the security parameter λ . Define a distribution $\mathcal{D}$ over $\mathcal{R}_q$. Sample $(\mathbf{A}, \mathbf{B}) \leftarrow \$ \mathcal{R}_q^{k \times n_r} \times \mathcal{R}_q^{\ell \times n_r}$ and output

$$\text{pp} = (k, n_r, \ell, q, \mathcal{D}, \mathbf{A}, \mathbf{B}).$$

- $\text{BDLOP.Com}((\mathbf{m}, \mathbf{r}), \text{pp}) \rightarrow (\mathbf{t}_A, \mathbf{t}_B)$. On input a message vector $\mathbf{m} \in \mathcal{R}_q^\ell$, randomness $\mathbf{r} \leftarrow \$ \mathcal{D}^{n_r}$, and public parameters pp , output the commitment $(\mathbf{t}_A, \mathbf{t}_B) \in \mathcal{R}_q^k \times \mathcal{R}_q^\ell$ defined by

$$\begin{bmatrix} \mathbf{t}_A \\ \mathbf{t}_B \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{r} + \begin{bmatrix} 0 \\ \mathbf{m} \end{bmatrix} \text{ mod } q.$$

- $\text{BDLOP.Verify}(\text{pp}, (\mathbf{m}, \mathbf{r}), (\mathbf{t}_A, \mathbf{t}_B)) \rightarrow b \in \{0, 1\}$. Check that

$$\begin{bmatrix} \mathbf{t}_A \\ \mathbf{t}_B \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{r} + \begin{bmatrix} 0 \\ \mathbf{m} \end{bmatrix} \text{ mod } q,$$

and that $\mathbf{r}$ is small. Output 1 if all conditions hold, and 0 otherwise.

Lemma 9 (Hiding and Binding of the BDLOP Commitment). *Let BDLOP be the commitment scheme defined above. Then, for parameters chosen according to λ :*

- BDLOP is computationally binding under the $\text{MSIS}_{k+\ell, n_r, B}$ assumption.
- BDLOP is computationally hiding under the $\text{MLWE}_{n_r - (k+\ell), k+\ell, \mathcal{D}}$ assumption.

A.5 Extractable BDLOP Commitment

The BDLOP commitment scheme allows committing to a message $\mathbf{m} \in \mathcal{R}_q^\ell$ using a short randomness $\mathbf{r}$, together with public random matrices $\mathbf{A} \in \mathcal{R}_q^{k \times m}$ and $\mathbf{B} \in \mathcal{R}_q^{\ell \times m}$, as

$$\begin{bmatrix} \mathbf{t}_A \\ \mathbf{t}_B \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{r} + \begin{bmatrix} 0 \\ \mathbf{m} \end{bmatrix} \text{ mod } q.$$

The binding property follows from the MSIS assumption on the matrix $\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}$, as finding two distinct openings yields a short non-zero vector in its kernel. The hiding property follows from the MLWE assumption, provided that $\mathbf{r}$ is sampled from a distribution such that

$$\left(\begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix}, \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{r} \right)$$

is computationally indistinguishable from uniform.

Extractability via simulated setup. For the purpose of the security proof of our blind signature, we rely on a simulated setup for the BDLOP commitment scheme that enables extraction. Concretely, in the reduction, the challenger generates the public matrices $(\mathbf{A}, \mathbf{B})$ as follows. It samples $\mathbf{A} \leftarrow \mathcal{R}_q^{k \times n_r}$ together with short matrices $\mathbf{S} \leftarrow \mathcal{D}_{\mathbf{B}}^{\ell \times k}$ and $\mathbf{E} \leftarrow \mathcal{D}_{\mathbf{B}}^{\ell \times n_r}$ and sets

$$\mathbf{B} := \mathbf{SA} + p\mathbf{E}.$$

The commitment algorithm itself remains unchanged, namely

$$\begin{bmatrix} \mathbf{t}_A \\ \mathbf{t}_B \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{B} \end{bmatrix} \mathbf{r} + \begin{bmatrix} 0 \\ \mathbf{m} \end{bmatrix} \bmod q.$$

Using the relation $\mathbf{B} = \mathbf{SA} + p\mathbf{E}$, we obtain

$$\mathbf{t}_B = \mathbf{St}_A + p\mathbf{Er} + \mathbf{m} \bmod q.$$

Therefore, by reducing modulo p , the error term vanishes and the reduction can recover the committed message as

$$\mathbf{m} = \mathbf{t}_B - \mathbf{St}_A \bmod p.$$

Indistinguishability of the setup. We argue that the above simulated setup is computationally indistinguishable from the real one under the MLWE assumption. In particular, any distinguisher between $(\mathbf{A}, \mathbf{B})$ sampled uniformly at random and $(\mathbf{A}, \mathbf{SA} + p\mathbf{E})$ for short $\mathbf{S}, \mathbf{E}$ can be used to construct an adversary against the MLWE problem. Therefore, this modification is indistinguishable to the adversary.

Remark 3 (Random-oracle-based setup). The simulated setup above is only used in the security reduction. In the real protocol, no trusted party is needed to generate the public matrices $(\mathbf{A}, \mathbf{B})$. Instead, these matrices can be derived from public seeds by using a hash function, modeled as a random oracle, to expand the seeds into pseudorandom elements of $\mathcal{R}_q$. Thus, all users can locally derive the same commitment parameters. The indistinguishability argument above ensures that replacing uniformly sampled matrices by programming the random oracle with matrices generated in the way above does not affect the adversary's view.

A.6 Non-Interactive Zero-Knowledge Proofs

We formally define NIZK and consider only transparent NIZK schemes without a trusted setup. Our definition is adapted from [BFM⁺25b].

A non-interactive zero-knowledge proof system for a family of language $\mathcal{L}$ which we denote NIZK is defined by the following algorithms:

- $\text{pp} \leftarrow \text{NIZK}_{\mathbf{L}}.\text{Setup}(1^\lambda)$. Given a security parameter λ , outputs public parameters pp
- $\pi \leftarrow \text{NIZK}_{\mathbf{L}}.\text{Prove}(\text{pp}, w, x)$. Given a witness w , a language $\mathbf{L} \in \mathcal{L}$ and a statement x such that $(x, w) \in L$, outputs a proof π .
- $1/0 \leftarrow \text{NIZK}_{\mathbf{L}}.\text{Verify}(\text{pp}, \pi, x)$. Given a proof π , a language $\mathbf{L} \in \mathcal{L}$ and a statement x , outputs 1 (“accept”) or 0 (“reject”).

The algorithm *Prove* is probabilistic, while the algorithm *Verify* is deterministic. The proof system satisfies the following properties:

- **Completeness.** For any language $\mathbf{L} \in \mathcal{L}$ and any statement-witness pair $(x, w) \in L$,

$$\Pr[\text{NIZK}_{\mathbf{L}}.\text{Verify}(\text{pp}, \pi, x) = 1 \mid \pi \leftarrow \text{NIZK}_{\mathbf{L}}.\text{Prove}(\text{pp}, w, x)] = 1.$$

- **ε -Zero-Knowledge.** There exists a probabilistic polynomial-time simulator Sim such that for any probabilistic polynomial-time distinguisher $\mathcal{A}$,

$$|\Pr[\mathcal{A}(\text{pp}, \pi) = 1 : \pi \leftarrow \text{NIZK}_L.\text{Prove}(\text{pp}, w, x)] - \Pr[\mathcal{A}(\pi) = 1 : \pi \leftarrow \text{Sim}(\text{pp}, L, x)]| \leq \varepsilon.$$

- **ε -Knowledge Soundness.** There exists a probabilistic polynomial-time extractor Ext such that for any probabilistic polynomial-time prover Prove^* and any language $L \in \mathcal{L}$, if

$$\Pr[\text{NIZK}_L.\text{Verify}(\text{pp}, \pi, x) = 1 : \pi \leftarrow \text{Prove}^*(\text{pp}, L, x)] \geq \varepsilon + \eta,$$

for some non-negligible function η , then $\Pr[(x, w) \in L : w \leftarrow \text{Ext}(\text{pp}, \text{Prove}^*, L, x)]$ is non-negligible.

A.7 Rank-1 Constraint Systems

Rank-one constraint systems (R1CS) are commonly used to express computations (*arithmetic circuits*) by encoding them into a suitable system of quadratic equations. Such encoding is particularly useful in arguments systems—in particular, for our purposes, *zero-knowledge Succinct Non-interactive ARguments of Knowledge* (zkSNARKs, [BCC⁺17]).

