Lattice-Based Rate-Limited Tokens in the Random Oracle Model

Victor Youdom Kemmoe [†], Anna Lysyanskaya [†], Ngoc Khanh Nguyen [‡]

[†]Brown University, US
{vyoudomk, anna}@cs.brown.edu
[‡]King's College London, UK
ngoc_khanh.nguyen@kcl.ac.uk

Abstract

An anonymous credential allows a user to prove that she is authorized in an anonymous and unlinkable fashion. A rate-limited token is an anonymous credential that can only be used a limited number of times in any particular context; this means that even though we do not know which users are gaining access to a resource, there is a limit to how many resources one user may consume. Such tokens are becoming an increasingly attractive way to balance privacy with authorized access. Although a general architecture for how to obtain rate-limited tokens from digital signatures, pseudorandom functions (PRFs), and non-interactive zero-knowledge proofs (NIZKs) has been known for over twenty years, efficiently instantiating it with post-quantum-secure signatures and proofs has, until now, remained an open problem.

In this work, we present the first lattice-based construction of rate-limited tokens and tackle the practical challenges associated with using lattice-based building blocks in this setting. A central difficulty lies in the absence of lattice-based PRFs that support efficient NIZK proofs of correct evaluation. We show that, in the random oracle model, a weak PRF—where adversaries are restricted to querying random inputs—suffices. We further present a weak PRF construction that both admits efficient NIZK proofs and remains secure—even when adversaries have partial control over the randomness—and extend this guarantee more generally to key-homomorphic PRFs.

Another contribution, which is of independent interest, is the first lattice-based construction of partially binding commitments, a primitive introduced by Goel et al. (Eurocrypt 2022) that was previously known only under discrete-log assumptions. We give a practical construction that enables succinct disjunctive proofs via a variant of the self-stacking compiler of Goel et al. Along the way, we develop a new technique for batching CNF proofs of Σ-protocols, which allows one to efficiently prove that a value is the output of a PRF on one of a set of inputs. As a direct application, this yields logarithmic-size lattice-based ring signatures based on Fiat–Shamir-with-Aborts Σ-protocols (Lyubashevsky, Eurocrypt 2012).

Finally, we observe for the first time that the anonymous counting tokens of Benhamouda, Raykova, and Seth (Asiacrypt 2023) can be obtained from anonymous rate-limited tokens. This yields a construction whose communication complexity is independent of the number of tokens that need to be issued.

1 Introduction

Privacy-preserving authentication tokens are becoming increasingly widespread; for example, they are used in iCloud Private Relay [TW24], Google's Private State Tokens [KLOR20], and Cloudflare's Privacy Pass [DGS$^+$18].

Rate-limited tokens [CL04, CHK$^+$06, CDLT25a] allow a user to obtain an entire "dispenser" of tokens in one transaction with the issuer, and then use them, unlinkably, one by one, up to N per any given context.

For example, a dispenser may be a newspaper subscription that allows a user to read up to N articles a day (in which case the context is today's date); or a subscription to a family of magazines that allows up to N articles from any publication per day (in which case the context consists of the name of the publication as well as today's date).

Camenisch, Hohenberger and Lysyanskaya (CHL) [CHL05] introduced the following approach for constructing rate-limited tokens. In order for a user to obtain a dispenser, the user and the issuer engage in a secure two-party protocol as a result of which the user receives the issuer's signature σ on the values (pk_U, k), where pk_U is the user's public key, while k is a random seed for a pseudorandom function.

In order to compute the ith token tk for a given context ctxt, the user and the token's recipient first agree on some nonce value that is unique for a given transaction. Next, the user computes the token's serial number $\mathsf{sn} = F_k(0, i, \mathsf{ctxt}) \bmod p$ and tag $\mathsf{tg} = \mathsf{pk}_U + \mathsf{nonce} \cdot F_k(1, i, \mathsf{ctxt}) \bmod p$ (where p is a prime that is a system parameter), as well as a zero-knowledge proof π that (0) sn and tg were computed correctly based on the values (pk_U, k) and i that are known to the user such that (1) the user knows the secret key sk_U corresponding to pk_U; (2) $1 \leq i \leq N$; and (3) the user knows the issuer's signature σ on (sk_U, k). The token is $\mathsf{tk} = (\mathsf{sn}, \mathsf{tg}, \pi)$. Its verification consists of the verification of the proof π; the system also requires that all tokens that have been redeemed so far be stored in an archive that is used to ensure that a token cannot be used more than once.

The reason that this scheme does not allow a user to spend tokens at a rate that's higher than N per context (in the sequel, we call this property of a token scheme *soundness*) is that the soundness of the proof system ensures that there exist only N distinct serial numbers per dispenser per context. Moreover, if two tokens from the same dispenser have the same serial number, then the two tags tg_1 and tg_2 allow one to recover the public key of the user to whom the dispenser was issued: one just needs to solve a system of two linear equations with two unknowns. Specifically, one can recover pk_U by computing $\mathsf{tg}_1 - \frac{\mathsf{nonce}_1}{\mathsf{nonce}_1 - \mathsf{nonce}_2}(\mathsf{tg}_1 - \mathsf{tg}_2)$.

The fact that the key k here is random as long as either the user or the issuer is honest ensures that tokens from different dispensers will never (except with negligible probability) have the same serial number. And the fact that a proof of knowledge of sk_U is part of the token ensures that, whenever the user associated with pk_U is honest and did not in fact double-spend a token, pk_U cannot be recovered from the double-spent token (this property is called "exculpability").

Although this general architecture is relatively simple, instantiating it in a way that yields a practical scheme is challenging. With a general-purpose zero-knowledge proof system such as Ligero [AHIV17], the proof size would be prohibitive. There is a rich literature on adapting it towards an instantiation with discrete-logarithm-based signatures and proof systems [CHL05, CHK$^+$06, BCKL09, CDLT25a] but those constructions do not provide post-quantum security; at best, Chairattana-Apirom et al. [CDLT25a] give a construction in which the user's privacy is unconditional (using a large truly random object instead of a PRF key k) while unforgeability of the tokens is based on a discrete-log-based assumption.

In this paper, we adapt this architecture to obtain a practical lattice-based construction. We use the BLNS [BLNS23a] signature that lends itself to lattice-based efficient protocols, and overcome the following challenges:

Challenge 1: a lattice-based PRF with an efficient NIZK. The CHL architecture requires an efficient proof of knowledge that a given value is the output of a pseudorandom function. Current efficient lattice-based PRFs that are secure in the standard model [BP14, BBL$^+$14, Kim20] follow a common structure: $F_{\mathbf{a},\mathbf{B},p}(x) = \left\lfloor \mathbf{a}^\top g(\mathbf{B}, x) \right\rceil_p$, where either $\mathbf{a}$ is public and $\mathbf{B}$ is secret, or vice versa, $\lfloor \cdot \rceil_p$ is a rounding function parametrize by an integer p, and g is a function that returns a matrix based on $\mathbf{B}$ and the input x. For instance, g may return a linear combination of columns of $\mathbf{B}$ with coefficients defined by x, or compute a chain of matrix multiplication followed by binary decomposition. However, proving a correct execution

of $F_{\mathbf{a},\mathbf{B},p}$ on a committed input x requires proving the correct execution of g, which induces large arithmetic circuits. As a result, when using the state-of-the-art lattice-based ZK framework of Lyubashevsky, Nguyen, and Plançon [LNP22], this leads to proof sizes on the order of megabytes (at the very least 3MB based on Albrecht and Gür asymptotics [AG24], see Section 7.5), unless one employs a succinct proof system, such as LaBRADOR [BS23], in conjunction to reduce the proof size at the cost of longer proving time.

In order to overcome this challenge, we note that for our application, a weak PRF — i.e. one that remains secure when the adversary can query it only on random inputs, rather than on inputs that are adversarially and adaptively chosen – can be sufficient. Instead of computing $\mathsf{sn} = F_k(0,i)$, we may have let $\mathsf{sn} = F_k(H(0,i))$, where H is a hash function that will be modeled as a random oracle in the proof of security. Obtaining a weak lattice-based PRF is a lot easier than a true PRF: Boneh et al. [BLMR13] showed that, under appropriate assumptions and methods for sampling a key $\mathbf{k}$, $F^*_{\mathbf{k}}(\mathbf{M}) = \lfloor \mathbf{M} \cdot \mathbf{k} \rceil_p$ is a weak PRF, where $\mathbf{M}$ is an $n \times m$ matrix with entries from $\mathbb{Z}_q$ for a large $q > p$, the key $\mathbf{k}$ is a vector where each entry is an integer whose absolute value is at most B, and for $a \in \mathbb{Z}_q$, $\lfloor a \rceil_p$ denotes rounding a to $\lfloor (p/q)a \rfloor$, while for $\mathbf{a} \in \mathbb{Z}_q^m$, this rounding operation is component-wise, i.e., $\lfloor \mathbf{a} \rceil_p = (\lfloor (p/q)a_1 \rfloor, \ldots, \lfloor (p/q)a_m \rfloor)$. The LNP [LNP22] proof system gives us a simple and short NIZK proof of knowledge of the key $\mathbf{k}$ such that $y = F^*_{\mathbf{k}}(\mathbf{M})$ for $\mathbf{M}$ known to the verifier. This is presented in Section 7.2.

Of course, which $\mathbf{M}$ is input to the PRF should not be known to the verifier; instead, in our construction, the verifier knows that $\mathbf{M}$ is in the set $L = \{H(0,1), \ldots, H(0,N)\}$. To prove that $\mathsf{sn} = F^*_{\mathbf{k}}(H(0,i))$ and $1 \le i \le N$, it is sufficient to provide a proof that the input to F is in L. In order to do that efficiently, we show a lattice-based batched OR proof (more on that below).

Challenge 2: ensuring that the adversary's PRF is collision-resistant. The CHL architecture requires that the PRF key k be truly random as long as either the user or the issuer is honest. This ensures that tokens from different dispensers should not have the same serial number. In the discrete-log-based setting, this is easy to achieve: the user can commit to a pre-key $k' \in \mathbb{Z}_q$, the issuer sends $k'' \leftarrow \mathbb{Z}_q$, and the final key is $k = k' + k''$. In the lattice-based setting it is not immediately clear how this approach could work, as adding two keys for the F^* PRF described above does not necessarily produce a valid PRF key.

In order to address this challenge, we first observe that F^* is key-homomorphic (other than for a small error that we ignore); i.e. $F^*_{\mathbf{k}_1+\mathbf{k}_2}(\mathbf{M}) = F^*_{\mathbf{k}_1}(\mathbf{M}) + F^*_{\mathbf{k}_2}(\mathbf{M})$. (Note that even if the key $\mathbf{k}_1 + \mathbf{k}_2$ is not B-bounded, $F^*_{\mathbf{k}_1+\mathbf{k}_2}(\mathbf{M})$ is still well-defined).

Next, we prove (Lemma 3.1) that for any key-homomorphic PRF F with a polynomial-size domain and a super-polynomial range, no two tokens from two different dispensers will have colliding serial numbers as long as either the user or the issuer is honest (we call this property weak collision resistance). More precisely, consider the experiment in which a malicious user j picks the key $\mathbf{k}_1^j$ and the honest issuer picks $\mathbf{k}_2^j$ (and reveals it to the adversary), and serial numbers of the user's tokens are computed as $\mathsf{sn}_{j,i} = F_{\mathbf{k}_1+\mathbf{k}_2}(H(0,i))$. Then with overwhelming probability, the adversary will never cause two users $j_1 \neq j_2$ to have tokens with the same serial number $\mathsf{sn}_{j_1,i_1} = F_{\mathbf{k}_1^{j_1}+\mathbf{k}_2^{j_1}}(H(0,i_1) = F_{\mathbf{k}_1^{j_2}+\mathbf{k}_2^{j_2}}(H(0,i_2)) = \mathsf{sn}_{j_2,i_2}$.

Challenge 3: ensuring exculpability. Anonymous rate-limited tokens (and similarly e-cash [CHL05]) must satisfy a property called *exculpability*, which requires that an adversary controlling an issuer and a set of users cannot frame an honest user for double-spending one of its tokens. Following the CHL architecture described above, the double spender's identity is recovered by computing

$$\mathsf{id}^* = G(\mathsf{sk}^*) = \mathsf{tg}_1 - \frac{\mathsf{nonce}_1}{\mathsf{nonce}_1 - \mathsf{nonce}_2}(\mathsf{tg}_1 - \mathsf{tg}_2) \tag{1.1}$$

where G is some public function and sk^* is the secret identity.

However, Bourse, Pointcheval, and Sanders [BPS19, Section 1.2] observed that id^* might not match the double spender's actual identity, even if tk_1 and tk_2 are well-formed. This issue stems from the fact that

sn_i and tg_i are typically derived from PRFs and scalars fully controlled by an adversary. In the discrete-logarithm setting, this can be resolved by extracting all secret information that went into computing sn_i and tg_i from π_i and converting equation (1.1) into a discrete logarithm solution. In the lattice setting, however, applying the same methodology will lead to a short integer solution (SIS) of very large norm, even if exact proofs are used. To address this issue, Bourse, Pointcheval, and Sanders assumed a strong collision-resistance property from PRFs: given a PRF F, it should be *hard* for an efficient adversary to output distinct key-input pairs $(k_i, x_i)_{i\in\{1,2\}}$ such that $F_{k_1}(x_1) = F_{k_2}(x_2)$. Subsequently, Deo et al. [DLNS20] addressed this issue by instead relying on a lossy trapdoor function (LTDF) [PW11] to extend the serial number of a token such that in case of double-spending, one can use the trapdoor of the LTDF to exactly recover the double spender's identity. Beyond increasing the token size, this approach may also enlarge the proofs appended to tokens, since one must prove the correct evaluation of the LTDF to establish token well-formedness.

Here we propose two solutions that do not require any extra assumptions on PRFs or extra primitives: the first solution is general and only requires a random oracle, while the second is tailored to lattice-based constructions. More specifically, in the first solution, for a token tk_i, we compute its nonce as $\mathsf{nonce}_i \leftarrow \mathsf{H}(\mathsf{sn}_i, \pi_{i|\mathsf{sn}_i})$, where $\pi_{i|\mathsf{sn}_i}$ is the proof of well-formedness of sn_i and H is a random oracle, and its tag as $\mathsf{tg}_i = G(\tilde{\mathsf{sk}}) + \mathsf{nonce}_i \cdot F_{k_i}(j)$, where k_i is a PRF key and j is a counter that was included in the computation of $\pi_{i|\mathsf{sn}_i}$. Under this formulation, for equation (1.1) to output an identity different from the double spender's identity, we must have, without loss of generality, $\mathsf{nonce}_2 \cdot \mathsf{a} = \mathsf{b}$, where $\mathsf{a} = (\mathsf{tg}_1 - \mathsf{id}^* - \mathsf{nonce}_1 \cdot F_{k_2}(j))$ and $\mathsf{b} = \mathsf{nonce}_1 \cdot (G(\tilde{\mathsf{sk}}) - \mathsf{id}^*)$ are fixed values. Finally, assuming that nonces are invertible (which is not hard to satisfy in both the discrete-log-based and lattice-based settings), we have that a and b are non-zero. Indeed, if $\mathsf{b} = 0$, then $G(\tilde{\mathsf{sk}}) = \mathsf{id}^*$, which contradicts the fact that we were not able to recover the double spender's identity. Moreover, conditioned on $\mathsf{b} \neq 0$, we have $\mathsf{a} \neq 0$. Therefore, querying H to find nonce_2 such that $\mathsf{nonce}_2 \cdot \mathsf{a} = \mathsf{b}$ amounts to sampling nonce_2 uniformly at random from the nonce space and hoping that it lands in the set of feasible solutions. Under appropriate assumptions (see Definition 2.1), this occurs with negligible probability. However, this approach (presented as part of our general construction in Section 6) requires outputting a separate proof of well-formedness for sn_i and tg_i.

For the second solution, which is tailored to lattice-based constructions (and is presented in the section on lattice-based instantiation, Section 7), we adopt an approach inspired by Chairattan-Apirom, Döttling, Lysyanskaya, and Tessaro [CDLT25a] by setting G (used in the computation of tg) to be an easily invertible function. In their work, G was set to be the identity function. In contrast, we set G to be the gadget matrix $\mathbf{G}$ [MP12]. Now, given id^* recovered from equation (1.1), we compute $\mathbf{v} = \mathbf{G}^{-1}(\mathsf{id}^*)$ and check if $\mathbf{Av} = \mathbf{t}_{\mathsf{id}^*}$, where $\mathbf{A}$ is a public matrix, and $\mathbf{t}_{\mathsf{id}^*}$ is a vector associated to id^*. Under this formulation, any adversary that successfully frame an honest user for double-spending can be turned into a solver for the (module) Short Integer Solution problem. By instantiating this construction, with a rate-limit of 256, we obtain tokens of size 899KB (see Section 7.5).

1.1 Partially binding vector commitments and batched OR-proofs

Goel et al. [GGHK22], and Avitabile et al. [ABFV22] introduced the partially binding vector commitment primitive. Such a commitment allows one to commit to a vector in such a way that this commitment can be equivocated in every position but one. This is a convenient building block for obtaining a succinct proof of knowledge of one of N witnesses [GGHK22] via the so-called self-stacking compiler for Σ-protocols. The (very high-level) idea is that the random-oracle based NIZK proof of knowledge of one of the witnesses $(w_1, \ldots, w_N)$ for statements $(x_1, \ldots, x_N)$ will implicitly consist of N transcripts (a_i, c_i, z_i) of a the Σ-protocol, one per statement. Yet these transcripts will be highly related, so that the challenge message is identical (i.e. $c_i = c\ \forall i$), the last messages of all N transcripts are identical (i.e. $z_i = z\ \forall i$), and the first message a_i of each transcript i can be computed from the instance, the challenge message, and the final

message (i.e. there is some function f such that $a_i = f(x_i, z, c)$). Then, to prove knowledge of one of the witnesses w_{i^*} using the partially binding commitment, the prover first computes the partially binding commitment cmt to some $(a'_1, \ldots, a'_N)$ such that it is only bound to the value $a'_{i^*} = a_{i^*}$ in position i^* that is computed correctly according to the Σ-protocol for the statement x_{i^*} and witness w_{i^*}. Next, the prover derives the challenge c via a call to the random oracle, and then computes the response $z = z_{i^*}$ using the Σ-protocol for the statement x_{i^*} and witness w_{i^*}. For $i \neq i^*$, it computes $a_i = f(x_i, z, c)$, and then derives a way to open C to $(a_1, \ldots, a_N)$. We will be using this approach to prove that the value sn was computed via a PRF on input a value from a list L (as explained above).

Several subtleties arise when adapting this approach, which we do in Section 5. The self-stacking compiler was introduced in the discrete-log-based setting where Σ-protocols have perfect correctness. In the lattice-based setting, in contrast, there is a small but non-negligible probability of correctness error; thus, the compiler needs to be re-analyzed to take that into consideration. Moreover, we give a construction of partially binding vector commitments that also admit a small but non-negligible correctness error. We show that they still work and allow us to obtain an efficient proof system for a disjunction using our version of the self-stacking compiler.

To give a better overview of how partially binding vector commitment works, let us consider the case of 1-out-of-2. Goel et al [GGHK22] proposed a construction based on Pedersen Commitment [Ped91], which works as follow: Let $\mathsf{crs} := (g, h) \leftarrow\!\$\ \mathbb{G}$, where $\mathbb{G}$ is a group of prime order p in which the discrete-log assumption holds. Now, given crs, sample $t \leftarrow\!\$\ \mathbb{Z}_p$, set $g_0 \leftarrow g^t$ and $g_1 \leftarrow h \cdot g_0^{-1}$. To commit to $(v_0, v_1) \in \mathbb{Z}_p^2$, compute $\mathsf{cmt} \leftarrow (g^{r_0} g_0^{v_0}, g^{r_1} g_1^{v_1})$, where $r_0, r_1 \leftarrow\!\$\ \mathbb{Z}_p^2$. The trapdoor t allows the committer to equivocate at position 0. Namely, for any desired value v'_o, the committer can open the position 0 to v'_0 by setting $r'_0 = r_0 + t(v_0 - v'_0)$ as part of its opening. However, if the committer also successfully equivocate at position 1, then given cmt and the distinct openings $((r_0, v_0), (r_1, v_1)), ((r'_0, v'_0), (r'_1, v'_1))$, such that $v_0 \neq v'_0$ and $v_1 \neq v'_1$, then from $((r_0, v_0), (r'_0, v'_0))$, we can recover the $\mathsf{dlog}_g g_0$ and from $((r_1, v_1), (r'_1, v'_1))$, we can recover $\mathsf{dlog}_g g_1$, but since $g_1 = h \cdot g_0^{-1}$, we can use $\mathsf{dlog}_g g_0$ and $\mathsf{dlog}_g g_1$ to compute $\mathsf{dlog}_g h$, thereby breaking the discrete log assumption. In the lattice-based setting, things are not so seamless.

One might try to reproduce the same machinery using Ajtai commitments [Ajt96]. Namely, set $\mathsf{crs} := (\mathbf{A}, \mathbf{B})$, compute $\mathbf{A}_0 = \mathbf{A} \cdot \mathbf{T}$, where $\mathbf{T}$ is a low-norm matrix random matrix, and $\mathbf{A}_1 = \mathbf{B} - \mathbf{A}_0$. As in the discrete-log based setting, one could then commit to $\mathbf{m}_0$ and $\mathbf{m}_1$ by computing $\mathsf{cmt} = (\mathbf{A}\mathbf{r}_0 + \mathbf{A}_0\mathbf{m}_0, \mathbf{A}\mathbf{r}_1 + \mathbf{A}_1\mathbf{r}_1)$. However, equivocating at position 0 to another message $\mathbf{m}'_0$ by returning $\mathbf{r}'_0 = \mathbf{r}_0 + \mathbf{T} \cdot (\mathbf{m}_0 - \mathbf{m}'_0)$ does not work directly. The distribution of $\mathbf{r}'_0$ differs from that of an honestly sampled opening, because of the shift by $\mathbf{T}(\mathbf{m}_0 - \mathbf{m}'_0)$. This distributional difference would prevent the composition with Σ-protocols and self-stacking compiler as described above, since the verifier could detect the position at which the prover equivocated.

We address this issue with Lyubashevsky's rejection-sampling technique [Lyu12]. However, using rejection sampling introduces the possibility of abort, which requires additional care. A second difficulty is that equivocation at both positions does not immediately allow us to obtain a solution to the Short Integer Solution problem (SIS). Indeed, given a commitment cmt and message-opening pairs that equivocate it at the both positions $((\mathbf{r}_0, \mathbf{m}_0), (\mathbf{r}_1, \mathbf{m}_1))$ and $((\mathbf{r}'_0, \mathbf{m}'_0), (\mathbf{r}'_1, \mathbf{m}'_1))$, one obtains $\mathbf{A}_0(\mathbf{r}_0 - \mathbf{r}'_0) + \mathbf{A}(\mathbf{m}_0 - \mathbf{m}'_0) = \mathbf{0}$ and $\mathbf{A}_1(\mathbf{r}_1 - \mathbf{r}'_1) + \mathbf{A}(\mathbf{m}_1 - \mathbf{m}'_1) = \mathbf{0}$. However, this does not directly give a useful SIS solution for the matrix $[\mathbf{A} \mid \mathbf{B}]$ since from the point of view of a reduction $\mathbf{A}_1$ could have the matrix that was trapdoored, unlike in the discrete-log setting, we do not recover a trapdoor for neither $\mathbf{A}_0$ nor $\mathbf{A}_1$. To solve this issue, we require the committer to include an OR-proof of knowledge of a low-norm matrix $\mathbf{T}$ such that $\mathbf{A} \cdot \mathbf{T} = \mathbf{A}_0 \vee \mathbf{A} \cdot \mathbf{T} = \mathbf{A}_1$. Now, if the committer can equivocate at both positions, we can rewind it and extract $\tilde{\mathbf{T}}$ such that $\mathbf{A} \cdot \tilde{\mathbf{T}} = \mathbf{A}_0$. For simplicity, we omit here the slack introduced by the extraction procedure. Using $\mathbf{A}_1(\mathbf{r}_1 - \mathbf{r}'_1) + \mathbf{A}(\mathbf{m}_1 - \mathbf{m}'_1) = \mathbf{0}$ and $\tilde{\mathbf{T}}$, we can now obtain a SIS solution for the matrix $[\mathbf{A} \mid \mathbf{B}]$.

Although the OR-proof allows us to extract an SIS solution, it also increases the size of the commitment key. This overhead becomes especially significant when the 1-out-of-2 construction is used to bootstrap a 1-out-of-2^k commitment, in a manner analogous to a Merkle tree. At a high level, the committer generates a pair of commitment-key matrices $(\mathbf{A}_{0,j}, \mathbf{A}_{1,j})$ for each $j \in [k]$. The index at which the committer is bound is represented by a string $s \in \{0,1\}^k$. For each level j, if s_j denotes the j-th bit of s, then the matrix corresponding to the opposite branch, namely $\mathbf{A}_{1-s_j,j}$ is generated with a trapdoor. However, this means that, for each level of tree, the committer must provide an OR-proof for $\mathbf{A} \cdot \mathbf{T}_k = \mathbf{A}_{0,k} \vee \mathbf{A} \cdot \mathbf{T}_k = \mathbf{A}_{1,k}$, which increases the commitment key by a factor of $2k$. To overcome this issue, we note that those statements can be viewed as a Conjunctive Normal Form (CNF) formula of the form $\wedge_{i\in[k]} (\mathbf{A} \cdot \mathbf{T}_k = \mathbf{A}_{0,k} \vee \mathbf{A} \cdot \mathbf{T}_k = \mathbf{A}_{1,k})$. Although prior work gives short proofs (logarithmic in the number of clauses) proofs for larger classes of CNF of Σ-protocols [ZLH+22, ABF+25], here by exploiting the structure of our specific CNF, most notably, the fact that the matrix $\mathbf{A}$ is shared across all clauses, we propose a batched Σ-protocol whose proof is practically constant-size, i.e., proving k clauses produce proof size that is roughly the same as that of a proof for a single clause.

As a bonus, we show in Section 5.3 that our partially binding vector commitment allows us to easily instantiate ring signatures from Lyubashevsky's identification protocol [Lyu12], similarly to how Goel et al. [GGHK22] obtained a ring signature from Schnorr's identification protocol using their discrete-log-based partially binding vector commitment. Although we do not expect the resulting ring signature to be as efficient as more tailored schemes [BKP20], the main advantage of our construction is that it can be used without changing the underlying signature scheme or system parameters.

1.2 Anonymous Counting Tokens

Benhamouda, Raykova, and Seth [BRS23] proposed a primitive called Anonymous Counting Token (ACT), which are anonymous tokens that enforce rate-limiting at the issuance level. Specifically, with ACTs, a user cannot obtain more than n tokens for a specific message (context) from the same issuer. This differs from anonymous rate-limited tokens, where a user can replenish its dispensers for a given context by sending a new request to an issuer. Their proposed construction followed the templates of a blind signature, with the following twist: (1) during registration, users commit to their PRF keys and send the commitments to the issuer, (2) during a blind-signature request, a user sends a blinded message, an evaluation of the PRF under its committed key on the blinded message, and a zero-knowledge proof of correct evaluation, (3) upon receiving the blinded message, the PRF evaluation, and the proof, the issuer uses the user's committed key to verify the correctness of the proof before issuing a blind signature. By the determinism of the PRF, the issuer will reject future queries that carry the same PRF outputs, hence achieving this notion of rate-limiting at issuance level. However, the main drawback of this construction is that, if the rate limit is $n > 1$, then the user must interact with the issuer n times, and during those interactions, it must prove that it has attached a committed counter $j \leq n$, so that the issuer can detect it if the user tries to request a token more than $n + 1$ times.

Here, we observe that by making the issuer of an anonymous rate-limited token stateful, we can obtain an ACT whose communication complexity is independent of the rate. Specifically, instead of a user committing to an identity, the user commits to a PRF key k_U. Then, during issuance, the user commits to the message/context m for which it wants to obtain tokens and a new PRF k_m. In addition, similarly to the approach of Benhamouda, Raykova, and Seth, the user produces a PRF evaluation of m under k_U, together with a proof of correct evaluation. Now, upon receiving the commitment to (m, k_m), the PRF output, and the proof of correct evaluation, the issuer uses the commited key k_U to check the validity of the proof and then issue a signature of (m, k_m). The user can then use k_m to derive all the tokens related to m that it needs.

Scheme	Round	Communication	Rate limit	Verification	Assumptions
BRS [BRS23]	2	$\mathfrak{n}\cdot\mathsf{poly}(\lambda)$	Issuance	Private	DCR, q-DDHI
			Redemption	Public	SXDH, q-DDHI
Our work (Sec. 8)	2	$\mathsf{poly}(\lambda)$	Issuance	Public	M-LWE, M-SIS

Table 1: Anonymous Counting Tokens. $\mathfrak{n}$ is the number of allowed tokens per messages. **Rate limit** indicates the stage at which a client is restricted from issuing more that $\mathfrak{n}$ tokens for a given message.

Organization. In Section 2, we introduce the necessary preliminaries. In Section 3, we introduce several building blocks and prove Lemma 3.1 (weak collision resistance for PRFs). In Section 4, we give a formal definition of anonymous rate-limited credentials. In Section 5, we present our partially binding vector commitment and batched proof for CNFs of Σ-protocols. In section 6, we show how to adapt the CHL architecture [CHL05] in the random-oracle model for weak PRFs to get a general construction that lends itself to a lattice-based instantiation. In Section 7 we instantiate our generic construction from lattice-based primitives and assumptions and provide further optimizations and concrete parameter estimates. In Section 8, we show how to adapt our generic anonymous rate limited construction from Section 6 to obtain an anonymous counting token.

2 Preliminaries

Notations. A function $\mathtt{f} : \mathbb{N} \to \mathbb{R}$ is said to be negligible if $\mathtt{f}(x) = x^{-\omega(1)}$. We use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function, and $\lambda \in \mathbb{N}$ to denote the security parameter. For any $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$. For a finite set $\mathcal{S}$, we use $|\mathcal{S}|$ to denote its cardinality, $x \leftarrow\!\!\$\ \mathcal{S}$ to denote sampling x uniformly at random from $\mathcal{S}$, and $U(\mathcal{S})$ to denote the uniform distribution over $\mathcal{S}$. Let $\mathsf{BitDecomp} : \mathbb{Z}_{\geq 0} \to \{0,1\}^*$ be the canonical binary decomposition function that returns the bit representation of an integer from the least significant bit to the most significant bit. We use bold lowercase letters to denote vectors/tuples (e.g., $\mathbf{a}$) and bold uppercase letters to denote matrices (e.g., $\mathbf{A}$). By default, all vectors are column vectors. Given a vector $\mathbf{a}$, we either use a_i or $\mathbf{a}[i]$ to denote its i-th component. Let $q, d \in \mathbb{N}$ such that d is a power of two. We use $\mathbb{Z}_q$ to denote the ring of integers modulo q, $R = \mathbb{Z}[X]/(X^d + 1)$ to denote the ring of integers of the $2d$-th cyclotomic field, and $R_q = R/qR$. $\mathbb{Z}_q^\times$ and $R_q^\times$ represents the set of units over $\mathbb{Z}_q$ and R_q, respectively. For any $f(X) = \sum_{i=0}^{d-1} f_i X^i \in R$, we define the ℓ_∞ and ℓ_p norms as follows:

$$\|f\|_\infty = \max_{0 \leq i < d} |f_i| \quad , \quad \|f\|_p = \sqrt[p]{|f_0|^p + \cdots + |f_{d-1}|^p}$$

For any $\mathbf{f} \in R^k$, with $k \in \mathbb{N}_{>1}$, we extend the ℓ_∞ and ℓ_p norms as follows:

$$\|\mathbf{f}\|_\infty = \max_{0 \leq i < k} \|\mathbf{f}[i]\|_\infty \quad , \quad \|\mathbf{f}\|_p = \sqrt[p]{\|\mathbf{f}[0]\|_p^p + \cdots + \|\mathbf{f}[k-1]\|_p^p}$$

Over $R_q \supseteq \mathbb{Z}_q$, $\|\cdot\|$ is defined with respect to coefficients in the range $(-q/2, q/2)$. By default, $\|\mathbf{f}\| = \|\mathbf{f}\|_2$. For $1 \leq p \leq \infty$, let $S_\beta^p = \{x \in R_q : \|x\|_p \leq \beta\}$. Additionally, for any $a > 0$, we define $\mathcal{B}_a$ to be the set of all polynomials in R with coefficients in $\{0, \ldots, a\}$.

Let $\mathcal{R}$ be a ring such that any free $\mathcal{R}$-module $\mathcal{R}^n$ is equipped with the ℓ_2-norm. We use $\mathbf{I}_k \in \mathcal{R}^{k \times k}$ to denote the identity matrix with entries in $\mathcal{R}$, and for any matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$, $s_1(\mathbf{A}) = \sup_{\mathbf{x}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}$ denotes its second operator norm and $\|\mathbf{A}\|_\infty = \max_{i,j} \|\mathbf{A}_{i,j}\|_\infty$.

Given two modulus $q, \hat{q} \in \mathbb{N}$ such that $q < \hat{q}$, the rounding function $\lfloor \cdot \rceil_q : \mathbb{Z}_{\hat{q}} \to \mathbb{Z}_q$ is defined by mapping $x \to \lfloor (q/\hat{q})x \rceil$ for all $x \in \mathbb{Z}_{\hat{q}}$. This function is extended to $R_{\hat{q}}$ and R_q by applying rounding operation coefficient-wise. We make use of the following lemma regarding $\lfloor \cdot \rceil_q$, which follows from straightforward algebra.

Lemma 2.1 ([LLNW17a, ESLR23]). *Let $q, \hat{q} \in \mathbb{N}$ such that $q \mid \hat{q}$. For any $x \in \mathbb{Z}_{\hat{q}}$ and $y \in \mathbb{Z}_q$ such that $\lfloor x \rceil_q = y$, there exists $e \in \{0, \ldots \hat{q}/q - 1\}$ such that $x - e \equiv \frac{\hat{q}}{q} y \bmod \hat{q}$.*

Given an A-module M, where A is a commutative ring, the annihilator of $m \in M$ over A is the set $\mathrm{Ann}_A(m) := \{a \in A : a \cdot m = 0_M\}$, where 0_M is the additive identity of M.

Definition 2.1 (Annihilator-Coset Evasive Set). *Let M be a finitely generated A-module, where A is a finite commutative ring. A set $\mathcal{S} \subseteq A$ is said to be ϵ-annihilator-coset evasive with respect to M if, for any $a \in A \setminus \{0\}$ and $m \in M \setminus \{0_M\}$ such that $a + \mathrm{Ann}_A(m)$ is nontrivial, $|\mathcal{S} \cap (a + \mathrm{Ann}_A(m))| \leq \epsilon |\mathcal{S}|$.*

Min-entropy and distance. Given a distribution $\mathcal{D}$ with finite support, its min-entropy is defined as $\mathrm{H}_\infty(\mathcal{D}) := -\log(\max_{x \in \mathrm{Supp}(\mathcal{D})} \mathcal{D}(x))$. Given two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, we write $\mathcal{D}_1 \overset{\mathrm{c}}{\approx} \mathcal{D}_2$ to denote that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable, and $\mathcal{D}_1 \overset{\mathrm{s}}{\approx} \mathcal{D}_2$ to denote that they are statistically indistinguishable.

For some of our reductions, we will only be able to claim that they run in expected polynomial time. However, by relying on the following lemma, one can convert these reductions into strict polynomial-time reductions at the cost of a polynomial loss in the overall success probability.

Lemma 2.2 ([BLNS23b], adapted). *Let $\{V_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of polynomial size (in λ) verification circuits; in other words, each circuit takes as input an instance x of length polynomial in λ and a witness w, also of polynomial length, and either accepts or rejects. Let $L_\lambda = \{x \mid \exists w \text{ such that } V_\lambda(x, w) = 1\}$. Let $\mathcal{D} = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of sampling algorithms; each algorithm D_λ samples instances from the language L_λ. Let $\mathcal{A}$ be an adversary such that, for any $\lambda \in \mathbb{N}$ and $x \leftarrow D_\lambda$, $\Pr[\mathsf{V}_\lambda(1^\lambda, x, \mathcal{A}(1^\lambda, x)) = 1] \geq \epsilon_\lambda$ and $\mathcal{A}$ runs in expected time T_λ. Then, there exists an adversary $\mathcal{B}$ that, with black-box oracle access to $\mathcal{A}$, runs in worst-case time at most $2T_\lambda/\epsilon_\lambda$, such that $\Pr[\mathsf{V}_\lambda(1^\lambda, x, \mathcal{B}^{\mathcal{A}}(1^\lambda, x)) = 1] \geq \epsilon_\lambda/2$.*

Proof. In brief, the proof follows by a Markov's inequality argument. For any $\lambda \in \mathbb{N}$, let $\mathcal{B}$ be the adversary that, on input $(1^\lambda, x)$, where $x \leftarrow D_\lambda$, runs $\mathcal{A}(1^\lambda, x)$ for at most $2T_\lambda/\epsilon_\lambda$ time and then outputs whatever $\mathcal{A}$ outputs within that time. Let $t_{\mathcal{A}}$ be the random variable that corresponds to $\mathcal{A}$'s run time. Then,

$$\begin{aligned}
\Pr[\mathsf{V}_\lambda(1^\lambda, x, \mathcal{B}^{\mathcal{A}}(1^\lambda, x)) = 1] &= \Pr[(\mathsf{V}_\lambda(1^\lambda, x, \mathcal{A}(1^\lambda, x)) = 1) \wedge (t_{\mathcal{A}} \leq 2T_\lambda/\epsilon_\lambda)] \\
&\geq \Pr[(\mathsf{V}_\lambda(1^\lambda, x, \mathcal{A}(1^\lambda, x)) = 1)] - \Pr[t_{\mathcal{A}} > 2T_\lambda/\epsilon_\lambda] \\
&\geq \epsilon_\lambda/2
\end{aligned}$$

The second inequality follows from Markov's inequality. □

2.1 Lattice Primer

In this subsection, we recall some important lemmas and theorems of lattice-based cryptography.