Namely, the R1CS *relation* is an NP-relation consisting of the statement/witness pairs $((\mathbb{F}, k, n, m, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{v}), \mathbf{w})$, where $\mathbb{F}$ is a finite field, k, n, m are the numbers of inputs, variables and constraints respectively ($k \leq n$), $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are $m \times (1 + n)$ -matrices over $\mathbb{F}$, and $\mathbf{v}, \mathbf{w}$ are vectors over $\mathbb{F}$ of length k and $n - k$ respectively, such that $(\mathbf{A}\mathbf{z}) \circ (\mathbf{B}\mathbf{z}) = \mathbf{C}\mathbf{z}$ for $\mathbf{z} = (1, \mathbf{v}, \mathbf{w})$. Here “ $\circ$ ” denotes the entry-wise product.

Several widely used argument systems take R1CS as their native input—for example, Aurora [BCR⁺19], Ligerio [AHIV17] and Orion [XZS22].

B Deferred Lemmas

Definition 9. For any $\mathbf{A} \in \mathcal{R}_q^{k \times \ell}$, let $D_{\ell, k, \mu, \nu}^{\text{bd-MLWE}}(\mathbf{A})$ be the distribution defined as $\{[\mathbf{A}\mathbf{s} + \mathbf{e}]_\nu | (\mathbf{s}, \mathbf{e}) \leftarrow D_\mu^\ell \times D_\mu^k\}$, that is, it samples from the $\text{MLWE}_{\ell, k, \mu}$ distribution with ν bits dropped.

Lemma 10 (cf. [DKM⁺24]). For any $\mu > 2n \cdot q^{\frac{1}{k+\ell} + \frac{2}{n\ell}}$ and $\nu < \log(q) - 2$, we have,

$$\Pr[H_\infty(D_{\ell, k, \mu, \nu}^{\text{bd-MLWE}}(\mathbf{A})) \geq n - 1] > 1 - 2^{-n+1}$$

Lemma 11 (cf. Lemma 3.2 [DKM⁺24]). Let ν, q be positive integers such that $q > 2$ and set $q_\nu = \lfloor q/2^\nu \rfloor$. Moreover, assume q and ν satisfy $q_\nu = \lfloor q/2^\nu \rfloor$, that is, q can be decompressed as $q = 2^\nu \cdot q_\nu + q_\perp$ for $q_\perp$ in $[0, 2^{\nu-1} - 1]$. Then for any $x \in \mathbb{Z}_q$, we have,

$$|x - 2^\nu \cdot \overline{[x]_\nu}| < 2^\nu - 1$$

and $\forall \mathbf{x}, \boldsymbol{\delta} \in \mathbb{Z}_q^n$, we have,

$$\|2^\nu \cdot (\overline{[\mathbf{x} + \boldsymbol{\delta}]_\nu} - \overline{[\mathbf{x}]_\nu}) \mod q\| \leq \|2^\nu \cdot \overline{[\boldsymbol{\delta}]_\nu} \mod q\| + \|\mathbf{1}\| \cdot f(\nu)$$

with,

$$f(\nu) = \begin{cases} 3 & \text{if } \nu = 1, \\ 6 & \text{if } \nu = 2, \\ 12 & \text{if } \nu = 3, \\ 2^\nu & \text{elsewise.} \end{cases}$$

C Correctness and Security of Raccoon*

C.1 Correctness of Raccoon*

Proof. The proof follows the same structure as the correctness proof of Raccoon in [DKM⁺24, Appendix C], with the additional shift introduced in Raccoon*. For completeness, we provide the full argument.

Correctness of the challenge verification is immediate, since the challenge c is recomputed deterministically from the verification equation. It therefore suffices to show that the norm bound is satisfied with overwhelming probability. Recall that $\hat{\mathbf{t}} = \mathbf{A}\mathbf{s} + \mathbf{e}_{\text{kg}}$, $\mathbf{t} = \lfloor \hat{\mathbf{t}} \rfloor_{\nu_t}$, $\mathbf{z} = c\mathbf{s} + \mathbf{r}' + \mathbf{r}$, $\hat{\mathbf{w}} = \mathbf{A}(\mathbf{r}' + \mathbf{r}) + (\mathbf{e}' + \mathbf{e})$, and $\mathbf{w} = \lfloor \hat{\mathbf{w}} \rfloor_{\nu_w}$. We first expand the intermediate value $\mathbf{y}$ computed during verification:

$$\begin{aligned} \mathbf{y} &:= \lfloor \mathbf{A}\mathbf{z} - 2^{\nu_t} \cdot c \cdot \bar{\mathbf{t}} \rfloor_{\nu_w} \\ &= \lfloor c\mathbf{A}\mathbf{s} + \mathbf{A}\mathbf{r}' + \mathbf{A}\mathbf{r} - 2^{\nu_t} \cdot c \cdot \bar{\mathbf{t}} \rfloor_{\nu_w} \\ &= \lfloor \mathbf{A}(\mathbf{r}' + \mathbf{r}) + \mathbf{e}' + \mathbf{e} + c(\hat{\mathbf{t}} - 2^{\nu_t} \cdot \lfloor \hat{\mathbf{t}} \rfloor_{\nu_t}) - c \cdot \mathbf{e}_{\text{kg}} - \mathbf{e}' - \mathbf{e} \rfloor_{\nu_w} \\ &= \lfloor \hat{\mathbf{w}} + c \cdot \mathbf{a}_t - \mathbf{a} \rfloor_{\nu_w}, \end{aligned}$$

where we rearranged terms to isolate rounding and noise contributions, and define $\mathbf{a}_t := \hat{\mathbf{t}} - 2^{\nu_t} \cdot \lfloor \hat{\mathbf{t}} \rfloor_{\nu_t}$ and $\mathbf{a} := c \cdot \mathbf{e}_{\text{kg}} + \mathbf{e}' + \mathbf{e}$.

We now bound the norm of the hint $\mathbf{h} = \mathbf{w} - \mathbf{y}$:

$$\begin{aligned} \|2^{\nu_w} \cdot \bar{\mathbf{h}} \bmod q\|_2 &= \|2^{\nu_w} \cdot (\bar{\mathbf{w}} - \bar{\mathbf{y}}) \bmod q\|_2 \\ &= \|2^{\nu_w} \cdot (\lfloor \hat{\mathbf{w}} \rfloor_{\nu_w} - \lfloor \hat{\mathbf{w}} + c \cdot \mathbf{a}_t - \mathbf{a} \rfloor_{\nu_w}) \bmod q\|_2 \\ &\leq \|2^{\nu_w} \cdot \overline{\lfloor c \cdot \mathbf{a}_t - \mathbf{a} \rfloor_{\nu_w}} \bmod q\|_2 + \sqrt{nk} \cdot 2^{\nu_w+1} \\ &\leq \|c \cdot \mathbf{a}_t - \mathbf{a}\|_2 + \sqrt{nk} \cdot 2^{\nu_w+1} \\ &\leq \|c \cdot \mathbf{a}_t\|_2 + \|\mathbf{a}\|_2 + \sqrt{nk} \cdot 2^{\nu_w+1}. \end{aligned}$$

We now bound each term separately. First, using the definition of $\mathbf{a}_t$ and rounding in Lemma 11, to get

$$\|c \cdot \mathbf{a}_t\|_2 \leq \|c\|_1 \cdot \sqrt{nk} \cdot 2^{\nu_t}.$$

It remains to bound $\|(\mathbf{z}, \mathbf{a})\|_2$:

$$\begin{aligned} \|(\mathbf{z}, \mathbf{a})\|_2 &= \|(c\mathbf{s} + \mathbf{r}' + \mathbf{r}, c \cdot \mathbf{e}_{\text{kg}} + \mathbf{e}' + \mathbf{e})\|_2 \\ &\leq \|c \cdot (\mathbf{s}, \mathbf{e}_{\text{kg}})\|_2 + \|(\mathbf{r}' + \mathbf{r}, \mathbf{e}' + \mathbf{e})\|_2 \\ &\leq \|c \cdot (\mathbf{s}, \mathbf{e}_{\text{kg}})\|_2 + \|(\mathbf{r}', \mathbf{e}')\|_2 + \|(\mathbf{r}, \mathbf{e})\|_2 \\ &\leq (\|c\|_1 \sigma_t + \sigma_{\mathbf{w}'} + \sigma_{\mathbf{w}}) \sqrt{n(k + \ell)}. \end{aligned}$$

Combining all bounds yields exactly the claimed norm bound B_2 , which completes the proof that the norm bound holds with overwhelming probability.

C.2 Unforgeability of Raccoon*

Proof. We prove unforgeability by reducing any forgery against Raccoon* to a forgery against Raccoon via a sequence of hybrid games.

Game $\mathbf{G}_0$ corresponds to the standard EUF-CMA experiment. In Game $\mathbf{G}_1$, we replace the random oracle $\mathbf{H}_c$ by a lazily sampled function. In Game $\mathbf{G}_2$, we simulate the signing oracle of Raccoon* using access to a signing oracle for Raccoon.

We show that the adversary's view remains identical or computationally indistinguishable between consecutive games, and that any successful forgery in Game $\mathbf{G}_2$ can be transformed into a valid EUF-CMA forgery against Raccoon.

G₀. Game **G₀** corresponds to the standard EUF-CMA experiment for the signature scheme **Raccoon^{*}**. The challenger samples a key pair

$$(\text{vk}, \text{sk}) \leftarrow \text{KGen}(\text{pp})$$

and gives the verification key vk to the adversary $\mathcal{A}$.

The adversary is given oracle access to:

- a signing oracle $\text{Sign}^*(\text{sk}, \cdot, \cdot, \cdot)$, which on input $(M, \mathbf{r}, \mathbf{e})$ returns a signature generated according to the honest signing algorithm of **Raccoon^{*}**, and
- a random oracle H_c .

The adversary $\mathcal{A}$ may adaptively query the signing oracle on at most Q_s inputs and the random oracle on at most Q_H inputs.

Finally, $\mathcal{A}$ outputs a pair (M^*, σ^*) . We say that $\mathcal{A}$ wins in **G₀** if

$$\text{Verify}(\text{vk}, M^*, \sigma^*) = 1 \quad \text{and} \quad M^* \notin \mathcal{Q}_s,$$

where $\mathcal{Q}_s$ denotes the set of messages queried to the signing oracle.

Let S_0 denote the event that $\mathcal{A}$ wins in game **G₀**.

G₁. Game **G₁** is identical to **G₀** except that the random oracle H_c is simulated via lazy sampling.

The challenger maintains a table $\mathcal{T}_{H_c}$, initially empty. Upon receiving a query $(\text{vk}, M, \mathbf{w})$, the challenger proceeds as follows:

- If $(\text{vk}, M, \mathbf{w}) \in \mathcal{T}_{H_c}$, it returns the previously stored value.
- Otherwise, it samples a fresh uniform challenge c from the challenge space, stores $\mathcal{T}_{H_c}(\text{vk}, M, \mathbf{w}) = c$, and returns c .

The signing oracle of **Raccoon^{*}** is unchanged.

Since the lazy-sampled random oracle is identically distributed to a truly random function, the adversary's view in **G₁** is identical to its view in **G₀**. Therefore,

$$\Pr[S_1] = \Pr[S_0].$$

G₂. In Game **G₂**, we replace the signing oracle of **Raccoon^{*}** by a simulator that uses access to a signing oracle for **Raccoon**.

Upon receiving a signing query $(M, \mathbf{r}, \mathbf{e})$, the challenger proceeds as follows:

1. Query the **Raccoon** signing oracle on M to obtain a valid signature $(c, \mathbf{z}, \mathbf{h})$.
2. Set

$$\mathbf{z}^* := \mathbf{z} + \mathbf{r}.$$

3. Sample

$$(\mathbf{r}', \mathbf{e}') \leftarrow \$ D_{\mathbf{w}'}^\ell \times D_{\mathbf{w}'}^k$$

and compute

$$\begin{aligned} \mathbf{w}^* &:= \mathbf{A}(\mathbf{r} + \mathbf{r}') + \mathbf{e}' + \mathbf{e}, \\ \mathbf{w} &:= \lfloor \mathbf{A}\mathbf{z} - 2^{\nu_t} \cdot \text{ct} \rfloor_{\nu_w} + \mathbf{h}, \\ \mathbf{h}^* &:= \mathbf{h} - \mathbf{w} + \mathbf{w}^*. \end{aligned}$$

4. If $(\text{vk}, M, \mathbf{w}^*) \in \mathcal{T}_{H_c}$, restart from Step 1. Otherwise, program the random oracle by setting

$$H_c(\text{vk}, M, \mathbf{w}^*) := c.$$

5. If

$$\|(\mathbf{z}^*, 2^{\nu_w} \mathbf{h}^*)\|_2 > B_2,$$

output $\perp$. Otherwise, return the signature

$$\sigma^* := (c, \mathbf{z}^*, \mathbf{h}^*).$$

Correctness of the simulation. We argue that the simulated signing oracle produces outputs that are identically distributed to those of the honest signing algorithm Sign^* .

First, the value $\mathbf{z}^* = c\mathbf{s} + \mathbf{r}' + \mathbf{r}$ has the same distribution as in Raccoon^* , since $\mathbf{z} = c\mathbf{s} + \mathbf{r}'$ in Raccoon . The value $\mathbf{w}^*$ is constructed exactly according to the definition of the commitment value in Raccoon^* , using freshly sampled randomness $(\mathbf{r}', \mathbf{e}')$ and the adversarial input $(\mathbf{r}, \mathbf{e})$.

Moreover, the value $\mathbf{h}^*$ is defined so that the verification equation holds by construction. Finally, the challenge c is consistently programmed into the random oracle at the point $(\text{vk}, M, \mathbf{w}^*)$.

Therefore, conditioned on no restart, the output distribution of the simulator is identical to that of the honest signing algorithm Sign^* .

Restart analysis. A restart occurs only if $(\text{vk}, M, \mathbf{w}^*)$ was already queried to the random oracle. Since at most Q_H oracle queries and Q_s signing queries occur, the probability of such a collision in any given attempt is at most

$$(Q_H + Q_s) \cdot 2^{-\lambda+1},$$

as shown in [DKM⁺24, Appendix C]. Hence, the expected number of repetitions is $1/(1 - \text{negl}(\lambda)(\lambda))$, and the simulator runs in expected polynomial time. Moreover, the restarts do not affect the distribution of the adversary's view.

Reduction to Raccoon. In Game $\mathbf{G}_2$, the signing oracle responds to queries by invoking the Raccoon signing oracle and applying the above transformation.

Since the verification algorithms of Raccoon and Raccoon^* coincide (for the same bound B_2), any successful forgery (M^*, σ^*) produced by $\mathcal{A}$ in $\mathbf{G}_2$ is also a valid EUF-CMA forgery against Raccoon on the same message.

Therefore, there exists a PPT adversary $\mathcal{B}$ such that

$$\mathcal{A}_{\text{EUF-CMA}}^{\text{Raccoon}^*}(\mathcal{A}) \leq \mathcal{A}_{\text{EUF-CMA}}^{\text{Raccoon}}(\mathcal{B}).$$

D Security Reduction of our Blind Signature

D.1 Blindness

Proof. We prove blindness of our protocol via a sequence of hybrid games.

$\mathbf{G}_0$. This game corresponds to the blindness experiment defined in Section 2.4. The adversary $\mathcal{A}$ selects two messages and a verification key. A random bit $b \in \{0, 1\}$ is chosen, and the challenger executes the protocol with the adversary using msg_b . At the end, the adversary receives signatures $(\text{Sig}_b, \text{Sig}_{1-b})$ and outputs a guess for b . Let S_0 denote the event that $\mathcal{A}$ correctly guesses b .

$\mathbf{G}_1$. In game $\mathbf{G}_1$, we modify the subroutine User_2 by replacing the proof π_1 with a simulated proof generated by the simulator of the NIZK scheme.

Let S_1 denote the event that $\mathcal{A}$ correctly guesses in $\mathbf{G}_1$. By the zero-knowledge property of π_1 , there exists a PPT adversary $\mathcal{B}$ such that

$$|\Pr[S_0] - \Pr[S_1]| \leq \text{Adv}_{\pi_1, \mathcal{B}}^{\text{ZK}}(\lambda).$$

G₂. In game **G₂**, we modify the subroutine **User₁** so that the commitment $C_{\text{msg}, \mathbf{r}, \mathbf{e}}$ is replaced by a commitment C^* to a uniformly random message from the message space of the commitment scheme.

Let S_2 denote the event that $\mathcal{A}$ correctly guesses in **G₂**. By the hiding property of the commitment scheme, there exists a PPT adversary $\mathcal{D}$ such that

$$|\Pr[S_1] - \Pr[S_2]| \leq \text{Adv}_{\text{CS}, \mathcal{D}}^{\text{HID}}(\lambda).$$

G₃. In game **G₃**, we modify the subroutine **User₃** so that it attempts to extract a witness from the proof π_2 after verification. If extraction fails, the subroutine aborts; otherwise it proceeds as in **G₂**.