Definition 2.2 (Gaussian function). *Given $\mathbf{c} \in \mathbb{R}^k$ and a Gaussian width parameter $\mathfrak{s} > 0$, the Gaussian function $\rho_{\mathfrak{s},\mathbf{c}} : \mathbb{R}^k \to \mathbb{R}$ centered at $\mathbf{c}$ is defined by $\rho_{\mathfrak{s},\mathbf{c}}(\mathbf{x}) = e^{-\pi \|\mathbf{x}-\mathbf{c}\|_2^2/\mathfrak{s}^2}$ for all $\mathbf{x} \in \mathbb{R}^k$. For any discrete set $\mathcal{X} \subset \mathbb{R}^k$, $\rho_{\mathfrak{s},\mathbf{c}}(\mathcal{X}) = \sum_{x \in \mathcal{X}} \rho_{\mathfrak{s},\mathbf{c}}(\mathbf{x})$.*

Definition 2.3 (Discrete Gaussian Distribution). *For any discrete set $\mathcal{X}$ admitting an efficient embedding ϕ to a subset of $\mathbb{R}^n$, the discrete Gaussian distribution over $\mathcal{X}$ centered at $\mathbf{c} \in \mathbb{R}^n$ with Gaussian width parameter $\mathfrak{s} > 0$ is defined as $D_{\mathcal{X},\mathfrak{s},\mathbf{c}}(x) = \rho_{\mathfrak{s},\mathbf{c}}(\phi(x))/\rho_{\mathfrak{s},\mathbf{c}}(\phi(\mathcal{X}))$ for all $x \in \mathcal{X}$. If $\mathbf{c} = \mathbf{0}$, we use $D_{\mathcal{X},\mathfrak{s}} = D_{\mathcal{X},\mathfrak{s},\mathbf{0}}$.*

Lattices. A k-dimensional lattice Λ is a discrete additive subgroup of $\mathbb{R}^k$, and its dual is defined by the set $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^k : \forall \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$. For any $\mathbf{A} \in R_q^{n \times m}$, the R-modules $\Lambda^\perp(\mathbf{A}) = \{\mathbf{s} \in R^m : \mathbf{A}\mathbf{s} = \mathbf{0} \bmod q\}$ and $\Lambda(\mathbf{A}^\top) = \{\mathbf{y} \in R^m : \exists \mathbf{s} \in R_q^n \wedge \mathbf{s}^\top \mathbf{A} \equiv \mathbf{y}^\top \bmod q\}$ are md-dimensional (module) lattices [LS15], and their respective duals satisfy the following equality: $(\Lambda^\perp(\mathbf{A}))^* = 1/q \cdot \Lambda(\mathbf{A}^\top)$. For any $\mathbf{c} \in R_q^m$, we use $\Lambda_{\mathbf{c}}^\perp(\mathbf{A}) = \{\mathbf{s} \in R^m : \mathbf{A}\mathbf{s} = \mathbf{c} \bmod q\}$. In general, for any $\lambda \in \mathbb{N}$ and polynomials $n(\cdot), m(\cdot), q(\cdot), d(\cdot)$, we use $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, $d = d(\lambda)$ to denote (module) lattice parameters that determine the ring R_q and the matrix $\mathbf{A}$.

Definition 2.4 (Smoothing Parameter [MR07])**.** *Given a k-dimensional lattice Λ and a positive real $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest real $\mathfrak{s} > 0$ such that $\rho_{1/\mathfrak{s},\mathbf{0}}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.*

Gadget Matrix [MP12]. Given $n, q \in \mathbb{N}, \mathfrak{b} \geq 2$, let $\tilde{q} = \lceil \log_{\mathfrak{b}} q \rceil$ $m = n\tilde{q}$ and

$$\mathbf{g}^\top = (1, \mathfrak{b}, \ldots, \mathfrak{b}^{\tilde{q}-1}) \in R_q^{1 \times \tilde{q}}$$

The gadget matrix of base $\mathfrak{b}$ is defined by $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^\top \in R_q^{n \times m}$. In addition, there exists an efficient function $\mathbf{G}^{-1} : R_q^{n \times t} \to R_q^{m \times t}$ such that for any $\mathbf{A} \in R_q^{n \times t}$, we have $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$. In brief, if $\mathfrak{b} = 2$, then $\mathbf{G}^{-1}$ decomposes each component of $\mathbf{A}$ into polynomials with binary coefficients (if $R_q = \mathbb{Z}_q$, this is simply binary decomposition).

Lemma 2.3 (Lattice Tail Bound)**.** *Let Λ be a full rank n-dimensional lattice. For any $\mathbf{c} \in \mathbb{R}^n$, $\varepsilon > 0$ and sufficiently large Gaussian width parameter $\mathfrak{s} \geq \eta_\varepsilon(\Lambda)$, we have:*

a. ([MR07, Lemma 4.4]) $\Pr_{\mathbf{x} \sim D_{\Lambda,\mathfrak{s},\mathbf{c}}} [\|\mathbf{x} - \mathbf{c}\|_2 > \mathfrak{s}\sqrt{n}] \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-n}$

b. ([Pei07, Corollary 5.3]) $\Pr_{\mathbf{x} \sim D_{\Lambda,\mathfrak{s},\mathbf{c}}} [\|\mathbf{x} - \mathbf{c}\|_\infty \geq \mathfrak{s} \cdot \log \sqrt{n}] \leq 2ne^{1-\pi \log^2 \sqrt{n}}$

Lemma 2.4 (Gaussian Rejection Sampling [Lyu12], adapted)**.** *Given $\alpha, T, m > 0$, let $V \subset R^m$ be a set of elements whose ℓ_2 norm is at most T. let $\mathfrak{s} \geq \max(\alpha T, \eta_\varepsilon(R^m))$, $M = \exp(\frac{\pi}{\alpha^2} + 1)$, and $\nu = 2\exp(\frac{-\alpha^2}{4\pi})$. Given $\mathbf{v} \in V$, sample $\mathbf{y} \leftarrow D_{R,\mathfrak{s}}^m$, and set $\mathbf{z} = \mathbf{y} + \mathbf{v}$. Consider the algorithm* RejSamp *that, on input $(\mathbf{z}, \mathbf{v}, \mathfrak{s}, M)$, output 1 with probability $\min\left(\frac{D_{R^m,\mathfrak{s}}(\mathbf{z})}{M \cdot D_{R^m,\mathfrak{s},\mathbf{v}}(\mathbf{z})}, 1\right)$, and 0 otherwise. Then,* RejSamp *outputs 1 with probability at least $(1-\nu)/M$, and conditioned on* RejSamp *returning 1, the statistical distance between the distribution of $\mathbf{z}$ and $D_{R,\mathfrak{s}}^m$ is at most ν.*

Lemma 2.5. *The second operator norm of a matrix satisfies the following bound:*

a. Given $\mathfrak{s} > 0$ and $\mathbf{A} \leftarrow (D_{R,\mathfrak{s}})^{n \times m}$, $s_1(\mathbf{A}) \leq \mathfrak{s}\sqrt{d} \cdot (\sqrt{n} + \sqrt{m} + \log \sqrt{d})$ with overwhelming probability [DM14, Fact 6].

b. Given $\mathbf{A} \leftarrow\!\!\$\ (S_1^\infty)^{n \times m}$, $s_1(\mathbf{R}) \leq \sqrt{2d/3}(\sqrt{n} + \sqrt{m} + \log \sqrt{d})$ with overwhelming probability [KLN25, Lemma A.2].

c. For any matrix $\mathbf{A} \in R^{n \times m}$, $s_1(\mathbf{A}) \leq d\sqrt{nm}\|\mathbf{A}\|_\infty$.

NTRU **lattice.** Let $f, g \in R_q$ such that f is invertible, and let $h = g/f$. The NTRU lattice is the lattice associated to the right kernel of the matrix $[h \mid 1] \in R_q^{1 \times 2}$, which correspond to the R-module $\Lambda^\top(h) = \{(u, v) \in R^2 : hu + v \equiv 0 \bmod q\}$. Let $\tilde{\eta}_\epsilon(\cdot) = \frac{1}{\sqrt{2\pi}}\eta_\varepsilon(\cdot)$ denote the scaled smoothing parameter.

Lemma 2.6 ([SS11, DN12, DLP14])**.** *There exists* ppt *algorithms* $\mathsf{NTRU} = (\mathsf{TrapGen}, \mathsf{SamplePre})$ *with the following syntax:*

- $\mathsf{TrapGen}(d, q) \to (h, \mathbf{T}_h)$*: Given* $d, q \in \mathbb{N}$*, the* TrapGen *algorithm outputs an element* $h \in R_q$ *and a trapdoor matrix* $\mathbf{T}_h \in R_q^{2\times 2}$*. In addition,* h *is statistically close to* $U(R_q^\times)$*.*
- $\mathsf{SamplePre}(h, \mathbf{T}_h, u, \mathfrak{s})$*: Given* h*, its trapdoor* $\mathbf{T}_h$*, a target* $u \in R_q$*, and a sufficiently large Gaussian width parameter* $\mathfrak{s} \geq \Theta(\sqrt{q}) \cdot \tilde{\eta}_\varepsilon(\mathbb{Z})$*, for some* $\varepsilon \in [0, 1)$*, the* SamplePre *algorithm samples* $\mathbf{v} \in R^2$ *such that* $[h \mid 1]\mathbf{v} = u$*. In addition, the statistical distance between* $\{\mathbf{v} \leftarrow \mathsf{SamplePre}(h, \mathbf{T}_h, u, \mathfrak{s})\}$ *and* $\{\mathbf{v} \leftarrow D_{\Lambda_u^\top(h),\mathfrak{s}}\}$ *is at most* $4d \cdot \varepsilon$*.*

Remark 2.1. Heuristically, it is sufficient to set $\mathfrak{s} \geq 1.17\sqrt{q} \cdot \tilde{\eta}_\varepsilon(\mathbb{Z})$ [DLP14]. In addition, given $\mathbf{g} \in R_q^m$ and $(h, \mathbf{T}_h)$, we can use standard lattice basis extension techniques [CHKP10] to sample $\mathbf{v} \in R^{m+2}$ using SamplePre such that $[h \mid \mathbf{g}^\top \mid 1]\mathbf{v} = u$ for some target $u \in R_q$, where $\mathbf{v}$ is statistically close $D_{\Lambda_u^\top([h|\mathbf{g}^\top|1]),\mathfrak{s}}$.

2.2 Hardness Assumptions

Definition 2.5 ($\mathsf{M\text{-}SIS}_{n,m,q,\beta}$ [LS15])**.** *For any* $\lambda \in \mathbb{N}$*, let* $n = n(\lambda)$*,* $m = m(\lambda)$*,* $d = d(\lambda)$*,* $q = q(\lambda)$*, and* $\beta = \beta(\lambda) < q$ *be lattice parameters. The Module Short Integer Solution (*M-SIS*) holds for* λ *if for any* ppt *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[(\mathbf{A}\mathbf{v} = \mathbf{0}) \wedge (0 < \|\mathbf{v}\| \leq \beta) \;\middle|\; \begin{matrix} \mathbf{A} \leftarrow\!\$\; R_q^{n\times m} \\ \mathbf{v} \leftarrow \mathcal{A}(1^\lambda, \beta, q, \mathbf{A}) \end{matrix}\right] \leq \mathsf{negl}(\lambda)$$

We also consider the M-SIS problem in its (Hermite) norm form where $\mathbf{A} = [\mathbf{I}_n \mid \bar{\mathbf{A}}]$, with $\bar{\mathbf{A}} \leftarrow\!\$\; R_q^{n\times(m-n)}$.

Definition 2.6 ($\mathsf{M\text{-}LWE}_{n,m,q,\chi}$ [LS15])**.** *For any* $\lambda \in \mathbb{N}$*, let* $n = n(\lambda)$*,* $m = m(\lambda)$*,* $d = d(\lambda)$*,* $q = q(\lambda)$ *be lattice parameters and* χ *be a probability distribution over* R*. The (decision) Module Learning With Errors (*M-LWE*) holds for* λ *with respect to* χ *if* $(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \overset{c}{\approx} (\mathbf{A}, \mathbf{u})$ *, where* $\mathbf{A} \leftarrow\!\$\; R_q^{m\times n}$*,* $\mathbf{s} \leftarrow \chi^n$*,* $\mathbf{e} \leftarrow \chi^m$*, and* $\mathbf{u} \leftarrow\!\$\; R_q^m$*.*

Definition 2.7 ($\mathsf{M\text{-}LWR}_{n,m,q,\hat{q},\chi}$ [BPR12, LW20])**.** *For any* $\lambda \in \mathbb{N}$*, let* $n = n(\lambda)$*,* $m = m(\lambda)$*,* $d = d(\lambda)$*,* $q = q(\lambda)$ *be lattice parameters,* $\hat{q} = \hat{q}(\lambda)$ *such that* $\hat{q} > q$ *and* χ *be a probability distribution over* R*. The (decision) Module Learning With Rounding (*M-LWR*) holds for* λ *with respect to* χ *if* $\left(\mathbf{A}, \lfloor \mathbf{A}\mathbf{s} \rceil_q\right) \overset{c}{\approx} (\mathbf{A}, \lfloor \mathbf{u} \rceil_q)$ *, where* $\mathbf{A} \leftarrow\!\$\; R_{\hat{q}}^{m\times n}$*,* $\mathbf{s} \leftarrow \chi^n$*, and* $\mathbf{u} \leftarrow\!\$\; R_{\hat{q}}^m$*.*

Remark 2.2. When $q|\hat{q}$, $\lfloor \mathbf{u} \rceil_q$ is uniform over R_q^m for any $\mathbf{u} \leftarrow\!\$\; R_{\hat{q}}^m$.

Definition 2.8 ($\mathsf{Int\text{-}NTRU\text{-}ISIS}_f$ [BLNS23b])**.** *For any* $\lambda \in \mathbb{N}$*, let* $n = n(\lambda)$*,* $m = m(\lambda)$*,* $d = d(\lambda)$*,* $q = q(\lambda)$ *be lattice parameters,* $\mathfrak{s} = \mathfrak{s}(\lambda)$ *be a Gaussian width parameter,* $\beta_0 = \beta_0(\lambda)$ *and* $\beta_1 = \beta_1(\lambda)$ *be norm bounds, and* $N = 2^{\omega(\log\lambda)}$*,* $\ell_0 = \ell_0(\lambda)$*,* $\ell_1 = \ell_1(\lambda)$ *be integers. Let* $f : [N] \to R_q$ *be an efficiently computable function. The* $\mathsf{Int\text{-}NTRU\text{-}ISIS}_f$ *holds for* λ *with respect to* $(f, \mathfrak{s}, \beta_0, \beta_1, \ell_0, \ell_1)$ *if for any* ppt *adversary* $\mathcal{A}$ *with oracle access to* $\mathcal{O}_{\mathsf{SamplePre}}$*,* $\Pr[\mathsf{Int\text{-}NTRU\text{-}ISIS}_f\text{-}\mathsf{Exp}^{\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{Int\text{-}NTRU\text{-}ISIS}_f\text{-}\mathsf{Exp}$ *and* $\mathcal{O}_{\mathsf{SamplePre}}$ *are defined in Fig. 1.*

3 Building Blocks

In this section, we recall the definitions and properties of the main building blocks that are central to this work. We defer the definitions of commitment schemes and digital signatures to Appendix A.

$\mathsf{Int\text{-}NTRU\text{-}ISIS}_f\text{-}\mathsf{Exp}^{\mathcal{A}}(1^\lambda)$:

$Q \leftarrow \emptyset$;
$(h, \mathbf{T}_h) \leftarrow \mathsf{NTRU.TrapGen}(d, q)$; $\mathbf{g} \leftarrow\!\$\ R_q^m$;
$\mathbf{a}^\top \leftarrow [h \mid \mathbf{g}^\top \mid 1]$; $\mathbf{c} \leftarrow\!\$\ R_q^{\ell_0+\ell_1}$;
$(\mathbf{x}^*, \mathbf{s}^*, \mathbf{r}^*, \mathbf{m}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{SamplePre}}}(1^\lambda, \mathbf{a}, \mathbf{c})$;
if $(\mathbf{m}^* \notin Q) \wedge \left(\mathbf{a}^\top \mathbf{s}^* = f(\mathbf{x}^*) + \mathbf{c}^\top [\mathbf{r}^{*\top} \mid \mathbf{m}^{*\top}]^\top\right)$
$\quad \wedge (\|\mathbf{s}^*\| \leq \beta_0) \wedge \left(\left\|[\mathbf{r}^{*\top} \mid \mathbf{m}^{*\top}]^\top\right\| \leq \beta_1\right)$
  return 1
else return 0

$\mathcal{O}_{\mathsf{SamplePre}}(\mathbf{r}, \mathbf{m})$:

if $([\mathbf{r}^{*\top} \mid \mathbf{m}^{*\top}]^\top > \beta_1)$:
  abort
$\mathbf{x} \leftarrow\!\$\ [N]$
$\mathbf{u} \leftarrow f(\mathbf{x}) + \mathbf{c}^\top [\mathbf{r}^\top \mid \mathbf{m}^\top]^\top$;
$\mathbf{s} \leftarrow \mathsf{NTRU.SamplePre}(\mathbf{a}, \mathbf{T}_h, \mathbf{u}, \mathfrak{s})$;
$Q \leftarrow Q \cup \{\mathbf{m}\}$;
return $(\mathbf{s}, \mathbf{x})$

Figure 1: Interactive $\mathsf{NTRU\text{-}ISIS}_f$ Experiment

3.1 Non-interactive Zero-knowledge proofs in the ROM

Definition 3.1 (NIZK). *Let $\mathfrak{c} : \mathbb{N} \to [0, 1]$ be a function. A non-interactive zero-knowledge (NIZK) proof for an NP relation $\mathcal{R}$ with completeness $\mathfrak{c}$ is a tuple of* ppt *algorithms $\Pi^{\mathcal{R}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: Given a security parameter $\lambda \in \mathbb{N}$, the* Setup *algorithm outputs a common reference string* crs*.*
- $\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \to \pi$*: Given a* crs *and a statement-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, and the* Prove *algorithm with access to a random oracle* H *outputs a proof π.*
- $\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x}, \pi) \to 1/0$*: Given a* crs*, a statement $\mathbb{x}$ and a proof π. The deterministic* Verify *algorithm with access to a random oracle* H *outputs 1 if π is a valid proof with respect to $\mathbb{x}$ and $\mathcal{R}$. Otherwise, it outputs 0.*

In addition, $\Pi^{\mathcal{R}}_{\mathsf{NIZK}}$ should satisfy the following properties:

- ***Completeness**: For any $\lambda \in \mathbb{N}$ and $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$, let $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\pi \leftarrow \mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x}, \mathbb{w})$. Then $\Pr[\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x}, \pi) = 1] \geq \mathfrak{c}(\lambda)$.*
- ***Zero-knowledge**: For any* ppt *distinguisher* D *with oracle access to* H*, there exists an efficient simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ and a negligible function* negl *such that for any $\lambda \in \mathbb{N}$, we have $\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\} \overset{c}{\approx} \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_1(1^\lambda)\}$ and*

$$\left| \begin{array}{l} \Pr\left[\mathsf{D}^{\mathsf{Prove}(\mathsf{crs},\cdot,\cdot)}(1^\lambda, \mathsf{crs}) = 1 | \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \\ \quad - \Pr\left[\mathsf{D}^{\mathsf{Sim}_2(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(1^\lambda, \mathsf{crs}) \mid (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_1(1^\lambda)\right] = 1 \end{array} \right| \leq \mathsf{negl}(\lambda)$$

where Sim_2 aborts if $(\mathbb{x}, \mathbb{w}) \notin \mathcal{R}$. Otherwise, it discards $\mathbb{w}$ and outputs π with just $\mathsf{crs}, \mathsf{td}, \mathbb{x}$ as inputs. In addition, it can reprogram H*.*

Remark 3.1 (Commit-and-Prove). We also consider commit-and-prove NIZK proofs. That is, the Prove algorithm first computes a commitment to the witness (or a portion of that witness, for example when the

witness is a vector/tuple, the commitment may be to a subset of that vector), and then the rest of the Prove algorithm computes the proof that the committed value is a valid witness (or subset thereof) for the relation; the output consists of the commitment and the proof. A useful property of the commit-and-prove paradigm is that commitments can be reused by multiple proofs. We refer the reader to the work of Escala and Groth for a formal definition [EG14].

Definition 3.2. *A non-interactive zero-knowledge (NIZK) proof* $\Pi^{\mathcal{R}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ *for an NP relation* $\mathcal{R}$ *is said to be a* $(2\mu+1)$*-round public-coin protocol if* $\mathsf{Prove}^{\mathsf{H}} = (\mathsf{P}_1, \mathsf{P}_2^{\mathsf{H}}, \ldots, \mathsf{P}_\mu^{\mathsf{H}})$*, where* P_1*, on input a* crs *and a statement-witness pair* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$*, outputs a message* a_1 *and a state* $\mathsf{st}_{\mathsf{P},1}$*, and* $\mathsf{P}_i^{\mathsf{H}}$*, on input a message* (a_{i-1}, c_{i-1}) *and a state* $\mathsf{st}_{\mathsf{P},i-1}$*, outputs a tuple* (c_i, a_i) *and a state* $\mathsf{st}_{\mathsf{P},i}$ *such that* $\mathsf{H}(\mathsf{crs}, \mathbb{x}, a_1, c_2, a_2, \ldots, a_{i-1}, c_{i-1}) = c_i$*. When* $\mu = 1$*, we say that* $\Pi^{\mathcal{R}}_{\mathsf{NIZK}}$ *is a* Σ*-protocol.*

Definition 3.3 (Knowledge Soundness)**.** *Let* $\Pi^{\mathcal{R}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ *be a NIZK proof system for an NP relation* $\mathcal{R}$*, and let* $\kappa : \mathbb{N} \times \mathbb{N} \to [0,1]$ *be a function.* $\Pi^{\mathcal{R}}_{\mathsf{NIZK}}$ *is said to be knowledge-sound with knowledge-soundness error* κ *if there exists a polynomial* q *and an extractor* Ext *satisfying one of the following criteria:*

- ***Rewindable extraction:*** *the extractor* Ext *runs in expected polynomial time. In addition, for any* $\lambda \in \mathbb{N}$*,* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* ppt *prover* P^* *that, on input* $(1^\lambda, \mathsf{crs})$*, outputs a valid statement-proof pair* $(\mathbb{x}, \pi)$ *with probability* ϵ_{P^*} *after performing* Q *queries to* H*, then,* Ext *with black-box rewinding access to* P^* *satisfies the following:* $\Pr[(\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x}, \pi) = 1) \wedge ((\mathbb{x}, \mathbb{w}^*) \in \mathcal{R}) : \mathbb{w}^* \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(\mathsf{crs}, \mathbb{x}, \pi)] \geq (\epsilon_{\mathsf{P}^*} - \kappa(\lambda, Q))/q(\lambda)$
- ***Straight-line extraction:*** *the extractor is a pair of* ppt *algorithms* $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$ *such that for any* $\lambda \in \mathbb{N}$*,* $\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\} \stackrel{c}{\approx} \{\mathsf{crs} : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Ext}_1\}$*. Moreover, for any* ppt *prover* P^**, that on input* $(1^\lambda, \mathsf{crs})$*, where* $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Ext}_1(1^\lambda)$*, outputs a valid statement-proof pair* $(\mathbb{x}, \pi)$ *with probability* ϵ_{P^*} *after performing* Q *queries to* H*, then* $\Pr[(\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x}, \pi) = 1) \wedge ((\mathbb{x}, \mathbb{w}^*) \in \mathcal{R}) : \mathbb{w}^* \leftarrow \mathsf{Ext}_2(\mathsf{crs}, \mathsf{td}, \mathbb{x}, \pi)] \geq (\epsilon_{\mathsf{P}^*} - \kappa(\lambda, Q))/q(\lambda)$*.*

In both cases, Ext *implements* H *for* P^* *and may reprogram* H *during its interaction with* P^**. In addition, if the first criterion holds, then we say that* $\Pi^{\mathcal{R}}_{\mathsf{NIZK}}$ *is knowledge-sound with rewindable extraction. If the second criterion holds, then we say that* $\Pi^{\mathcal{R}}_{\mathsf{NIZK}}$ *is knowledge-sound with straight-line extraction.*

In this work, we focus on knowledge-sound NIZK proof systems with negligible knowledge-soundness error, assuming an appropriate hardness assumption. Furthermore, unless explicitly stated otherwise, all proof systems are assumed to be knowledge sound.

Remark 3.2. We also deal with NIZK that have *relaxed* knowledge-soundness, i.e., given an NP relation $\mathcal{R}$, the extractor will only be able to extract a statement-witness pair $(\mathbb{x}, \mathbb{w}) \in \tilde{\mathcal{R}} \supseteq \mathcal{R}$. More specifically, we consider NIZK with relaxed knowledge-soundness where the relaxed relation $\tilde{\mathcal{R}}$ is of the form $\tilde{\mathcal{R}} = \mathcal{R} \times \mathcal{S}$ where $\mathcal{S}$ is the slack space.

LNP framework. Lyubashevsky, Nguyen, and Plancon [LNP22] proposed a knowledge-sound NIZK commit-and-prove proof system with rewindable extraction, which we henceforth denoted by LNP. The proof system allows one to prove quadratic relations over elements of R_q^m. More specifically, given $\mathbf{s} \in R_q^m$, let $\tilde{\mathbf{s}} := (\mathbf{s}, \theta(\mathbf{s})) \in R_q^{2m}$, where $\theta : R_q \to R_q$ is the automorphism sending X to X^{-1}. Also, for $x \in R_q$, denote $\mathsf{ct}(x) \in \mathbb{Z}_q$ to be the constant term of x. Then LNP alows one to commit-and-prove knowledge of $\tilde{\mathbf{s}}$ that satisfies relations of the following form:

- **Quadratic equations**: Given a quadratic function $f : R_q^{2m} \to R_q$ of the form

$$\mathbf{x} \mapsto \mathbf{x}^\intercal \mathbf{Q}_2 \mathbf{x} + \mathbf{x}^\intercal \mathbf{q}_1 + q_0, \quad \text{for public } \mathbf{Q}_2, \mathbf{q}_1, q_0, \tag{3.1}$$

prove that $f(\tilde{\mathbf{s}}) = 0$.

- **Vanishing constant terms**: Given a quadratic function $F : R_q^{2m} \to R_q$ of the form Equation 3.1, prove that $\mathsf{ct}(F(\tilde{\mathbf{s}})) = 0$.
- **Exact ℓ_2-norm**: Given $\mathbf{D} \in R_q^{k \times 2m}$, $\mathbf{u} \in R_q^k$, and a norm bound β, prove that $\|\mathbf{D}\tilde{\mathbf{s}} - \mathbf{u}\| \leq \beta$.
- **Approximate ℓ_∞-norm**: Given $\mathbf{D} \in R_q^{k \times 2m}$, $\mathbf{u} \in R_q^k$, and a norm bound β, prove that $\|\mathbf{D}\tilde{\mathbf{s}} - \mathbf{u}\|_\infty \leq \gamma \cdot \beta$, where γ is a relaxation constant.
- **Binary vectors**: Given $\mathbf{D} \in R_q^{k \times 2m}$, and $\mathbf{u} \in R_q^k$, prove that $\mathbf{D}\tilde{\mathbf{s}} - \mathbf{u} \in \mathcal{B}_1^k$.

Bootle, Lyubashevsky, Nguyen, and Sorniotti [BLNS23b, Section 6] showed that, using the Katsumata's transform [Kat21], LNP can be made efficiently straight-line extractable. This transform modifies the common reference string of the underlying proof system to include a trapdoor that enables extraction of the committed values. We denote this version of LNP by $\mathsf{LNP}_{\mathsf{str}}$.

3.2 Signature with Efficient Protocols

We recall the notion of *signature with efficient protocols* (SEP), introduced by Camenisch and Lysyanskaya [CL02]. Informally, an SEP is a digital signature paired with a commitment scheme Π_{Com}, such that, in addition to issuing signatures on messages, one can: (1) obliviously issue signatures on a message m by signing commitments to m generated by Π_{Com}; and (2) *efficiently* give a zero-knowledge proof of knowledge of a message and a valid signature on that message.

In the literature [CL02, CL04, LMPY16, LLM+16, JRS23, BLNS23a], there is no formal definition of an SEP, and one of the main obstacles toward obtaining such a definition is the difficulty of capturing the notion of *forgery* when an adversary can issue signature request on *well-formed* commitments and only present proofs of knowledge of valid message-signature pairs (see Jeudy et al. [JRS23, Section 5] for an extended discussion). SEPs generally serve as building blocks for other primitives such as anonymous credential, group-based signature, etc., and are required to satisfy three requirements: (1) the underlying signature should be secure against Existential Unforgeability under Chosen Message Attack (EUF-CMA), (2) the associated commitment scheme should admit an efficient and sound NIZK proof of knowledge of an opening to guarantee the *well-formedness* of commitments, and (3) there should be an efficient and sound (commit-and-prove) NIZK proof of knowledge of a valid message-signature pair. Usually, these three requirements are enough to help argue about the security of primitives that use an SEP, and this will also be the case for us.

Here, we do not attempt to provide a formal definition of an SEP. Our goal is to *formally* distill the algorithms that constitute an SEP based on prior works [CL02, LLM+16, JRS23, BLNS23a] and the three requirements that we mentioned in the previous paragraph. It is worth noting that in general, the *plain* signature algorithm simply issues signatures on commitments of messages. This allows to easily and efficiently obtain signatures on commitments by committing to messages using hidden randomness. Moreover, we will focus on SEPs that can handle vectors of messages.

Given a commitment scheme $\Pi_{\mathsf{Com}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ for committing to a vector of messages, consider the relation

$$\mathcal{R}_{\mathsf{Com}}(\mathsf{crs}_{\mathsf{Com}}) := \left\{ \begin{matrix} \mathbb{x} := (c_1, c_2), \\ \mathbb{w} := (m_1, m_2, o_1, o_2) \end{matrix} : \begin{matrix} \mathsf{Open}(\mathsf{crs}_{\mathsf{Com}}, c_1, m_1, o_1) = 1 \wedge \\ \mathsf{Open}(\mathsf{crs}_{\mathsf{Com}}, c_2, (m_1, m_2), o_2) = 1 \end{matrix} \right\} \tag{3.2}$$

with associated NIZK proof $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$. Note that if $c_2 = \perp$, i.e., $\mathbb{x} := c_1$, then $\mathcal{R}_{\mathsf{Com}}$ corresponds to the usual relation related to proving well-formedness of a commitment.

Definition 3.4 (Signature with efficient protocols)**.** *Let* Π_{Com} *be a linearly homomorphic commitment scheme that admits an efficient NIZK PoK* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$ *for the relation* $\mathcal{R}_{\mathsf{Com}}$*. A signature with efficient protocols associated with* Π_{Com} *is a tuple of* ppt *algorithms* $\Pi_{\mathsf{SEP}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{SignCom}, \mathsf{Verify}, \mathsf{OblSign})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: *Given the security parameter* $\lambda \in \mathbb{N}$*, the* Setup *algorithm computes* $\mathsf{crs}_{\mathsf{Com}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Setup}(1^\lambda)$*, which internally calls* $\Pi_{\mathsf{Com}}.\mathsf{Setup}$*. It then generates a common reference string* crs'*, sets* $\mathsf{crs} := (\mathsf{crs}', \mathsf{crs}_{\mathsf{Com}})$ *and outputs* crs*.*
- $\mathsf{Gen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$: *Given the* crs*, the* Gen *algorithm outputs a public-secret key pair* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\mathsf{Sign}(\mathsf{crs}, \mathsf{sk}, m) \to \sigma$: *Given* crs*, a secret key* sk*, and a message* m*, the* Sign *algorithm outputs a signature* σ*. Specifically,* Sign *first computes* $(c, o) \leftarrow \Pi_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, m)$*. It then issues a signature* $\sigma' \leftarrow \mathsf{SignCom}(\mathsf{crs}, \mathsf{sk}, c)$ *and returns* $\sigma := (\sigma', o)$*.*
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, m, \sigma) \to 1/0$: *Given* crs*, a public key* pk*, a message* m*, and a signature* σ*, the deterministic* Verify *outputs* 1 *if* σ *is a valid signature on* m *w.r.t* pk*. Else it outputs* 0*.*
- $\mathsf{OblSign}(\mathtt{S}(\mathsf{crs}, \mathsf{sk}, \tilde{c}) \rightleftharpoons \mathtt{R}(\mathsf{crs}, \tilde{c}, m)) \to (\pi_{\mathsf{Open}}; (\sigma, r))$: *This is an interactive protocol between a signer* S*, holding* $(\mathsf{crs}, \mathsf{sk}, \tilde{c})$*, and a receiver* R*, holding* $(\mathsf{crs}, \tilde{c}, m, o_{\tilde{c}})$*, where* $\tilde{c}$ *is a commitment to* m *with opening* $o_{\tilde{c}}$*. Specifically, we consider a protocol that is round-optimal, consisting of two moves, and proceeds as follows:*
  1. $\mathtt{R}(\mathsf{crs}, \mathsf{c}, m, o_{\tilde{c}})$: *Compute* $(c, o) \leftarrow \Pi_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, m)$ *and* $\pi_{\mathsf{Open}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (\mathsf{c}, c), \mathbb{w} := (m, o, o_{\mathsf{c}}))$*. Send* (c, π_{Open}) *to* S*.*
  2. $\mathtt{S}(\mathsf{crs}, \mathsf{sk}, \mathsf{c}, c, \pi_{\mathsf{Open}})$: *If* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (\mathsf{c}, c), \pi_{\mathsf{Open}}) = 0$*, send* $\perp$ *to* R*. Else, compute* $(c_0, o_0) \leftarrow \Pi_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, 0)$ *and* $c' \leftarrow c + c_0$*. Then compute* $\sigma' \leftarrow \mathsf{SignCom}(\mathsf{sk}, c')$*, and send* $\sigma := (\sigma', o_0)$ *to* R*.*
  3. $\mathtt{R}(m, o, \sigma)$: *Given* $(\sigma', o_0) := \sigma$*, use the linear homomorphic property of* Π_{Com} *to derive* $o' \leftarrow o + o_0$*, and set the signature* $\sigma'' := (\sigma', o')$*.*

In addition, Π_{SEP} *should satisfy the following properties:*

- ***Correctness and Security of the underlying signature****: The underlying digital signature* $\Pi_{\mathsf{SEP}}|_\Sigma = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{SignCom}, \mathsf{Verify})$ *is correct and EUF-CMA-secure for any* ppt *adversary. (see Appendix A for a definition of digital signatures).*
- ***Commit-and-prove NIZK proof of knowledge for message-signature pairs****: There exists an efficient commit-and-prove NIZK proof* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for the relation*

$$\mathcal{R}_{M,\Sigma}(\mathsf{crs}) = \left\{\mathbb{x} := \mathsf{pk}, \mathbb{w} := (m, \sigma) : \Pi_{\mathsf{SEP}}.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, m, \sigma) = 1\right\} \tag{3.3}$$

  where the Prove *algorithm outputs, in addition to a proof* π*, a commitment to the message* m *underlying the signed message-signature pair. Note that* Prove *could also commit to the entire witness* $\mathbb{w}$*, but for simplicity it is sufficient to commit to* m*, especially since* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *is assumed to be knowledge sound.*

Remark 3.3. To facilitate the composition of OblSign with other protocols, we require that $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$ is knowledge-sound with straight-line extraction.