Let S_3 denote the event that $\mathcal{A}$ correctly guesses in **G₃**. The view differs from **G₂** only if extraction fails. By knowledge soundness of π_2 , there exists a PPT adversary $\mathcal{K}$ such that

$$|\Pr[S_2] - \Pr[S_3]| \leq \text{Adv}_{\pi_2, \mathcal{K}}^{\text{KS}}(\lambda).$$

G₄. In game **G₄**, we modify the subroutine **User₃** so that, using the extracted witness $(\mathbf{s}, \mathbf{r}')$, it recomputes

$$\mathbf{z}^* = c\mathbf{s} + \mathbf{r}' + \mathbf{r},$$

where $\mathbf{r}$ is the blinding randomness chosen in **User₁**.

This expression is identical to the one computed in the honest execution, hence the distribution of $\mathbf{z}$ remains unchanged.

Let S_4 denote the event that $\mathcal{A}$ correctly guesses in **G₄**. We therefore have

$$\Pr[S_3] = \Pr[S_4].$$

G₅. In game **G₅**, we modify the subroutine **User₂** by replacing the ciphertext ctx_c with an encryption of an independently sampled value c^* .

Let S_5 denote the event that $\mathcal{A}$ correctly guesses in **G₅**. By IND-CPA security of the homomorphic encryption scheme **HE**, there exists a PPT adversary $\mathcal{S}$ such that

$$|\Pr[S_4] - \Pr[S_5]| \leq \text{Adv}_{\text{HE}, \mathcal{S}}^{\text{IND-CPA}}(\lambda).$$

G₆. In the final game **G₆**, all components generated by the user are independent of the challenger's bit b : the proofs are simulated, the commitment hides the message, the ciphertext encrypts an independent value, the response $\mathbf{z}$ is computed from extracted values and fresh blinding randomness, independent of the challenge bit.

Hence, the adversary's view is independent of b , except for the statistical deviation introduced by the blinding mechanism. The only dependence on signer randomness arises from $\mathbf{r}'$, which is masked by the larger blinding noise $\mathbf{r}$.

By standard bounds on discrete Gaussian distributions, we have

$$\Delta(r + r', r) \leq O\left(\frac{\|r'\|}{\sigma_w}\right).$$

In particular, for $\sigma_w \gg \|r'\|$, this yields statistical distance at most ϵ_{Stat} .

Therefore,

$$\left|\Pr[S_5] - \frac{1}{2}\right| \leq \epsilon_{\text{Stat}}.$$

Final bound. Combining the bounds from all hybrids, we conclude that for any PPT adversary $\mathcal{A}$,

$$\mathcal{A}_{\text{BLD}}^{\text{BRaccoon}}(\mathcal{A}) \leq \text{Adv}_{\pi_1, \mathcal{B}}^{\text{ZK}}(\lambda) + \text{Adv}_{\text{CS}, \mathcal{D}}^{\text{HID}}(\lambda) + \text{Adv}_{\pi_2, \mathcal{K}}^{\text{KS}}(\lambda) + \text{Adv}_{\text{HE}, \mathcal{S}}^{\text{IND-CPA}}(\lambda) + \epsilon_{\text{Stat}}.$$

This concludes the proof of blindness.

E Poseidon Hash Function

Poseidon from [GKR⁺21] is a sponge-based hash function instantiated with a permutation $\text{Poseidon} : \mathbb{F}_p^t \rightarrow \mathbb{F}_p^t$ defined over a prime field $\mathbb{F}_p$, where t denotes the state width. The permutation consists of $\text{Total}_{\text{round}} = \text{F}_{\text{round}} + \text{P}_{\text{round}}$ rounds structured as

$$\underbrace{\text{Full rounds}}_{\text{F}_{\text{round}}/2} \circ \underbrace{\text{Partial rounds}}_{\text{P}_{\text{round}}} \circ \underbrace{\text{Full rounds}}_{\text{F}_{\text{round}}/2}.$$

Each round applies the following operations in sequence: addition of round constants, application of a non-linear S-box layer, and multiplication by a maximum distance separable (MDS) matrix $M \in \mathbb{F}_p^{t \times t}$. The non-linear layer is defined by the power map

$$S(x) = x^\alpha,$$

where $\alpha \geq 3$ satisfies $\gcd(\alpha, p-1) = 1$. The S-box is applied to all t state elements, whereas during partial rounds it is applied only to the first state element, the remaining $t-1$ elements being processed linearly.

The hash function itself is obtained via a standard sponge construction with capacity c and rate $r = t - c$. For a target security level λ , the capacity is chosen such that $c \geq 2\lambda$.

F Discussion on Table 1

Table 1 combines values taken directly from the literature with a small number of estimates and recomputations. We detail below, for each compared work, the source of the reported data.

[AKSY22]. We consider the “full-fledged blind” variant described in [AKSY22, Appendix A, Fig. 5].

In this variant the public values $\mathbf{A}$ and PKE.pk are derived from random oracles. For the concrete size estimates, we use the instantiation of [AKSY22, Section 4.4]. The issuing transcript is the exchange of $\mathbf{t}$ and $\mathbf{y}$; concretely, only the first component $\mathbf{y}_1$ of $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2)$ is sent, since $\mathbf{y}_2$ is publicly recoverable from $\mathbf{t}$ and the verification key $\mathbf{vk} = C$. This gives a communication cost of 1.37 KB. The final signature is (π, ct) , whose size is estimated as 45.19 KB. Finally, $\mathbf{vk}$ is one element of $\mathbb{Z}[X]/(7213, X^{512} + 1)$, so that

$$|\mathbf{vk}| = 512 \cdot 13/8 = 832 \text{ bytes} \approx 0.8 \text{ KB}.$$

The scheme supports an unbounded, i.e. polynomially many, number of signing queries per public key.

[dPK22]. We take the concrete parameters re-evaluated in [HSS24, §4.2]: the original NTRU parameter set was later observed to fall short of the claimed 128-bit security level due to attacks on overstretched-NTRU parameters [ABD16, CJL16], while the updated parameters restore the intended security level. This leaves the pre-signature and final signature sizes unchanged and only mildly increases the communication cost. We also account for the induced increase in the verification key, which consists of three elements of $\mathcal{R}_q$.

[AGJ⁺24]. For this anonymous-credential system, which refines the approach of [JRS23], we use the size estimates reported in [AGJ⁺24, Tables 8.1, 8.4–8.6 and Algorithm 6.3]. Since revealing the underlying SEP signature would make presentations linkable to the issuing session, we count the non-interactive credential-showing proof as the blind signature analogue, i.e., as the object given as input to public verification; its size is 79.58 KB. The communication cost counts the issuing/signing transcript exchanged between the user and the issuer, and is estimated as

$$2.38 + 35.99 + 6.81 = 45.18 \text{ KB},$$

where 2.38 KB estimates the commitment by the size of an element of the same type, namely upk ; 35.99 KB is the issuance proof size; and 6.81 KB is the size of the emitted SEP signature sent by the issuer in [AGJ⁺24, Algorithm 6.3].

[BLNS23a]. The relevant data are reported in [BLNS23a, Tab. 3 and Fig. 4]. We estimate the communication cost by counting the full signing transcript exchanged between the user and the signer, namely the user’s first message $(\mathbf{c}, \mathbf{ct}, \pi_1)$ and the signer’s response $\mathbf{s}$. Thus, the communication cost is

$$|\pi_1| + |\mathbf{ct}| + |\mathbf{c}| + |\mathbf{s}|.$$

For the parameters of [BLNS23a, Tab. 3], with $q = 7933$ and ring degree $d = 512$, an element of $\mathcal{R}_q$ requires approximately $512 \cdot 13/8 \simeq 832$ bytes, i.e., about 0.8 KB. Hence $|\mathbf{c}| \simeq 0.8$ KB, while $\mathbf{s} \in \mathcal{R}_q^2$ contributes about 1.6 KB. The ciphertext size $|\mathbf{ct}|$ depends on the chosen IND-CPA encryption scheme. The bound on the number Q_s of signing queries is governed by [BLNS23a, Thms. 3.2 and 3.3]; in our table, we set $Q_s = 2^{20}$ as a conservative choice.

[JS25b] We used the authors’ parameter-estimation script⁸. The script confirms the transcript and blind signature sizes reported in [JS25b, Table 5.5], and additionally yields the verification-key size and the maximum number of signatures per public key.