3.3 Pseudo-random Functions

Definition 3.5. *A pseudo-random function (PRF) with input domain $\mathcal{X}$ and output domain $\mathcal{Y}$ is a tuple of* ppt *algorithms $\Pi_{\mathsf{PRF}} = (\mathsf{Gen}, \mathsf{Eval})$ with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to k$*: Given a security parameter $\lambda \in \mathbb{N}$, the* Gen *algorithm outputs a secret key k.*
- $\mathsf{Eval}(k, x) \to y$*: Given a secret key k and an input $x \in \mathcal{X}$, the deterministic* Eval *algorithm outputs $y \in \mathcal{Y}$.*

Generally, a PRF Π_{PRF} is said to be secure if a ppt distinguisher with oracle access to Π_{PRF} cannot distinguish it from a truly random function defined over the same input-output domain pair, except with negligible probability. However, in this work, we consider a weaker notion called *weak pseudo-randomness*, where the distinguisher sends all its queries at once, instead of sending them adaptively. Note that such a PRF can be made adaptively secure in the random oracle model by first hashing a query and then evaluating the PRF on the resulting hash digest.

Definition 3.6 (Weak Pseudo-randomness)**.** *A PRF Π_{PRF}, defined over input-output domain pair $(\mathcal{X}, \mathcal{Y})$, is said to be weakly secure if for any* ppt *distinguisher* D*, there exists a negligible function* negl *such that for any $\lambda \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{r}(x_1,\ldots,x_n) \leftarrow \mathsf{D}(1^\lambda); b \leftarrow\!\$\ \{0,1\}; k \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda);\\ f \leftarrow\!\$\ \mathsf{Func}[\mathcal{X},\mathcal{Y}];\\ \textit{if } b = 0 : (y_1,\ldots,y_n) \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Eval}(k,(x_1,\ldots,x_n));\\ \textit{if } b = 1 : (y_1,\ldots,y_n) \leftarrow f((x_1,\ldots,x_n));\\ b' \leftarrow \mathsf{D}(y_1,\ldots,y_n) :\\ b = b'\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Remark 3.4 (High Min-entropy of $\Pi_{\mathsf{PRF}}.\mathsf{Gen}$). For Π_{PRF} to be secure, whether adaptively or non-adaptively (weak pseudo-randomness), it is necessary that, for any $\lambda \in \mathbb{N}$, we have $\mathrm{H}_\infty\big(\mathsf{PRF}.\mathsf{Gen}(1^\lambda)\big) \geq \omega(\log \lambda)$. Otherwise, one can easily distinguish it from a random function by testing one of the most likely keys output by $\Pi_{\mathsf{PRF}}.\mathsf{Gen}$.

Definition 3.7 (Almost Key-Homomorphic PRFs [BLMR13], adapted)**.** *Given a PRF Π_{PRF} defined over input-output domain pair $(\mathcal{X}, \mathcal{Y})$, let $\mathcal{K}$ be the support of the distribution defined by $\Pi_{\mathsf{PRF}}.\mathsf{Gen}$. In addition, suppose $(\mathcal{K}, \odot)$ is module and that $(\mathcal{Y}, \boxdot, \|\cdot\|)$ is a normed module. Π_{PRF} is said to be a secure δ-almost key-homomorphic PRF, with $\delta \geq 0$, if it is (non-) adaptively secure, and if for any $k_1, k_2 \in \mathcal{K}$ and $x \in \mathcal{X}$, $\Pi_{\mathsf{PRF}}.\mathsf{Eval}(k_1 \odot k_2, x) \boxdot e = \Pi_{\mathsf{PRF}}.\mathsf{Eval}(k_1, x) \boxdot \Pi_{\mathsf{PRF}}.\mathsf{Eval}(k_2, x)$, where $e \in \mathcal{Y}$ is such that most $\|e\| \leq \delta$. If $\delta = 0$, then Π_{PRF} is (perfectly) key-homomorphic.*

Given a weakly secure PRF Π_{PRF} defined over an input-output domain pair $(\mathcal{X}, \mathcal{Y})$ and a hash function $\mathsf{H} : \{0,1\}^* \to \mathcal{X}$ modeled as a random oracle, we use $\Pi^{\mathsf{H}}_{\mathsf{PRF}} := (\mathsf{Gen}, \mathsf{Eval}^{\mathsf{H}})$, where $\mathsf{Eval}^{\mathsf{H}}(k, x) := \Pi_{\mathsf{PRF}}(k, \mathsf{H}(x))$, to denote an adaptively secure instantiation of Π_{PRF} in the random oracle model.

Composition with NIZK proofs. In addition, we focus on PRFs whose output domains are modules over commutative rings and whose outputs can be efficiently proven using NIZK proofs. Specifically, let Π_{PRF} be a PRF with output domain $\mathcal{Y}$, where $\mathcal{Y}$ is an A-module for some ring A. We require that there exist an SEP Π_{SEP} and an associated commit-and-prove NIZK for message-signature pairs, $\Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}}$, such that, given an output $(c_{M,\Sigma}, \pi_{M,\Sigma}, o_{M,\Sigma})$ of $\Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}}.\mathsf{Prove}^{\mathsf{H}}$ on an input $(c_{\mathsf{SEP}}, o_{\mathsf{SEP}}, m, k, \sigma)$, where c_{SEP} is a

commitment to (m, k), σ is a signature on c_{SEP}, and k is a key for Π_{PRF}, and given a commitment c_j to an index $j \in [n]$ generated by some commitment scheme Π_{Com}, there exists an efficient NIZK argument of knowledge for the relation $\mathcal{R}_{\mathsf{1Many}}(\mathsf{crs}, c_{M,\Sigma}, c_j) := \{\mathcal{R}_{\mathsf{sn}}, \mathcal{R}_{\mathsf{tg}}, \mathcal{R}_{\mathsf{sn}} \wedge \mathcal{R}_{\mathsf{tg}}\}$, where

$$\begin{aligned}
\mathcal{R}_{\mathsf{sn}}(\mathsf{crs}, c_{M,\Sigma}, c_j) := \left\{ \begin{array}{l} \mathbb{x} := \left(y_1, \{x_i\}_{i\in[n]}\right), \mathbb{w} := ((m,k), o_{M,\Sigma}, j, o_j) : \\ (\Pi^{M,\Sigma}_{\mathsf{Com}}.\mathsf{Open}\left(\mathsf{crs}, c_{M,\Sigma}, (m,k), o_{M,\Sigma}\right) = 1) \wedge j \in [n] \wedge \\ (\Pi_{\mathsf{Com}}.\mathsf{Open}(\mathsf{crs}, c_j, j, o_j) = 1) \wedge (y_1 = \mathsf{Eval}^{\mathsf{H}}(k, x_j)) \end{array} \right\} \\
\mathcal{R}_{\mathsf{tg}}(\mathsf{crs}, c_{M,\Sigma}, c_j) := \left\{ \begin{array}{l} \mathbb{x} := \left(a, y_2, \{x'_i\}_{i\in[n]}\right), \mathbb{w} := ((m,k), o_{M,\Sigma}, j, o_j) : \\ (\Pi^{M,\Sigma}_{\mathsf{Com}}.\mathsf{Open}\left(\mathsf{crs}, c_{M,\Sigma}, (m,k), o_{M,\Sigma}\right) = 1) \wedge j \in [n] \wedge \\ (\Pi_{\mathsf{Com}}.\mathsf{Open}(\mathsf{crs}, c_j, j, o_j) = 1) \wedge (y_2 = F(m) + a \cdot \mathsf{Eval}^{\mathsf{H}}(k, x'_j)) \end{array} \right\}
\end{aligned} \tag{3.4}$$

where $\Pi^{M,\Sigma}_{\mathsf{Com}}$ is the commitment associated to $\Pi^{M,\Sigma}_{\mathsf{NIZK}}$, $a \in A$, and F is a (collision-resistant) function whose codomain is $\mathcal{Y}$.

Remark 3.5. We stress that, although $\Pi_{\mathsf{PRF}}.\mathsf{Eval}$ in equation (3.4) invokes a hash function H modeled as a random oracle, the NIZK proof does not need to certify correct evaluation of H. Since all candidate inputs to Π_{PRF} are part of the statement, it suffices to prove that y_i is the output of Π_{PRF} on one of them, i.e., an "OR" statement. Moreover, if Π_{Com} is linearly homomorphic commitment (LHC), then checking that $j \in [n]$ can be reduced to proving that there exists $i \in [n]$ such that $c_j - \Pi_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, i; \perp)$ is a commitment to zero, which is again an "OR" statement.

Weak Collision Resistance. Given a secure PRF Π_{PRF}, one would hope that it is also a *good* collision resistance function. Indeed, this is the case when a ppt adversary is granted oracle access to Π_{PRF}. However, in our setting, we require that Π_{PRF} remains collision resistant even when the adversary is given the key. Bourse, Pointcheval, and Sanders [BPS19] considered a strong notion of collision resistance, requiring security even when the adversary has full control over the keys. In contrast, we propose a slightly weaker notion that holds for key-homomorphic PRFs: we require that a key-homomorphic PRF remains collision resistant when the adversary can partially influence the key and when the PRF's domain is of polynomial size. We show below that this weaker notion follows directly from the security of the underlying key-homomorphic PRF.

Definition 3.8 (Weak Collision Resistance)**.** *Let $\Pi_{\mathsf{PRF}} = (\mathsf{Gen}, \mathsf{Eval})$ be a key-homomorphic PRF with polynomial-size domain $\mathcal{X}$ and superpolynomial-size codomain $\mathcal{Y}$. Let $\mathcal{S} \subset \mathcal{Y}$ be an efficiently testable set. Π_{PRF} is said to be weakly collision resistant with respect to $\mathcal{S}$ if for any* ppt *adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_{\mathcal{K}}$, there exists a negligible function* negl *such that, for any $\lambda \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{r} (k_1, k_2, k'_1, k'_2, x_1, x_2) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{K}}}(1^\lambda) : \\ \left((k_1, k_2), (k'_1, k'_2) \in \mathcal{Q}\right) \wedge \\ (\mathsf{Eval}(k_1 \odot k_2, x_1) \boxdot \mathsf{Eval}(k'_1, x_2)^{-1} \boxdot \mathsf{Eval}(k'_2, x_2)^{-1}) \in \mathcal{S} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Where the oracle $\mathcal{O}_{\mathcal{K}}$, on input k^*_1, executes $k^*_2 \leftarrow \mathsf{Gen}(1^\lambda)$, adds (k^*_1, k^*_2) to $\mathcal{Q}$, and returns k^*_2.

Remark 3.6. In Definition 3.8, we explicitly required the domain of the PRF to be of polynomial size and its codomain to be of superpolynomial size. This is because, under these conditions, we can easily rule out certain pathological cases of PRFs with *bad* sets, where querying the PRF on an input from such a set forces the function to degenerate. This follows from the fact that, by a simple hybrid argument, a PRF defined over a polynomial-size domain and superpolynomial-size codomain must be injective except with negligible probability.

Remark 3.7 (Relaxing the requirement on Key-Homomorphism). We note that the key-homomorphism requirement in the above definition can be relaxed if, instead, one considers the concatenation of PRF outputs. Namely, given $k_1, k_2 \in \mathcal{K}$, where, without loss of generality, k_2 is provided by the oracle $\mathcal{O}_{\mathsf{ck}}$, and input $x \in \mathcal{X}$, the output is $F_{k_1}(x) \| F_{k_2}(x)$. Alternatively, if $(\mathcal{Y}, \boxdot)$ is a finite module, one can ouput $F_{k_1}(x) \boxdot F_{k_2}(x)$. In both cases, weak collision resistance follows readily from the fact that k_2 is honestly generated and $\mathcal{X}$ is of polynomial size.

Lemma 3.1. *Let Π_{PRF} be a key-homomorphic PRF with key space $\mathcal{K}$, polynomial-size domain $\mathcal{X}$ and superpolynomial-size codomain $\mathcal{Y}$. Let $\mathcal{S} \subset \mathcal{Y}$ be any efficiently testable set such that, for any $\lambda \in \mathbb{N}$, $|\mathcal{Y}| \geq 2^{\omega(\log \lambda)} \cdot |\mathcal{S}|$. If Π_{PRF} is secure, then it is weakly collision resistant with respect to $\mathcal{S}$.*

Proof. We proceed by contraposition. Suppose $\lambda \in \mathbb{N}$, and let $\mathcal{A}$ be a ppt adversary that, after Q key queries, outputs a weak collision on Π_{PRF} with non-negligible probability $\epsilon_{\mathcal{A}}$. We use $\mathcal{A}$ to build a ppt adversary $\mathcal{B}$ that breaks Π_{PRF}'s security as follows:

- At initialization, $\mathcal{B}$ samples $i \leftarrow\!\!\$\ [Q]$. Then, it answers the next $i-1$ key queries of $\mathcal{A}$ as $\mathcal{O}_{\mathcal{K}}$ and maintains a set $\mathcal{Q}$ where it stores query-answer pairs.
- Upon receiving $\mathcal{A}$'s i-th key query with value $k_1^{(i)} \in \mathcal{K}$, $\mathcal{B}$'s stops responding to $\mathcal{A}$. Next, it queries its function oracle $\mathcal{O}_{\Pi_{\mathsf{PRF}}}$ on each $x \in \mathcal{X}$ and build the sets $\mathcal{Z} := \{\mathsf{Eval}(k_1^{(j)} \odot k_2^{(j)}, x_1) \boxdot (\Pi_{\mathsf{PRF}}.\mathsf{Eval}(k_1^{(i)}, x_2))^{-1} \boxdot (\mathcal{O}_{\Pi_{\mathsf{PRF}}}(x_2))^{-1} : x_1, x_2 \in \mathcal{X}, (k_1^{(j)}, k_2^{(j)}) \in \mathcal{Q}\}$.
- $\mathcal{B}$ outputs 1 if there exists $a \in \mathcal{Z}$ such that $a \in \mathcal{S}$. Else, it outputs 0.

Note that $\mathcal{Z}$ is of polynomial size since $\mathcal{Q}$ and $\mathcal{X}$ are polynomial in size. If $\mathcal{O}_{\Pi_{\mathsf{PRF}}}$ implements Π_{PRF}, then in case $\mathcal{B}$ guesses the correct query, $\mathcal{Z}$ will contains an elment in $\mathcal{S}$ with probability $\epsilon_{\mathcal{A}}/Q$ since the key used by $\mathcal{O}_{\Pi_{\mathsf{PRF}}}$ is perfectly close to a key that $\mathcal{B}$ would have returned to $\mathcal{A}$ as part of the weakly collision experiment. However, if $\mathcal{O}_{\Pi_{\mathsf{PRF}}}$ implements a truly random function $f : \mathcal{X} \to \mathcal{Y}$, then $\mathcal{Z}$ will contains an element in $\mathcal{S}$ with probability at most $\frac{(Q-1)|\mathcal{S}||\mathcal{X}|^2}{|\mathcal{Y}|} \leq \frac{(Q-1)|\mathcal{X}|^2}{2^{\omega(\log \lambda)}}$. Therefore, the advantage of $\mathcal{B}$ in distinguishing Π_{PRF} from f is at least $|\frac{\epsilon_{\mathcal{A}}}{Q} - \frac{(Q-1)|\mathcal{X}|^2}{2^{\omega(\log \lambda)}}|$, which is non-negligible. □

Corollary 3.0.1. *Let Π_{PRF} be a secure δ-almost key-homomorphic PRF with polynomial size domain $\mathcal{X}$ and superpolynomial size codomain $\mathcal{Y}$. Let $\mathcal{S}_\delta \subset \mathcal{Y}$ be the set of elements in $\mathcal{Y}$ with norm at most δ. In addition, suppose for any $\lambda \in \mathbb{N}$, $|\mathcal{Y}| \geq 2^{\omega(\log \lambda)} \cdot |\mathcal{S}_\delta|$. Then, Π_{PRF} is weakly collision resistant with respect to $\mathcal{S}_\delta$.*

We note that Libert et al. [LLNW17b, Lemma 6] proved, in a non-black-box way, that Boneh et al.'s Learning-With-Rounding-based key-homomorphic PRF [BLMR13] is weakly collision resistant by explicitly relying on the uniformity of the keys. In contrast, Corollary 3.0.1 allows us to obtain the same result in a black-box way when the norm over $\mathcal{Y}$ is the ℓ_∞-norm and $\delta = 1$, thereby enabling the use of keys drawn from high min-entropy distributions that are not necessarily uniform and for which Learning-With-Rounding holds.

4 Privacy-Preserving Rate-limited tokens

In this section we recall the definition of anonymous rate-limited tokens (ARLTs) based on the work of Camenisch et al. [CHK+06] and Chairattana-Apirom et al. [CDLT25a]. In addition, we make the observation that the anonymous counting tokens introduced by Benhamouda et al. [BRS23] can be seen as instantiation of ARLTs for fixed contexts.

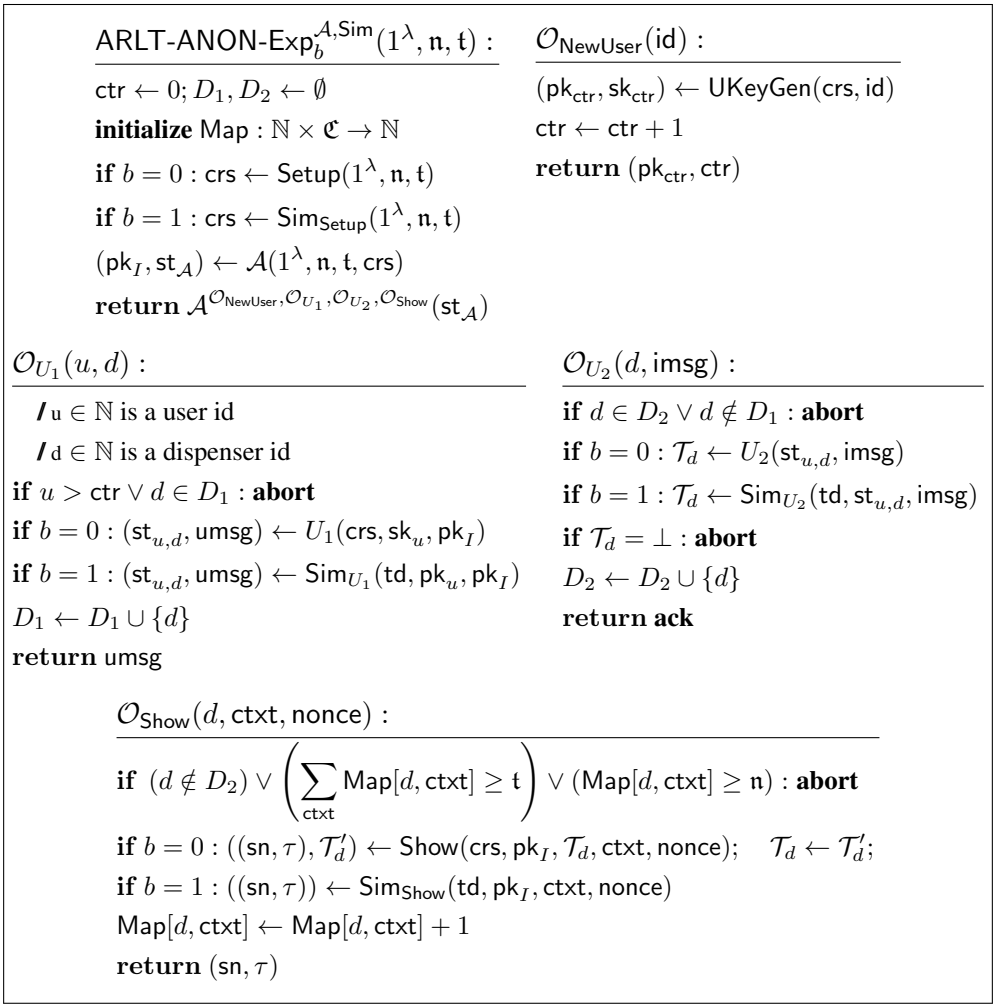

Figure 2: Π_{ARLT}'s anonymity Experiment

Definition 4.1 (ARLT)**.** *An anonymous rate-limited token (ARLT) with context space* $\mathfrak{C}$*, nonce space* $\mathfrak{N}$*, and user's identity space* ID *is a tuple of* ppt *algorithms* $\Pi_{\mathsf{ARLT}} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify}, \mathsf{Identify})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda,\mathfrak{n},\mathfrak{t}) \to \mathsf{crs}$: *Given a security parameter* $\lambda \in \mathbb{N}$*, a rate limit* $\mathfrak{n} \in \mathbb{N}$ *for all contexts in* $\mathfrak{C}$ *and a total rate limit* $\mathfrak{t} \in \mathbb{N}$ *such that* $\mathfrak{n} = \mathsf{poly}(\lambda)$*,* $\mathfrak{t} = \mathsf{poly}(\lambda)$*, and* $\mathfrak{t} \geq \mathfrak{n}$*, the* Setup *algorithm outputs a common reference string* crs*.*
- $\mathsf{IKeyGen}(\mathsf{crs}) \to (\mathsf{pk}_I, \mathsf{sk}_I)$: *Given* crs*, the issuer key generation algorithm* IKeyGen *outputs a public-secret key pair* $(\mathsf{pk}_I, \mathsf{sk}_I)$*.*
- $\mathsf{UKeyGen}(\mathsf{crs}, \mathsf{id}_U) \to (\mathsf{pk}_U, \mathsf{sk}_U)$: *Given* crs *and* $\mathsf{id}_U \in \mathsf{ID}$*, the user key generation algorithm* UKeyGen *outputs a public-secret key pair* $(\mathsf{pk}_U, \mathsf{sk}_U)$*.*
- $\mathsf{Issue}\,(I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U) \rightleftharpoons U(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I)) \to (\perp; \mathcal{T})$: *This an interactive protocol between an issuer* I *with inputs* $(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U)$ *and a user* U *with inputs* $(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I)$*. At the end of the interaction,* I *outputs* $\perp$ *and* U *outputs a token dispenser* $\mathcal{T}$*. Similar to Camenisch et al. [CHK+06] and*

Chairattana-Apirom et al. [CDLT25b], we focus on a round-optimal (two-move) Issue *protocol with the following structure:*

1. $U_1(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I) \to (\mathsf{st}_U, \mathsf{umsg})$.
2. $I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U, \mathsf{umsg}) \to \mathsf{imsg}$.
3. $U_2(\mathsf{st}_U, \mathsf{imsg}) \to \mathcal{T}$.

- $\mathsf{Show}(\mathsf{crs}, \mathsf{pk}_I, \mathcal{T}, \mathsf{ctxt}, \mathsf{nonce}) \to ((\mathsf{sn}, \tau), \mathcal{T}')$: *Given the* crs*, an issuer public key* pk_I*, a token dispenser* $\mathcal{T}$*, a context* $\mathsf{ctxt} \in \mathfrak{C}$*, and a nonce* $\mathsf{nonce} \in \mathfrak{N}$*, the* Show *algorithm outputs a serial number-token pair* (sn, τ) *and an updated token dispenser* $\mathcal{T}'$.

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau, \mathsf{ctxt}, \mathsf{nonce}) \to 1/0$*: Given the* crs*, an issuer public key* pk_I*, a serial number-token pair* (sn, τ)*, a context* $\mathsf{ctxt} \in \mathfrak{C}$*, and a nonce* $\mathsf{nonce} \in \mathfrak{N}$*, the deterministic* Verify *algorithm outputs* 1 *is* (sn, τ) *is valid with respect to* $(\mathsf{pk}_I, \mathsf{ctxt} \in \mathfrak{C}, r)$*. Otherwise, it outputs* 0.

- $\mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau, \tau', \mathsf{ctxt}, \mathsf{nonce}, \mathsf{nonce}') \to \mathsf{pk}_U$*: Given the* crs*, an issuer public key* pk_I*, a serial number* sn *with associated tokens* τ *and* τ'*, a context* $\mathsf{ctxt} \in \mathfrak{C}$*, and nonces* $\mathsf{nonce}, \mathsf{nonce}' \in \mathfrak{N}$ *such that* (sn, τ) *is valid with respect to* $(\mathsf{pk}_I, \mathsf{ctxt}, \mathsf{nonce})$ *and* (sn, τ') *is valid with respect to* $(\mathsf{pk}_I, \mathsf{ctxt}, \mathsf{nonce}')$*, the deterministic* Identify *algorithm outputs a user public key* pk_U.

In addition, Π_{ARLT} *should satisfy the following properties:*

- ***Correctness****: For any* $\lambda \in \mathbb{N}$*,* $\mathfrak{n} = \mathsf{poly}(\lambda)$*,* $\mathfrak{t} = \mathsf{poly}(\lambda)$*, with* $\mathfrak{t} \geq \mathfrak{n}$*,* $\mathsf{id}_U \in \mathsf{ID}$*, tuple of contexts* $(\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_\mathfrak{t}) \in \mathfrak{C}^\mathfrak{t}$ *such that no context value that appears more than* $\mathfrak{n}$ *times, and tuple of nonces* $(\mathsf{nonce}_1, \ldots, \mathsf{nonce}_\mathfrak{t}) \in \mathfrak{N}^\mathfrak{t}$*,*

$$\Pr\left[\begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t}); \\ (\mathsf{pk}_I, \mathsf{sk}_I, \mathsf{st}_I) \leftarrow \mathsf{IKeyGen}(\mathsf{crs}); \\ (\mathsf{pk}_U, \mathsf{sk}_U) \leftarrow \mathsf{UKeyGen}(\mathsf{crs}, \mathsf{id}_U); \\ (\bot; \mathcal{T}_0) \leftarrow \mathsf{Issue}\left(I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U) \rightleftharpoons U(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I)\right); \\ ((\mathsf{sn}_i, \tau_i), \mathcal{T}_i) \leftarrow \mathsf{Show}(\mathsf{crs}, \mathsf{pk}_I, \mathcal{T}_{i-1}, \mathsf{ctxt}_i, \mathsf{nonce}_i) \forall i \in [\mathfrak{t}] : \\ \exists j \in [\mathfrak{t}], \mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}_j, \tau_j, \mathsf{ctxt}_j, \mathsf{nonce}_j) = 0 \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- ***Anonymity****: It should be hard for a* ppt *adversary that can corrupt issuers and users to learn "meaningful" information about honest users' tokens beyond what is revealed through interaction with those tokens. More formally, for any* ppt *adversary* $\mathcal{A}$ *with oracle access to* $\mathcal{O} = (\mathcal{O}_{\mathsf{NewUser}}, \mathcal{O}_{U_1}, \mathcal{O}_{U_2}, \mathcal{O}_{\mathsf{Show}})$*, there exist an efficient simulator* $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{U_1}, \mathsf{Sim}_{U_2}, \mathsf{Sim}_{\mathsf{Show}})$ *and a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$*,* $\mathfrak{n} = \mathfrak{n}(\lambda)$*,* $\mathfrak{t} = \mathfrak{t}(\lambda)$ *such that* $\mathfrak{t} \geq \mathfrak{n}$*,*

$$\left|\begin{array}{l} \Pr[\mathsf{ARLT\text{-}ANON\text{-}Exp}_0^{\mathcal{A},\mathsf{Sim}}(1^\lambda, \mathfrak{n}, \mathfrak{t}) = 1] \\ \quad - \Pr[\mathsf{ARLT\text{-}ANON\text{-}Exp}_1^{\mathcal{A},\mathsf{Sim}}(1^\lambda, \mathfrak{n}, \mathfrak{t}) = 1] \end{array}\right| \leq \mathsf{negl}(\lambda)$$

where ARLT-ANON-Exp *and* $\mathcal{O}$ *are defined in Fig. 2.* $\mathcal{O}_{\mathsf{NewUser}}$ *allows to register an honest user;* $\mathcal{O}_{U_1}$ *and* $\mathcal{O}_{U_2}$ *allow to issue a token dispenser to an honest user; and* $\mathcal{O}_{\mathsf{Show}}$ *allows to obtain tokens from an honest token dispenser* $\mathcal{T}_d$*, indexed by* $d \in \mathbb{N}$*, with the restriction that at most* $\mathfrak{n}$ *tokens can be request per context and at most* $\mathfrak{t}$ *tokens can be issued per dispenser.*

- ***Linkability****: It should be hard for a* ppt *adversary that can corrupt users to output tokens corresponding to a double-spending such that* Identify *function does not return the identity of one of those corrupted users. More formally, for any* ppt *adversary with oracle access to $\mathcal{O}_I$, there exists a negligible function* negl *such that for any $\lambda \in \mathbb{N}$, $\mathfrak{t} = \mathfrak{t}(\lambda)$, $\mathfrak{n} = \mathfrak{n}(\lambda)$ such that $\mathfrak{t} \geq \mathfrak{n}$,*

$$\Pr\left[\begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t}); \\ (\mathsf{pk}_I, \mathsf{sk}_I, \mathsf{st}_I) \leftarrow \mathsf{IKeyGen}(\mathsf{crs}); \\ (\mathsf{ctxt}, (\mathsf{sn}_i, \tau_i, \mathsf{nonce}_i)_{i\in\{0,1\}}) \leftarrow \mathcal{A}^{\mathcal{O}_I}(1^\lambda, \mathsf{crs}, \mathsf{pk}_I) \\ \mathsf{pk}_U^* \leftarrow \mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}_0, \tau_0, \tau_1, \mathsf{ctxt}, \mathsf{nonce}_0, \mathsf{nonce}_1) : \\ \mathsf{sn}_0 = \mathsf{sn}_1 \wedge \mathsf{nonce}_0 \neq \mathsf{nonce}_1 \wedge \mathsf{pk}_U^* \notin Q \cup \{\perp\} \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Here, $\mathcal{O}_I$ answers to $\mathcal{A}$'s queries on input $(\mathsf{pk}_U, \mathsf{umsg})$ by executing the issuer algorithm $I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U, \mathsf{umsg})$. Moreover, it maintains a set Q containing all user public keys used by $\mathcal{A}$.

- ***Exculpability****: It should be hard for a* ppt *adversary that can corrupt issuers and users to frame an honest user for double-spending. More formally, for any* ppt *adversary $\mathcal{A}$ with oracle access to $\mathcal{O} = (\mathcal{O}_{\mathsf{NewUser}}, \mathcal{O}_{U_1}, \mathcal{O}_{U_2}, \mathcal{O}_{\mathsf{Show}})$ there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any $\lambda \in \mathbb{N}$, $\mathfrak{n} = \mathsf{poly}(\lambda)$, $\mathfrak{t} = \mathsf{poly}(\lambda)$ such that $\mathfrak{t} \geq \mathfrak{n}$,*

$$\Pr[\mathsf{ARLT\text{-}EXCULP\text{-}Exp}^{\mathcal{A}}(1^\lambda, \mathfrak{n}, \mathfrak{t}) = 1] \leq \mathsf{negl}(\lambda)$$

where ARLT-EXCULP-Exp *and $\mathcal{O}$ are defined in Fig. 3. $\mathcal{O}$'s description is similar to the one in the anonymity experiment for the case $b = 0$ (cf. Fig. 2).*

- ***Soundness****: It should be hard for a* ppt *adversary to forge a serial number-token pair. Following Camenisch et al. [CHK+06], we formalize this notion by requiring the existence of an extractor that outputs functions explaining the serial numbers produced by the adversary. More specifically, there exists an extractor* Ext *running in expected polynomial time, such that for any $\lambda \in \mathbb{N}$ and* ppt *adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_I$,* Ext *with black-box rewindable access to $\mathcal{A}$ satisfies the following:*

$$\Pr\left[\begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t}); \\ (\mathsf{pk}_I, \mathsf{sk}_I, \mathsf{st}_I) \leftarrow \mathsf{IKeyGen}(\mathsf{crs}); \\ (\mathsf{ctxt}, \mathsf{sn}, \tau, \mathsf{nonce}) \leftarrow \mathcal{A}^{\mathcal{O}_I}(1^\lambda, \mathsf{crs}, \mathsf{pk}_I); \\ f_1, \ldots, f_{|Q|} \leftarrow \mathsf{Ext}^{\mathcal{A}}(\mathsf{crs}, \mathsf{ctxt}, \mathsf{sn}, \tau, \mathsf{nonce}) : \\ (\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau, \mathsf{ctxt}, \mathsf{nonce}) = 1) \wedge \\ (\forall i \in [|Q|], j \in [\mathfrak{n}], f_i(\mathsf{ctxt}, j) \neq \mathsf{sn}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Here, $\mathcal{O}_I$ answers to $\mathcal{A}$'s queries on input $(\mathsf{pk}_U, \mathsf{umsg})$ by executing the issuer algorithm $I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U, \mathsf{umsg})$. Moreover, it maintains a set Q containing all user public keys used by $\mathcal{A}$. Finally, when Ext *interacts with $\mathcal{A}$, it simulates $\mathcal{O}_I$ for $\mathcal{A}$.*

Remark 4.1 (Total rate limit). It is not always necessary to fix the total rate limit $\mathfrak{t}$ during the execution of Setup. In fact, the construction of Camenisch et al. [CHK+06] based on bilinear pairings does not impose an explicit total rate limit. In that case, correctness can be adapted to hold for any arbitrarily chosen polynomially bounded limit.

Remark 4.2 (Anonymous Counting Tokens). Benhamouda, Raykova, and Seth [BRS23] introduced anonymous counting tokens (ACTs), which allow a user to obtain a fixed number of tokens on a message from an

$\underline{\mathsf{ARLT\text{-}EXCULP\text{-}Exp}^{\mathcal{A}}(1^\lambda, \mathfrak{n}, \mathfrak{t}) :}$

$\mathsf{ctr} \leftarrow 0; D_1, D_2 \leftarrow \emptyset$

initialize $\mathsf{Map} : \mathbb{N} \times \mathfrak{C} \to \mathbb{N}$

$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t})$

$(\mathsf{pk}_I, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathfrak{n}, \mathfrak{t}, \mathsf{crs})$

$(\mathsf{ctxt}, (\mathsf{sn}_i, \tau_i, \mathsf{nonce}_i)_{i\in\mathbb{Z}_2}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{NewUser}}, \mathcal{O}_{U_1}, \mathcal{O}_{U_2}, \mathcal{O}_{\mathsf{Show}}}(\mathsf{st}_{\mathcal{A}})$

if $(\mathsf{sn}_0 \neq \mathsf{sn}_1) \vee (\mathsf{nonce}_0 = \mathsf{nonce}_1)$: **abort**

$\mathsf{pk}' \leftarrow \mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}_0, \tau_0, \tau_1, \mathsf{ctxt}, \mathsf{nonce}_0, \mathsf{nonce}_1)$

return $(\exists j \in [\mathsf{ctr}] : \mathsf{pk}' = \mathsf{pk}_j)$

$\wedge\ (\forall i \in \mathbb{Z}_2 : \mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}_0, \tau_i, \mathsf{ctxt}, \mathsf{nonce}_i) = 1)$

$\underline{\mathcal{O}_{\mathsf{NewUser}}(\mathsf{id}) :}$

$(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{UKeyGen}(\mathsf{crs}, \mathsf{id})$

$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$

return $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{ctr})$

$\underline{\mathcal{O}_{U_1}(u, d) :}$

if $u > \mathsf{ctr} \vee d \in D_1$: **abort**

$(\mathsf{st}_{u,d}, \mathsf{umsg}) \leftarrow U_1(\mathsf{crs}, \mathsf{sk}_u, \mathsf{pk}_I)$

$D_1 \leftarrow D_1 \cup \{d\}$

return umsg

$\underline{\mathcal{O}_{\mathsf{Show}}(d, \mathsf{ctxt}, \mathsf{nonce}) :}$

if $(d \notin D_2) \vee \left(\sum_{\mathsf{ctxt}} \mathsf{Map}[d, \mathsf{ctxt}] \geq \mathfrak{t}\right)$

$\quad \vee\ (\mathsf{Map}[d, \mathsf{ctxt}] \geq \mathfrak{n})$: **abort**

$((\mathsf{sn}, \tau), \mathcal{T}'_d) \leftarrow \mathsf{Show}(\mathsf{crs}, \mathsf{pk}_I, \mathcal{T}_d, \mathsf{ctxt}, \mathsf{nonce})$

$\mathcal{T}_d \leftarrow \mathcal{T}'_d$

$\mathsf{Map}[d, \mathsf{ctxt}] \leftarrow \mathsf{Map}[d, \mathsf{ctxt}] + 1$

return (sn, τ)

$\underline{\mathcal{O}_{U_2}(d, \mathsf{imsg}) :}$

if $d \in D_2 \vee d \notin D_1$: **abort**

$\mathcal{T}_d \leftarrow U_2(\mathsf{st}_{u,d}, \mathsf{imsg})$

if $\mathcal{T}_d = \perp$: **abort**

$D_2 \leftarrow D_2 \cup \{d\}$

return ack

Figure 3: Π_{ARLT}'s exculpability Experiment

issuer without revealing the message. Later, the user can redeem these tokens while preventing the issuer from linking redemption to the issuance session. Their definition is closely related to blind signatures.

We observe that ACTs can be obtained from ARLTs by treating the message as the context and making the issuer stateful. In contrast to ARLTs, where a user can always obtain fresh tokens for a given context by generating a new wallet, in ACTs, wallets are bound to a specific context and cannot be *replenished*. To enforce this, we follow the approach of Benhamouda, Raykova, and Seth and incorporate a PRF evaluation of the message (context) into a token request (user's first message during the execution of the Issue protocol). This allow the issuer to reject any issuance request that contains a previously seen PRF value. We provide a complete definition of ACTs in Section A.1.

5 Partially-Binding Vector Commitment From Lattices

We recall the definition of partially-binding vector commitment based on the works of Goel et al. [GGHK22], and Avitabile et al. [ABFV22].

Definition 5.1 (1-out-of-ℓ Binding Vector Commitment)**.** *A 1-out-of-ℓ binding vector commitment for a message space $\mathcal{M}^\ell$ is a tuple of* ppt *algorithms $\Pi_{\mathsf{PBVC}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Equiv})$ with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: Given a security parameter $\lambda \in \mathbb{N}$, the setup algorithm outputs the public parameters* pp.

- $\mathsf{Gen}(\mathsf{pp}, i) \to (\mathsf{ck}, \mathsf{ek})$*: Given* pp *and an index* $i \in \mathbb{Z}_\ell$*, the* Gen *algorithm outputs a commitment key* ck *and an equivocation key* ek*.*

- $\mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}) \to (\mathsf{cmt}, o)$*: Given* pp*,* ck*, and a vector* $\mathbf{v} \in \mathcal{M}^\ell$*, the* Commit *algorithm outputs a commitment* cmt *and an opening* o*.*

- $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}, o) \to 1/0$*: Given* pp*,* ck*, a commitment* cmt*, a vector* $\mathbf{v} \in \mathcal{M}^\ell$*, and an opening* o*, the deterministic* Open *algorithm outputs* 1 *if* o *is a valid opening for the vector* $\mathbf{v}$ *with respect to the commitment* cmt*.*

- $\mathsf{Equiv}(\mathsf{pp}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}') \to o'$*: Given* pp*,* ek*, a vector* $\mathbf{v} \in \mathcal{M}^\ell$ *with associated commitment* cmt *and opening* o*, a vector* $\mathbf{v}'$ *such that* $\mathbf{v}[i] = \mathbf{v}'[i]$*, where* $i \in \mathbb{Z}_\ell$ *is the index that was used to generated* ek*, the* Equiv *algorithm outputs an opening* o' *such that* $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}', o') = 1$*.*

In addition, Π_{PBVC} *should satisfy the following properties:*

- ***ϵ-Partial Equivocation****: Intuitively, given a vector* $\mathbf{v} \in \mathcal{M}^\ell$ *with associated commitment* cmt *and opening* o*, and any vector* $\mathbf{v}' \in \mathcal{M}^\ell$ *such that* $\mathbf{v}[i] = \mathbf{v}'[i]$*, where* i *is the index on which* Π_{PBVC} *is binding, the* Equiv *algorithm outputs an opening* o' *attesting that* cmt *is a valid commitment to* $\mathbf{v}'$ *with probability* $1 - \epsilon(\lambda)$ *for some small* ϵ*. If* ϵ *is non-negligible, then this notion of equivocation is relaxed compared to equivocable commitments defined in prior work [GGHK22]; but our definition is more general because it allows for any* ϵ *that is noticeably smaller than* 1*.*

  Formally, for $\epsilon(\lambda) < 1 - \frac{1}{\lambda^c}$ *for some constant* c*, a commitment scheme is* ϵ*-partially equivocable if for any* $\lambda \in \mathbb{N}$*,* $i \in \mathbb{Z}_\ell$*, and* $\mathbf{v}, \mathbf{v}' \in \mathcal{M}^\ell$ *such that* $\mathbf{v}[i] = \mathbf{v}'[i]$*,*

$$\Pr\left[\begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(\mathsf{pp}, i); \\ (\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}); \\ o' \leftarrow \mathsf{Equiv}(\mathsf{pp}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}') : \\ \mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}', o') \neq 1 \end{array}\right] \leq \epsilon(\lambda)$$

- ***Partial Binding****: Intuitively, there must exist an index* i *(this can be generalized to* k *indices,* $i_1, \ldots, i_k$*) on which the adversary should not be able to equivocate (except with negligible probability). This index is fixed at the time when the commitment key* ck *is chosen.*

  The security game requires that for every index i*, the adversary produce a commitment* cmt_i *and a way to open it to two vectors* $\mathbf{v}_i$ *and* $\mathbf{v}'_i$ *that differ in position* i *— thereby showing that no position* i *is protected from equivocation. More formally, for any* ppt *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); \\ \left(\mathsf{ck}, \{(\mathsf{cmt}_i, \mathbf{v}_i, o_i, \mathbf{v}'_i, o'_i)\}_{i \in \mathbb{Z}_\ell}\right) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp}) : \\ \forall i \in \mathbb{Z}_\ell, \left(\mathbf{v}_i[i] \neq \mathbf{v}'_i[i]\right) \\ \wedge \left(\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}_i, \mathbf{v}_i, o_i) = \mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}_i, \mathbf{v}'_i, o'_i) = 1\right) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- ***Hiding****: The notion of hiding is stronger than that of a regular commitment. In addition to requiring that the commitment* cmt *hide its committed value, a commitment key* ck *should also hide its binding index, and opening values produced by the* Commit *and* Equiv *algorithms should be indistinguishable. Formally, the following properties should hold:*

$\mathsf{PBVC\text{-}Hiding\text{-}Exp}^{\mathcal{A}}(1^\lambda, 0)$:	$\mathsf{PBVC\text{-}Hiding\text{-}Exp}^{\mathcal{A}}(1^\lambda, 1)$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
$(i, j, \mathbf{v}, \mathbf{v}', \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$	$(i, j, \mathbf{v}, \mathbf{v}', \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pp})$
if $\mathbf{v}[i] \neq \mathbf{v}'[i]$: **abort**	**if** $\mathbf{v}[i] \neq \mathbf{v}'[i]$: **abort**
$(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)$;	$(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(\mathsf{pp}, j)$;
$(\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v})$;	$(\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}')$
$o' \leftarrow \mathsf{Equiv}(\mathsf{pp}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}')$;	**return** $\mathcal{A}(\mathsf{st}_{\mathcal{A}}, \mathsf{ck}, \mathsf{cmt}, (o, 1))$
if $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}', o') \neq 1$:	
return $\mathcal{A}(\mathsf{st}_{\mathcal{A}}, \mathsf{ck}, \mathsf{cmt}, (o', 0))$	
return $\mathcal{A}(\mathsf{st}_{\mathcal{A}}, \mathsf{ck}, \mathsf{cmt}, (o', 1))$	

Figure 4: PBVC's Hiding Experiment

- *For any* $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $i \in \mathbb{Z}_\ell$, $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)$, *and* $\mathbf{v}, \mathbf{v}' \in \mathcal{M}^\ell$, $\{\mathsf{cmt} : (\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v})\} \overset{c}{\approx} \{\mathsf{cmt} : (\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}')\}$.
- *Let* F *be the event that* Equiv *outputs a faulty opening. Then, for any* ppt *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$*,*

$$\left| \begin{array}{l} \Pr[(\mathsf{PBVC\text{-}Hiding\text{-}Exp}^{\mathcal{A}}(1^\lambda, 0) = 1) \wedge \neg \mathsf{F}] \\ \quad - \Pr[(\mathsf{PBVC\text{-}Hiding\text{-}Exp}^{\mathcal{A}}(1^\lambda, 1) = 1) \wedge \neg \mathsf{F}] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

Where PBVC-Hiding-Exp *is defined in Fig. 4.*

Remark 5.1 (On the failure event in Hiding). Note that our formal definition of hiding deviates slightly from standard definitions in the cryptographic literature, since it explicitly exposes the failure event. This is necessary because our ϵ-partial equivocation definition allows ϵ, the failure probability of equivocation, to be noticeable. Thus, this failure event cannot be hidden inside a negligible error term. However, this is not an issue for the application we have in mind: upon failure of Equiv, the process can be restarted, ensuring that only commitments and openings obtained from successful equivocations are output. This is analogous to the case of non-aborting honest-verifier zero-knowledge [Kat21].

Definition 5.2 (Stackable Σ-protocol [GGHK22], adapted)**.** *A* Σ*-protocol* $\Pi = (\mathsf{Setup}, (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V})$ *with challenge distribution* $\mathcal{C}$ *for an NP relation* $\mathcal{R}$ *is said to be stackable if it satisfies the following properties:*

- ***Extended HVZK****: There exists a deterministic simulator* Sim *and an efficiently sampleable distribution* $\mathcal{Z}$ *such that for any* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$*,* $\lambda \in \mathbb{N}$*, and* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*, the following distributions are computationally indistinguishable:*

$$\left\{ (a, c, z) : \begin{array}{c} (a, \mathsf{st}_{\mathsf{P}}) \leftarrow \mathsf{P}_1(\mathsf{crs}, \mathbb{x}, \mathbb{w}), \\ c \leftarrow \mathcal{C}, z \leftarrow \mathsf{P}_2(\mathsf{st}_{\mathsf{P}}, a, c) \end{array} \right\} \overset{c}{\approx} \left\{ (a, c, z) : \begin{array}{c} c \leftarrow \mathcal{C}, z \leftarrow \mathcal{Z}(\mathbb{x}, c) \\ a \leftarrow \mathsf{Sim}(\mathsf{crs}, \mathbb{x}, c, z) \end{array} \right\}$$

- ***Recycable third message****: Intuitively, for any valid proof* $\pi := (a, c, z)$ *generated for a valid statement* $\mathbb{x}$*, the third message* z *should reveal no information about* $\mathbb{x}$*. More specifically, there exists an*

efficiently samplable $\mathcal{Z}$ such that for any $(\mathbb{x}_1, \mathbb{w}_1), (\mathbb{x}_2, \mathbb{w}_2) \in \mathcal{R}$, $\lambda \in \mathbb{N}$, and $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, the following distributions are computationally indistinguishable:

$$\{z : c \leftarrow \mathcal{C}, z \leftarrow \mathcal{Z}(\mathbb{x}_1, c)\} \overset{c}{\approx} \left\{ z : \begin{array}{c} (a, \mathsf{st}_\mathsf{P}) \leftarrow \mathsf{P}_1(\mathsf{crs}, \mathbb{x}_2, \mathbb{w}_2), \\ c \leftarrow \mathcal{C}, z \leftarrow \mathsf{P}_2(\mathsf{st}_\mathsf{P}, a, c) \end{array} \right\}$$

GGHK showed that, if the Σ-protocol satisfies Definition 5.2, then a proof for the disjunction of statements can be compressed, or "stacked". The first message of the stacked protocol will be a commitment cmt (a short one) to a vector $\mathbf{v}$ that only binds the prover to the first message a_i for the Σ-protocol for the statement $\mathbb{x}_i$ for which the prover knows the witness. The second message is the challenge c. The third message z is the same for each instance of the Σ-protocol because of third-message recyclability property. To verify, the first messages $a_1, \ldots a_\ell$ are recovered, using the fact that the simulator is deterministic, from the instances $\mathbb{x}_1, \ldots, \mathbb{x}_\ell$, the challenge c and the third message z. The prover must supply the verifier with the opening o' that will attest to the fact that cmt has indeed bound the prover to a_i, for some i. More formally:

Compiler 5.1 (Self-Stacking Compiler [GGHK22])**.** Let $\Pi = (\mathsf{Setup}, (\mathsf{P}_1, \mathsf{P}_2), V)$ be a stackable Σ-protocol with simulator Sim (see Definition 5.2) for an NP relation $\mathcal{R}$ with associated language $\mathcal{L}_\mathcal{R}$ and $\Pi_{\mathsf{PBVC}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Equiv})$ be a 1-out-of-ℓ partial binding vector commitment (see Definition 5.1). GGHK's compiler to obtain a Σ-protocol $\Pi' = (\mathsf{Setup}, (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V})$ for the disjunctive NP relation $\mathcal{R}' := \vee_{i \in [\ell]} \mathcal{R}$ works as follows:

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$, compute $\mathsf{crs}_\Pi \leftarrow \Pi.\mathsf{Setup}(1^\lambda)$ and $\mathsf{pp}_{\Pi_{\mathsf{PBVC}}} \leftarrow \Pi_{\mathsf{PBVC}}.\mathsf{Setup}(1^\lambda)$. Return $\mathsf{crs} := \mathsf{crs}_\Pi \| \mathsf{pp}_{\Pi_{\mathsf{PBVC}}}$.
- $\mathsf{P}_1(\mathsf{crs}, \tilde{\mathbb{x}} := (\mathbb{x}_0, \ldots, \mathbb{x}_{\ell-1}), \tilde{\mathbb{w}} := (i, \mathbb{w}_i))$: Given $\mathsf{crs}_\Pi \| \mathsf{pp}_{\Pi_{\mathsf{PBVC}}} := \mathsf{crs}$, a statement $(\mathbb{x}_0, \ldots, \mathbb{x}_{\ell-1}) := \tilde{\mathbb{x}} \in \mathcal{L}_R^\ell$, and witness $(i, \mathbb{w}_i) := \tilde{\mathbb{w}}$ such that $i \in \mathbb{Z}_\ell$ and $(\mathbb{x}_i, \mathbb{w}_i) \in R$, proceed as follows:
  1. Compute $(\mathsf{ck}, \mathsf{ek}) \leftarrow \Pi_{\mathsf{PBVC}}.\mathsf{Gen}(\mathsf{pp}_{\Pi_{\mathsf{PBVC}}}, i)$.
  2. Compute $(a_i, \mathsf{st}_\mathsf{P}) \leftarrow \Pi.\mathsf{P}_1(\mathsf{crs}_\Pi, \mathbb{x}_i, \mathbb{w}_i)$.
  3. Set $\mathbf{v} := (\perp^{(0)}, \ldots, \perp^{(i-1)}, a_i, \perp^{(i+1)}, \ldots, \perp^{(\ell-1)})$.
  4. Compute $(\mathsf{cmt}, o) \leftarrow \Pi_{\mathsf{PBVC}}.\mathsf{Commit}(\mathsf{pp}_{\Pi_{\mathsf{PBVC}}}, \mathsf{ck}, \mathbf{v})$.
  5. Return $\tilde{a} := (\mathsf{ck}, \mathsf{cmt})$, and $\tilde{\mathsf{st}_\mathsf{P}} := (\mathsf{st}_\mathsf{P}, \mathsf{crs}, \mathsf{ek}, a_i, i, o)$.
- $\mathsf{P}_2(\tilde{\mathsf{st}_\mathsf{P}}, \tilde{a}, c)$: Given a prover state $(\mathsf{st}_\mathsf{P}, \mathsf{crs}, \mathsf{ek}, a_i, i, o) := \tilde{\mathsf{st}_\mathsf{P}}$, a message $(\mathsf{ck}, \mathsf{cmt}) := \tilde{a}$, and a challenge c, proceed as follows:
  1. Compute $z \leftarrow \Pi.\mathsf{P}_2(\mathsf{st}_\mathsf{P}, a_i, c)$.
  2. For $j \in \{0, \ldots, \ell - 1\} \setminus \{i\}$ compute $a'_j \leftarrow \mathsf{Sim}(\mathsf{crs}_\Pi, \mathbb{x}_j, c, z)$.
  3. Set $\mathbf{v}' := (a'_0, \ldots, a'_{i-1}, a_i, a'_{i-1}, \ldots, a'_{\ell-1})$. (Recall that $\mathbf{v} := (\perp^{(0)}, \ldots, \perp^{(i-1)}, a_i, \perp^{(i+1)}, \ldots, \perp^{(\ell-1)})$.)
  4. Compute $o' \leftarrow \Pi_{\mathsf{PBVC}}.\mathsf{Equiv}(\mathsf{pp}_{\Pi_{\mathsf{PBVC}}}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}')$.
  5. Return $\tilde{z} := (z, o')$.
- $\mathsf{V}(\mathsf{crs}, \tilde{\mathbb{x}}, \tilde{\pi} := (\tilde{a}, c, \tilde{z}))$: Given $\mathsf{crs}_\Pi \| \mathsf{pp}_{\Pi_{\mathsf{PBVC}}} := \mathsf{crs}$, a statement $(\mathbb{x}_0, \ldots, \mathbb{x}_{\ell-1}) := \tilde{\mathbb{x}} \in \mathcal{L}_R^\ell$, and a proof $(\tilde{a}, c, \tilde{z}) := \tilde{\pi}$, proceed as follows:
  1. Parse $(\mathsf{ck}, \mathsf{cmt}) := \tilde{a}$ and $(z, o) := \tilde{z}$.
  2. For $i \in \{0, \ldots, \ell - 1\}$, compute $a_i \leftarrow \mathsf{Sim}(\mathsf{crs}_\Pi, \mathbb{x}_i, c, z)$.

3. Set $\mathbf{v} := (a_0, \ldots, a_{\ell-1})$.
4. Return $(\Pi_{\mathsf{PBVC}}.\mathsf{Open}(\mathsf{pp}_{\Pi_{\mathsf{PBVC}}}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}, o) = 1) \wedge (\bigwedge_{i \in \{0,\ldots,\ell-1\}} \Pi.\mathsf{V}(\mathsf{crs}_{\Pi}, \mathbb{x}_i, \pi_i := (a_i, c, z)) = 1)$.

Theorem 5.1 (Self-stacking [GGHK22], reworded)**.** *Let $\Pi = (\mathsf{Setup}, (\mathsf{P}_1, \mathsf{P}_2), V)$ be a stackable Σ-protocol (see Definition 5.2) for an NP relation $\mathcal{R}$, and $\Pi_{\mathsf{PBVC}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Equiv})$ be a 1-out-of-ℓ partial binding vector commitment (see Definition 5.1). The Σ-protocol $\Pi' = (\mathsf{Setup}, (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V})$ for the disjunctive NP relation $\mathcal{R}' := \bigvee_{i \in [\ell]} \mathcal{R}$ obtained by applying GGHK's compiler (see Compiler 5.1) is HVZK, and $\mathsf{CC}(\Pi') = \mathsf{CC}(\Pi) + |\mathsf{ck}| + |\mathsf{cmt}| + |o|$, where $|\mathsf{ck}|, |\mathsf{cmt}|$, and $|o|$ denote the sizes of commitment keys, commitments, and openings generated by Π_{PBVC}, respectively. Furthermore, if Π is special-sound, then so is Π'.*

Remark 5.2. In Theorem 5.1, GGHK considered a 1-out-of-ℓ PBVC satisfying ϵ-Partial equivocation, where ϵ is negligible in the security parameter. In our setting, however, ϵ is noticeably smaller than 1 but not negligible, and we show that this level of partial equivocation still suffices.

5.1 Construction

Our starting point is the 1-out-of-2 PBVC construction of Goel et al. [GGHK22], which is an adaptation of Pedersen Commitment [Ped91]. In their construction, a committer holding a bit $b \in \{0, 1\}$, to which it wishes to be bound, is given random group elements $\mathsf{pp} := (h, f) \in \mathbb{G}^2$, where $\mathbb{G}$ is a group in which the discrete logarithm assumption is *hard*, and then generates a commitment key $\mathsf{ck} := (g_0, g_1) \in \mathbb{G}^2$ such that $g_0 g_1 = f$ and $g_{1-b} = h^t$ for some random $t \in \mathbb{Z}_{|\mathbb{G}|}$. To commit to a vector $\mathbf{v} := (v_0, v_1) \in \mathbb{Z}^2_{|\mathbb{G}|}$, the committer samples $(r_0, r_1) \leftarrow\!\!\$\ \mathbb{Z}^2_{|\mathbb{G}|}$ and outputs $\mathsf{cmt}_{\mathbf{v}} := (h^{r_0} g_0^{v_0}, h^{r_1} g_1^{v_1})$. Given knowledge of t, it is easy to see that the committer can equivocate $\mathsf{cmt}_{\mathbf{v}}$ to any vector $\mathbf{v}' := (v'_0, v'_1)$ such that $v_b = v'_b$. However, equivocating on both positions will lead to a recovery of $\mathsf{dlog}_h g_0$ and $\mathsf{dlog}_h g_1$, which is sufficient to solve $\mathsf{dlog}_h f$.

One could attempt to port this construction to the lattice setting by replacing Pedersen Commitment with Ajtai Commitment [Ajt96, GGH96] as follows: (1) set $\mathsf{pp} := [\mathbf{A} \mid \mathbf{B}]$, where $[\mathbf{A} \mid \mathbf{B}]$ is a random matrix $[\mathbf{A} \mid \mathbf{B}]$ for which the M-SIS assumption is *hard*, (2) set $\mathsf{ck} := (\mathbf{D}_0, \mathbf{D}_1)$ such that $\mathbf{D}_0 + \mathbf{D}_1 = \mathbf{B}$ and $\mathbf{D}_{1-b} = \mathbf{AT}$, where $\mathbf{T}$ is a high-entropy low-norm matrix, and (3) commit to a low-norm vector $\mathbf{v} := (\mathbf{v}_0, \mathbf{v}_1)$ by sampling high-entropy low-norm vectors $\mathbf{r}_0, \mathbf{r}_1$ and computing $\mathsf{cmt} := (\mathbf{A}\mathbf{r}_0 + \mathbf{D}_0\mathbf{v}_0, \mathbf{A}\mathbf{r}_1 + \mathbf{D}_1\mathbf{v}_1)$. Clearly, with $\mathbf{T}$, index $1 - b$ is equivocable. However, although an equivocation on both positions allows to recover an M-SIS solution for matrices $[\mathbf{A} \mid \mathbf{D}_0]$ and $[\mathbf{A} \mid \mathbf{D}_1]$, it is not enough to recover an M-SIS solution for the matrix $[\mathbf{A} \mid \mathbf{B}]$ since $\mathbf{D}_0$ and $\mathbf{D}_1$ are generated by the committer. To patch this issue, we require that the committer includes a proof of knowledge π_{ck} for the statement "$\exists \mathbf{T} : (\mathbf{D}_0 = \mathbf{AT} \vee \mathbf{D}_1 = \mathbf{AT}) \wedge \|\mathbf{T}\|_\infty \leq \beta$", where β is an a priori norm-bound on entries of $\mathbf{T}$, to the commitment key. Now, using π_{ck}, we can *extract* $\mathbf{T}$ and convert M-SIS solutions for $[\mathbf{A} \mid \mathbf{D}_0]$ and $[\mathbf{A} \mid \mathbf{D}_1]$ into an M-SIS solution for $[\mathbf{A} \mid \mathbf{B}]$. In addition, thanks to π_{ck} we can reduce the size of the randomness needed to compute cmt to one vector instead of two, i.e., $\mathsf{cmt} := \mathbf{Ar} + \mathbf{D}_0\mathbf{v}_0 + \mathbf{D}_1\mathbf{v}_1$.

Let $\mathcal{S} := \{\bar{c} : \exists c_1, c_2 \in \mathcal{B}_1, \bar{c} = c_1 - c_2\} \setminus \{0\}$ be a slack space. Let $\Pi^{\mathsf{OR}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ be a relaxed knowledge-sound (see remark 3.2) Σ-protocol for the relation $\mathcal{R}_{\mathsf{OR}} := \{\mathbb{x} := (\mathbf{A}, \mathbf{D}_0, \mathbf{D}_1, \beta), \mathbb{w} := \mathbf{T} : (\mathbf{D}_0 = \mathbf{AT} \vee \mathbf{D}_1 = \mathbf{AT}) \wedge \|\mathbf{T}\|_\infty \leq \beta\}$ that admits a knowledge extractor Ext with knowledge error κ_{OR} for the corresponding relaxed relation $\tilde{\mathcal{R}}_{\mathsf{OR}} := \{\mathbb{x} := (\mathbf{A}, \mathbf{D}_0, \mathbf{D}_1, \beta'), \mathbb{w} := (\bar{c}, \mathbf{T}) : \bar{c} \in \mathcal{S} \wedge (\bar{c}\mathbf{D}_0 = \mathbf{AT} \vee \bar{c}\mathbf{D}_1 = \mathbf{AT}) \wedge \|\mathbf{T}\|_\infty \leq \beta'\}$ (see Section 5.2 for a concrete instantiation). We give the full description of our 1-out-of-2 PBVC, using $\Pi^{\mathcal{R}_{\mathsf{OR}}}_{\mathsf{NIZK}}$, below.

Construction 5.1 (1-out-of-2 Binding Commitment)**.** Let $\mathsf{RejSamp}$ be the algorithm defined in Lemma 2.4. For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be an integer, $\mathfrak{b} \geq 2$, and $n = n(\lambda)$, $m = n\lceil \log_{\mathfrak{b}} q \rceil$, $d = d(\lambda)$ be lattice

parameters. Let $\mathfrak{s} = \mathfrak{s}(\lambda)$ be a Gaussian width parameter, $\beta = \beta(\lambda)$ be a norm bounds, and $M = M(\lambda)$ be a rejection constant. Let $H : 0,1^* \to \{0,1\}^{\leq p(\lambda)}$ be a random oracle. We construct a 1-out-of-2 binding commitment for the message space $\mathcal{M}^2 = (R_q^n)^2$ as follows:

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$, sample $\bar{\mathbf{A}} \leftarrow\$ R_q^{n\times(m-n)}$, $\mathbf{B} \leftarrow\$ R_q^{n\times m}$. Set $\mathbf{A} \leftarrow [\mathbf{I}_n \mid \bar{\mathbf{A}}]$. Compute $\mathsf{crs}_{\mathsf{OR}} \leftarrow \Pi^{\mathcal{R}_{\mathsf{OR}}}_{\mathsf{NIZK}}.\mathsf{Setup}(1^\lambda)$. Return $\mathsf{pp} := (\mathbf{A}, \mathbf{B}, \mathsf{crs}_{\mathsf{OR}})$.
- $\mathsf{Gen}(\mathsf{pp}, b)$: Given $(\mathbf{A}, \mathbf{B}, \mathsf{crs}_{\mathsf{OR}}) := \mathsf{pp}$ and $b \in \{0,1\}$, proceed as follows:
  1. Sample $\mathbf{T} \leftarrow (S_1^\infty)^{m\times m}$.
  2. Compute $\mathbf{D}_{1-b} \leftarrow \mathbf{A}\cdot\mathbf{T}$ and $\mathbf{D}_b \leftarrow \mathbf{B} - \mathbf{D}_{1-b}$.
  3. Compute $\pi_{\mathsf{ck}} \leftarrow \Pi^{\mathcal{R}_{\mathsf{OR}}}_{\mathsf{NIZK}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{OR}}, \mathbb{x} := (\mathbf{A}, \mathbf{D}_0, \mathbf{D}_1, 1), \mathbb{w} := \mathbf{T})$.
  4. Return $\mathsf{ck} := (\mathbf{D}_0, \pi_{\mathsf{ck}})$ and $\mathsf{ek} := (\mathbf{D}_0, \pi_{\mathsf{ck}}, \mathbf{T}, b)$.
- $\mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v})$: Given $(\mathbf{A}, \mathbf{B}, \mathsf{crs}_{\mathsf{OR}}) := \mathsf{pp}$, $(\mathbf{D}_0, \pi_{\mathsf{ck}}) := \mathsf{ck}$, and $\mathbf{v} \in \mathcal{M}^2$, proceed as follows:
  1. Parse $(\mathbf{v}_0, \mathbf{v}_1) := \mathbf{v}$, and compute $\mathbf{D}_1 \leftarrow \mathbf{B} - \mathbf{D}_0$.
  2. If $\Pi^{\mathcal{R}_{\mathsf{OR}}}_{\mathsf{NIZK}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{OR}}, \mathbb{x} := (\mathbf{A}, \mathbf{D}_0, \mathbf{D}_1, 1), \pi_{\mathsf{ck}}) = 0$, **abort**.
  3. Sample $\mathbf{r} \leftarrow D^m_{R,\mathfrak{s}}$.
  4. Compute $\mathbf{t} \leftarrow \mathbf{A}\mathbf{r} + \mathbf{D}_0 \cdot \mathbf{G}^{-1}(\mathbf{v}_0) + \mathbf{D}_1 \cdot \mathbf{G}^{-1}(\mathbf{v}_1)$.
  5. Return $\mathsf{cmt} := \mathbf{t}$ and $o := \mathbf{r}$.
- $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}, o)$: Given $(\mathbf{A}, \mathbf{B}, \mathsf{crs}_{\mathsf{OR}}) := \mathsf{pp}$, $(\mathbf{D}_0, \pi_{\mathsf{ck}}) := \mathsf{ck}$, $\mathbf{t} := \mathsf{cmt}$, $\mathbf{v} \in \mathcal{M}^2$, and $\mathbf{r} := o$, proceed as follows:
  1. $\mathbf{t}' \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}; \mathbf{r})$.
  2. Return $(\mathbf{t}' = \mathbf{t}) \wedge (\|\mathbf{r}\| \leq \beta)$.
- $\mathsf{Equiv}(\mathsf{pp}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}')$: Given $(\mathbf{A}, \mathbf{B}, \mathsf{crs}_{\mathsf{OR}}) := \mathsf{pp}$, $(\mathbf{D}_0, \pi_{\mathsf{ck}}, \mathbf{T}, b) := \mathsf{ek}$, $\mathbf{v} \in \mathcal{M}^2$, $\mathbf{t} := \mathsf{cmt}$, $\mathbf{r} := o$, and $\mathbf{v}' \in \mathcal{M}^2$, proceed as follows:
  1. If $\mathsf{Open}(\mathsf{pp}, \mathsf{ck} := (\mathbf{D}_0, \pi_{\mathsf{ck}}), \mathsf{cmt}, \mathbf{v}, o) = 0$ or $\mathbf{v}_b \neq \mathbf{v}'_b$, **abort**.
  2. Compute $\mathbf{D}_1 \leftarrow \mathbf{B} - \mathbf{D}_0$.
  3. Compute $\mathbf{d} \leftarrow \mathbf{T} \cdot (\mathbf{G}^{-1}(\mathbf{v}_{1-b}) - \mathbf{G}^{-1}(\mathbf{v}'_{1-b}))$ and $\mathbf{r}' \leftarrow \mathbf{r} + \mathbf{d}$.
  4. If $\mathsf{RejSamp}(\mathbf{r}', \mathbf{d}, \mathfrak{s}, M) = 0$, Output $o' := \perp$. Otherwise, output $o' := \mathbf{r}'$.

Theorem 5.2 (Partial Binding)**.** *For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be a prime, $\mathfrak{b} \geq 2$, and $n = n(\lambda)$, $m = n\lceil \log_{\mathfrak{b}} q \rceil$, $d = d(\lambda)$ be lattice parameters. Let $\varepsilon = \mathsf{negl}(\lambda)$, $\mathfrak{s} \geq \eta_\varepsilon(R^m)$ and $\beta = \mathfrak{s}\sqrt{md}$. Let β' be a bound on $\Pi^{\mathcal{R}_{\mathsf{OR}}}_{\mathsf{NIZK}}$ relaxed witness and κ_{OR} be the knowledge error of $\Pi^{\mathcal{R}_{\mathsf{OR}}}_{\mathsf{NIZK}}$'s extractor. Let $\beta'' = d\sqrt{4(\beta + \beta' m(\mathfrak{b}-1)\sqrt{md})^2 + (\mathfrak{b}-1)^2 md}$. Assume $(\mathfrak{b}-1)d < q/2$ and that the normal-form $\mathsf{M\text{-}SIS}_{n,2m-n,q,\beta''}$ assumption holds. Then, Construction 5.1 is partial binding (see Definition 5.1).*

Proof. We proceed by contraposition. Let $\mathcal{A}$ be a ppt adversary that breaks the partial binding property of Construction 5.1 with non-negligible success probability $\epsilon_{\mathcal{A}} > \kappa_{\mathsf{OR}}$. We construct a ppt adversary $\mathcal{B}$ that breaks the normal-form $\mathsf{M\text{-}SIS}_{n,2m-n,q,\beta''}$ success with probability at least $\epsilon_{\mathcal{A}} - \kappa_{\mathsf{OR}}$, as follows:

$\mathcal{B}$ plays the normal-form $\mathsf{M\text{-}SIS}_{n,2m-n,q,\beta''}$ experiment:

- $\mathcal{B}$ samples $[\bar{\mathbf{A}} \mid \mathbf{B}] \hookleftarrow\$ R_q^{n\times(2m-n)}$, where $\bar{\mathbf{A}} \in R_q^{n\times(m-n)}$ and $\mathbf{B} \in R_q^{n\times m}$. Then, $\mathcal{B}$ computes $\mathsf{crs}_{\mathsf{OR}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}.\mathsf{Setup}(1^\lambda)$. Finally, it sends $\mathsf{pp} := ([\mathbf{I}_n \mid \bar{\mathbf{A}}], \mathbf{B}, \mathsf{crs}_{\mathsf{OR}})$ to $\mathcal{A}$.
- Once $\mathcal{B}$ receives $(\mathsf{ck}, \{\mathsf{cmt}_i, \mathbf{v}_i, o_i, \mathbf{v}'_i, o'_i\}_{i\in\{0,1\}})$ from $\mathcal{A}$ such that $\mathbf{v}_i[i] \neq \mathbf{v}'_i[i]$ and $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}_i, \mathbf{v}_i, o_i) = \mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}_i, \mathbf{v}'_i, o'_i) = 1$, it proceeds as follows:
  - Parse $(\mathbf{D}_0, \pi_{\mathsf{ck}}) := \mathsf{ck}$, and compute $\mathbf{D}_1 \leftarrow \mathbf{B} - \mathbf{D}_0$. Then, run $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}$'s extractor on $\mathcal{A}$ with $(\mathsf{crs}, \mathbb{x} := ([\mathbf{I}_n \mid \bar{\mathbf{A}}], \mathbf{D}_0, \mathbf{D}_1))$ as input to obtain the relaxed witness $(\bar{c}, \tilde{\mathbf{T}})$ such that $\bar{c} \in \mathcal{S}$ and $||\tilde{\mathbf{T}}||_\infty \leq \beta'$. Without loss of generality, suppose $j = 1$ was the position on which $\mathcal{A}$ was bound, i.e., $\bar{c} \cdot \mathbf{D}_0 = \mathbf{A} \cdot \tilde{\mathbf{T}}$.
  - Given that $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}_1, \mathbf{v}_1, o_1) = \mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}_1, \mathbf{v}'_1, o'_1) = 1$ with $\mathbf{r}_1 := o_1$, $\mathbf{r}'_1 := o'_1$, and $\mathbf{v}_1[1] \neq \mathbf{v}'_1[1]$, we have

$$[\mathbf{I}_n \mid \bar{\mathbf{A}}]\bar{\mathbf{r}} + \mathbf{D}_0\bar{\mathbf{v}}_{1,0} + \mathbf{D}_1\bar{\mathbf{v}}_{1,1} = \mathbf{0}^n \tag{5.1}$$

where $\bar{\mathbf{r}} = \mathbf{r}_1 - \mathbf{r}'_1$, $\bar{\mathbf{v}}_{1,0} = \mathbf{G}^{-1}(\mathbf{v}_{1,0}) - \mathbf{G}^{-1}(\mathbf{v}'_{1,0})$ and $\bar{\mathbf{v}}_{1,1} = \mathbf{G}^{-1}(\mathbf{v}_{1,1}) - \mathbf{G}^{-1}(\mathbf{v}'_{1,1})$. By replacing $\mathbf{D}_0$ and $\mathbf{D}_1$ by their expressions in equation (5.1), we have:

$$\begin{aligned}
&\bar{c} \cdot [\mathbf{I}_n \mid \bar{\mathbf{A}}]\bar{\mathbf{r}} + [\mathbf{I}_n \mid \bar{\mathbf{A}}] \cdot \tilde{\mathbf{T}} \cdot \bar{\mathbf{v}}_{1,0} + (\bar{c} \cdot \mathbf{B} - [\mathbf{I}_n \mid \bar{\mathbf{A}}] \cdot \tilde{\mathbf{T}})\bar{\mathbf{v}}_{1,1} = \mathbf{0}^n \\
\implies &\big[[\mathbf{I}_n \mid \bar{\mathbf{A}}] \mid \mathbf{B}\big] \underbrace{\begin{bmatrix} \bar{c} \cdot \bar{\mathbf{r}} + \tilde{\mathbf{T}}(\bar{\mathbf{v}}_{1,0} - \bar{\mathbf{v}}_{1,1}) \\ \bar{c} \cdot \bar{\mathbf{v}}_{1,1} \end{bmatrix}}_{\mathbf{s}} = \mathbf{0}^n
\end{aligned}$$

and since $\bar{\mathbf{v}}_{1,1} \neq \mathbf{0}^n$, there exists a component $v_{1,1}^{(i)} \neq 0$, for some $i \in [n]$, such that $\left\|v_{1,1}^{(i)}\right\|_\infty \leq \mathfrak{b} - 1$. This implies that $\left\|\bar{c} \cdot v_{1,1}^{(i)}\right\|_\infty \leq (\mathfrak{b} - 1)d < q/2$. Therefore, $\bar{c} \cdot v_{1,1}^{(i)}$ holds over R without wrap-around. Since R is an integral domain and $\bar{c} \neq 0$, it follows that $\bar{c} \cdot v_{1,1}^{(i)} \neq 0$. Hence, $\bar{c} \cdot \bar{\mathbf{v}}_{1,1} \neq \mathbf{0}^n$, which implies that $\mathbf{s} \neq \mathbf{0}^n$.

$$\begin{aligned}
\|\mathbf{s}\| &= \sqrt{\left(\|\bar{c} \cdot \bar{\mathbf{r}}\| + \left\|\tilde{\mathbf{T}}(\bar{\mathbf{v}}_{1,0} - \bar{\mathbf{v}}_{1,1})\right\|\right)^2 + \|\bar{c} \cdot \bar{\mathbf{v}}_{1,1}\|^2} \\
&\leq \sqrt{\left(\|\bar{c}\|_1 \cdot \|\bar{\mathbf{r}}\| + s_1(\tilde{\mathbf{T}}) \cdot \|\bar{\mathbf{v}}_{1,0} - \bar{\mathbf{v}}_{1,1}\|\right)^2 + \|\bar{c}\|_1^2 \cdot \|\bar{\mathbf{v}}_{1,1}\|^2} \\
&\leq \sqrt{\left(2d\beta + \beta' dm(2(\mathfrak{b} - 1)\sqrt{md})\right)^2 + d^3(\mathfrak{b} - 1)^2 m} = \beta''
\end{aligned}$$

Where the second inequality follows from lemmata 2.3, and 2.5.c, and the fact that $\bar{c}$ is the difference of polynomials with binary coefficients by definition.