[HKLN20] We used the signature size from [HKLN20, Fig. 9], the number of signatures per public key specified at the end of [HKLN20, §6], and the sizes computed in [AHJ21, Table 1].

[KLR25] The sizes were computed using a script that the authors kindly provided in a private communication.

We also used [JS25a, Table 5.4] and [KLR25, Table 1] to cross-check some of the entries in our table.

G SampleInBall Procedure from [PFW23]

The **SampleInBall** procedure from [PFW23] deterministically maps a seed c to a challenge vector $C \in \{-1, 0, 1\}^d \subset \mathbb{F}_q^d$ containing exactly ω non-zero entries in $\{-1, 1\}$. Conditioned on no abort, the output distribution is uniform over all such vectors.

The algorithm initializes C as the all-zero vector. It then iteratively selects ω distinct positions using a truncated inside-out variant of the Fisher–Yates shuffle. At iteration i , an index j is sampled uniformly in $\{0, \dots, d - i\}$ (via rejection sampling), the entries C_{d-i} and C_j are swapped, and C_j is assigned a random sign. This ensures that exactly ω positions contain non-zero values and that their locations and signs are uniformly distributed.

In our instantiation, the randomness used for index selection and sign generation is derived from a Poseidon-based XOF applied to c . We assume that ω is a multiple of 8, which allows signs to be extracted in batches of eight. This choice slightly simplifies the arithmetic constraints and yields a more efficient R1CS representation.

⁸ <https://github.com/truncatedsampler/truncated-sampler/tree/main/scripts>

Algorithm 1 SampleInBall using Poseidon as an XOF

```

1: function SAMPLEINBALL(seed)
2:    $C \leftarrow 0 \in \{-1, 0, 1\}^d \subset \mathbb{F}_q^d$ 
3:    $(h_1, \dots, h_\omega, h_{\omega+1}, \dots, h_{\omega+\omega/8}) \leftarrow \text{PoseidonXOF}(\text{seed})$ 
4:    $\vec{s\tilde{g}} \leftarrow \text{GenSigns}(h_{\omega+1}, \dots, h_{\omega+\omega/8})$ 
5:   if  $\vec{s\tilde{g}} = \perp$  then
6:     Abort
7:   end if
8:   for  $i \leftarrow 1$  to  $\omega$  do
9:      $q^* \leftarrow q - (q \bmod (d - i + 1))$ 
10:    if  $h_i \geq q^*$  then
11:      Abort
12:    end if
13:     $j \leftarrow h_i \bmod (d - i + 1)$ 
14:    swap $(C_{d-i}, C_j)$ 
15:     $C_j \leftarrow (-1)^{s\tilde{g}_i}$ 
16:  end for
17:  return  $C$ 
18: end function

```

Algorithm 2 GenSigns

```

1: function GENSIGNS( $h_1, \dots, h_{\omega/8}$ )
2:    $\vec{s\tilde{g}} \leftarrow \varepsilon \in \{0, 1\}^*$ 
3:    $q^* \leftarrow q - (q \bmod 2^8)$ 
4:   for  $i \leftarrow 1$  to  $\omega/8$  do
5:     if  $h_i \geq q^*$  then
6:       return  $\perp$ 
7:     end if
8:      $h' \leftarrow h_i \bmod 2^8$ 
9:      $(h'_0, \dots, h'_7) \leftarrow \text{ConvBase2}(h')$ 
10:     $\vec{s\tilde{g}} \leftarrow \vec{s\tilde{g}} \parallel (h'_0, \dots, h'_7)$ 
11:  end for
12:  return  $\vec{s\tilde{g}}$ 
13: end function

```

H Detailed description of $\mathbf{L}_1$ and $\mathbf{L}_2$

We write $\mathbf{vk} = (A_{\text{sig}}, t)$ for the Raccoon verification key and $\mathbf{ek} = (A_{\text{bgv}}, b_{\text{bgv}})$ for the BGV encryption key. In the optimized instantiation, the Raccoon relations, the BGV ciphertext equations, and the BDLOP commitments are all expressed over the same modulus q . Hence,

$$b_{\text{bgv}} = A_{\text{bgv}} s_{\text{bgv}} + h e_{\text{bgv}} \pmod{q},$$

where h is the BGV plaintext modulus. The parameter search chooses q and h so that BGV decryption and the extractable BDLOP simulation do not wrap around modulo q ; therefore no lifting quotient is needed in the statements below.

The user computes

$$\mu = H(\text{msg}, \mathbf{vk}),$$

where μ is a 2λ -bit digest. Note that we use the seed representation of $\mathbf{vk}$, thus in the estimate its considered as a 2λ -bit digest.

For a BDLOP commitment with parameters (A_i, B_i) and short randomness γ , we use the shorthand

$$\text{BDLOP.Com}_i(m; \gamma) = (t_A, t_B) \iff \begin{cases} t_A = A_i \gamma \pmod{q}, \\ t_B = B_i \gamma + m \pmod{q}. \end{cases}$$

In the concrete instantiation, the first commitment is the extractable first-round commitment, while the second one is the linking commitment between the lattice and arithmetic proof components.

$$\mathcal{L}_{1,1} = \left\{ \begin{array}{l} ((t_A^0, t_B^0), (t_A^1, t_B^1), \text{ctx}_c, w', \mathbf{vk}, \mathbf{ek}, \text{pp}) : \\ \exists \text{msg}, \mu, r, e, w, c, e_w, \gamma_0, \gamma_1, \rho_c, \eta_{c,1}, \eta_{c,2} \text{ s.t.} \\ t_A^0 = A_0 \gamma_0 \pmod{q}, \\ t_B^0 = B_0 \gamma_0 + (\text{msg}, r, e) \pmod{q}, \\ \|\gamma_0\| \leq B_{\gamma,0}, \\ t_A^1 = A_1 \gamma_1 \pmod{q}, \\ t_B^1 = B_1 \gamma_1 + (w, \mu, c) \pmod{q}, \\ \|\gamma_1\| \leq B_{\gamma,1}, \\ w' + A_{\text{sig}} r + e = 2^{\nu_w} w + e_w \pmod{q}, \\ \|(r, e)\|_2 \leq B_w, \quad \|e_w\|_\infty < 2^{\nu_w-1}, \\ \text{ctx}_c = (u_c, v_c), \\ u_c = A_{\text{bgv}}^T \rho_c + h \eta_{c,1} \pmod{q}, \\ v_c = b_{\text{bgv}}^T \rho_c + h \eta_{c,2} + c \pmod{q}, \\ (\rho_c, \eta_{c,1}, \eta_{c,2}) \in \mathcal{B}_{\text{enc}}. \end{array} \right\}.$$

$$\mathcal{L}_{1,2} = \left\{ \begin{array}{l} ((t_A^1, t_B^1), \text{pp}, \mathbf{vk}) : \\ \exists w, \mu, c, \gamma_1 \text{ s.t.} \\ t_A^1 = A_1 \gamma_1 \pmod{q}, \\ t_B^1 = B_1 \gamma_1 + (w, \mu, c) \pmod{q}, \\ \|\gamma_1\| \leq B_{\gamma,1}, \\ \mu = \text{Poseidon}(\mathbf{vk}, \text{msg}), \\ \rho = \text{Poseidon}(\mu, w), \\ c = \text{SampleInBall}(\rho). \end{array} \right\}.$$

Thus, $\mathbf{L}_{1,2}$ checks only the arithmetic challenge-generation part: one Poseidon hash absorbing the private digest μ and the committed value w , followed by the deterministic **SampleInBall** procedure. The binding property of the linking BDLOP commitment guarantees that the same (w, μ, c) is used in $\mathbf{L}_{1,1}$ and $\mathbf{L}_{1,2}$.

$$\mathcal{L}_2 = \left\{ \begin{array}{l} (\text{ctx}_{z'}, \text{ctx}_c, w', \text{vk}, \text{ek}, \text{pp}) : \\ \exists r', e', s, e_{kg}, e_t, (u_{r,j}, v_{r,j}, \rho_j, \eta_{j,1}, \eta_{j,2})_{j=1}^\ell \text{ s.t.} \\ w' = A_{\text{sig}} r' + e' \pmod{q}, \\ \|(r', e')\|_2 \leq B_{w'}, \\ A_{\text{sig}} s + e_{kg} = 2^{\nu_t} t + e_t \pmod{q}, \\ \|(s, e_{kg})\|_2 \leq B_t, \quad \|e_t\|_\infty < 2^{\nu_t-1}, \\ \text{ctx}_c = (u_c, v_c), \quad \text{ctx}_{z'} = (\text{ctx}_{z'}^{(1)}, \dots, \text{ctx}_{z'}^{(\ell)}), \\ \text{ctx}_{z'}^{(j)} = (u_{z,j}, v_{z,j}), \\ u_{r,j} = A_{\text{bgv}}^T \rho_j + h \eta_{j,1} \pmod{q}, \\ v_{r,j} = b_{\text{bgv}}^T \rho_j + h \eta_{j,2} + r'_j \pmod{q}, \\ (\rho_j, \eta_{j,1}, \eta_{j,2}) \in \mathcal{B}_{\text{fl}}, \\ u_{z,j} = s_j u_c + u_{r,j} \pmod{q}, \\ v_{z,j} = s_j v_c + v_{r,j} \pmod{q}, \quad \forall j \in [\ell]. \end{array} \right\}.$$