$\mathcal{B}$ fails to obtain a solution for the normal form M-SIS if it cannot extract $(\bar{c}, \tilde{\mathbf{T}})$ from $\mathcal{A}$ by using $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}$'s extractor, and this happens with probability at most κ_{OR}. This completes the proof. □

Theorem 5.3 (ϵ-Partial Equivocation)**.** *For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be a prime, $\mathfrak{b} \geq 2$, and $n = n(\lambda)$, $m = n\lceil \log_{\mathfrak{b}} q \rceil$, $d = d(\lambda)$ be lattice parameters. Let $M = \exp(\pi/\lambda + 1)$, and $T = (\mathfrak{b}-1)d\sqrt{2m/3}(2\sqrt{m} + \log\sqrt{d})$. Let $\varepsilon = \mathsf{negl}(\lambda)$, $\mathfrak{s} \geq \max(O(\sqrt{\lambda})T, \eta_\varepsilon(R^m))$. Let $\epsilon = 1 - (\frac{1}{M} + \mathsf{negl}(\lambda))$. Then, Construction 5.1 is ϵ-partially equivocable (see Definition 5.1).*

Proof. Let $(\mathsf{cmt}, \mathbf{r} := o)$ be the outputs of the Commit algorithm some input $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathsf{crs}_{\mathsf{OR}}), \mathsf{ck} = (\mathbf{D}_0, \pi_{\mathsf{ck}})$, and $\mathbf{v} \in \mathcal{M}^2$. Without loss of generality, suppose ck is binding on index 1, i.e., $\mathbf{D}_0 = \mathbf{A} \cdot \mathbf{T}$ for some $\mathbf{T} \leftarrow\!\!\$\ (S_1^\infty)^{m \times m}$. Let $\mathbf{v}' \in \mathcal{M}^2$ such that $\mathbf{v}'_1 = \mathbf{v}_1$. Then, by executing the algorithm Equiv on $(\mathbf{v}, \mathsf{cmt}, o, \mathbf{v}')$, we obtain a new opening $\mathbf{r}' = \mathbf{r} + \mathbf{T}(\mathbf{G}^{-1}(\mathbf{v}_0) - \mathbf{G}^{-1}(\mathbf{v}'_0)) \neq \perp$ with probability at least $1/M + \mathsf{negl}(\lambda)$ by Lemma 2.4, since $\left\|\mathbf{T}(\mathbf{G}^{-1}(\mathbf{v}_0) - \mathbf{G}^{-1}(\mathbf{v}'_0))\right\| \leq s_1(\mathbf{T})\left\|\mathbf{G}^{-1}(\mathbf{v}_0) - \mathbf{G}^{-1}(\mathbf{v}'_0)\right\| \leq (\mathfrak{b}-1)d\sqrt{2m/3}(2\sqrt{m} + \log\sqrt{d}) = T$, where the second inequality follows from Lemma 2.5.b and the fact that $\mathbf{G}^{-1}(\mathbf{v}_0)$ and $\mathbf{G}^{-1}(\mathbf{v}_0)$ have entries in $\{0, \ldots, \mathfrak{b}-1\}$. Finally, it follows by inspection that $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}', o' := \mathbf{r}') = 1$. This completes the proof. □

Theorem 5.4 (Hiding)**.** *For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be a prime, $\mathfrak{b} \geq 2$, and $n = n(\lambda)$, $m = n\lceil \log_{\mathfrak{b}} q \rceil$, $d = d(\lambda)$ be lattice parameters. Let $M = \exp(\pi/\lambda + 1)$, and $T = (\mathfrak{b}-1)d\sqrt{2m/3}(2\sqrt{m} + \log\sqrt{d})$. Let $\varepsilon = \mathsf{negl}(\lambda)$, $\mathfrak{s} \geq \max(O(\sqrt{\lambda})T, \eta_\varepsilon(R^m))$. Suppose the $\mathsf{M\text{-}LWE}_{m-n,n,q,U(S_1^\infty)}$ and $\mathsf{M\text{-}LWE}_{m-n,n,q,D_{R,\mathfrak{s}}}$ assumptions hold. Further suppose that $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}$ is zero-knowledge. Then Construction 5.1 is hiding (see Definition 5.1).*

Proof. The hiding property of commitments follows readily from the hardness of the $\mathsf{M\text{-}LWE}_{m-n,n,q,D_{R,\mathfrak{s}}}$ assumption. Here, we focus on proving that commitment keys do not reveal the binding index, and that openings do not reveal whether they were generated by Commit or Equiv. To do so, we proceed via a sequence of hybrids that interpolate between the experiments $\mathsf{PBVC\text{-}Hiding\text{-}Exp}_0$ and $\mathsf{PBVC\text{-}Hiding\text{-}Exp}_1$ (see Fig. 4). Let $\mathsf{Sim}_{\mathsf{OR}} = (\mathsf{Setup}, \mathsf{Prove})$ be the simulator of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}$.

- Hyb_0: This is $\mathsf{PBVC\text{-}Hiding\text{-}Exp}_0$.
  - Given a ppt adversary $\mathcal{A}$, the challenger generates pp and sends it to $\mathcal{A}$. Then, it receives $(i, j, \mathbf{v}, \mathbf{v}')$ from $\mathcal{A}$, with $i, j \in \{0, 1\}$.
  - If $\mathbf{v}[i] \neq \mathbf{v}[i']$, the challenger aborts. Otherwise, it computes $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(\mathsf{pp}, i)$; $(\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v})$; $o' \leftarrow \mathsf{Equiv}(\mathsf{pp}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}')$.
  - If $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}', o') \neq 1$, it returns $(\mathsf{ck}, \mathsf{cmt}, (o', 0))$ to $\mathcal{A}$. Otherwise, it returns $(\mathsf{ck}, \mathsf{cmt}, (o', 1))$.
- Hyb_1: Similar to Hyb_0. However, instead of committing to $\mathbf{v}$, the challenger commits to $\mathbf{v}'$, i.e., it computes $(\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}')$.

  Let $\mathbf{d} \in R_q^m$ be the shift computed during the execution of Equiv. In Hyb_0, $0 \leq \|\mathbf{d}\| \leq T$, whereas in Hyb_1, $\mathbf{d} = \mathbf{0}^m$. In Hyb_0, Equiv succeeds with probability at least $(1-\nu)/M$ by Lemma 2.4, where $\nu = \mathsf{negl}(\lambda)$, whereas in Hyb_1, it succeeds with probability $1/M$. Moreover, conditioned on Equiv succeeding, in both hybrids, given the output $(\mathsf{ck}, \mathsf{cmt}, (o', 1))$, where $\mathbf{t} := \mathsf{cmt}$ and $\mathbf{r}' := o'$, we have $\mathbf{A}\mathbf{r}' + [\mathbf{D}_0 \mid \mathbf{D}_1]\mathbf{v}' = \mathbf{t}$. Lemma 2.4 further implies that the marginal distributions of $\mathbf{r}'$ in the two hybrids are within statistical distance ν. Since $\mathbf{t}$ is determined by $\mathbf{r}'$ and $\mathbf{v}'$, the joint distributions of $(\mathsf{ck}, \mathbf{t}, \mathbf{r}')$, conditioned on success, are also within statistical distance ν. Therefore, the statistical distance between Hyb_0 and Hyb_1 is at most $O(\nu)$.
- Hyb_2: Similar to Hyb_1. However, the challenger commits to $\mathbf{v}'$, i.e., it computes $(\mathsf{cmt}, o) \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}')$, and outputs $(\mathsf{ck}, \mathsf{cmt}, o)$. Conditioned on Equiv succeeding in Hyb_1, it follows that $\mathsf{Hyb}_1 \overset{\mathfrak{s}}{\approx} \mathsf{Hyb}_2$. Note that here we do not account for the failure probability of Equiv in Hyb_1, since our goal is to show indistinguishability conditioned on Equiv not failing.
- Hyb_3: Similar to Hyb_2. However, during Setup the challenger computes $(\mathsf{crs}_{\mathsf{OR}}, \mathsf{td}_{\mathsf{OR}}) \leftarrow \mathsf{Sim}_{\mathsf{OR}}.\mathsf{Setup}(1^\lambda)$, and during Gen, after sampling $\mathbf{T} \leftarrow\!\!\$\ (S_1^\infty)^{m \times m}$ and computing $\mathbf{D}_{1-i} \leftarrow \mathbf{A} \cdot \mathbf{T}$ and $\mathbf{D}_i \leftarrow \mathbf{B} - \mathbf{D}_{1-i}$, it computes $\pi_{\mathsf{ck}} \leftarrow \mathsf{Sim}_{\mathsf{OR}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{OR}}, \mathsf{td}_{\mathsf{OR}}, \mathbb{x} := (\mathbf{A}, \mathbf{D}_i, \mathbf{D}_{1-i}, 1))$.

  $\mathsf{Hyb}_2 \overset{\mathfrak{c}}{\approx} \mathsf{Hyb}_3$ by the zero-knowledge property of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}$.

- Hyb_4: Similar to Hyb_3. However, during Gen, the challenger samples $\mathbf{D}_i \leftarrow\!\!\$\ R_q^{n\times m}$. $\mathsf{Hyb}_4 \stackrel{c}{\approx} \mathsf{Hyb}_3$ by the hardness $\mathsf{M\text{-}LWE}_{m-n,n,q,U(S_1^\infty)}$.
- Hyb_5: Similar to Hyb_4. However, during Gen, the challenger discards i and samples $\mathbf{D}_{1-j} \leftarrow\!\!\$\ R_q^{n\times m}$, then computes $\mathbf{D}_j \leftarrow \mathbf{B} - \mathbf{D}_{1-j}$. Clearly, Hyb_5 is identical to Hyb_4.
- Hyb_6: Similar to Hyb_5. However, during Gen, the challenger samples $\mathbf{T} \leftarrow\!\!\$\ (S_1^\infty)^{m\times m}$ and computes $\mathbf{D}_{1-j} \leftarrow \mathbf{A}\cdot\mathbf{T}$ and $\mathbf{D}_j \leftarrow \mathbf{B} - \mathbf{D}_{1-j}$. Next, it computes $\pi_{\mathsf{ck}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{OR}}, \mathbb{x} := (\mathbf{A}, \mathbf{D}_j, \mathbf{D}_{1-j}, 1), \mathbb{w} := \mathbf{T})$.

  $\mathsf{Hyb}_5 \stackrel{c}{\approx} \mathsf{Hyb}_6$ by the hardness of $\mathsf{M\text{-}LWE}_{m-n,n,q,U(S_1^\infty)}$ and the zero-knowledge property of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{OR}}}$. In addition, Hyb_6 corresponds to $\mathsf{PBVC\text{-}Hiding\text{-}Exp}_1$.

This completes the proof. □

We note that Construction 5.1 can be made unconditionally statistically hiding, at the cost of larger parameters, by replacing the M-LWE assumption with the Regularity Lemma [LPR13, Corollary 7.5] and by relying on the smudging lemma [AJL+12, Lemma 1] instead of rejection sampling.

From 1-out-of-2 to 1-out-of-2^k. Goel et al. [GGHK22] showed how to use a 1-out-of-2 PBVC to instantiate a 1-out-of-2^k PBVC. We briefly recall the main idea and provide a full instantiation based on Construction 5.1 in Appendix B. The approach is to use the 1-out-of-2 PBVC to commit to a full binary tree of depth k as follows: (1) the committer generates k commitment keys, where each commitment key is binding with respect to each bit of the index $i \in \mathbb{Z}_{2^k}$ to which the committer wishes to be bound, i.e., the commitment keys encode a path from the root to the i-th leaf; (2) the committer commits to the tree by generating one commitment randomness per level, i.e., for each pair on the same level, the same randomness is use to generate their commitments, which serve as values for the next level; and (3) the committer outputs the root of the tree as the commitment. Equivocation is achieved by appropriately equivocating commitments from the leaf level to the root.

Note that directly instantiating a 1-out-of-2^k PBVC via Construction 5.1 requires proving well-formedness of the commitment key at each level, for a total of k proofs. In the next section, we show how to batch these proofs to obtain a single proof for all k levels.

Remark 5.3 (Rejection Sampling in 1-out-of-2^k). Our 1-out-of-2 PBVC construction makes use of rejection sampling during equivocation. If applied to obtain the 1-out-of-2^k construction above, it would require applying rejection sampling at each level of the tree, thereby decreasing the probability of no-abort exponentially in k. Instead, one can defer the rejection sampling until after all tree levels have been equivocated. By Lemma 2.4, we can keep the failure probability independent of k, at the cost of increasing the Gaussian width parameter by a factor of $\sqrt{k}$.

5.2 Reducing Commitment Keys' Size and Batching OR-proofs

This subsection focuses on the OR-proof π_{ck}. Recall that we want to prove knowledge of a matrix $\mathbf{T} \in R_q^{m\times m}$ such that

$$(\mathbf{A}\cdot\mathbf{T} = \mathbf{D}_0 \vee \mathbf{A}\cdot\mathbf{T} = \mathbf{D}_1) \wedge \|\mathbf{T}\|_\infty \le \beta \tag{5.2}$$

where matrices $\mathbf{A}, \mathbf{D}_0, \mathbf{D}_1 \in R_q^{n\times m}$ are public. Our main protocol will show how to efficiently prove k such OR-proofs simultaneously with communication size (almost) sublinear in k.

Warm-up. Our starting point to prove (5.2) is combining the amortized Fiat-Shamir-with-Aborts protocol [Lyu12] with the folklore framework to prove disjunctions shown in [CDS94]. Concretely, suppose $\mathbf{A} \cdot \mathbf{T} = \mathbf{D}_b$ for a bit $b \in \{0,1\}$. The prover starts by picking $\mathbf{Y}_b \leftarrow D_{\mathfrak{s}}^{m \times m}$ from a discrete Gaussian distribution and computing $\mathbf{W}_b := \mathbf{A}\mathbf{Y}_b$. Then, the prover picks a uniformly random challenge $c_{1-b} \leftarrow \mathcal{B}_1$, samples $\mathbf{Z}_{1-b} \leftarrow D_{\mathfrak{s}}^{m \times m}$ and computes $\mathbf{W}_{1-b} := \mathbf{A}\mathbf{Z}_{1-b} - c_{1-b}\mathbf{D}_{1-b}$. The prover sends $(\mathbf{W}_0, \mathbf{W}_1)$ to the verifier. Upon receiving a challenge $c \leftarrow \mathcal{B}_1$ back, the prover computes $c_b := c \oplus c_{1-b} \in \mathcal{B}_1$, where the XOR operation is performed component-wise,[1] and computes $\mathbf{Z}_b := \mathbf{Y}_b + c_b\mathbf{T}$ and performs rejection sampling on $\mathbf{Z}_b$. Finally, the prover outputs the tuple $(c_0, c_1, \mathbf{Z}_0, \mathbf{Z}_1)$. The verifier then checks whether

$$c_0 \oplus c_1 = c \quad \text{and} \quad \mathbf{A}\mathbf{Z}_j = \mathbf{W}_j + c_j\mathbf{D}_j, \quad \|\mathbf{Z}_j\|_\infty \le \beta' \quad \text{for } j \in \{0,1\}.$$

Batching. We move on to our main protocol, which focuses on simultaneously proving knowledge of k matrices $\mathbf{T}^{(i)} \in R_q^{m \times m}$ that satisfy

$$\left(\mathbf{A} \cdot \mathbf{T}^{(i)} = \mathbf{D}_0^{(i)} \vee \mathbf{A} \cdot \mathbf{T}^{(i)} = \mathbf{D}_1^{(i)}\right) \wedge \|\mathbf{T}^{(i)}\|_\infty \le \beta \quad \text{for } i \in [k]. \tag{5.3}$$

To this end, we define the batched OR relation

$$\mathcal{R}_{k\text{-OR}} := \left\{ \begin{array}{c} \mathbb{x} := (\mathbf{A}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_{i\in[k]}, \beta), \\ \mathbb{w} := (\mathbf{T}^{(i)})_{i\in[k]} \end{array} : \begin{array}{c} \forall i \in [k], \left\|\mathbf{T}^{(i)}\right\|_\infty \le \beta \quad \wedge \\ (\mathbf{D}_0^{(i)} = \mathbf{A}\mathbf{T}^{(i)} \vee \mathbf{D}_1^{(i)} = \mathbf{A}\mathbf{T}^{(i)}) \end{array} \right\}$$

and the relaxed variant

$$\tilde{\mathcal{R}}_{k\text{-OR}} := \left\{ \begin{array}{c} \mathbb{x} := (\mathbf{A}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_{i\in[k]}, \beta), \\ \mathbb{w} := (\bar{c}^{(i)}, \mathbf{T}^{(i)})_{i\in[k]} \end{array} : \begin{array}{c} \forall i \in [k], \left\|\mathbf{T}^{(i)}\right\|_\infty \le \beta \wedge \bar{c}^{(i)} \in \mathcal{S}_1^\infty \wedge \\ (\bar{c}^{(i)}\mathbf{D}_0^{(i)} = \mathbf{A}\mathbf{T}^{(i)} \vee \bar{c}^{(i)}\mathbf{D}_1^{(i)} = \mathbf{A}\mathbf{T}^{(i)}) \end{array} \right\}.$$

The main strategy can be summarized as follows. Let us first consider the naive approach, where the prover performs k Σ-protocols from the single instance case in parallel. That is, the prover in the first round sends $(\mathbf{W}_0^{(i)}, \mathbf{W}_1^{(i)})_{i\in[k]}$, obtains k challenges $c^{(1)}, \ldots, c^{(k)} \leftarrow \mathcal{B}_1$, and responds with $(c_0^{(i)}, c_1^{(i)}, \mathbf{Z}_0^{(i)}, \mathbf{Z}_1^{(i)})_{i\in[k]}$. Then, we have

$$c_0^{(i)} \oplus c_1^{(i)} = c^{(i)}, \quad \mathbf{A}\mathbf{Z}_j^{(i)} = \mathbf{W}_j^{(i)} + c_j^{(i)}\mathbf{D}_j^{(i)}, \quad \|\mathbf{Z}_j\|_\infty \le \beta'$$

for $j \in \{0,1\}$ and $i \in [k]$. Our key observation here is by summing the k linear relations we obtain:

$$\mathbf{A}\left(\underbrace{\sum_{i=1}^{k} \mathbf{Z}_j^{(i)}}_{\mathbf{Z}_j :=}\right) = \left(\underbrace{\sum_{i=1}^{k} \mathbf{W}_j^{(i)}}_{\mathbf{W}_j :=}\right) + \sum_{i=1}^{k} \mathbf{D}_j^{(i)} c_j^{(i)}.$$

Hence, in our protocol we will directly send the sums $(\mathbf{W}_0, \mathbf{W}_1, \mathbf{Z}_0, \mathbf{Z}_1)$. Unfortunately, we could not compress the messages related to the challenges, and thus the prover needs to send all $(c_0^{(i)}, c_1^{(i)})_{i\in[k]}$, thus making the communication complexity linear in k. We provide the full non-interactive protocol below.

[1] We note that a slightly more efficient method is presented in [dPLNS17], which allows the challenge space to contain *sparse* binary polynomials.

Construction 5.2 (Batching OR-proofs)**.** For any $\lambda \in \mathbb{N}$, let $q \in \mathbb{N}$ be prime, $n = n(\lambda)$, $m > n$, $d = d(\lambda)$ be lattice parameters. Let $\mathfrak{s} = \mathfrak{s}(\lambda)$ be a Gaussian width parameter, $\beta = \beta(\lambda), \beta' = \beta'(\lambda)$ be norm bounds, and $M = M(\lambda)$ be a rejection constant. Let $B \geq 1$ be a "repetition" parameter. Denote by $\mathsf{H} : \{0,1\}^* \to \mathcal{B}_1^k$ the random oracle. We construct a non-interactive zero-knowledge proof for the relation $\mathcal{R}_{k\text{-}\mathsf{OR}}$ as follows.

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$ and the batching parameter $k \geq 1$, sample $\bar{\mathbf{A}} \leftarrow\!\!\$\ R_q^{n \times (m-n)}$. Return $\mathsf{pp} := \mathbf{A} \leftarrow [\mathbf{I}_n \mid \bar{\mathbf{A}}]$.

- $\mathsf{Prove}^{\mathsf{H}}(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, (\mathbf{T}^{(i)})_i)$: Given public parameters $\mathbf{A} := \mathsf{pp}$, statement $(\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i$, and witness $(\mathbf{T}^{(i)})_i$, proceed as follows:

  1. Let $t = 0$ and $\mathfrak{b} = 0$.
  2. While $\mathfrak{b} = 0$ and $t \leq B$:
  3. (a) For $i \in [k]$:
     i. Let $b_i \in \{0,1\}$ be the bit such that $\mathbf{A}\mathbf{T}^{(i)} = \mathbf{D}_{b_i}^{(i)}$.
     ii. Sample $\mathbf{Y}_{b_i}^{(i)} \leftarrow D_{\mathfrak{s}}^{m \times m}$
     iii. Compute $\mathbf{W}_{b_i}^{(i)} := \mathbf{A}\mathbf{Y}_{b_i}^{(i)}$.
     iv. Sample $c_{1-b_i}^{(i)} \leftarrow \mathcal{B}_1$ and $\mathbf{Z}_{1-b_i}^{(i)} \leftarrow D_{\mathfrak{s}}^{m \times m}$.
     v. Compute $\mathbf{W}_{1-b_i}^{(i)} := \mathbf{A}\mathbf{Z}_{1-b_i}^{(i)} - c_{1-b_i}^{(i)}\mathbf{D}_{1-b_i}^{(i)}$.
  (b) Compute $\mathbf{W}_0 := \sum_{i=1}^k \mathbf{W}_0^{(i)}$ and $\mathbf{W}_1 := \sum_{i=1}^k \mathbf{W}_1^{(i)}$.
  (c) Compute $(c^{(i)})_{i \in [k]} = \mathsf{H}(\mathbf{A}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, \mathbf{W}_0, \mathbf{W}_1)$.
  (d) For $i \in [k]$:
     i. Compute $c_{b_i}^{(i)} := c^{(i)} \oplus c_{1-b_i}^{(i)}$.
     ii. Compute $\mathbf{Z}_{b_i}^{(i)} := \mathbf{Y}_{b_i}^{(i)} + c_{b_i}^{(i)}\mathbf{T}^{(i)}$.
  (e) Compute $\mathbf{Z}_0 := \sum_{i=1}^k \mathbf{Z}_0^{(i)}$ and $\mathbf{Z}_1 := \sum_{i=1}^k \mathbf{Z}_1^{(i)}$.
  (f) $\mathfrak{b} \leftarrow \mathsf{RejSamp}((\mathbf{Z}_{b_i}^{(i)})_{i \in [k]}, (c_{b_i}^{(i)}\mathbf{T}^{(i)})_{i \in [k]}, \mathfrak{s}, M)$.
  (g) $t = t + 1$.
  4. If $\mathfrak{b} = 0$, then output $\pi_{\mathsf{ck}} := \perp$.
  5. Else, return $\pi_{\mathsf{ck}} := ((c_0^{(i)}, c_1^{(i)})_{i \in [k]}, \mathbf{Z}_0, \mathbf{Z}_1)$.

- $\mathsf{Verify}^{\mathsf{H}}(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, \pi_{\mathsf{ck}})$: Given public parameters $\mathbf{A} := \mathsf{pp}$, statement $(\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i$ and a proof $\pi_{\mathsf{ck}} := ((c_0^{(i)}, c_1^{(i)})_{i \in [k]}, \mathbf{Z}_0, \mathbf{Z}_1)$, proceed as follows:

  1. If $\|\mathbf{Z}_0\|_\infty > \beta'$ or $\|\mathbf{Z}_1\|_\infty > \beta'$, return 0.
  2. If for some $i \in [k]$, $c_0^{(i)} \notin \mathcal{B}_1$ or $c_1^{(i)} \notin \mathcal{B}_1$, return 0.
  3. For $i \in [k]$, compute $c^{(i)} := c_0^{(i)} \oplus c_1^{(i)}$.
  4. If $(c^{(i)})_{i \in [k]} = \mathsf{H}(\mathbf{A}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, \mathbf{A}\mathbf{Z}_0 - \sum_{i=1}^k c_0^{(i)}\mathbf{D}_0^{(i)}, \mathbf{A}\mathbf{Z}_1 - \sum_{i=1}^k c_1^{(i)}\mathbf{D}_1^{(i)})$, return 1. Otherwise, output 0.

Lemma 5.1 (Completeness)**.** *Let $d = O(\lambda)$, $M = \exp(\pi/\lambda + 1)$, $B = 2M\lambda$ and set $\mathfrak{s} = O(\beta m d\sqrt{\lambda})$ and $\beta' = \mathfrak{s} \cdot \sqrt{k} \cdot \log\sqrt{m^2 d}$. Then, the NIZK in Construction 5.2 satisfies completeness with the completeness function $\mathfrak{c}(\lambda) := \mathsf{negl}(\lambda)$.*

Proof. As Item 4 holds by following our analysis above, we only focus on the other terms. As for Item 2, all partitioned challenges $(c_0^{(i)}, c_0^{(i)})_{i\in[k]}$ are indeed over $\mathcal{B}_1$. Finally, to show Item 1 holds, we first take an upper-bound on the "secret" vector being masked in the rejection sampling. Namely, for each $i \in [k]$ we have

$$\|c_{b_i}^{(i)} \cdot \mathbf{T}^{(i)}\|_\infty \leq \|c_{b_i}^{(i)}\|_1 \cdot \|\mathbf{T}^{(i)}\|_\infty \leq \beta d$$

and thus

$$\|c_{b_i}^{(i)} \cdot \mathbf{T}^{(i)}\|_2 \leq \|c_{b_i}^{(i)} \cdot \mathbf{T}^{(i)}\|_\infty \cdot m \leq \beta m d =: T.$$

Now, since $\mathfrak{s} = O(\sqrt{\lambda}) \cdot T$, we obtain that the prover does not abort in single iteration of the loop in Item 2 with probability $\frac{1}{M} - \mathsf{negl}(\lambda)$ by Lemma 2.4. Hence, with probability

$$\left(1 - \frac{1}{M} + \mathsf{negl}(\lambda)\right)^B \leq \left(1 - \frac{1}{2M}\right)^{2M\lambda} \approx \exp(-\lambda)$$

the prover does not output $\pi_{\mathsf{ck}} = \perp$. Moreover, by rejection sampling, conditioned on not aborting, matrices $\mathbf{Z}_0, \mathbf{Z}_1$ are statistically close to discrete Gaussians $D_{\mathfrak{s}\sqrt{k}}^{m\times m}$. Hence, by Lemma 2.3, the probability that $\|\mathbf{Z}_0\|_\infty > \beta'$ or $\|\mathbf{Z}_1\|_\infty > \beta'$, where $\beta' = \mathfrak{s} \cdot \sqrt{k} \cdot \log\sqrt{m^2 d}$, is negligible. Thus, the statement holds. □

As noted before, we will show knowledge soundness with respect to the relaxed relation $\tilde{\mathcal{R}}_{k\text{-}\mathsf{OR}}$.

Lemma 5.2 (Knowledge Soundness)**.** *The NIZK in Construction 5.2 is knowledge-sound (w.r.t the relation $\tilde{\mathcal{R}}_{k\text{-}\mathsf{OR}}$) with knowledge soundness error $\frac{k\cdot(Q+1)}{2^d}$.*

Proof. We first prove that the underlying Σ-protocol satisfies a notion called *coordinate-wise special soundness* [FMN24]. Concretely, suppose we are given $k+1$ three-message transcripts

$$\mathsf{tr}_\ell := \left((\mathbf{W}_0, \mathbf{W}_1), (c^{(\ell,i)})_{i\in[k]}, ((c_0^{(\ell,i)}, c_1^{(\ell,i)})_{i\in[k]}, \mathbf{Z}_0^{(\ell)}, \mathbf{Z}_1^{(\ell)})\right)$$

such that for $\ell, i \in [k]$ and $j \in \{0,1\}$, the following conditions are satisfied

$$\begin{aligned}
\forall \gamma \in [k]\backslash\{\ell\}, \quad c^{(\ell,\gamma)} = c^{(0,\gamma)} \quad \text{and} \quad c^{(\ell,\ell)} &\neq c^{(0,\ell)} \\
c_0^{(\ell,i)} \in \mathcal{B}_1^m, c_1^{(\ell,i)} &\in \mathcal{B}_1 \\
c^{(\ell,i)} = c_0^{(\ell,i)} \oplus c_1^{(\ell,i)} & \\
\|\mathbf{Z}_j^{(\ell)}\|_\infty &\leq \beta' \\
\mathbf{A}\mathbf{Z}_j^{(\ell)} - \sum_{i=1}^k \mathbf{D}_j^{(i)} c_j^{(\ell,i)} &= \mathbf{W}_j.
\end{aligned}$$

Informally, the first line says that the k-dimensional vectors $(c^{(\ell,i)})_{i\in[k]}$ and $(c^{(0,i)})_{i\in[k]}$ differ in exactly the ℓ-th coordinate. We show how this is crucial for extraction.

Fix $\ell \in [k]$ and focus on tr_0 and tr_ℓ. Then, for $j \in \{0,1\}$ we have

$$\mathbf{A}\mathbf{Z}_j^{(\ell)} - \sum_{i=1}^k \mathbf{D}_j^{(i)} c_j^{(\ell,i)} = \mathbf{W}_j = \mathbf{A}\mathbf{Z}_j^{(0)} - \sum_{i=1}^k \mathbf{D}_j^{(i)} c_j^{(0,i)}.$$

Since $(c^{(\ell,i)})_{i\in[k]}$ and $(c^{(0,i)})_{i\in[k]}$ differ in exactly the ℓ-th coordinate, we have that

$$\mathbf{A}(\mathbf{Z}_j^{(\ell)} - \mathbf{Z}_j^{(0)}) = (c^{(\ell,\ell)} - c^{0,\ell}) \cdot \mathbf{D}_j^{(\ell)}.$$

Hence, we define the extracted (relaxed) witness

$$\mathbf{T}^{(\ell)} := \mathbf{Z}_j^{(\ell)} - \mathbf{Z}_j^{(0)} \quad \text{and} \quad \bar{c}^{(\ell)} := c^{(\ell,\ell)} - c^{(0,\ell)} \in \mathcal{S}_1^\infty.$$

Crucially, note that $\|\mathbf{T}^{(\ell)}\|_\infty \leq 2\beta'$ and $\bar{c}^{(\ell)}$ is non-zero. Therefore, we deduce that from the $k+1$ valid transcripts, which satisfy the coordinate-wise special soundness property, a valid witness $(\bar{c}^{(\ell)}, \mathbf{T}^{(\ell)})_{\ell\in[k]}$ is extracted.

Finally, one can build an extractor $\mathcal{E}$, which retrieves such $k+1$ transcripts from the prover $\mathcal{P}^*$ using the "abstract sampling game" strategy from [FMN24, Lemma 8.1]. The algorithm $\mathcal{E}$ invokes $\mathcal{P}^*$ in expectation $1 + k \cdot (Q+1)$ times, and succeeds with probability $\varepsilon - (Q+1) \cdot k/2^d$, which concludes the proof. □

Lemma 5.3 (Zero-Knowledge)**.** *Let $d = O(\lambda)$, $M = \exp(\pi/\lambda + 1)$, $B = 2M\lambda$ and set $\mathfrak{s} = O(\beta md\sqrt{\lambda})$ and $\beta' = \mathfrak{s} \cdot \sqrt{k} \cdot \log\sqrt{m^2 d}$. Then, the NIZK in Construction 5.2 satisfies honest-verifier zero-knowledge.*

Proof. We follow the proof strategy from [BLNS23b, DFPS23]. Let D be a PPT distinguisher against the honest-verifier zero-knowledge game which makes Q_π calls to the prover and Q_H queries to the random oracle. We prove the statement by hybrid argument, where for each algorithm $\mathcal{P}_i$ we will modify the prover algorithm.

$\mathcal{P}_0(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, (\mathbf{T}^{(i)})_i)$: We define the algorithm to be exactly $\mathsf{Prove}^\mathsf{H}$, apart from the fact that $\mathcal{P}_0$ manually simulates the random oracle queries and keeps the responses for consistency. Clearly, we have

$$\left| \begin{array}{r} \Pr\left[\mathsf{D}^{\mathsf{Prove}^\mathsf{H}(\cdot)}(1^\lambda, \mathsf{crs}) = 1 | \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \\ - \Pr\left[\mathsf{D}^{\mathcal{P}_0(\cdot)}(1^\lambda, \mathsf{crs}) \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \end{array} \right| = 0.$$

$\mathcal{P}_1(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, (\mathbf{T}^{(i)})_i)$: This algorithm, as opposed to $\mathcal{P}_0$, just after Item 2, samples a challenge $(c^{(i)})_{i\in[k]} \leftarrow \mathcal{B}_1^m$, and then later in Item 3c it programs the random oracle H so that:

$$\mathsf{H}(\mathbf{A}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, \mathbf{W}_0, \mathbf{W}_1) := (c^{(i)})_{i\in[k]}$$

without checking whether a hash output was already assigned to the input $(\mathbf{A}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, \mathbf{W}_0, \mathbf{W}_1)$ – we call this a "collision". Note that at most $B \cdot Q_\pi + Q_\mathsf{H}$ such tuples will ever be set during the experiment, which implies that there will be at most $B \cdot Q_\pi + Q_\mathsf{H}$ such $\mathbf{W}_{b_1}$.

Then, if we parse

$$\mathbf{Y}_{b_1}^{(1)} := \begin{bmatrix} \mathbf{Y}_{b_1,0}^{(1)} \\ \mathbf{Y}_{b_1,1}^{(1)} \end{bmatrix} \leftarrow D_\mathfrak{s}^{m\times m}$$

then for any image $\mathbf{U} \in R_q^{n\times m}$, by recalling $\mathbf{A} = [\mathbf{I}_n \mid \bar{\mathbf{A}}]$, we have

$$\begin{aligned} \Pr[\mathbf{W}_{b_1} = \mathbf{U}] &= \Pr[\mathbf{A}\mathbf{Y}_{b_1}^{(1)} = \mathbf{U} - \sum_{i=2}^k \mathbf{W}_{b_1}^{(i)}] \\ &= \Pr[\mathbf{Y}_{b_1,0}^{(1)} = \mathbf{U} - \sum_{i=2}^k \mathbf{W}_{b_1}^{(i)} - \bar{\mathbf{A}}\mathbf{Y}_{b_1,1}^{(1)}] \\ &\leq 2^{-nmd} \end{aligned}$$

where for the last equation we used the min-entropy of a discrete Gaussian [Lyu12, Lemma 4.4]. Hence, if $\mathcal{P}_1$ is accessed Q_π times and the probability of getting a collision each time is at most $(B \cdot Q_\pi + Q_\mathsf{H})2^{-nmd}$

by the union bound, then the probability that a collision occurs after calling the prover Q_π times is at most $BQ_\pi(B \cdot Q_\pi + Q_{\mathsf{H}})2^{-nmd}$. Therefore,

$$\left| \begin{array}{l} \Pr\left[\mathsf{D}^{\mathcal{P}_0(\cdot)}(1^\lambda, \mathsf{crs}) = 1 | \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \\ \quad - \Pr\left[\mathsf{D}^{\mathcal{P}_1(\cdot)}(1^\lambda, \mathsf{crs}) \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \end{array} \right| \leq B \cdot Q_\pi(B \cdot Q_\pi + Q_{\mathsf{H}})2^{-nmd}$$

which is negligible.

$\mathcal{P}_2(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, (\mathbf{T}^{(i)})_i)$: This algorithm behaves almost identically as $\mathcal{P}_1$, but instead of computing $\mathbf{W}_{b_i}^{(i)} := \mathbf{A}\mathbf{Y}_{b_i}^{(i)}$, it sets $\mathbf{W}_{b_i}^{(i)} := \mathbf{A}\mathbf{Z}_{b_i}^{(i)} - c_{b_i}^{(i)}\mathbf{D}_{b_i}^{(i)}$. From the verification equations we deduce that $\mathcal{P}_2$ behaves identically as $\mathcal{P}_1$ and thus

$$\left| \begin{array}{l} \Pr\left[\mathsf{D}^{\mathcal{P}_1(\cdot)}(1^\lambda, \mathsf{crs}) = 1 | \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \\ \quad - \Pr\left[\mathsf{D}^{\mathcal{P}_2(\cdot)}(1^\lambda, \mathsf{crs}) \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \end{array} \right| = 0.$$

$\mathcal{P}_3(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, (\mathbf{T}^{(i)})_i)$: This algorithm differs from $\mathcal{P}_2$ in computing each $\mathbf{Z}_{b_i}^{(i)}$. Namely, instead of setting $\mathbf{Z}_{b_i}^{(i)} = \mathbf{Y}_{b_i}^{i} + c^{(i)}\mathbf{T}^{(i)}$ and running

$$\mathfrak{b} \leftarrow \mathsf{RejSamp}((\mathbf{Z}_{b_i}^{(i)})_{i\in[k]}, (c_{b_i}^{(i)}\mathbf{T}^{(i)})_{i\in[k]}, \mathfrak{s}, M) = 0,$$

$\mathcal{P}_3$ directly samples $\mathbf{Z}_{b_i}^{(i)} \leftarrow D_{\mathfrak{s}}^{m\times m}$ and picks $\mathfrak{b} = 1$ with probability $1/M$ and $\mathfrak{b} = 0$ with probability $1 - 1/M$. In this hybrid, we invoke Lemma 2.4 at most $B \cdot Q_\pi$ times and thus

$$\left| \begin{array}{l} \Pr\left[\mathsf{D}^{\mathcal{P}_2(\cdot)}(1^\lambda, \mathsf{crs}) = 1 | \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \\ \quad - \Pr\left[\mathsf{D}^{\mathcal{P}_3(\cdot)}(1^\lambda, \mathsf{crs}) \mid \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] = 1 \end{array} \right| \leq B \cdot Q_\pi \cdot \exp(-\lambda).$$

The final observation is that $\mathcal{P}_3(\mathsf{pp}, (\mathbf{D}_0^{(i)}, \mathbf{D}_1^{(i)})_i, (\mathbf{T}^{(i)})_i)$ does not use the witness $(\mathbf{T}^{(i)})_i$. Hence, we can define the zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ where $\mathsf{Sim}_1 = \mathsf{Setup}$ and $\mathsf{Sim}_2 = \mathcal{P}_3$. Hence, the statement follows by the hybrid argument. □ □

Communication efficiency. Size of the proof π_{ck} can be upper-bounded by

$$\underbrace{2 \cdot k \cdot d}_{|(c_0^{(i)})_i|+|(c_1^{(i)})_i|} + \underbrace{2 \cdot m^2 \cdot d \cdot \log(2\beta')}_{|\mathbf{Z}_0|+|\mathbf{Z}_1|}.$$

Although the proof contains a linear number of "decomposed" challenges $(c_0^{(i)})_i$ and $(c_1^{(i)})_i$, they are all bits and thus we do not expect them to be a significant part of the proof size when instantiated concretely.

5.3 Application: Ring Signature From Fiat-Shamir with Abort

GGHK sketched how an identification protocol based on a stackable Σ-protocol can be used to instantiate a ring signature in the Fiat-Shamir paradigm [FS86]. Specifically, they applied their compiler (see Compiler 5.1), together with their discrete-log-based 1-out-of-2 PBVC, to Schnorr's identification protocol [Sch89], obtaining a ring signature whose signature size is logarithmic in the size of the ring. In this section, we show that Lyubashevsky's identification protocol [Lyu12] is stackable. Recall this is a Σ-protocol with aborts and it forms the basis for NIST post-quantum signatures candidates such as Dilithium [DKL$^+$18] and Racoon [dPKPR24]. We can then apply our 1-out-of-2 PBVC (see Construction 5.1) to obtain an M-SIS-based ring signature à la GGHK.

Construction 5.3 (Σ-protocol with abort template)**.** For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be an integer, and $n = n(\lambda)$, $d = d(\lambda)$, $m = m(\lambda)$ be lattice parameters. Let $\mathfrak{s} = \mathfrak{s}(\lambda)$ be a Gaussian-width parameter, let $\beta_1 = \beta_1(\lambda)$ and $\beta_2 = \beta_2(\lambda)$ be norm-bound. Let $\mathcal{C} \subset R_q$ such that $|\mathcal{C}| \geq 2^{\omega(\log \lambda)}$ and for any $c \in \mathcal{C}$, $\|c\|_1 \leq \beta_2$. Let $M = M(\lambda)$ be a rejection constant. Let $\mathsf{H} : \{0,1\}^* \to \mathcal{C}$ be a random oracle. Consider the relation $\mathcal{R} = \{\mathbb{x} := (\mathbf{A}, \mathbf{t}, \beta_1), \mathbb{w} := \mathbf{s} : \mathbf{A} \cdot \mathbf{s} = \mathbf{t} \wedge \|\mathbf{s}\| \leq \beta_1\}$

- $\mathsf{P}_1(\mathbb{x} := (\mathbf{A}, \mathbf{t}, \beta_1), \mathbb{w} := \mathbf{s})$: Given $(\mathbf{A}, \mathbf{t}) \in R_q^{n \times m}$ and $\mathbf{t} \in R_q^n$, and a witness $\mathbf{s} \in R_q^m$ such that $\|\mathbf{s}\| \leq \beta_1$, proceed as follows: Sample $\mathbf{y} \leftarrow D^m_{R,\mathfrak{s}}$, return $\mathbf{w} \leftarrow \mathbf{A} \cdot \mathbf{y}$ as a and state $\mathsf{st}_\mathsf{P} := \mathbf{y}$.
- $\mathsf{P}_2^\mathsf{H}(\mathbb{x} := (\mathbf{A}, \mathbf{t}, \beta_1), \mathbb{w} := \mathbf{s}, a, \mathsf{st}_\mathsf{P})$: Given $\mathbb{x}$, $\mathbb{w}$, $\mathbf{w} := a$ and $\mathbf{y} := \mathsf{st}_\mathsf{P}$, proceed as follows:
  1. Compute $c \leftarrow \mathsf{H}(\mathbb{x}, a)$.
  2. Compute $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{s}$.
  3. If $\mathsf{RejSamp}(\mathbf{z}, c \cdot \mathbf{s}, \mathfrak{s}, M) = 0$, return $\perp$. Otherwise, return $\mathbf{z}$.
- $\mathsf{V}^\mathsf{H}(\mathbb{x} := (\mathbf{A}, \mathbf{t}, \beta_1), \pi)$: Given $\mathbb{x}$, a proof $(a, c, \mathbf{z}) := \pi$, where $\mathbf{w} := a$, proceed as follows: If $\mathbf{A} \cdot \mathbf{z} = \mathbf{w} + c\mathbf{t}$ and $\|\mathbf{z}\| \leq \mathfrak{s}\sqrt{md}$, return 1. Otherwise, return 0.

We refer the reader to Lyubashevsky [Lyu12] for the proofs of correctness, zero-knowledge, and soundness. Here, we focus on showing that, conditioned on no abort, the above construction satisfies extended HVZK and Recycable third message.

Theorem 5.5. *For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be an integer, and $n = n(\lambda)$, $d = d(\lambda)$, $m = m(\lambda)$ be lattice parameters. Let $\varepsilon = \mathsf{negl}(\lambda)$, let $T = \beta_1 \cdot \beta_2$, and let $\mathfrak{s} = \max\{\eta_\varepsilon(R^m), O(\sqrt{\lambda})T\}$. Let $M = \exp(\pi/\lambda + 1)$. Then, conditioned on P_2 not aborting, Construction 5.3 satisfies extended HVZK and Recycable third message (see Definition 5.2)*

Proof. Consider the distribution $\mathcal{Z} = D^m_{R,\mathfrak{s}}$. Gentry, Peikert, and Vaikuntanathan [GPV07, Section 4.2] gave an efficient sampler for $\mathcal{Z}$. Now, consider the following simulator:

- Upon receiving $\mathbb{x} := (\mathbf{A}, \mathbf{t}, \beta_1)$, $c \leftarrow \mathcal{C}$, and $\mathbf{z} \leftarrow \mathcal{Z}$: return $\mathbf{w} \leftarrow \mathbf{A}\mathbf{z} - c\mathbf{t}$ as a.

Note that for any witness $\mathbf{s}$ satisfying the above statement, we have $\|c\mathbf{s}\| \leq \|c\|_1\|\mathbf{s}\| \leq \beta_1\beta_2$. Therefore, conditioned on P_2 not aborting, Lemma 2.4 implies that $(a, c, \mathbf{z})$ produced by $(\mathsf{P}_1, \mathsf{P}_2^\mathsf{H})$ is within negligible statistical distance of $(a', c', \mathbf{z}')$ produced by the simulator on input $\mathbf{z}' \leftarrow \mathcal{Z}$. Hence, Construction 5.3 satisfies extended HVZK. Again, by Lemma 2.4, the marginal distribution of $\mathbf{z}$ in both tuples is independent of the statement. Therefore, Construction 5.3 has recyclable third message. □

Beyond Σ-protocols. In subsequent work, Goel et al. [GHKS23] showed that PBVCs can be used to obtain disjunctive proofs whose size is smaller than that of the canonical construction of Cramer, Damgard and Schoenmakers [CDS94] for $(2\mu + 1)$-round protocols, with $\mu > 1$, provided that the prover's messages can be separated into statement-independent and statement-dependent components. In brief, the prover commits to the statement-dependent messages using a PBVC and reuses the statement-independent messages across the different branches of the disjunctive proof. We believe that our PBVC can serve as a post-quantum alternative in such settings.

6 Anonymous Rate-Limited Tokens in the ROM

In this section, we present a generic construction for anonymous rate-limited tokens based on a weak PRF in the random oracle model (ROM). Our construction follows the template establish by Camenisch

et al. [CHL05, CHK+06]: (1) the issuer uses a signature with efficient protocols to obliviously sign on a user's identity and PRF's key; and (2) to redeem a token, the user computes a serial number and a tag by evaluating a PRF on a counter, and then proves knowledge of a signature on its identity and PRF key used to compute the serial number and tag, in addition to proving that the counter is withing a certain range and that the serial number and tag are well-formed.

Our departure from the template of Camenisch et al. lies in how we handle proving the correct evaluation of the PRF on a counter within a given range. Since we use a weak PRF, the input to the PRF must be the output of a random oracle in order to obtain adaptive security. However, because we must also prove correct PRF evaluation, this creates a challenge at the level of the underlying proof system. To address this issue, we use a one-out-of-many approach to prove that a purported PRF output corresponds to evaluating the PRF on one out of many random values, where each such value is an output of the random oracle. In our setting, these values correspond to the random oracle evaluated over the range of the counter.

Another way in which we depart from the template established by Camenisch et al. in this generic construction is in how we generate the nonce used to compute the tag. Instead of simply using the provided nonce, which could be maliciously generated to frame honest users for double-spending, we use the a random oracle to derive the nonce for the tag computation from the provided nonce and the proof of well-formedness of the serial number. Later, in Section 7, when instantiating the construction from Lattice-based assumptions, we revert to simply using the provided nonce and instead modify the tag computation so that framing honest users of double-spending is still hard.

Construction 6.1. For any $\lambda \in \mathbb{N}$, let $\mathfrak{t} = \mathfrak{t}(\lambda)$ and $\mathfrak{n} = \mathfrak{n}(\lambda)$ such that $\mathfrak{t} \geq \mathfrak{n}$. Let $\delta = \delta(\lambda)$ be an error bound and $\nu = \mathsf{negl}(\lambda)$ a relative bound. Let $c \geq 1$ be a constant. Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\leq p(\lambda)}$ be a random oracle, where p is a polynomial. Consider the following primitives:

- Let $\Pi_{\mathsf{PRF}} = (\mathsf{Gen}, \mathsf{Eval}^{\mathsf{H}})$ be a weak δ-almost key-homomorphic PRF (see Definition 3.7) with key space $\mathcal{K}$, and output space $\mathcal{Y}$, where $\mathcal{Y}$ is a finitely generated A-module for some finite commutative ring A.
- Let $\mathcal{S} \subseteq A^\times$ be an efficiently testable ν-annihilator-coset evasive set with respect to $\mathcal{Y}$ (see Definition 2.1), such that $|\mathcal{S}| \geq 2^{\omega(\log \lambda)}$, and for all distinct $s_1, s_2 \in \mathcal{S}$, $s_1 - s_2 \in A^\times$.
- Let $\Pi_{\mathsf{SEP}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{SignCom}, \mathsf{Verify})$ be a signature with efficient protocols over message space $\mathcal{M}_{\mathsf{SEP}}$, where $\mathcal{K} \subseteq \mathcal{M}_{\mathsf{SEP}}$. Suppose Π_{SEP} is associated with a LHC $\Pi^{\mathsf{SEP}}_{\mathsf{Com}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$, together with NIZK schemes $\Pi^{\mathcal{R}_{\mathsf{Com}}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ and $\Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ for the relations $\mathcal{R}_{\mathsf{Com}}$ and $\mathcal{R}_{M,\Sigma}$, respectively (see equations (3.2), (3.3)).
- Let $\Pi^{\mathcal{R}_{\mathsf{1Many}}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ be a NIZK scheme for the relation $\mathcal{R}_{\mathsf{1Many}}$(see equation 3.4) defined with respect to Π_{PRF}, Π_{SEP}, and where the function F correspond to $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$.

We construct an anonymous rate-limited credential with context space $\mathfrak{C} = \{0,1\}^{c \log \lambda}$, nonce space $\mathfrak{N} = \mathcal{S}$, and user's identity space $\mathsf{ID} = \mathcal{M}_{\mathsf{SEP}}$ as follows:

- $\mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t})$: Given λ, the rate limit per context $\mathfrak{n}$, and the total limit $\mathfrak{t}$, proceed as follows:
  1. Compute $\mathsf{crs}_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{Setup}(1^\lambda)$.
  2. Compute $\mathsf{crs}_{M,\Sigma} \leftarrow \Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}}.\mathsf{Setup}(1^\lambda, \mathsf{crs}_{\mathsf{SEP}})$;
  3. Compute $\mathsf{crs}_{\mathsf{1Many}} \leftarrow \Pi^{\mathcal{R}_{\mathsf{1Many}}}_{\mathsf{NIZK}}.\mathsf{Setup}(1^\lambda, \mathsf{crs}_{M,\Sigma})$.
  4. Return $\mathsf{crs} := (\mathfrak{n}, \mathfrak{t}, \mathsf{crs}_{\mathsf{SEP}}, \mathsf{crs}_{M,\Sigma}, \mathsf{crs}_{\mathsf{1Many}})$.
- $\mathsf{IKeyGen}(\mathsf{crs})$: Given crs, proceed as follows:

1. Compute $(\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{Gen}(\mathsf{crs})$.
2. Return $(\mathsf{pk}_I, \mathsf{sk}_I)$.

- $\mathsf{UKeyGen}(\mathsf{crs}, \mathsf{id}_U)$: Given crs, and $\mathsf{id}_U \in \mathcal{M}_{\mathsf{SEP}}$ proceed as follows:

1. Compute $(c_U, o_U) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, \mathsf{id}_U)$.
2. Set $\mathsf{pk}_U := c_U$ and $\mathsf{sk}_U := (\mathsf{id}_U, o_U)$.
3. Return $(\mathsf{pk}_U, \mathsf{sk}_U)$.

- $\mathsf{Issue}.U_1(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I)$: Given crs, sk_U, and pk_I, proceed as follows:

1. Compute $\hat{k} \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda)$.
2. Compute $(c_{\mathsf{SEP}}, o_{\mathsf{SEP}}) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, (\mathsf{sk}_U, \hat{k}))$.
3. Compute $\pi_{\mathsf{Open}} \leftarrow \Pi^{\mathcal{R}_{\mathsf{Com}}}_{\mathsf{NIZK}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}), \mathbb{w} := (\mathsf{sk}_U, \hat{k}, o_{\mathsf{SEP}}))$.
4. Return $\mathsf{st}_U := (\mathsf{pk}_I, \hat{k}, o_{\mathsf{SEP}}, \mathsf{sk}_U)$ and $\mathsf{umsg} := (c_{\mathsf{SEP}}, \pi_{\mathsf{Open}})$.

- $\mathsf{Issue}.I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U, \mathsf{umsg})$: Given crs, sk_I, pk_U, and $(c_{\mathsf{SEP}}, \pi_{\mathsf{Open}}) := \mathsf{umsg}$, proceed as follows:

1. If $\Pi^{\mathcal{R}_{\mathsf{Com}}}_{\mathsf{NIZK}}.\mathsf{Verify}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}), \pi_{\mathsf{Open}}) = 0$: **abort**.
2. Compute $\tilde{k} \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda)$.
3. Compute $c'_{\mathsf{SEP}} \leftarrow c_{\mathsf{SEP}} + \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, (\mathbf{0}, \tilde{k}); \perp)$. ▶ *Shift c_{SEP} by a deterministic commitment to $(\mathbf{0}, \tilde{k})$*
4. Compute $\sigma'_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{SignCom}(\mathsf{crs}, \mathsf{sk}_I, c'_{\mathsf{SEP}})$.
5. Return $\mathsf{imsg} := (\tilde{k}, \sigma'_{\mathsf{SEP}})$.

- $\mathsf{Issue}.U_2(\mathsf{st}_U, \mathsf{imsg})$: Given $(\mathsf{pk}_I, \hat{k}, o_{\mathsf{SEP}}, \mathsf{sk}_U) := \mathsf{st}_U$, $(\tilde{k}, \sigma'_{\mathsf{SEP}}) := \mathsf{imsg}$, proceed as follows:

1. Compute $k \leftarrow \hat{k} + \tilde{k}$.
2. Parse $(\sigma'', o_0) := \sigma'_{\mathsf{SEP}}$ and compute $\sigma_{\mathsf{SEP}} \leftarrow (\sigma'', o_{\mathsf{SEP}} + o_0)$.
3. If $\Pi_{\mathsf{SEP}}.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, (\mathsf{sk}_U, k), \sigma_{\mathsf{SEP}}) = 0$: **abort**.
4. Initialize an empty map $\mathsf{Map} : \mathfrak{C} \to \{\perp\} \cup \mathbb{N}$.
5. Return $\mathcal{T} := (\mathsf{sk}_U, k, \sigma_{\mathsf{SEP}}, \mathsf{Map})$.

- $\mathsf{Show}(\mathsf{crs}, \mathsf{pk}_I, \mathcal{T}, \mathsf{ctxt}, \mathsf{nonce})$: Given crs, pk_I, $\mathcal{T}$, $\mathsf{ctxt} \in \mathfrak{C}$, nonce, proceed as follows:

1. Parse $(\mathsf{sk}_U, k, \sigma_{\mathsf{SEP}}, \mathsf{Map}) := \mathcal{T}$.
2. If $(\mathsf{Map}[\mathsf{ctxt}] > \mathfrak{n})$ or $(\sum_{\mathsf{ctxt} \in \mathsf{Map}} \mathsf{Map}[\mathsf{ctxt}] > \mathfrak{t})$: **abort**.
3. Compute $(c_{M,\Sigma}, \pi_{M,\Sigma}, o_{M,\Sigma}) \leftarrow \Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := \mathsf{pk}_I, \mathbb{w} := (\mathsf{sk}_U, k, \sigma_{\mathsf{SEP}}))$.
4. Set $j \leftarrow \mathsf{Map}[\mathsf{ctxt}] + 1$ and compute $(c_j, o_j) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, j)$.
5. Compute $\mathsf{sn} \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k, 0 \| \mathsf{ctxt} \| j)$.
6. Compute $\pi_{\mathsf{sn}} \leftarrow \Pi^{\mathcal{R}_{\mathsf{1Many}}}_{\mathsf{NIZK}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, \mathsf{sn}, \{x_i\}_{i \in [\mathfrak{n}]}), \mathbb{w} := ((\mathsf{sk}_U, k), o_{M,\Sigma}, j, o_j))$, *where* $x_i = 0 \| \mathsf{ctxt} \| i$. ▶ $\Pi^{\mathcal{R}_{\mathsf{1Many}}}_{\mathsf{NIZK}}$ *is used only to prove $\mathcal{R}_{\mathsf{sn}}$ in equation (3.4), given $c_{M,\Sigma}$, c_j, and $y_1 = \mathsf{sn}$.*
7. Compute $a \leftarrow \mathsf{H}(\mathsf{nonce}, \pi_{\mathsf{sn}})$, with $a \in \mathcal{S}$.

8. Compute $\mathsf{tg} \leftarrow \mathsf{pk}_U + a \cdot \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k, 1\|\mathsf{ctxt}\|j)$.
9. Compute $\pi_{\mathsf{tg}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, a, \mathsf{tg}, \{x'_i\}_{i\in[\mathfrak{n}]}), \mathbb{w} := ((\mathsf{sk}_U, k), o_{M,\Sigma}, j, o_j))$, *where* $x'_i = 1\|\mathsf{ctxt}\|i$. ▶ $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ *is used only to prove* $\mathcal{R}_{\mathsf{tg}}$ *in equation (3.4), given* $c_{M,\Sigma}$, c_j, *and* $y_2 = \mathsf{tg}$.
10. Update $\mathsf{Map}[\mathsf{ctxt}] \leftarrow \mathsf{Map}[\mathsf{ctxt}] + 1$.
11. Return $\mathsf{sn}, \tau := (\mathsf{tg}, c_j, c_{M,\Sigma}, \pi_{M,\Sigma}, \pi_{\mathsf{sn}}, \pi_{\mathsf{tg}})$ and $\mathcal{T}' := (\mathsf{sk}_U, k, \sigma_{\mathsf{SEP}}, \mathsf{Map})$.

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau, \mathsf{ctxt}, \mathsf{nonce})$: Given crs, pk_I, sn, $\mathsf{ctxt} \in \mathfrak{C}$, τ and nonce, proceed as follows:
  1. Parse $(\mathsf{tg}, c_j, c_{M,\Sigma}, \pi_{M,\Sigma}, \pi_{\mathsf{sn}}, \pi_{\mathsf{tg}}) := \tau$.
  2. Compute $a \leftarrow \mathsf{H}(\mathsf{nonce}, \pi_{\mathsf{sn}})$ and check if $a \in \mathcal{S}$. If not, return 0.
  3. Return $\left(\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_I, c_{M,\Sigma}), \pi_{M,\Sigma}) = 1\right)$
     $\wedge \left(\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, \mathsf{sn}, \{x_i\}_{i\in[\mathfrak{n}]}), \pi_{\mathsf{sn}}) = 1\right)$
     $\wedge \left(\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, a, \mathsf{tg}, \{x'_i\}_{i\in[\mathfrak{n}]}), \pi_{\mathsf{tg}}) = 1\right)$
     where $x_i = 0\|\mathsf{ctxt}\|i, x'_i = 1\|\mathsf{ctxt}\|i$.

- $\mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau_0, \tau_1, \mathsf{ctxt}, \mathsf{nonce}_0, \mathsf{nonce}_1)$: Given crs, sn, τ_0, τ_1, $\mathsf{ctxt} \in \mathfrak{C}$, nonce_0 and nonce_1, proceed as follows:
  1. If $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau_0, \mathsf{ctxt}, \mathsf{nonce}_0) = 0$ or $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}, \tau_1, \mathsf{ctxt}, \mathsf{nonce}_1) = 0$ or $(\mathsf{nonce}_0 = \mathsf{nonce}_1)$: **abort**.
  2. Parse $(\mathsf{tg}_b, c_{j_b}, c_{M,\Sigma_b}, \pi_{M,\Sigma_b}, \pi_{\mathsf{sn}_b}, \pi_{\mathsf{tg}_b}) := \tau_b$, with $b \in \{0, 1\}$.
  3. Compute $a_b \leftarrow \mathsf{H}(\mathsf{nonce}_b, \pi_{\mathsf{sn}_b})$.
  4. Compute $\mathsf{pk}'_U \leftarrow \mathsf{tg}_0 - \frac{a_0}{a_0 - a_1} \cdot (\mathsf{tg}_0 - \mathsf{tg}_1)$.
  5. Return pk'_U.

Remark 6.1 (Users' IDs). For users' identities, it suffices for the identity space ID to be a sufficiently large subset of $\mathcal{M}_{\mathsf{SEP}}$, so that users can *commit* to their identities and obtain signatures them. In addition, a separate scheme Π_{Com}, different from $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$, can be used to commit to identities, as long as it interacts nicely with $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$ and $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$. Furthermore, if Π_{Com} is such that computing a deterministic commitment to a message m by evaluating $\Pi_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, m; \perp)$ is equivalent to evaluating a one-way function (e.g. [Ped91, Ajt96, DF02]), then users can use random ids and deterministically commit to them.

Remark 6.2 (Domain Separation). Although we are using a single random oracle for all our building blocks, we note that standard domain separation techniques can be applied to obtain different random oracles tailored to each building block.

The correctness of construction 6.1 follows directly from inspection and from the correctness of the underlying primitives. Therefore, we focus on proving that it satisfies anonymity, soundness, linkability, and exculpability.

Theorem 6.1 (Anonymity)**.** *Assume* H *is a random oracle. If* Π_{PRF} *is secure,* $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$ *is hiding,* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$, $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, *and* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ *are zero-knowledge, then Construction 6.1 satisfies the anonymity property (see Definition 4.1).*

Proof. Let $\lambda \in \mathbb{N}$, $\mathfrak{n} = \mathfrak{n}(\lambda)$, and $\mathfrak{t} = \mathfrak{t}(\lambda)$. Let $\mathsf{Sim}_{\mathsf{Com}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}})$, $\mathsf{Sim}_{M,\Sigma} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}})$, $\mathsf{Sim}_{\mathsf{1Many}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}})$ be the zero-knowledge simulator of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$, $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, and $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$, respectively. We describe our simulator for the anonymity game:

Simulator 6.1. The simulator keeps track of and answers to random oracle queries. In addition, it implements the random oracle for other simulators that it calls.

- $\mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, \mathfrak{n}, \mathfrak{t})$: Given λ, the rate limit per context $\mathfrak{n}$, and the total limit $\mathfrak{t}$, proceed as follows:
  1. Compute $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{Com}}) \leftarrow \mathsf{Sim}_{\mathsf{Com}}.\mathsf{Setup}(1^\lambda)$.
  2. Compute $\mathsf{crs}_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{Setup}(1^\lambda; \mathsf{crs}_{\mathsf{Com}})$. ▶ *Here, the simulator sets* $\mathsf{crs}_{\mathsf{Com}}$ *of the commitment associated to* Π_{SEP} *to be the one it computed*
  3. Compute $(\mathsf{crs}_{M,\Sigma}, \mathsf{td}_{M,\Sigma}) \leftarrow \mathsf{Sim}_{M,\Sigma}.\mathsf{Setup}(1^\lambda, \mathsf{crs}_{\mathsf{SEP}})$.
  4. Compute $(\mathsf{crs}_{\mathsf{1Many}}, \mathsf{td}_{\mathsf{1Many}}) \leftarrow \mathsf{Sim}_{\mathsf{1Many}}.\mathsf{Setup}(1^\lambda, \mathsf{crs}_{M,\Sigma})$.
  5. Initialize an empty state $\mathsf{st}_{\mathsf{Sim}}$.
  6. Set $\mathsf{crs} := (\mathfrak{n}, \mathfrak{t}, \mathsf{crs}_{\mathsf{SEP}}, \mathsf{crs}_{M,\Sigma}, \mathsf{crs}_{\mathsf{1Many}})$ and $\mathsf{td} := (\mathsf{td}_{\mathsf{Com}}, \mathsf{td}_{M,\Sigma}, \mathsf{td}_{\mathsf{1Many}})$.
- $\mathsf{Sim}_{\mathsf{UKeyGen}}(\mathsf{crs}, \mathsf{id}_U)$: Given crs and $\mathsf{id}_U \in \mathcal{M}_{\mathsf{SEP}}$, proceed as follows:
  1. Compute $(c_U, o_U) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, \mathbf{0})$. Then return c_U as pk_U.
- $\mathsf{Sim}_{U_1}(\mathsf{td}, \mathsf{pk}_U, \mathsf{pk}_I)$: Given the trapdoor td, pk_U, and pk_I, proceed as follows:
  1. Compute $(c_{\mathsf{SEP}}, o_{\mathsf{SEP}}) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, (\mathbf{0}, \mathbf{0}))$.
  2. Compute $\pi_{\mathsf{Open}} \leftarrow \mathsf{Sim}_{\mathsf{Com}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathsf{td}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}))$.
  3. Store $(\mathsf{pk}_I, \mathsf{pk}_U, o_{\mathsf{SEP}})$ in $\mathsf{st}_{\mathsf{Sim}}$.
  4. Return $\mathsf{umsg} := (c_{\mathsf{SEP}}, \pi_{\mathsf{Open}})$.
- $\mathsf{Sim}_{U_2}(\mathsf{td}, \mathsf{imsg})$: Given the trapdoor td and $(\tilde{k}, \sigma'_{\mathsf{SEP}}) := \mathsf{imsg}$, proceed as follows:
  1. Retrieve $(\mathsf{pk}_I, \mathsf{pk}_U, o_{\mathsf{SEP}})$ corresponding to imsg from $\mathsf{st}_{\mathsf{Sim}}$. ▶*This can be made formal by using a session id to keep track of which* Sim_{U_1}*'s message correspond to* imsg.
  2. Parse $(\sigma'', o_0) := \sigma'_{\mathsf{SEP}}$ and compute $\sigma_{\mathsf{SEP}} \leftarrow (\sigma'', o_{\mathsf{SEP}} + o_0)$.
  3. If $\Pi_{\mathsf{SEP}}.\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, (\mathbf{0}, \tilde{k}), \sigma_{\mathsf{SEP}}) = 0$: return $(\mathcal{T} := \bot)$.
  4. Return **ack**.
- $\mathsf{Sim}_{\mathsf{Show}}(\mathsf{td}, \mathsf{pk}_I, \mathsf{ctxt}, \mathsf{nonce})$: Given the trapdoor td, pk_I, ctxt, and nonce, proceed as follows:
  1. Sample $\mathsf{sn} \leftarrow\!\!\$\ \mathcal{Y}$ and $\mathsf{tg} \leftarrow\!\!\$\ \mathcal{Y}$.
  2. Compute $(c_{M,\Sigma}, \pi_{M,\Sigma}) \leftarrow \mathsf{Sim}_{M,\Sigma}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathsf{td}, \mathbb{x} := \mathsf{pk}_I)$.
  3. Compute $(c_j, o_j) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, \mathbf{0})$.
  4. Compute $\pi_{\mathsf{sn}} \leftarrow \mathsf{Sim}_{\mathsf{1Many}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathsf{td}, \mathbb{x} := (c_{M,\Sigma}, c_j, \mathsf{sn}, \{x_i\}_{i\in[\mathfrak{n}]}))$, where $x_i = 0\|\mathsf{ctxt}\|i$.
  5. Compute $a \leftarrow \mathsf{H}(\mathsf{nonce}, \pi_{\mathsf{sn}})$, with $a \in \mathcal{S}$.
  6. Compute $\pi_{\mathsf{tg}} \leftarrow \mathsf{Sim}_{\mathsf{1Many}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, a, \mathsf{tg}, \{x'_i\}_{i\in[\mathfrak{n}]}))$, where $x'_i = 0\|\mathsf{ctxt}\|i$.
  7. Return $\mathsf{sn}, \tau := (\mathsf{tg}, c_j, c_{M,\Sigma}, \pi_{M,\Sigma}, \pi_{\mathsf{sn}}, \pi_{\mathsf{tg}})$.

Let $\mathcal{A}$ be a ppt that plays the anonymity experiment (see Definition 4.1). We proceed via a sequence of hybrids:

- Hyb_0: This is the real anonymity experiment:

1. At initialization, the challenger computes $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t})$ and sends crs to $\mathcal{A}$. Then, $\mathcal{A}$ outputs pk_I, the issuer public key.
2. Upon receiving a NewUser, U_1, U_2, or Show query from $\mathcal{A}$, the challenger replies by executing UKeyGen, Issue.U_1, Issue.U_2, and Show algorithm, respectively.

- Hyb_1: Similar to Hyb_0. However, during initialization, the challenger computes $(\mathsf{crs}_{\mathsf{1Many}}, \mathsf{td}_{\mathsf{1Many}}) \leftarrow \mathsf{Sim}_{\mathsf{1Many}}.\mathsf{Setup}(1^\lambda, \mathsf{crs}_{M,\Sigma})$. Next, during the execution of a Show query, the challenger uses $\mathsf{Sim}_{\mathsf{1Many}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{1Many}}, \mathsf{td}_{\mathsf{1Many}}, \cdot)$ to compute π_{sn} and π_{tg}.

  $\mathsf{Hyb}_1 \overset{c}{\approx} \mathsf{Hyb}_0$ by the zero-knowledge property of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$.

- Hyb_2: Simlar to Hyb_1. However, during initialization, the challenger computes $(\mathsf{crs}_{M,\Sigma}, \mathsf{td}_{M,\Sigma}) \leftarrow \mathsf{Sim}_{M,\Sigma}.\mathsf{Setup}(1^\lambda, \mathsf{crs}_{\mathsf{SEP}})$. Next, during the execution of a Show query, the challenger uses $\mathsf{Sim}_{M,\Sigma}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}_{M,\Sigma}, \mathsf{td}_{M,\sigma}, \cdot)$ to compute $c_{M,\Sigma}, \pi_{M,\Sigma}$.

  $\mathsf{Hyb}_2 \overset{c}{\approx} \mathsf{Hyb}_1$ by the zero-knowledge property of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$.

- Hyb_3: Similar to Hyb_2. However, during initialization, the challenger computes $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{Com}}) \leftarrow \mathsf{Sim}_{\mathsf{Com}}.\mathsf{Setup}(1^\lambda)$, then it uses $\mathsf{crs}_{\mathsf{Com}}$ to initialize SEP by computing $\mathsf{crs}_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{Setup}(1^\lambda; \mathsf{crs}_{\mathsf{Com}})$. Next, during the execution of a U_1 query, the challenger uses $\mathsf{Sim}_{\mathsf{Com}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{Com}}, \cdot)$ to compute π_{Open}.

  $\mathsf{Hyb}_3 \overset{c}{\approx} \mathsf{Hyb}_2$ by the zero-knowledge property of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$.

- Hyb_4: Similar to Hyb_3. However, during the execution of a Show query, the challenger samples $\mathsf{sn} \leftarrow\!\!\$\ \mathcal{Y}$ and $\mathsf{tg} \leftarrow\!\!\$\ \mathcal{Y}$.

  $\mathsf{Hyb}_4 \overset{c}{\approx} \mathsf{Hyb}_3$ by the security of Π_{PRF}.

- Hyb_5: Similar to Hyb_4. However, during the execution of a NewUser, U_1, or Show query, the challenger replaces all $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$ commitments by commitments to zero.

  $\mathsf{Hyb}_5 \overset{c}{\approx} \mathsf{Hyb}_4$ by the hiding property of $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$. Note that in Hyb_5, the challenger executes the simulator 6.1, and this correspond to the simulated anonymity experiment.