I R1CS Encoding of the Language $\mathbf{L}_{1,2}$

This section describes the R1CS used to encode the arithmetic part of $\mathbf{L}_{1,2}$. The circuit contains four components: the verification of a linking BDLOP commitment, the computation of the digest $\mu = \text{H}(\text{seed}_{\text{vk}}, \text{msg})$, the computation of the challenge hash from $(\mu, \mathbf{w})$, and the verification of the `SampleInBall` procedure; see Appendix G. Throughout this section, n denotes the ring degree and ω denotes the Hamming weight of the challenge vector.

In the circuit, the verification key is represented by a public seed $\text{seed}_{\text{vk}} \in \{0, 1\}^{2\lambda}$, from which the verification key is expanded outside the circuit. The message msg is represented as a vector of field elements. The circuit first checks

$$\mu = \text{Poseidon}(\text{seed}_{\text{vk}}, \text{msg}),$$

where μ is a 2λ -bit digest. It then checks

$$\rho = \text{Poseidon}(\mu, \mathbf{w}), \quad c = \text{SampleInBall}(\rho),$$

where $\mathbf{w}$ is the rounded commitment value and c is the challenge. The value μ is also included in the linking BDLOP commitment, so the arithmetic and lattice statements refer to the same digest.

I.1 R1CS for the BDLOP commitment

We first encode the computation of a BDLOP commitment

$$(\mathbf{t}_A, \mathbf{t}_B) = (\mathbf{A}\gamma, \mathbf{B}\gamma + \mathbf{m}) \in \mathcal{R}_q^{n_A} \times \mathcal{R}_q^{n_B},$$

where $\mathbf{A} \in \mathcal{R}_q^{n_A \times m_A}$ and $\mathbf{B} \in \mathcal{R}_q^{n_B \times m_A}$ are public, $\gamma \in \mathcal{R}_q^{m_A}$ is the commitment randomness, and $\mathbf{m} \in \mathcal{R}_q^{n_B}$ is the committed message. In the language $\mathbf{L}_{1,2}$, this message is

$$\mathbf{m} = (\mathbf{w}, \mu, c).$$

As in Boschini et al. [BCOS20], the commitment equations are checked in the NTT basis. For a vector or matrix x , we write $\hat{x}$ for its NTT representation. The private inputs $\mathbf{m}$ and γ are supplied both in the standard basis and in the NTT basis. The circuit therefore checks the commitment equations in the NTT basis and the consistency between the standard and NTT representations of the private inputs.

Table 7. Protocol notation used in the R1CS encoding of $\mathbf{L}_{1,2}$.

Notation	Meaning
n	Ring degree of $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$.
q	Modulus of the R1CS field and of the ring $\mathcal{R}_q$ used for the arithmetic checked in $\mathbf{L}_{1,2}$.
ℓ_q	Bit length of the modulus, $\ell_q = \lceil \log_2(q) \rceil$.
ω	Hamming weight of the challenge vector output by SampleInBall .
seed_{vk}	Public seed representing the verification key inside the circuit. It has bit length 2λ .
msg	Message representation used in the circuit. In our estimates, msg is represented by 256 field elements.
μ	Private digest computed as $\mu = \text{Poseidon}(\text{seed}_{\text{vk}}, \text{msg})$ and included in the linking commitment.
$\mathbf{A} \in \mathcal{R}_q^{n_{\mathbf{A}} \times m_{\mathbf{A}}}$	Public matrix used in the first component of the linking BDLOP commitment.
$\mathbf{B} \in \mathcal{R}_q^{n_{\mathbf{B}} \times m_{\mathbf{A}}}$	Public matrix used in the second component of the linking BDLOP commitment.
$n_{\mathbf{A}}$	Number of rows of $\mathbf{A}$, hence the number of ring elements in $\mathbf{t}_{\mathbf{A}}$.
$n_{\mathbf{B}}$	Number of rows of $\mathbf{B}$, hence the number of ring elements in $\mathbf{t}_{\mathbf{B}}$ and in the committed message $\mathbf{m}$.
$m_{\mathbf{A}}$	Number of columns of $\mathbf{A}$ and $\mathbf{B}$, hence the number of ring elements in the commitment randomness γ .
$\gamma \in \mathcal{R}_q^{m_{\mathbf{A}}}$	Randomness/opening of the linking BDLOP commitment.
$\mathbf{m} \in \mathcal{R}_q^{n_{\mathbf{B}}}$	Message committed in the linking BDLOP commitment. In our use case, $\mathbf{m} = (\mathbf{w}, \mu, c)$.
$(\mathbf{t}_{\mathbf{A}}, \mathbf{t}_{\mathbf{B}})$	Linking BDLOP commitment, defined as $\mathbf{t}_{\mathbf{A}} = \mathbf{A}\gamma$ and $\mathbf{t}_{\mathbf{B}} = \mathbf{B}\gamma + \mathbf{m}$.
$\widehat{x}$	NTT representation of a ring element, vector, or matrix x .
$u \in [n]$	Index of an NTT coordinate. We use u instead of ω to avoid a collision with the challenge weight.
φ	Primitive root used to define the NTT representation.
sg_i	Sign bit used in the i -th non-zero coefficient of the challenge.
C	Challenge vector constructed by SampleInBall . It has exactly ω non-zero coefficients.

Table 8. Constraint-count notation for the BDLOP commitment and Poseidon hashes in the R1CS encoding of $\mathsf{L}_{1,2}$.

Notation	Description
β_γ	Infinity-norm bound on the coefficients of the commitment randomness γ .
ℓ_γ	Bit length used to range-check γ , defined as $\ell_\gamma = \lceil \log_2(\beta_\gamma) \rceil$.
t_{std}	State size of the Poseidon permutation used for the two hash computations.
r_{std}	Rate of the Poseidon permutation, i.e. the number of absorbed field elements per permutation.
F_{std}	Number of full rounds in the Poseidon permutation.
P_{std}	Number of partial rounds in the Poseidon permutation.
α	S-box exponent of the Poseidon permutation.
ℓ_α	Multiplicative-depth estimate for the S-box $x \mapsto x^\alpha$, defined as $\ell_\alpha = \lceil \log_2(\alpha) \rceil$.
L_{vk}	Number of field elements used to represent seed_{vk} . Since seed_{vk} has bit length 2λ , one may take $L_{\text{vk}} = \lceil 2\lambda/\ell_q \rceil$.
L_{msg}	Number of field elements used to represent the message. In our estimates, $L_{\text{msg}} = 256$.
L_μ	Number of field elements used to represent the digest μ . Since μ has bit length 2λ , one may take $L_\mu = \lceil 2\lambda/\ell_q \rceil$.
L_{w}	Number of field elements used to represent the rounded value $\mathbf{w}$ after the compression induced by ν_w .
L_{dig}	Number of field elements absorbed by the digest hash, $L_{\text{dig}} = L_{\text{vk}} + L_{\text{msg}}$.
L_{chal}	Number of field elements absorbed by the challenge hash, $L_{\text{chal}} = L_\mu + L_{\text{w}}$.
Π_{Poseidon}	Number of R1CS constraints required for one Poseidon permutation.
$N_{\text{BDLOP}}^{\text{R1CS}}$	Number of R1CS constraints used to verify the linking BDLOP commitment.
$N_{\text{Poseidon}}^{\text{R1CS}}(L)$	Number of R1CS constraints used to hash an input of length L field elements with Poseidon.

Table 9. Constraint-count notation for SampleInBall and total R1CS size in the encoding of $\mathsf{L}_{1,2}$.