This completes the proof. □

Theorem 6.2 (Soundness)**.** *Assume* H *is a random oracle. Suppose further that:*

- *In* Π_{SEP}*, the underlying signature is unforgeable, and the associated commitment,* $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$*, is binding (see Definition 3.4).*
- *The commitment underlying the commit-and-prove NIZK* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *is binding.*
- $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$ *is knowledge-sound with straight-line extraction;* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *and* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ *are knowledge-sound with rewindable extraction; and all three proof systems have negligible knowledge-soundness error.*

Proof. At a high level, we show that if an adversary outputs a serial number that was not generated by evaluating Π_{PRF} on one of the signed PRF keys and a given ctxt, then the adversary must have broken Π_{SEP}'s unforgeability. Let $\lambda \in \mathbb{N}, \mathfrak{n} = \mathfrak{n}(\lambda)$, and $\mathfrak{t} = \mathfrak{t}(\lambda)$. Let $\mathsf{Ext}_{\mathsf{Com}} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$ be the extractor for $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$, and let κ_{Com} be its knowledge-soundness error. Consider the following extractor $\mathsf{Ext} = (\mathsf{Ext}_{\mathsf{Setup}}, \mathsf{Ext}_{\mathsf{Issue}})$ for the soundness game:

- $\mathsf{Ext}_{\mathsf{Setup}}(1^\lambda, \mathfrak{n}, \mathfrak{t})$: Given $\lambda, \mathfrak{n}$, and $\mathfrak{t}$, execute $\mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t})$, with the following modification: compute $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{Com}}) \leftarrow \mathsf{Ext}_{\mathsf{Com}}.\mathsf{Ext}_1(1^\lambda)$, and then compute $\mathsf{crs}_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{Setup}(1^\lambda; \mathsf{crs}_{\mathsf{Com}})$. Finally, store $\mathsf{td}_{\mathsf{Com}}$ in memory.
- $\mathsf{Ext}_{\mathsf{Issue}}(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U, \mathsf{umsg})$: Given the crs, sk_I, pk_U, and $(c_{\mathsf{SEP}}, \pi_{\mathsf{Open}}) := \mathsf{umsg}$, proceed as follows.
  - First compute $(\tilde{k}, \sigma'_{\mathsf{SEP}}) \leftarrow \mathsf{Issue}.I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{pk}_U, \mathsf{umsg})$.
  - Then extract $\mathbb{w}_U \leftarrow \mathsf{Ext}_{\mathsf{Com}}.\mathsf{Ext}_2(\mathsf{crs}, \mathsf{td}_{\mathsf{Com}}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}), \pi_{\mathsf{Open}})$.
  - If $\mathbb{w}_U = \perp$, **abort**.
  - Otherwise, parse $(\mathsf{sk}_U, \hat{k}, o_{\mathsf{SEP}}) := \mathbb{w}_U$, store $(\mathsf{sk}_U, \hat{k}, o_{\mathsf{SEP}}, \tilde{k}, \sigma'_{\mathsf{SEP}})$ in memory, and return $(\tilde{k}, \sigma'_{\mathsf{SEP}})$.

Let $\mathcal{A}$ be a ppt adversary that, after making Q_{Iss} Issue queries, outputs a tuple $(\mathsf{ctxt}, \mathsf{sn}, \tau, \mathsf{nonce})$ such that sn was not generated by any of the PRF keys it received in response to those issue queries. We show, via a sequence of hybrids, that this occurs with negligible probability.

- Hyb_0 : In this hybrid, $\mathcal{A}$ interacts with an issuer running Issue. Let E_0 be the event that $\mathcal{A}$ outptus a valid tuple $(\mathsf{ctxt}, \mathsf{sn}, \tau, \mathsf{nonce})$ such that sn was not computed using one of the issued PRFs keys.
- Hyb_1 : Same has hybrid Hyb_0. However, Setup is replaced by $\mathsf{Ext}_{\mathsf{Setup}}$ and Issue is replaced by $\mathsf{Ext}_{\mathsf{Issue}}$.

  Let E_1 be the event that $\mathcal{A}$ outputs a valid tuple such that for all extracted tuples $(\mathsf{sk}_{U_i}, \hat{k}_i, o_{\mathsf{SEP}_i}, \tilde{k}_i, \sigma'_{\mathsf{SEP}_i})$, with $i \in [Q_{\mathsf{Iss}}]$, and $j \in [\mathfrak{n}]$, $\mathsf{sn} \neq \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_i, 0 \| \mathsf{ctxt} \| j)$, where $k_i = \hat{k}_i + \tilde{k}_i$. By the properties of $\mathsf{Ext}_{\mathsf{Com}}$,

$$\Pr[\mathsf{E}_1] \geq \Pr[\mathsf{E}_0] - \epsilon_{\mathsf{dist},\mathsf{crs}_{\mathsf{Com}}} - Q_{\mathsf{Iss}} \cdot \kappa_{\mathsf{Com}} \tag{6.1}$$

  where $\epsilon_{\mathsf{dist},\mathsf{crs}_{\mathsf{Com}}}$ is the advantage of a reduction, built from $\mathcal{A}$, in distinguishing $\mathsf{crs}_{\mathsf{Com}}$ in Hyb_0 from $\mathsf{crs}_{\mathsf{Com}}$ in Hyb_1.

Now, we analyze E_1. Let $\mathsf{Ext}_{M,\Sigma}$, $\mathsf{Ext}_{\mathsf{1Many}}$ be the extractors for $\Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}}$, and $\Pi^{\mathcal{R}_{\mathsf{1Many}}}_{\mathsf{NIZK}}$, respectively, and let $\kappa_{M,\Sigma}$, and κ_{1Many} be their knowledge-soundness errors. Consider the following extraction process:

- Parse $(\mathsf{tg}, c_j, c_{M,\Sigma}, \pi_{M,\Sigma}, \pi_{\mathsf{sn}}, \pi_{\mathsf{tg}}) := \tau$ and compute $a \leftarrow \mathsf{H}(\mathsf{nonce}, \pi_{\mathsf{sn}})$.
- Run $\mathbb{w}_{M,\Sigma} \leftarrow \mathsf{Ext}^{\mathcal{A}}_{M,\Sigma}(\mathsf{crs}, \mathbb{x} := \mathsf{pk}_I, (c_{M,\Sigma}, \pi_{M,\Sigma}))$. If $\mathbb{w}_{M,\Sigma} = \perp$, abort. Else, parse $(\mathsf{sk}^*_U, k^*, \sigma^*_{\mathsf{SEP}}, o^*_{M,\Sigma}) := \mathbb{w}_{M,\Sigma}$.
- Rewind $\mathcal{A}$ to the point where it outputs its response.
- Run $\mathbb{w}_{\mathsf{1Many},\mathsf{sn}} \leftarrow \mathsf{Ext}^{\mathcal{A}}_{\mathsf{1Many}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, \mathsf{sn}, \{x_i\}_{i\in[n]}), \pi_{\mathsf{sn}})$, where $x_i = 0 \| \mathsf{ctxt} \| i$. If $\mathbb{w}_{\mathsf{1Many},\mathsf{sn}} = \perp$, abort. Else, parse $(\bar{\mathsf{sk}}_U, \bar{k}, \bar{j}, \bar{o}_{M,\Sigma}, \bar{o}_j) := \mathbb{w}_{\mathsf{1Many},\mathsf{sn}}$.
- Rewind $\mathcal{A}$ to the point where it outputs its response.
- Run $\mathbb{w}_{\mathsf{1Many},\mathsf{tg}} \leftarrow \mathsf{Ext}^{\mathcal{A}}_{\mathsf{1Many}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma}, c_j, a, \mathsf{tg}, \{x'_i\}_{i\in[n]}), \pi_{\mathsf{tg}})$, where $x'_i = 1 \| \mathsf{ctxt} \| i$. If $\mathbb{w}_{\mathsf{1Many},\mathsf{tg}} = \perp$, abort. Else parse $(\check{\mathsf{sk}}_U, \check{k}, \check{j}, \check{o}_{M,\Sigma}, \check{o}_j) := \mathbb{w}_{\mathsf{1Many},\mathsf{tg}}$.

Note that this extraction procedure does not incur nested rewinding, since $\mathcal{A}$ is never rewound during its interaction with the issuer. Conditioned on the success of $\mathsf{Ext}_{M,\Sigma}$ and $\mathsf{Ext}_{\mathsf{1Many}}$, it follows that by the binding property of the commitment underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, we have $(\mathsf{sk}_U^*, k^*) = (\bar{\mathsf{sk}}_U, \bar{k}) = (\check{\mathsf{sk}}_U, \check{k})$, and by the binding property of $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$, we have $\bar{j} = \check{j} = j^*$, where $j^* \in [\mathfrak{n}]$. In addition, by the knowledge-soundness of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$, we have $\mathsf{sn} = \mathsf{PRF.Eval}^{\mathsf{H}}(k^*, 0\|\mathsf{ctxt}\|j^*)$, which implies that (sk_U^*, k^*) was not signed by the issuer during its initial interaction with $\mathcal{A}$. Now, we argue that, even after rewinding $\mathcal{A}$, (sk_U^*, k^*), together with σ_{SEP}^*, is still a valid forgery against Π_{SEP}. To this end, we build a reduction $\mathcal{B}$ against the unforgeability of Π_{SEP} that works as follows:

- $\mathcal{B}$ plays the role of the challenger in Hyb_1 and sets the issuer public key to be the public of the Π_{SEP}'s EUF-CMA challenger.
- For each issue query on $\mathsf{umsg} := (c_{\mathsf{SEP}}, \pi_{\mathsf{Open}})$ with respect to a public key pk_U, $\mathcal{B}$ computes $\mathbb{w}_U \leftarrow \mathsf{Ext}_{\mathsf{Com}}.\mathsf{Ext}_2(\mathsf{crs}, \mathsf{td}_{\mathsf{Com}}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}), \pi_{\mathsf{Open}})$. If $\mathbb{w}_U = \bot$, $\mathcal{B}$ outputs $\bot$. Otherwise, $\mathcal{B}$ parses $(\mathsf{sk}_U, \hat{k}, o_{\mathsf{SEP}}) := \mathbb{w}_U$, samples $\tilde{k} \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda)$, and forwards $(\mathsf{sk}_U, \hat{k} + \tilde{k})$ to Π_{PRF}'s unforgeability challenger. Upon receiving $\sigma_{\mathsf{SEP}} := (\sigma, o)$ from EUF-CMA challenger, $\mathcal{B}$ derives $\sigma'_{\mathsf{SEP}} \leftarrow (\sigma, o - o_{\mathsf{SEP}})$ and forward $(\tilde{k}, \sigma'_{\mathsf{SEP}})$ to $\mathcal{A}$ as answer to its Issue query.
- When $\mathcal{A}$ outputs its response, $\mathcal{B}$ executes the extraction process described above and returns $((\mathsf{sk}_U^*, k^*), \sigma_{\mathsf{SEP}}^*)$ as its candidate forgery.

Let $r_{M,\Sigma}$ and r_{1Many} be the number of times that $\mathsf{Ext}_{M,\Sigma}$ and $\mathsf{Ext}_{\mathsf{1Many}}$ rewind $\mathcal{A}$, respectively. Since $\mathcal{A}$ may keep issuing Issue queries during those rewindings, $\mathcal{B}$ answers all such queries successfully except with probability at most $(r_{M,\Sigma} + r_{\mathsf{1Many}}) \cdot Q_{\mathsf{Iss}} \cdot \kappa_{\mathsf{Com}}$. Now, (sk_U^*, k^*) appears in one of the i-th subsequent Issue queries performed by $\mathcal{A}$ if $\mathcal{B}$ samples $\tilde{k}_i \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda)$ such that $\tilde{k}_i = k^* - \hat{k}_i$, where $\hat{k}_i$ is $\mathcal{A}$'s contribution. By the high-minentropy of $\Pi_{\mathsf{PRF}}.\mathsf{Gen}$ (see Remark 3.4), this happens with probability at most $2^{-\omega(\log\lambda)}$. Since $\mathcal{A}$ can perform at most $(r_{M,\Sigma} + r_{\mathsf{1Many}}) \cdot Q_{\mathsf{Iss}}$ Issue queries during extraction, it follows that, except with probability $(r_{M,\Sigma} + r_{\mathsf{1Many}}) \cdot Q_{\mathsf{Iss}} \cdot 2^{-\omega(\log\lambda)}$, (sk_U^*, k^*) was not queried to the EUF-CMA challenger. Therefore,

$$\begin{aligned}\Pr[\mathcal{B}\ \mathsf{wins}] \geq \Pr[\mathsf{E}_1] - \kappa_{M,\sigma} - 2 \cdot \kappa_{\mathsf{1Many}} - \epsilon_{\mathsf{Com},M,\Sigma} - \epsilon_{\mathsf{Com,SEP}} \\ - (r_{M,\Sigma} + r_{\mathsf{1Many}}) \cdot Q_{\mathsf{Iss}} \cdot (2^{-\omega(\log\lambda)} + \kappa_{\mathsf{Com}})\end{aligned} \tag{6.2}$$

Where $\epsilon_{\mathsf{Com},M,\Sigma}$ and $\epsilon_{\mathsf{Com,SEP}}$ denote the success probabilities of reductions, built from $\mathcal{A}$, in breaking the binding of the commitment underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ and of $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$, respectively. By combining equations (6.1) and (6.2), and by the unforgeability of Π_{SEP}, it follows that Construction 6.1 is sound. Note that $r_{M,\Sigma}$ and r_{1Many} are only upper-bounded by a polynomial in expectation. However, by a truncation argument using Markov's inequality, they can be made strictly upper-bounded by a polynomial, at the cost of a polynomial loss in $\mathcal{B}$'s success probability (see Lemma 2.2). □

Theorem 6.3 (Linkability)**.** *Assume* H *is a random oracle. Suppose* Π_{PRF} *is a secure* δ*-almost key homomorphic PRF such that, for any* $\lambda \in \mathbb{N}$, $|\mathcal{Y}| \geq 2^{\omega(\log\lambda)}|\mathcal{S}_\delta|$*, where* $\mathcal{Y}$ *is the codomain of* Π_{PRF} *and* $\mathcal{S}_\delta$ *is the set of elements in* $\mathcal{Y}$ *with norm at most* δ*. Suppose further that:*

- *In* Π_{SEP}*, the underlying signature is unforgeable, and the associated commitment,* $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$*, is binding (see Definition 3.4).*
- *The commitment underlying the commit-and-prove NIZK* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *is binding.*
- $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$ *is knowledge-sound with straight-line extraction;* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *and* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ *are knowledge-sound with rewindable extraction; and all three proof systems have negligible knowledge-soundness error.*

Then Construction 6.1 is linkable (see Definition 4.1).

Proof. Let $\lambda \in \mathbb{N}$ and $\mathfrak{n} = \mathfrak{n}(\lambda)$. Let $\mathcal{A}$ be a ppt adversary that, after performing a total of Q_{H} queries to H and Q_{Iss} Issue requests, outputs a tuple $(\mathsf{ctxt}, (\mathsf{sn}_b, \tau_b, \mathsf{nonce}_b)_{b\in\{0,1\}})$ that breaks the linkability requirement, i.e., the $\mathsf{Identify}$ algorithm, when executed on this tuple, returns an identity not associated with $\mathcal{A}$. We show that this happens with negligible probability. At a high level, the proof first establishes the well-formedness of the PRF keys via a sequence of witness extractions, and then invokes Lemma 3.1 to argue that any double-spending event yielding a key not associated with $\mathcal{A}$ happens only with negligible probability.

Let $\mathsf{Ext}_{\mathsf{Com}} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$, $\mathsf{Ext}_{M,\Sigma}$, $\mathsf{Ext}_{\mathsf{1Many}}$ be the extractors for $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$, $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, and $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$, respectively, and let κ_{Com}, $\kappa_{M,\Sigma}$, and κ_{1Many} be their knowledge-soundness errors. We proceed via sequence of hyrids:

- Hyb_0: This is the original linkability experiment. Let E_0 be the event that $\mathcal{A}$ outputs a tuple that breaks the linkability property in this hybrid.
- Hyb_1: Same as Hyb_0. However during Setup, first compute $(\mathsf{crs}_{\mathsf{Com}}, \mathsf{td}_{\mathsf{Com}}) \leftarrow \mathsf{Ext}_{\mathsf{Com}}.\mathsf{Ext}_1(1^\lambda)$, and then $\mathsf{crs}_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{Setup}(1^\lambda; \mathsf{crs}_{\mathsf{Com}})$. Let E_1 be the event that $\mathcal{A}$ outputs a tuple that breaks the linkability property in this hybrid. By the properties of $\mathsf{Ext}_{\mathsf{Com}}$,

$$\Pr[\mathsf{E}_1] \geq \Pr[\mathsf{E}_0] - \epsilon_{\mathsf{dist},\mathsf{crs}_{\mathsf{Com}}} \tag{6.3}$$

  where $\epsilon_{\mathsf{dist},\mathsf{crs}_{\mathsf{Com}}}$ is the advantage of a reduction, built from $\mathcal{A}$, in distinguishing $\mathsf{crs}_{\mathsf{Com}}$ in Hyb_0 from $\mathsf{crs}_{\mathsf{Com}}$ in Hyb_1.

Now, we analyze E_1. Upon receiving $(\mathsf{ctxt}, (\mathsf{sn}_b, \tau_b, \mathsf{nonce}_b)_{b\in\{0,1\}})$ from $\mathcal{A}$ during the linkability experiment, we apply the same extraction procedure described in the proof of Theorem 6.2:

- For $b \in \{0, 1\}$, proceed as follows:
  - Parse $(\mathsf{tg}_b, c_{j_b}, c_{M,\Sigma_b}, \pi_{M,\Sigma_b}, \pi_{\mathsf{sn}_b}, \pi_{\mathsf{tg}_b}) := \tau_b$.
  - Compute $a_b \leftarrow \mathsf{H}(\mathsf{nonce}_b, \pi_{\mathsf{sn}_b})$, and if $a_0 = a_1$, abort.
  - Run $\mathbb{w}_{M,\Sigma_b} \leftarrow \mathsf{Ext}_{M,\Sigma}^{\mathcal{A}}(\mathsf{crs}, \mathbb{x} := \mathsf{pk}_I, (c_{M,\Sigma_b}, \pi_{M,\Sigma_b}))$. If $\mathbb{w}_{M,\Sigma_b} = \bot$, abort. Else, parse $(\mathsf{sk}_{U_b}, k_b, \sigma_{\mathsf{SEP}_b}, o_{M,\Sigma_b}) := \mathbb{w}_{M,\Sigma}$.
  - Rewind $\mathcal{A}$ to the point where it outputs its response.
  - Run $\mathbb{w}_{\mathsf{1Many},\mathsf{sn}_b} \leftarrow \mathsf{Ext}_{\mathsf{1Many}}^{\mathcal{A}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma_b}, c_{j_b}, \mathsf{sn}_b, \{x_i\}_{i\in[n]}), \pi_{\mathsf{sn}_b})$, where $x_i = 0\|\mathsf{ctxt}\|i$. If $\mathbb{w}_{\mathsf{1Many},\mathsf{sn}_b} = \bot$, abort. Else, parse $(\tilde{\mathsf{sk}}_{U_b}, \tilde{k}_b, \tilde{j}_b, \tilde{o}_{M,\Sigma_b}, \tilde{o}_{j_b}) := \mathbb{w}_{\mathsf{1Many},\mathsf{sn}_b}$.
  - Rewind $\mathcal{A}$ to the point where it outputs its response.
  - Run $\mathbb{w}_{\mathsf{1Many},\mathsf{tg}_b} \leftarrow \mathsf{Ext}_{\mathsf{1Many}}^{\mathcal{A}}(\mathsf{crs}, \mathbb{x} := (c_{M,\Sigma_b}, c_{j_b}, a_b, \mathsf{tg}_b, \{x'_i\}_{i\in[n]}), \pi_{\mathsf{tg}_b})$, where $x'_i = 1\|\mathsf{ctxt}\|i$. If $\mathbb{w}_{\mathsf{1Many},\mathsf{tg}_b} = \bot$, abort. Else, parse $(\hat{\mathsf{sk}}_{U_b}, \hat{k}_b, \hat{j}_b, \hat{o}_{M,\Sigma_b}, \hat{o}_{j_b}) := \mathbb{w}_{\mathsf{1Many},\mathsf{tg}_b}$.

As previously stated, this procedure does not incur nested rewinding since $\mathcal{A}$ is never rewound during its interaction with the issuer. In addition, by a union bound over the collision of the a_b's and the failure of each underlying extraction, the probability that the extraction procedure aborts is upper bounded by $Q_{\mathsf{H}}^2 \cdot 2^{-\omega(\log\lambda)} + 2(\kappa_{M,\Sigma} + 2\kappa_{\mathsf{1Many}})$, where the first summand comes from the fact that $a_b \in \mathcal{S}$ and $|\mathcal{S}| \geq 2^{\omega(\log\lambda)}$. Let Bad denote this failure event.

Now let Forge be the event that $\mathcal{A}$ breaks the unforgeability of the signature underlying Π_{SEP}. Then, conditioned on $\overline{\mathsf{Forge}}$ and $\overline{\mathsf{Bad}}$, (sk_{U_b}, k_b) must be a message signed by the issuer during the linkability experiment. By the execution of the Issue protocol, this implies that there exists $(k'_b, c_{\mathsf{SEP}_b}, \pi_{\mathsf{Open}_b}, \mathsf{pk}_{U_b})$ such

that $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Verify}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_{U_b}, c_{\mathsf{SEP}_b}), \pi_{\mathsf{Open}_b}) = 1$ and $c'_{\mathsf{SEP}_b} \leftarrow c_{\mathsf{SEP}_b} + \Pi_{\mathsf{Com}}^{\mathsf{SEP}}.\mathsf{Commit}(\mathsf{crs}, (\mathbf{0}, k'_b); \perp)$ is the commitment that was signed, where k'_b was honestly generated by the issuer. By running $\mathsf{Ext}_{\mathsf{Com}}.\mathsf{Ext}_2$ with $(\mathsf{crs}, \mathsf{td}_{\mathsf{Com}}, (\mathsf{pk}_{U_b}, c_{\mathsf{SEP}_b}), \pi_{\mathsf{Open}_b})$ as inputs, we extract $(\mathsf{sk}'_{U_b}, k''_b, o'_{\mathsf{SEP}_b})$ such that $k_b = k'_b + k''_b$ and $\mathsf{sk}'_{U_b} = \mathsf{sk}_{U_b}$, unless $\mathsf{Ext}_{\mathsf{Com}}$ fails or $\mathcal{A}$ breaks the binding of $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$. Let $\mathsf{F}_{\mathsf{Ext}_{\mathsf{com}}} \vee \mathsf{BrkBind}_{\mathsf{SEP}}$ be this event. By the law of total probability, it follows that

$$\begin{aligned}
\Pr[\mathsf{E}_1] &\leq \Pr[\mathsf{Bad}] + \Pr[\mathsf{Forge}, \overline{\mathsf{Bad}}] + \Pr[\mathsf{F}_{\mathsf{Ext}_{\mathsf{com}}} \vee \mathsf{BrkBind}_{\mathsf{SEP}}, \overline{\mathsf{Forge}}, \overline{\mathsf{Bad}}] \\
&\quad + \Pr[\mathsf{E}_1, \overline{\mathsf{F}_{\mathsf{Ext}_{\mathsf{Com}}}}, \overline{\mathsf{BrkBind}_{\mathsf{SEP}}}, \overline{\mathsf{Forge}}, \overline{\mathsf{Bad}}] \\
&\leq Q_{\mathsf{H}}^2 \cdot 2^{-\omega(\log \lambda)} + 2(\kappa_{M,\Sigma} + 2\kappa_{\mathsf{1Many}} + \epsilon_{\mathsf{Forge}} + \kappa_{\mathsf{Com}} + \epsilon_{\mathsf{Com,SEP}}) \\
&\quad + \Pr[\mathsf{E}_1, \overline{\mathsf{F}_{\mathsf{Ext}_{\mathsf{Com}}}}, \overline{\mathsf{BrkBind}_{\mathsf{SEP}}}, \overline{\mathsf{Forge}}, \overline{\mathsf{Bad}}] \qquad (6.4)
\end{aligned}$$

where $\epsilon_{\mathsf{Forge}}$ is the success probability of a reduction, built from $\mathcal{A}$, in the unforgeability experiment for the signature scheme underlying Π_{SEP} (see the proof of Theorem 6.2 for an extended argument), and $\epsilon_{\mathsf{Com,SEP}}$ is the success probability of a reduction, built from $\mathcal{A}$, in the binding experiment for $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$. Note that, in both the unforgeability and binding experiments, reductions run in expected poly-time, since they invoke $\mathsf{Ext}_{M,\Sigma}$, $\mathsf{Ext}_{\mathsf{1Many}}$. However, via a runtime-truncation argument based on Markov's inequality, these reductions can be transformed into strict poly-time adversaries at the cost of a polynomial loss in the final success probability (see Lemma 2.2).

Conditioned on $\overline{\mathsf{F}_{\mathsf{Ext}_{\mathsf{Com}}}}$, $\overline{\mathsf{BrkBind}_{\mathsf{SEP}}}$, $\overline{\mathsf{Forge}}$, and $\overline{\mathsf{Bad}}$, it follows that, unless $\mathcal{A}$ breaks the binding of the commitment underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, $(\mathsf{sk}_{U_b}, k_b) = (\tilde{\mathsf{sk}}_{U_b}, \tilde{k}_b) = (\hat{\mathsf{sk}}_{U_b}, \hat{k}_b)$. Moreover, unless $\mathcal{A}$ breaks the binding of $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$, we have $\tilde{j}_b = \hat{j}_b = j_b$, where $j_b \in [\mathfrak{n}]$. Therefore, $\mathsf{sn}_b = \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_b, 0 \| \mathsf{ctxt} \| j_b)$ and $\mathsf{tg}_b = \mathsf{pk}_{U_b} + a_b \cdot \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_b, 1 \| \mathsf{ctxt} \| j_b)$. We consider the following cases. First, suppose that $k_0 \neq k_1$. Since $\mathsf{sn}_0 = \mathsf{sn}_1$, this implies a collision on Π_{PRF}. However, since $k_b = k'_b + k''_b$, where k'_b is sampled honestly, and the domain over which Π_{PRF} is evaluated has size $2\mathfrak{n}|\mathfrak{C}| = \mathsf{poly}(\lambda)$, Lemma 3.1 implies that this happens with probability at most $Q_{\mathsf{Iss}} \cdot \epsilon_{\mathsf{PRF}} + \mathsf{negl}(\lambda)$. Second, suppose $k_0 = k_1$. There are two subcases. If $\mathsf{sk}_{U_0} = \mathsf{sk}_{U_1}$, then $\mathsf{Identify}$ returns a public key associated to $\mathcal{A}$. If $\mathsf{sk}_{U_0} \neq \mathsf{sk}_{U_1}$, then two different users were issued the same key, but this case happens with negligible probability by the high min-entropy of $\Pi_{\mathsf{PRF}}.\mathsf{Gen}$ (see Remark 3.4). Therefore,

$$\Pr[\mathsf{E}_1, \overline{\mathsf{F}_{\mathsf{Ext}_{\mathsf{Com}}}}, \overline{\mathsf{BrkBind}_{\mathsf{SEP}}}, \overline{\mathsf{Forge}}, \overline{\mathsf{Bad}}] \leq 2(\epsilon_{\mathsf{Com},M,\Sigma} + \epsilon_{\mathsf{Com,SEP}}) + Q_{\mathsf{Iss}} \cdot \epsilon_{\mathsf{PRF}} + \mathsf{negl}(\lambda) \qquad (6.5)$$

where $\epsilon_{\mathsf{Com},M,\Sigma}$ is the success probability of a reduction, built from $\mathcal{A}$, against the binding property of the commitment underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, and ϵ_{PRF} is the advantage of a reduction, built from $\mathcal{A}$, that tries to distinguish Π_{PRF} from a random function. By combining equations (6.3) to (6.5), and using the assumptions of the theorem, it follows that the probability that $\mathcal{A}$ breaks the linkability property of Construction 6.1 is negligible. □

Theorem 6.4 (Exculpability)**.** *Assume* H *is a random oracle. Suppose* Π_{PRF} *is a secure* δ*-almost key homomorphic PRF such that, for any* $\lambda \in \mathbb{N}$, $|\mathcal{Y}| \geq 2^{\omega(\log \lambda)}|\mathcal{S}_\delta|$*, where* $\mathcal{Y}$ *is the codomain of* Π_{PRF} *and* $\mathcal{S}_\delta$ *is the set of elements in* $\mathcal{Y}$ *with norm at most* δ*. Suppose further that:*

- *In* Π_{SEP}*, the underlying signature is unforgeable, and the associated commitment,* $\Pi_{\mathsf{Com}}^{\mathsf{SEP}}$*, is hiding and binding (see Definition 3.4).*
- *The commitment underlying the commit-and-prove NIZK* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *is binding.*
- $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}$ *is knowledge-sound with straight-line extraction;* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ *and* $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ *are knowledge-sound with rewindable extraction; and all three proof systems zero-knowledge and have negligible knowledge-soundness error.*

Then Construction 6.1 satisfies the exculpability property (see Definition 4.1).

Proof. Let $\lambda \in \mathbb{N}$, $\mathfrak{n} = \mathfrak{n}(\lambda)$, and $\mathfrak{t} = \mathfrak{t}(\lambda)$. Let $\mathsf{Sim} = (\mathsf{Sim}_{\mathsf{Setup}}, \mathsf{Sim}_{\mathsf{UKeyGen}}, \mathsf{Sim}_{U_1}, \mathsf{Sim}_{U_2}, \mathsf{Sim}_{\mathsf{Show}})$ be the simulator from simulator 6.1. Let $\mathcal{A}$ be a ppt adversary that, after performing Q_{H} random oracle queries, breaks the exculpability property of Construction 6.1. We show, via a sequence of hybrids, that this happen with negligible probability.

- Hyb_0: This is the Exculpability experiment:

  1. The challenger computes $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{t})$ and sends crs to $\mathcal{A}$. Then, $\mathcal{A}$ outputs pk_I, the issuer public key.
  2. Upon receiving a NewUser, U_1, U_2, or Show query from $\mathcal{A}$, the challenger replies by executing UKeyGen, Issue.U_1, Issue.U_2, and Show algorithm, respectively.
  3. $\mathcal{A}$ outputs $(\mathsf{ctxt}, (\mathsf{sn}_b, \tau_b, \mathsf{nonce}_b)_{b\in\{0,1\}})$.
  4. If $\mathsf{sn}_0 = \mathsf{sn}_1$ and $\mathsf{Identify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{sn}_0, \tau_0, \tau_1, \mathsf{ctxt}, \mathsf{nonce}_0, \mathsf{nonce}_1)$ returns a user key that was returned in response to a NewUser query, then challenger outputs 1. Else, it outputs 0.

  Let E_0 be the event that the challenger outputs 1 in this hybrid.

- Hyb_1: Similar to Hyb_0. However, during initialization, the challenger computes $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}_{\mathsf{Setup}}(1^\lambda, \mathfrak{n}, \mathfrak{t})$ and sends crs to $\mathcal{A}$. Next, the challenger answers to $\mathcal{A}$'s queries using $\mathsf{Sim}_{\mathsf{UKeyGen}}$, Sim_{U_1}, Sim_{U_2}, and $\mathsf{Sim}_{\mathsf{Show}}$.

  Let E_1 be the event that the challenger outputs 1 in this hybrid. By Theorem 6.1, it follows that

$$\Pr[\mathsf{E}_0] \leq \Pr[\mathsf{E}_1] + \epsilon_{\mathsf{ANON}}, \tag{6.6}$$

  where ϵ_{ANON} is the advantage of a reduction $\mathcal{B}_{\mathsf{ANON}}(\mathcal{A})$ against the Anonymity Experiment of Construction 6.1.

Now, we analyze E_1. Parse $(\mathsf{tg}_b, c_{j_b}, c_{M,\Sigma_b}, \pi_{M,\Sigma_b}, \pi_{\mathsf{sn}_b}, \pi_{\mathsf{tg}_b}) := \tau_b$, with $b \in \{0,1\}$, and let $a_b \leftarrow \mathsf{H}(\mathsf{nonce}_b, \pi_{\mathsf{sn}_b})$. Note that Identify fails to outputs a public key if $a_0 = a_1$, but this happens with probability at most $Q_{\mathsf{H}}^2/2^{\omega(\log\lambda)}$, since $|\mathcal{S}| \geq 2^{\omega(\log\lambda)}$. Now, conditioned on $a_0 \neq a_1$, if pk' is the user key returned by the Identify algorithm, then $\mathsf{pk}' = \mathsf{tg}_0 - \frac{a_0}{a_0 - a_1}(\mathsf{tg}_0 - \mathsf{tg}_1)$, which implies

$$(a_0 - a_1) \cdot \mathsf{pk}' = a_0 \cdot \mathsf{tg}_1 - a_1 \cdot \mathsf{tg}_0 \tag{6.7}$$

By using the extraction procedure describe in the proofs of Theorems 6.2 and 6.3 on $(\mathsf{ctxt}, \mathsf{sn}_1, \tau_1, \mathsf{nonce}_1)$, we extract $(\mathsf{sk}_{U_1}, k_1, \mathsf{pk}_{U_1}, j_1)$, with $j_1 \in [\mathfrak{n}]$, such that $\mathsf{sn}_1 = \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_1, 0\|\mathsf{ctxt}\|j_1)$ and $\mathsf{tg}_1 = \mathsf{pk}_{U_1} + a_1 \cdot \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_1, 1\|\mathsf{ctxt}\|j_1)$. Moreover, by parsing $(\mathsf{id}_{U_1}, o_{U_1}) := \mathsf{sk}_{U_1}$, $(\mathsf{id}_{U_1}, o_{U_1})$ is a valid opening to pk_{U_1} with respect to $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$. Let $\mathtt{s}_1 = \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_1, 1\|\mathsf{ctxt}\|j_1)$. By plugging the expression of tg_1 in equation (6.7), we have

$$a_0(\mathsf{pk}' - \mathsf{pk}_{U_1}) = a_1(a_0 \cdot \mathtt{s}_1 - \mathsf{tg}_0 + \mathsf{pk}') \tag{6.8}$$

Let $\mathtt{a} = a_0(\mathsf{pk}' - \mathsf{pk}_{U_1})$ and $\mathtt{b} = (a_0 \cdot \mathtt{s}_1 - \mathsf{tg}_0 + \mathsf{pk}')$. We recall that $\mathtt{a}, \mathtt{b} \in \mathcal{Y}$, where $\mathcal{Y}$ is a finitely generated A-module over a finite commutative ring A. Thus, for E_1 to happen, it suffices to find a_1 such that $a_1 \cdot \mathtt{b} = \mathtt{a}$. Indeed, without loss of generality, since $a_0 = \mathsf{H}(\mathsf{nonce}_0, \pi_{\mathsf{sn}_0})$, the proof π_{sn_0} of well-formedness of sn_0 fixes all the variables used to compute tg_0. In addition, to compute a_1, we need to compute π_{sn_1} first, and this fixes pk_{U_1} and $\mathtt{s}_1$. Therefore, $\mathtt{a}$ and $\mathtt{b}$ are fixed, and since a_1 is the output of a random oracle, finding such an a_1 is equivalent to sampling a_1 uniformly from $\mathcal{S}$, the nonce space, conditioned on $a_1 \cdot \mathtt{b} = \mathtt{a}$.

Note that if $\mathtt{a} = \mathtt{b} = \mathbf{0}$, then this is trivial. However, if $\mathtt{a} = \mathbf{0}$, then $\mathsf{pk}' = \mathsf{pk}_{U_1}$ since $a_0 \in A^\times$. This implies that sk_{U_1} is a valid opening for pk'. Hence, we can use $\mathcal{A}$ to build an (expected) poly-time adversary that either breaks the hiding property of $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$, if sk_{U_1} is the opening used to create pk', or the binding property of $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$, if it is different. This is done by substituting one of the users' keys returned in response to a NewUser query with a challenge from the hiding/binding experiment for $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$, incuring a multiplicative loss of $1/Q_{\mathsf{NewUser}}$ in the reduction's success probability, where Q_{NewUser} is the number of NewUser queries performed by $\mathcal{A}$. Thus, under the hiding and binding property of $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$, $\mathtt{a} \neq \mathbf{0}$ with overwhelming probability.

Conditioned on $\mathtt{a} \neq \mathbf{0}$, it follows that $\mathtt{b} \neq \mathbf{0}$; otherwise there does not exist an $a_1 \in \mathcal{S}$ that satisfies equation (6.8). Let $\tilde{a} \in A$ be a candidate solution, i.e, $\tilde{a} \cdot \mathtt{b} = \mathtt{a}$. Then the set of all solutions to equation (6.8) is $\tilde{a} + \mathrm{Ann}_A(\mathtt{b})$, where $\mathrm{Ann}_A(\mathtt{b}) \neq A$ since $\mathtt{b} \neq \mathbf{0}$. Since $a_1 \in \mathcal{S}$, we have $a_1 \in \mathcal{S} \cap (\tilde{a} + \mathrm{Ann}_A(\mathtt{b}))$. Hence the probability that $\mathcal{A}$ finds a_1 is at most $Q_{\mathsf{H}} \cdot \frac{|\mathcal{S} \cap (\tilde{a} + \mathrm{Ann}_A(\mathtt{b}))|}{|\mathcal{S}|} \leq Q_{\mathsf{H}} \cdot \nu$, since $\mathcal{S}$ is a ν-annihilator-coset evasive set (see Definition 2.1). Now, since pk' is one of the honest-user public keys, and there are at most Q_{NewUser} honest-users, the probability that $\mathcal{A}$ finds a_1 for any of those keys is upper bounded by $Q_{\mathsf{NewUser}} \cdot Q_{\mathsf{H}} \cdot \nu$, by a union bound. Therefore,

$$\Pr[\mathsf{E}_1] \leq Q_{\mathsf{H}}^2 \cdot 2^{-\omega(\log \lambda)} + \kappa + \epsilon_{\mathsf{dist},\mathsf{crs}_{\mathsf{Com}}} + 2Q_{\mathsf{NewUser}} \cdot \epsilon_{\mathsf{Com},\mathsf{Hid},\mathsf{SEP}} + Q_{\mathsf{NewUser}} \cdot Q_{\mathsf{H}} \cdot \nu \tag{6.9}$$

Where κ is an upper bound on the sum of the knowledge errors of $\Pi^{\mathcal{R}_{\mathsf{Com}}}_{\mathsf{NIZK}}$, $\Pi^{\mathcal{R}_{M,\Sigma}}_{\mathsf{NIZK}}$, and $\Pi^{\mathcal{R}_{\mathsf{1Many}}}_{\mathsf{NIZK}}$. In addition, $\epsilon_{\mathsf{dist},\mathsf{crs}_{\mathsf{Com}}}$ is the advantage of a reduction, built from $\mathcal{A}$, in distinguishing crs generated by $\mathsf{Ext}_{\mathsf{Com}}$ from crs generated by Setup, and $\epsilon_{\mathsf{Com},\mathsf{Hid},\mathsf{SEP}}$ is the success probability of an (expected) poly-time reduction, built from $\mathcal{A}$, that tries to break either the binding or the hiding property of $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}$. By combining equations (6.6) and (6.9), and using the assumptions of the theorem, it follows that Construction 6.1 satisfies exculpability. □

Remark 6.3. Note that in equation (6.9), the factor 2 on $\epsilon_{\mathsf{Com},\mathsf{Hid},\mathsf{SEP}}$ can be dropped if $\Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}$ is assumed to be a one-way function for deterministic commitments, and ids are random. In this case, instead of having the reduction guess whether it should break hiding or binding, the reduction simply tries to break one-wayness.