Notation	Description
t_{XOF}	State size of the Poseidon-XOF permutation used inside SampleInBall .
r_{XOF}	Rate of the Poseidon-XOF permutation.
F_{XOF}	Number of full rounds in the Poseidon-XOF permutation.
P_{XOF}	Number of partial rounds in the Poseidon-XOF permutation.
α'	S-box exponent of the Poseidon-XOF permutation.
$\ell_{\alpha'}$	Multiplicative-depth estimate for the XOF S-box, defined as $\ell_{\alpha'} = \lceil \log_2(\alpha') \rceil$.
N_{XOF}	Number of XOF output elements required by SampleInBall , namely $N_{\text{XOF}} = \omega + \lceil \omega/8 \rceil$.
$N_{\text{SampleInBall}}^{\text{R1CS}}$	Number of R1CS constraints used to verify SampleInBall .
$N_{\mathsf{L}_{1,2}}^{\text{R1CS}}$	Total number of R1CS constraints used for the language $\mathsf{L}_{1,2}$.

For every NTT coordinate $u \in [n]$, the commitment equations become

$$\widehat{\mathbf{t}}_{\mathbf{A}a}[u] = \sum_{j=1}^{m_{\mathbf{A}}} \widehat{\mathbf{A}}_{a,j}[u] \widehat{\gamma}_j[u], \quad a \in [n_{\mathbf{A}}],$$

and

$$\widehat{\mathbf{t}}_{\mathbf{B}b}[u] = \sum_{j=1}^{m_{\mathbf{A}}} \widehat{\mathbf{B}}_{b,j}[u] \widehat{\gamma}_j[u] + \widehat{\mathbf{m}}_b[u], \quad b \in [n_{\mathbf{B}}].$$

Since $\mathbf{A}$ and $\mathbf{B}$ are public, all products above have one public input. Thus each equation can be embedded as one linear R1CS constraint. The $\mathbf{t}_{\mathbf{A}}$ equations contribute $n n_{\mathbf{A}}$ constraints and the $\mathbf{t}_{\mathbf{B}}$ equations contribute $n n_{\mathbf{B}}$ constraints.

We must also ensure that the NTT values supplied for γ and $\mathbf{m}$ are consistent with their standard-basis coefficients. For each polynomial $v = \sum_{j=0}^{n-1} v_j X^j \in \mathcal{R}_q$, the circuit recomputes

$$\widehat{v}[u] = \sum_{j=0}^{n-1} v_j \varphi^{ju}, \quad u \in [n],$$

where φ denotes the chosen primitive root used by the NTT. This costs n constraints per private polynomial. Hence the consistency checks contribute

$$n(m_{\mathbf{A}} + n_{\mathbf{B}})$$

constraints.

Finally, binding of the linking commitment requires the opening randomness γ to be bounded. Let β_{γ} be an infinity-norm bound on the coefficients of γ , and set

$$\ell_{\gamma} = \lceil \log_2(\beta_{\gamma}) \rceil.$$

The circuit enforces the binary decomposition, or in the binary case directly the booleanity, of the coefficients of γ . This contributes

$$n m_{\mathbf{A}}(1 + \ell_{\gamma})$$

constraints.

Therefore, the total number of constraints for the BDLOP commitment is

$$N_{\text{BDLOP}}^{\text{R1CS}} = n(n_{\mathbf{A}} + 2n_{\mathbf{B}} + 2m_{\mathbf{A}} + \ell_{\gamma} m_{\mathbf{A}}). \quad (4)$$

I.2 R1CS for Poseidon

The language $\mathcal{L}_{1,2}$ checks two Poseidon hash computations:

$$\mu = \text{Poseidon}(\text{seed}_{\text{vk}}, \text{msg}), \quad \rho = \text{Poseidon}(\mu, \mathbf{w}).$$

The first hash binds the message representation to the digest μ , while the second hash derives the seed used by `SampleInBall` from the committed digest and the rounded commitment value.

Let

$$L_{\text{dig}} = L_{\text{vk}} + L_{\text{msg}}, \quad L_{\text{chal}} = L_{\mu} + L_{\mathbf{w}}$$

be the corresponding input lengths in field elements. Let t_{std} be the Poseidon state size, r_{std} its rate, F_{std} and P_{std} the numbers of full and partial rounds, and α the S-box exponent. We set

$$\ell_{\alpha} = \lceil \log_2(\alpha) \rceil.$$

With our constraint accounting, each affine layer contributes one constraint per state element, and each S-box $x \mapsto x^\alpha$ contributes ℓ_α multiplicative constraints. Thus one Poseidon permutation costs

$$\Pi_{\text{Poseidon}} = 2t_{\text{std}}(F_{\text{std}} + P_{\text{std}}) + t_{\text{std}}F_{\text{std}}\ell_\alpha + P_{\text{std}}\ell_\alpha.$$

If the input length is L field elements, the number of permutations is $\lceil L/r_{\text{std}} \rceil$. Hence

$$N_{\text{Poseidon}}^{\text{R1CS}}(L) = \left\lceil \frac{L}{r_{\text{std}}} \right\rceil \Pi_{\text{Poseidon}}. \quad (5)$$

For $L_{1,2}$, this formula is applied twice, once with $L = L_{\text{dig}}$ and once with $L = L_{\text{chal}}$.

I.3 R1CS for SampleInBall

We now encode the computation of **SampleInBall**, which derives a sparse challenge vector of Hamming weight ω . The procedure uses **PoseidonXOF** to generate enough field elements to sample both the indices and the signs of the non-zero coefficients.

The signs are packed eight per XOF output element. Therefore, the number of XOF output elements required is

$$N_{\text{XOF}} = \omega + \left\lceil \frac{\omega}{8} \right\rceil.$$

The first ω elements are used for index sampling, while the remaining $\lceil \omega/8 \rceil$ elements provide the sign bits.

Let t_{XOF} be the Poseidon-XOF state size, r_{XOF} its rate, F_{XOF} and P_{XOF} the numbers of full and partial rounds, and α' the XOF S-box exponent. Set

$$\ell_{\alpha'} = \lceil \log_2(\alpha') \rceil.$$

One XOF permutation costs

$$\Pi_{\text{XOF}} = 2t_{\text{XOF}}(F_{\text{XOF}} + P_{\text{XOF}}) + t_{\text{XOF}}F_{\text{XOF}}\ell_{\alpha'} + P_{\text{XOF}}\ell_{\alpha'}.$$

Thus the XOF part contributes

$$\left\lceil \frac{N_{\text{XOF}}}{r_{\text{XOF}}} \right\rceil \Pi_{\text{XOF}}$$

constraints.

The circuit also checks the modular reductions used to extract the indices and the sign bits. Let

$$\ell_q = \lceil \log_2(q) \rceil.$$

For each XOF element, we enforce the corresponding modular decomposition with ℓ_q bit constraints and one quotient constraint, for a total of $N_{\text{XOF}}(\ell_q + 1)$ constraints.

Finally, the circuit checks the swaps performed by **SampleInBall**. For each of the ω iterations, selector variables identify the sampled index. The booleanity of the selectors, the reconstruction of the sampled index, and the state update cost $3n + 3$ constraints per iteration. The sign assignment

$$C_j = 1 - 2\text{sg}_i$$

adds one constraint per non-zero coefficient.

Altogether, the number of constraints for **SampleInBall** is

$$\begin{aligned} N_{\text{SampleInBall}}^{\text{R1CS}} &= \left\lceil \frac{\omega + \lceil \omega/8 \rceil}{r_{\text{XOF}}} \right\rceil \Pi_{\text{XOF}} \\ &\quad + \left(\omega + \left\lceil \frac{\omega}{8} \right\rceil \right) (\ell_q + 1) + \omega(3n + 3) + \omega. \end{aligned} \quad (6)$$

We do not add separate constraints checking that the sampled XOF values fall below the rejection thresholds used by **SampleInBall**. These checks correspond to the rejection behavior of the sampling algorithm: a transcript using rejected values would not reproduce the public challenge vector accepted by the verification equation. Adding such inequalities would only increase the R1CS size by additional range-check constraints.

I.4 Total constraint count for $\mathbf{L}_{1,2}$

The total constraint count estimated for $\mathbf{L}_{1,2}$ is

$$N_{\mathbf{L}_{1,2}}^{\text{R1CS}} = N_{\text{BDLOP}}^{\text{R1CS}} + N_{\text{Poseidon}}^{\text{R1CS}}(L_{\text{dig}}) + N_{\text{Poseidon}}^{\text{R1CS}}(L_{\text{chal}}) + N_{\text{SampleInBall}}^{\text{R1CS}}. \quad (7)$$

J Methodology for the instantiation

Bounding methodology. All bounds used in the proof statements are chosen so that the relations can be checked by the underlying lattice proof system while preserving correctness of the protocol computations. We distinguish two kinds of bounds. First, global ℓ_2 -bounds are used for short module vectors sampled from binary or Gaussian-like distributions, such as signing randomness, blinding randomness, and BDLOP openings. These bounds are used in the LNP statements and in the corresponding SIS/MLWE parameter selection. Second, coefficient-wise ℓ_∞ -bounds are used whenever correctness depends on the absence of modular wrap-around, for example in rounding errors, plaintext decoding, extractable commitments, and BGV decryption.

For a vector of ρ ring elements over $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, binary randomness is bounded by

$$B_{\text{bin}}(\rho) = \sqrt{n\rho},$$

while Gaussian randomness with width σ is bounded by

$$B_{\text{gauss}}(\rho, \sigma) = e^{1/4} \sigma \sqrt{n\rho}.$$

When a binding argument compares two valid openings, the relevant SIS witness is their difference. Thus, for Gaussian openings we use the conservative bound $2B_{\text{gauss}}$, whereas for binary openings we use the binary bound implemented in our parameter search.

The user computes

$$\mu = H(\text{seed}_{\text{vk}}, \text{msg}),$$

where seed_{vk} is a public 2λ -bit seed representing the verification key inside the circuit, and μ is a 2λ -bit digest. In the first-round extractable commitment, the user commits to

$$m_0 = (\text{msg}, \mu, r, e),$$

while the linking commitment used between the lattice and arithmetic proofs commits to

$$m_1 = (\mathbf{w}, \mu, c).$$

The arithmetic proof checks the digest computation and the challenge generation:

$$\mu = \text{Poseidon}(\text{seed}_{\text{vk}}, \text{msg}), \quad \rho = \text{Poseidon}(\mu, \mathbf{w}), \quad c = \text{SampleInBall}(\rho).$$

Thus, the R1CS contains two Poseidon hashes: one binding the message and the verification-key seed to μ , and one deriving the challenge from $(\mu, \mathbf{w})$. The binding property of the two BDLOP commitments ensures that the same digest μ is used in the lattice and arithmetic statements.

Rounding relations are rewritten linearly by introducing explicit rounding-error variables. For example,

$$\mathbf{w} = \lfloor \mathbf{w}' + A_{\text{sig}} r + e \rfloor_{\nu_{\mathbf{w}}}$$

is represented as

$$\mathbf{w}' + A_{\text{sig}} r + e = 2^{\nu_{\mathbf{w}}} \mathbf{w} + e_{\mathbf{w}} \pmod{q}, \quad \|e_{\mathbf{w}}\|_\infty < 2^{\nu_{\mathbf{w}}-1}.$$

The public-key rounding relation is handled analogously:

$$A_{\text{sig}} s + e_{\text{kg}} = 2^{\nu_{\mathbf{t}}} \mathbf{t} + e_{\mathbf{t}} \pmod{q}, \quad \|e_{\mathbf{t}}\|_\infty < 2^{\nu_{\mathbf{t}}-1}.$$

This makes the statements linear over the ring, which is the form required by LNP.

For the BGV component, the plaintext bound is derived from the value recovered after decryption. In our protocol this value is

$$z' = c \cdot s + r',$$

so we use

$$B_{\text{plain}} = B_c B_s + B_{r'},$$

where $B_c = \|c\|_1 = \omega$, B_s bounds the signing-key component, and $B_{r'}$ bounds the signer randomness. The plaintext scale h is then chosen so that

$$h > 2B_{\text{plain}},$$

ensuring that the plaintext does not wrap modulo h . The ciphertext modulus is chosen so that the BGV decryption residual does not wrap:

$$B_{\text{plain}} + hB_{\text{final}} < q/2,$$

where B_{final} includes the evaluated encryption noise and the flooding noise.

In earlier formulations, different relations lived over different moduli: the Raccoon relations modulo q , the BGV ciphertexts modulo a ciphertext modulus Q , and the extractable BDLOP commitment potentially modulo another modulus. This would require lifting equations into a common proof modulus. Concretely, a relation $F = 0 \pmod{q}$ proved modulo Q would have to be rewritten as

$$F - q\kappa = 0 \pmod{Q},$$

with an explicit quotient witness κ and a no-wrap condition ensuring that the lifted equality represents the intended integer equality. These quotient witnesses increase both the witness size and the proof size.

To avoid this overhead, we choose a common modulus q large enough for all linear lattice relations used in the concrete proofs. The same modulus is used for the Raccoon linear relations, the BGV ciphertext ring, and the extractable BDLOP commitment. The parameter search enforces the correctness conditions directly:

$$B_{\text{plain}} + hB_{\text{final}} < q/2$$

for BGV decryption, and

$$B_{\text{msg}} + h_{\text{ext}}B_{E\gamma} < q/2$$

for extraction from the simulated BDLOP setup. Thus, neither the BGV residual nor the extractable BDLOP extraction equation wraps around modulo q . With this choice, all lattice-linear equations are checked over the same ring R_q , and no lifting quotient κ is needed in the proof statements.

Table 10. Bounds used in the lattice relations and BDLOP commitments.

Symbol	Meaning	Bound / condition
B_t	Bound for the signing-key opening (s, c_{kg})	$B_t = e^{1/4} \sigma_t \sqrt{n(k+\ell)}$
$B_{w'}$	Bound for the signer first-round randomness (r', e')	$B_{w'} = e^{1/4} \sigma_{w'} \sqrt{n(k+\ell)}$
B_w	Bound for the user blinding randomness (r, e)	$B_w = e^{1/4} \sigma_w \sqrt{n(k+\ell)}$
B_{e_w}	Infinity bound for the commitment-rounding error	$\ e_w\ _\infty < 2^{\nu_w-1}$
B_{e_t}	Infinity bound for the public-key rounding error	$\ e_t\ _\infty < 2^{\nu_t-1}$
B_2^*	Final Raccoon verification bound	$\ (z, 2^{\nu_w} h)\ _2 \leq B_2^*$
$B_{\gamma,i}$	BDLOP opening bound for commitment i	Binary: $\sqrt{n\rho_i}$, Gaussian: $e^{1/4} \sigma_\gamma \sqrt{n\rho_i}$
$B_{\Delta_{\gamma,i}}$	BDLOP binding/MSIS difference bound	Binary: $B_{\gamma,i}$, Gaussian: $2B_{\gamma,i}$
B_{msg}	Coefficient bound for the external message representation	In our R1CS estimate, msg is represented by 256 field elements
B_μ	Coefficient bound for the digest μ	μ is a 2λ -bit string encoded over the proof field
B_{msg}	Coefficient bound for the extractable round-1 committed message	For $m_0 = (\text{msg}, \mu, r, e)$, conservatively $B_{\text{msg}} = \max\{B_{\text{msg}}, B_\mu, B_w\} = B_w$
B_{E_γ}	Extraction noise in the simulated BDLOP setup	$B_{E_\gamma} = B_{E_\gamma,0}^{\text{row}}$
h_{ext}	Plaintext scale for extractable BDLOP	$h_{\text{ext}} > 2B_{\text{msg}}$
q	Common modulus for the lattice statements	$B_{\text{msg}} + h_{\text{ext}} B_{E_\gamma} < q/2$

Table 11. Bounds used for the MLWE-BGV encryption relations.

Symbol	Meaning	Bound / condition
B_c	Challenge bound	$B_c = \omega$, where $\omega = \ c\ _1$
B_s	Plaintext-side signing-key bound	Usually $B_s = B_t$
$B_{r'}$	Plaintext-side signer randomness bound	Usually $B_{r'} = B_{w'}$
B_{plain}	Bound on decrypted plaintext $c \cdot s + r'$	$B_{\text{plain}} = B_c B_s + B_{r'}$
h	BGV plaintext scale	$h > 2B_{\text{plain}}$
B_{enc}	Fresh BGV encryption noise	$B_{\text{enc}} = B_{e_{\text{pk}}} B_\rho + B_{\eta_2} + B_{s_{\text{he}}} B_{\eta_1}$
B_{eval}	Noise after evaluating $f_s(C, M) = s \odot C + M$	$B_{\text{eval}} = B_s B_{\text{in}} + B_{\text{add}}$
B_{fl}	Noise-flooding bound	Chosen so that flooding statistically hides evaluation noise
B_{final}	Final BGV decryption-noise bound	$B_{\text{final}} = B_{\text{eval}} + B_{\text{fl}}$
q	Common BGV ciphertext modulus	$B_{\text{plain}} + h B_{\text{final}} < q/2$