7 Instantiating ARLT in the ROM from Lattices

7.1 Instantiating Π_{SEP}

To instantiate Π_{SEP}, we use the signature with efficient protocols that underlies the anonymous credential system of Bootle, Lyubashevsky, Nguyen, and Sorniotti. [BLNS23a], henceforth referred to as BLNS.

As highlighted in Definition 3.4, a signature with efficient protocols needs a LHC scheme Π_{Com} that admits an efficient NIZK PoK of an opening. In the case of BLNS, the commitment is the Ajtai Commitment [Ajt96, GGH96] instantiated with two vectors $(\mathbf{c}_0, \mathbf{c}_1) \in R_q^{\ell_0} \times R_q^{\ell_1}$, where ℓ_0 represents the randomness length and ℓ_1 represents the message length. The relation $\mathcal{R}_{\mathsf{Com}}$ that needs to be satisfied by an opening is defined in equation (7.1), and BLNS provides an efficient straight-line extractable NIZK PoK $\Pi^{\mathcal{R}_{\mathsf{Com}}}_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ in the random oracle model, based on $\mathsf{LNP}_{\mathsf{str}}$. To simply the exposition in this section, we assume the existence of $\Pi^{\mathcal{R}_{\mathsf{Com}}}_{\mathsf{NIZK}}$ and refer the reader to BLNS [BLNS23b, Section 6] for a full description.

$$\mathcal{R}_{\mathsf{Com}}(\mathbf{c}_0, \mathbf{c}_1) := \left\{ \mathbb{x} := (t, \beta), \mathbb{w} := (\mathbf{m}, \mathbf{r}) : \begin{matrix} (t = \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m}) \\ \wedge (\|[\mathbf{m} \mid \mathbf{r}]\| \leq \beta) \end{matrix} \right\} \tag{7.1}$$

The signature scheme underlying BLNS works as follows. To sign a low-norm message $\mathbf{m} \in R_q^{\ell_1}$, a signer holding an NTRU vector $[h \mid \mathbf{g}^\top \mid 1]$ as public key, where $(h, \mathbf{g}^\top) \in R_q \times R_q^m$, and an NTRU trapdoor $\mathbf{T}_h$ as secret key, proceeds as follows: (1) sample a low-norm high-entropy vector $\mathbf{r} \in R_q^{\ell_0}$ and an integer $\mathtt{x} \leftarrow\!\!\$\ [N]$, where N is a public parameter, and (2) use $\mathbf{T}_h$ to sample a low-norm vector $\mathbf{s}$ such that

$$[h \mid \mathbf{g}^\top \mid 1]\mathbf{s} = f_{\mathsf{bin}}(\mathtt{x}) + \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m} \tag{7.2}$$

where $f_{\mathsf{bin}} : [N] \to R_q$ is defined as $f_{\mathsf{bin}}(\mathtt{x}) := \mathsf{coef}^{-1}(B \cdot \mathsf{BitDecomp}(\mathtt{x} - 1))$ and B is a public parameter chosen as follows: $B \leftarrow\!\!\$\ \mathbb{Z}_q^{d \times \log N}$. In other words, f_{bin} converts an integer into an element of the ring R_q by multiplying the matrix B by the vector that corresponds to $\mathtt{x}$'s binary decomposition, and then viewing the resulting vector as an element of R_q by thinking of each position as a coefficient. The resulting signature is $\sigma := (\mathbf{s}, \mathtt{x}, \mathbf{r})$. The unforgeability of the scheme relies on the hardness of the Interactive $\mathsf{NTRU\text{-}ISIS}_{f_{\mathsf{bin}}}$ assumption (see Definition 2.8). With respect to the syntax highlighted in Definition 3.4, one can see $\mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m}$ as a commitment to $\mathbf{m}$, and the process of sampling $\mathtt{x}$ and $\mathbf{s}$ as signing on this commitment.

The process of obliviously signing a message is similar to the one described above. The main difference is that a committer/receiver holding a message $\mathbf{m}$ computes an Ajtai commitment $t = \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m}$ and a proof π_{Open} attesting that t satisfies $\mathcal{R}_{\mathsf{Com}}$ (see eq. (7.1)). The receiver then sends (t, π_{Open}) to a signer, who samples a low-norm vector $\mathbf{s}$ and $\mathtt{x} \in [N]$ such that equation (7.2) is satisfied with respect to t. In this case, the resulting signature *pre-signature* is $\sigma' := (\mathbf{s}, \mathtt{x})$, and the final one is $\sigma := (\mathbf{s}, \mathtt{x}, \mathbf{r})$. Note this process slightly differs from the syntax highlighted in Definition 3.4, in that the signer does not re-randomized the commitment before signing it. This is because, in the case of BLNS, the act of sampling $\mathtt{x}$ and computing $f_{\mathsf{bin}}(\mathtt{x}) + t$ can be interpreted as a form of re-randomization of t.

Finally, one needs to be able to prove knowledge of a valid message-signature pair, which, in the case of BLNS, boils down to satisfying the relation $\mathcal{R}_{M,\Sigma}$ defined in equation (7.3). BLNS provided a efficient rewind-extractable commit-and-prove NIZK $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$, based on LNP, to handle that relation [BLNS23b, Section 5]. Here, we will simply assume its existence.

$$\mathcal{R}_{M,\Sigma}(\mathbf{c}_0, \mathbf{c}_1, B) := \left\{ \begin{matrix} \mathbb{x} := (h, \mathbf{g}, \beta_0, \beta_1), \\ \mathbb{w} := (\mathbf{m}, \mathbf{s}, \mathtt{x}, \mathbf{r}) \end{matrix} : \begin{matrix} [h \mid \mathbf{g}^\top \mid 1]\mathbf{s} = f_{\mathsf{bin}}(\mathtt{x}) + \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m} \\ \wedge (\|\mathbf{s}\| \le \beta_0) \wedge (\|[\mathbf{m} \mid \mathbf{r}]\| \le \beta_1) \end{matrix} \right\} \tag{7.3}$$

Note that in the context of Construction 6.1, $\mathbf{m}$ in the above equation corresponds to a concatenation of the user's secret key and the PRF key.

We provide a complete description of the BLNS SEP scheme below.

Construction 7.1 (BLNS's SEP [BLNS23a])**.** Let $\mathsf{NTRU} = (\mathsf{TrapGen}, \mathsf{SamplePre})$ be the algorithms defined in Lemma 2.6, and $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}} = (\mathsf{Setup}, \mathsf{Prove}^{\mathsf{H}}, \mathsf{Verify}^{\mathsf{H}})$ be a NIZK with straight-line extraction for relation $\mathcal{R}_{\mathsf{Com}}$ (see eq. (7.1)). For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be a prime, and $n = n(\lambda)$, $d = d(\lambda)$, $m = m(\lambda)$ be lattice parameters. Let $\mathfrak{s} = \mathfrak{s}(\lambda)$ be a Gaussian-width parameter. Let $\psi_0 = \psi_0(\lambda)$, $\psi_1 = \psi_1(\lambda)$, and $\beta = \beta(\lambda)$ be norm bounds, and let $N = 2^{\omega(\log \lambda)}$ be an integer. Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\le p(\lambda)}$ be a random oracle, where p is some polynomial. Let $\ell_0 = \ell_0(\lambda)$ be the randomness length, and let $\ell_1 = \ell_1(\lambda)$ be the message length. $\mathsf{BLNS}_{\mathsf{SEP}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sign}, \mathsf{SignCom}, \mathsf{OblSign}, \mathsf{Verify})$, with message space $\mathcal{M}_{\mathsf{SEP}} = (S_{\psi_1}^{\infty})^{\ell_1}$, works as follows:

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$, proceed as follows:
  1. Sample $B \leftarrow\!\!\$\ \mathbb{Z}_q^{d \times \log N}$.
  2. Sample $(\mathbf{c}_0, \mathbf{c}_1) \leftarrow R_q^{\ell_0} \times R_q^{\ell_1}$.
  3. Compute $\mathsf{crs}_{\mathsf{Com}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Setup}(1^\lambda, (\mathbf{c}_0, \mathbf{c}_1))$.
  4. Return $\mathsf{crs} := (\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1)$.

- $\mathsf{Gen}(\mathsf{crs})$: Given $(\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1) := \mathsf{crs}$, proceed as follows
  1. $(h, \mathbf{T}_h) \leftarrow \mathsf{NTRU.TrapGen}(d, q)$.
  2. Sample $\mathbf{g} \leftarrow\!\$\ R_q^m$.
  3. Return $\mathsf{pk} := (h, \mathbf{g})$ and $\mathsf{sk} := (\mathbf{T}_h, h, \mathbf{g})$.
- $\mathsf{SignCom}(\mathsf{crs}, \mathsf{sk}, t)$: Given $(\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1) := \mathsf{crs}$, $(\mathbf{T}_h, h, \mathbf{g}) := \mathsf{sk}$ and $t \in R_q$, proceed as follows:
  1. Sample $\mathtt{x} \leftarrow\!\$\ [N]$ and compute $u \leftarrow f_{\mathsf{bin}}(\mathtt{x}) + t$.
  2. Compute $\mathbf{s} \leftarrow \mathsf{NTRU.SamplePre}([h \mid \mathbf{g}^\top \mid 1], \mathbf{T}_h, u, \mathfrak{s})$.
  3. Return $\sigma' := (\mathbf{s}, \mathtt{x})$.
- $\mathsf{Sign}(\mathsf{crs}, \mathsf{sk}, \mathbf{m})$: Given $(\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1) := \mathsf{crs}$, $(\mathbf{T}_h, h, \mathbf{g}) := \mathsf{sk}$ and $\mathbf{m} \in \mathcal{M}_{\mathsf{SEP}}$, proceed as follows:
  1. Sample $\mathbf{r} \leftarrow\!\$\ (S_{\psi_0}^{\infty})^{\ell_0}$ and compute $t \leftarrow \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m}$.
  2. Compute $(\mathbf{s}, \mathtt{x}) \leftarrow \mathsf{SignCom}(\mathsf{crs}, \mathsf{sk}, t)$.
  3. Return $\sigma := (\mathbf{s}, \mathtt{x}, \mathbf{r})$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}, \mathbf{m}, \sigma)$: Given $(\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1) := \mathsf{crs}$, $(h, \mathbf{g}) := \mathsf{pk}$, $\mathbf{m} \in \mathcal{M}_{\mathsf{SEP}}$, and $(\mathbf{s}, \mathtt{x}, \mathbf{r}) := \sigma$, proceed as follows:
  1. If $[h \mid \mathbf{g}^\top \mid 1]\mathbf{s} = f_{\mathsf{bin}}(\mathtt{x}) + \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m} \wedge \|\mathbf{s}\| \le \mathfrak{s}\sqrt{(2+m)d} \wedge \|\mathbf{r}\| \le \psi_0\sqrt{\ell_0 d}$, return 1.
  2. Otherwise, return 0.
- $\mathsf{OblSign}(S(\mathsf{crs}, \mathsf{sk}) \rightleftharpoons R(\mathsf{crs}, \mathbf{m}))$
  - $\mathtt{R}(\mathsf{crs}, \mathbf{m})$, the receiver $\mathtt{R}$, on input , proceeds as follows:
    1. Parse $(\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1) := \mathsf{crs}$.
    2. Sample $\mathbf{r} \leftarrow\!\$\ (S_{\psi}^{\infty})^{\ell_r}$.
    3. Compute $t \leftarrow \mathbf{c}_0^\top \mathbf{r} + \mathbf{c}_1^\top \mathbf{m}$.
    4. Compute $\pi_{\mathsf{Open}} \leftarrow \Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{Com}}, \mathbb{x} := (t, \beta), \mathbb{w} := (\mathbf{r}, \mathbf{m}))$.
    5. Send (t, π_{Open}) to $\mathtt{S}$ and keep $o := (\mathbf{r})$.
  - $\mathtt{S}(\mathsf{crs}, \mathsf{sk}, t, \pi_{\mathsf{Open}})$: the signer $\mathtt{S}$, on input , proceeds as follows:
    1. Parse $(\mathsf{crs}_{\mathsf{Com}}, B, \mathbf{c}_0, \mathbf{c}_1) := \mathsf{crs}$.
    2. If $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{Com}}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}_{\mathsf{Com}}, \mathbb{x} := (t, \beta), \pi_{\mathsf{Open}}) = 0$: return $\perp$.
    3. Return $\mathsf{SignCom}(\mathsf{crs}, \mathsf{sk}, t)$.

Theorem 7.1 ([BLNS23b], adapted)**.** *For any $\lambda \in \mathbb{N}$, let $q = q(\lambda)$ be a prime, and $n = n(\lambda)$, $d = d(\lambda)$, $m = m(\lambda)$ be lattice parameters. let $\varepsilon = \mathsf{negl}(\lambda)$, let $\mathfrak{s} \ge 1.17\sqrt{q} \cdot \eta_\varepsilon(R^2)$. Under the hardness of the $\mathsf{Int\text{-}NTRU\text{-}ISIS}_f$ assumption with parameters $(f_{\mathsf{bin}}, \mathfrak{s}, \mathfrak{s}\sqrt{(m+2)d}, \sqrt{(\psi_0^2\ell_0 + \psi_1^2\ell_1)d}, \ell_0, \ell_1)$ (see Definition 2.8), $\mathsf{BLNS}_{\mathsf{SEP}}$ is correct except with negligible probability and is* EUF-CMA.

Remark 7.1 (Message and Randomness Spaces of BLNS). Note that the message space $\mathcal{M}_{\mathsf{SEP}}$ of $\mathsf{BLNS}_{\mathsf{SEP}}$ need not be exactly $(S_{\psi_1}^{\infty})^{\ell_1}$. It suffices to take $\mathcal{M}_{\mathsf{SEP}}$ to be any efficiently testable subset of $R_q^{\ell_1}$ with bounded ℓ_2-norm. Similarly, the commitment randomness need not be sampled from $(S_{\psi_{\ell_0}}^{\infty})^{\ell_0}$: it suffices to use any efficiently samplable distribution χ over R for which $\mathsf{M\text{-}LWE}_{\ell_0-1,\ell_0,q,\chi}$ holds.

7.2 Instantiating Π_{PRF}

We instantiate our PRF using the M-LWR-based weak key-homomorphic PRF which was highlighted by Boneh et al. [BLMR13] and used as the basis for the verifiable random function of Esgin et al. [ESLR23]. In brief, this PRF can be seen as a lattice-based counter-part of the weak Decisional Diffie-Hellman-based PRF of Naor, Pinkas, Reingold [NPR99]. Given a PRF key $\mathbf{k}$, the PRF evaluation on an input x returns $\lfloor \mathsf{H}(x) \cdot \mathbf{k} \rfloor_q$, where H is random oracle that maps inputs to matrices and q is a public parameter.

Construction 7.2 (PRF)**.** For any $\lambda \in \mathbb{N}$, let $q_0 = q_0(\lambda)$, $q = q(\lambda)$ be a prime, and $\hat{q} = \hat{q}(\lambda)$ be an integer such that $\hat{q} = q_0 \cdot q$. Let $n = n(\lambda)$, $m = m(\lambda)$, and $d = d(\lambda)$ be lattice parameters. Let $H : \{0,1\}^* \to R_{\hat{q}}^{n \times m}$ be a random oracle. $\Pi_{\mathsf{PRF}} = (\mathsf{Gen}, \mathsf{Eval})$, with key space $\mathcal{K} := \chi^m$, where χ is a distribution over R, is defined as follows:

- $\mathsf{Gen}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$, sample $\mathbf{k} \leftarrow \mathcal{K}$.
- $\mathsf{Eval}^{\mathsf{H}}(\mathbf{k}, x)$: Given a key $\mathbf{k}$ and an input $x \in \{0,1\}^*$, return $\lfloor \mathsf{H}(x) \cdot \mathbf{k} \rfloor_q$.

Theorem 7.2 ([ESLR23], adapted)**.** *For any $\lambda \in \mathbb{N}$, let $\mathfrak{t} = \mathfrak{t}(\lambda)$ be an upper-bound on the number of Π_{PRF}'s evaluations. If $\mathsf{M\text{-}LWR}_{m,\mathfrak{t}n,q,\hat{q},\chi}$ holds, then Construction 7.2 is a weak PRF (see Definition 3.6).*

Theorem 7.2 directly follows from the fact that each Π_{PRF} evaluation can be viewed as releasing n M-LWR samples across $\mathfrak{t}$ queries.

Theorem 7.3 (Key-Homomorphism)**.** *For any $\lambda \in \mathbb{N}$, let $\mathfrak{t} = \mathfrak{t}(\lambda)$ be an upper-bound on the number of Π_{PRF}'s evaluations. Then Construction 7.2 is 1-almost-key-homomorphic (see Definition 3.7) with respect to the ℓ_∞-norm under the hardness of $\mathsf{M\text{-}LWR}_{m,\mathfrak{t}n,q,\hat{q},\chi}$ and $\mathsf{M\text{-}LWR}_{m,\mathfrak{t}n,q,\hat{q},\chi*\chi}$, where $\chi * \chi$ denotes the convolution of χ with itself.*

Proof. Given $\lambda \in \mathbb{N}$, let $\hat{q} = \hat{q}(\lambda)$ and $q = q(\lambda)$ be such that $q \mid \hat{q}$, and let $q_0 = \hat{q}/q$. The argument follows from the fact that for any $x, y \in \mathbb{Z}_{\hat{q}}$, there exists $e \in \{0,1\}$ such that $\lfloor x + y \rfloor_q = \lfloor x \rfloor_q + \lfloor y \rfloor_q + e$. Indeed, by writing $x = \tilde{x} q_0 + r_1$ and $y = \tilde{y} q_0 + r_2$, where $0 \le r_1, r_2 < q_0$, we have $\lfloor x + y \rfloor_q = \lfloor (q/\hat{q})(x + y) \rfloor = \tilde{x} + \tilde{y} + \lfloor (r_1 + r_2)/q_0 \rfloor$. Letting $e = \lfloor (r_1 + r_2)/q_0 \rfloor$, it follows that $e \in \{0,1\}$ since $0 \le (r_1 + r_2)/q_0 < 2$. Moreover, $\lfloor x \rfloor_q = \tilde{x}$ and $\lfloor y \rfloor_q = \tilde{y}$.
Now, given an input $x \in \{0,1\}^*$, and keys $\mathbf{k}_1, \mathbf{k}_2 \leftarrow \chi^m$, the above claim implies that $\lfloor \mathsf{H}(x) \cdot (\mathbf{k}_1 + \mathbf{k}_2) \rfloor_q = \lfloor \mathsf{H}(x) \cdot \mathbf{k}_1 \rfloor + \lfloor \mathsf{H}(x) \cdot \mathbf{k}_2 \rfloor + \mathbf{e}$, where $\mathbf{e} \in \mathcal{B}_1^n$. Moreover, $\lfloor \mathsf{H}(x) \cdot (\mathbf{k}_1 + \mathbf{k}_2) \rfloor_q$ is pseudo-random by the hardness of $\mathsf{M\text{-}LWR}_{m,\mathfrak{t}n,q,\hat{q},\chi*\chi}$. □

The above theorem can be generalized to the case of a convolution of χ with itself α times by induction. However, for our setting, $\alpha = 2$ suffices.

By Corollary 3.0.1, Construction 7.2 is weak collision resistant with respect to the set $\mathcal{S}_\delta = \mathcal{B}_1^n$ if we restrict our input domain to $\{0,1\}^{c \log \lambda}$, for some $c > 0$, since $|R_q^n|/|\mathcal{B}_1^n| = (q/2)^{nd} \in 2^{\omega(\log \lambda)}$.

7.3 Instantiating $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$

In this section, we describe how to instantiate the proof system $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ such that it is compatible with our ARLT construction (see Construction 6.1) where the underlying SEP is instantiated using $\mathsf{BLNS}_{\mathsf{SEP}}$, and the underlying Π_{PRF} is instantiated using Construction 7.2.

Users's Key Generation. We first describe a concrete key-generation procedure for users, since this will be needed during the execution of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$. Although in Construction 6.1, we considered a setting where users commit to their identities, here, to simplify the exposition, we assume that users' identities are sampled uniformly at random from $\mathcal{B}_{\mathfrak{b}-1}^{\ell_s}$, where $\mathfrak{b} \geq 2$, $\mathcal{B}_{\mathfrak{b}-1}$ is the set of polynomial in R_q with coefficients in $\{0, \ldots, \mathfrak{b}-1\}$, and ℓ_s is the length of a user identity. Specifically, the Setup of Construction 6.1 outputs a matrix $\mathbf{A}_s = [\mathbf{I}_n \mid \bar{\mathbf{A}}_s]$, where $\bar{\mathbf{A}}_s \leftarrow\!\$\ R_q^{n\times(\ell_s-n)}$, in addition to other common reference strings. Given $\mathbf{A}_s$, a user U samples $\mathsf{sk}_U \leftarrow \mathcal{B}_{\mathfrak{b}-1}^{\ell_s}$ and outputs $\mathsf{pk}_U \leftarrow \mathbf{A}_s \cdot \mathsf{sk}_U$ as its public key. Clearly, pk_U hides sk_U under the hardness of the $\mathsf{M\text{-}LWE}_{\ell_s-n,n,q,\mathcal{B}_{\mathfrak{b}-1}}$ assumption. In addition, in the oblivious signing algorithm of $\mathsf{BLNS}_{\mathsf{SEP}}$ (see Construction 7.1) the relation $\mathcal{R}_{\mathsf{Com}}$ can easily be upgraded to show that sk_U is an "opening" to pk_U and a sub-opening to a commitment $t := \mathbf{c}_0\mathbf{r} + \mathbf{c}_1 \cdot [\mathsf{sk}_U^\top \mid \hat{\mathbf{k}}^\top]^\top$, where $\hat{\mathbf{k}}$ is the user's contribution for the PRF key.

Now, we are ready to describe our instantiation of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$. First, suppose that the rate limit $\mathfrak{n}$ satisfies $\mathfrak{n} < 2^{\tau d}$, where $\tau > 0$ is any constant such that τd is an integer. Then, for any $j \in [\mathfrak{n}]$, $\mathsf{BitDecomp}(j)$ can be represented by an element $\mathbf{j} \in \mathcal{B}_1^{\ell_j}$, where $\ell_j = 1$ if $0 < \tau \leq 1$, and $\ell_j = \tau$ if $\tau > 1$. This is done by interpreting the bits of $\mathsf{BitDecomp}(j)$ as coefficients of polynomials in R_q.

Now, consider a user U holding a token dispenser $\mathcal{T} := (\mathsf{sk}_U, \mathbf{k}, \sigma_{\mathsf{SEP}}, \mathsf{Map})$, where $\sigma_{\mathsf{SEP}} := (\mathbf{s}, \mathtt{x}, \mathbf{r})$ is a $\mathsf{BLNS}_{\mathsf{SEP}}$ signature on $(\mathsf{sk}_U, \mathbf{k})$. Suppose that U wants to redeem a token on a context $\mathsf{ctxt} \in \{0,1\}^{c\log\lambda}$ and a nonce $\mathsf{nonce} \in \{0,1\}^{\leq p(\lambda)}$, where $c > 0$ and p is some polynomial. Let q and $\hat{q}$ be the modulus used by Π_{PRF} and let $q_0 = \hat{q}/q$. Following construction 6.1, U proceeds as follows:

- Commit to $j \in [\mathfrak{n}]$ by computing $t_j = \mathbf{c}_0^\top \mathbf{r}_j + \mathbf{c}_1^\top \mathbf{j}$, where $\mathbf{r}_j$ is sampled from the same distribution used by $\mathsf{BLNS}_{\mathsf{SEP}}$'s commitment, and $\mathbf{j}$ is padded with zeros if needed to match the length of $\mathbf{c}_1$.
- Compute $\mathsf{sn} \leftarrow \lfloor \mathsf{H}(0\|\mathsf{ctxt}\|j) \cdot \mathbf{k} \rceil_q$
- Compute $\mathsf{tg} \leftarrow \mathsf{pk}_U + a\lfloor \mathsf{H}(1\|\mathsf{ctxt}\|j) \cdot \mathbf{k} \rceil_q$, where a is derived by hashing the proof of well-formedness of sn, concatenated with nonce. For the time being, we assume that a is given and return to this point later.

Let $\mathbf{X}_j = \mathsf{H}(0\|\mathsf{ctxt}\|j)$ and $\mathbf{X}'_j = \mathsf{H}(1\|\mathsf{ctxt}\|j)$. Proving the well-formedness of sn and tg is equivalent to proving, over $R_{\hat{q}}$, that

$$\begin{aligned} q_0 \cdot \mathsf{sn} &= \mathbf{X}_j \cdot \mathbf{k} - \mathbf{e}_{\mathsf{sn}} \\ q_0 \cdot \mathsf{tg} &= q_0(\mathsf{pk}_U + a \cdot \mathbf{X}'_j \cdot \mathbf{k}) - a \cdot \mathbf{e}_{\mathsf{tg}} \end{aligned} \tag{7.4}$$

where $\mathbf{e}_{\mathsf{sn}}, \mathbf{e}_{\mathsf{tg}} \in \mathcal{B}_{q_0-1}^n$ by Lemma 2.1. For this reason our proofs are over $R_{\hat{q}}$ instead of R_q. Moreover, to ensure that no wraparound occurs in the computation of sn and tg, we need to prove that $\mathbf{e}_{\mathsf{sn}}$ and $\mathbf{e}_{\mathsf{tg}}$ indeed lie in $\mathcal{B}_{q_0-1}^n$, which is equivalent to proving *exactly* that $\|\mathbf{e}_{\mathsf{sn}}\|_\infty, \|\mathbf{e}_{\mathsf{tg}}\|_\infty \leq q_0 - 1$. Since our goal is to remain compatible with the instantiation of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ used by $\mathsf{BLNS}_{\mathsf{SEP}}$, which relies on LNP, and since LNP only supports approximate ℓ_∞-norm, we change the representation of $\mathbf{e}_{\mathsf{sn}}$ and $\mathbf{e}_{\mathsf{tg}}$. Specifically, let $\tilde{\mathbf{G}}_\gamma = \mathbf{I}_n \otimes [0, 1, \ldots, 2^{\gamma-2}, \omega]$, where $\omega = q_0 - 2^{\gamma-1}$ and $q_0 \leq 2^\gamma$, and let $\tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}} \in \mathcal{B}_1^{n\lceil\log q_0\rceil}$ such that $\mathbf{e}_{\mathsf{sn}} = \tilde{\mathbf{G}}_\gamma \cdot \tilde{\mathbf{e}}_{\mathsf{sn}}$ and $\mathbf{e}_{\mathsf{tg}} = \tilde{\mathbf{G}}_\gamma \cdot \tilde{\mathbf{e}}_{\mathsf{tg}}$. Using $\tilde{\mathbf{G}}_\gamma$ instead of the traditional base-two gadget matrix implies that some binary decompositions are not unique. However, as Esgin et al. noted [ESLR23], this is not an issue, and this representation still ensures that $\|\mathbf{e}\|_\infty \leq q_0 - 1$. Thus, equation (7.4) is equivalent to

$$\begin{aligned} q_0 \cdot \mathsf{sn} &= \mathbf{X}_j \cdot \mathbf{k} - \tilde{\mathbf{G}}_\gamma \cdot \tilde{\mathbf{e}}_{\mathsf{sn}} \\ q_0 \cdot \mathsf{tg} &= q_0(\mathsf{pk}_U + a \cdot \mathbf{X}'_j \cdot \mathbf{k}) - a \cdot \tilde{\mathbf{G}}_\gamma \cdot \tilde{\mathbf{e}}_{\mathsf{tg}} \end{aligned} \tag{7.5}$$

We can now instantiate $\mathcal{R}_{\mathsf{1Many}} = \{\mathcal{R}_{\mathsf{sn}}, R_{\mathsf{tg}}\}$ as defined in equation 3.4. To simplify the exposition, in what follows we omit the component of the commitment scheme underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ and assume that the

witness is indeed committed:

$$\mathcal{R}_{\mathsf{sn}} := \left\{ \begin{array}{l} \mathbb{x} := \left(t_j, \mathsf{sn}, \{\mathbf{X}_i\}_{i\in[\mathfrak{n}]}\right), \mathbb{w} := ((\mathsf{sk}_U, \mathbf{k}, \tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}}), \mathbf{j}, \mathbf{r}_j) : \\ j \in [\mathfrak{n}] \wedge (t_j - \mathbf{c}_0^\top \mathbf{j} = \mathbf{c}_1^\top \mathbf{r}_j) \wedge \tilde{\mathbf{e}}_{\mathsf{sn}} \in \mathcal{B}_1^{n\lceil \log q_0 \rceil} \\ \wedge \left(q_0 \cdot \mathsf{sn} = [\mathbf{X}_j \mid -\tilde{\mathbf{G}}_\gamma] \cdot [\mathbf{k}^\top \mid \tilde{\mathbf{e}}_{\mathsf{sn}}^\top]^\top\right) \end{array} \right\}$$

$$\mathcal{R}_{\mathsf{tg}} := \left\{ \begin{array}{l} \mathbb{x} := \left(t_j, a, \mathsf{tg}, \{\mathbf{X}'_j\}_{i\in[\mathfrak{n}]}\right), \mathbb{w} := ((\mathsf{sk}_U, \mathbf{k}, \tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}}), \mathbf{j}, \mathbf{r}_j) : \\ j \in [\mathfrak{n}] \wedge (t_j - \mathbf{c}_0^\top \mathbf{j} = \mathbf{c}_1^\top \mathbf{r}_j) \wedge \tilde{\mathbf{e}}_{\mathsf{tg}} \in \mathcal{B}_1^{n\lceil \log q_0 \rceil} \\ \wedge \left(q_0 \cdot \mathsf{tg} = [q_0\mathbf{A}_s \mid aq_0 \cdot \mathbf{X}'_j \mid -a \cdot \tilde{\mathbf{G}}_\gamma] \cdot [\mathsf{sk}_U^\top \mid \mathbf{k}^\top \mid \tilde{\mathbf{e}}_{\mathsf{tg}}^\top]^\top\right) \end{array} \right\}$$

Where $\mathbf{X}_i = \mathsf{H}(0\|\mathsf{ctxt}\|i)$ and $\mathbf{X}'_i = \mathsf{H}(1\|\mathsf{ctxt}\|i)$.

Proving that $\tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}} \in \mathcal{B}_1^{n\lceil \log q_0 \rceil}$ can be handled by $\mathsf{BLNS}_{\mathsf{SEP}}$'s instantiation of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, which relies on LNP. That is, the relation $\mathcal{R}_{M,\Sigma}$ defined in equation (7.3) can be upgraded to prove knowledge of vectors $\tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}}$ over $R_{\hat{q}}$ with binary coefficients, while ensuring that these vectors are included in the commitment.

Finally, observe that the dependence on the hidden index j can be handled by rewritting each relation as a disjunction over all $i \in [\mathfrak{n}]$. For each possible index i, the corresponding branch consists of a matrix-vector relation of the following form:

$$\mathbb{x}_{i,\mathsf{sn}} := \left(\begin{bmatrix} \mathbf{c}_1^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & q_0\mathbf{X}_i & -\tilde{\mathbf{G}}_\gamma \end{bmatrix}, \begin{pmatrix} t_j - \mathbf{c}_0^\top \mathbf{i} \\ q_0 \mathsf{sn} \end{pmatrix}\right) \quad \mathbb{x}_{i,\mathsf{tg}} := \left(\begin{bmatrix} \mathbf{c}_1^\top & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & q_0\mathbf{A}_s & aq_0\mathbf{X}'_i & -a\tilde{\mathbf{G}}_\gamma \end{bmatrix}, \begin{pmatrix} t_j - \mathbf{c}_0^\top \mathbf{i} \\ q_0 \mathsf{tg} \end{pmatrix}\right)$$

Therefore $\mathcal{R}_{\mathsf{sn}}$ and $\mathcal{R}_{\mathsf{tg}}$ can be proven by using a disjunctive extension of Lyubashevsky's Fiat-Shamir with abort Σ-protocol [Lyu12] and the resulting proofs can be compressed using our PBVC (see Section 5) so that the final proof size is logarithmic in $\mathfrak{n}$.

Candidate ν-annihilator-coset evasive. To ensure that our instantiation of Construction 6.1 satisfies the exculpability property (see Definition 4.1), we need the nonce space $\mathfrak{N}$ to satisfy ν-annihilator-coset evasiveness (see Definition 2.1), where $\nu = \mathsf{negl}(\lambda)$. In addition, $\mathfrak{N}$ must be a superpolynomial-size subset of the unit group of our ring, which in this case is R_q. Recall that this is not over $R_{\hat{q}}$, since output space of Π_{PRF} is defined over R_q. We also require that, for any distinct $a_0, a_1 \in \mathfrak{N}$, $a_0 - a_1$ is invertible. In equation (7.4), we assumed that $a = \mathsf{H}(\mathsf{nonce}, \pi_{\mathsf{sn}})$ was given. We now show how to instantiate it for the case $q \equiv 5 \bmod 8$. For such moduli, Lyubashevsky and Seiler [LS18, Corollary 1.2] showed that $R_q \cong \mathbb{F}_{q^{d/2}} \times \mathbb{F}_{q^{d/2}}$, where the two copies of $\mathbb{F}_{q^{d/2}}$ are isomorphic but not identical. Let ϕ be a fixed isomorphism between these two copies. We set $\mathfrak{N} = \{(\alpha, \phi(\alpha)) : \alpha \in \mathbb{F}_{q^{d/2}}^\times\}$. Then $\mathfrak{N} \subseteq R_q^\times$, $|\mathfrak{N}| = q^{d/2} - 1$, and $a_0 - a_1 \in R_q^\times$ for all distinct $a_0, a_1 \in \mathfrak{N}$. It remains to show annihilator-coset evasiveness. Let $b \in R_q \setminus \{0\}$ and $\mathbf{m} \in R_q^n \setminus \{\mathbf{0}^n\}$ such that $b + \mathrm{Ann}_{R_q}(\mathbf{m})$ is a nontrivial coset. Let $\mathbf{t} \in R_q^n \setminus \{\mathbf{0}^n\}$ such that $b \cdot \mathbf{m} = \mathbf{t}$. Then there exists $i \in [n]$ such that, $\bar{b} \cdot \bar{m}_i = \bar{t}_i \neq 0$ over $\mathbb{F}_{q^{d/2}}$, where $\bar{b}, \bar{m}_i, \bar{t}_i$ are the projections of b, m_i, t_i over $\mathbb{F}_{q^{d/2}}$, respectively.

$$\Pr_{a \leftarrow\$ \mathfrak{N}}[a \in (b + \mathrm{Ann}_{R_q}(\mathbf{m}))] = \Pr_{a \leftarrow\$ \mathfrak{N}}[a \cdot \mathbf{m} = \mathbf{t}] \leq \Pr_{a \leftarrow\$ \mathfrak{N}}[\bar{a} = \bar{m}_i^{-1}\bar{t}_i \in \mathbb{F}_{q^{d/2}}^\times] \leq \frac{1}{q^{d/2} - 1}$$

Therefore, $\mathfrak{N}$ is a $1/(q^{d/2} - 1)$-annihilator-coset evasive and satisfies the required algebraic properties.

In practice, given a fourth-root of unity $\zeta_4 \in \mathbb{Z}_q^\times$, we can hash into $\mathfrak{N}$ by sampling a nonzero polynomial $p \in \mathbb{Z}_q[X]/(X^{d/2} - \zeta_4)$ and returning the inverse CRT of $(p(X), p(X^3))$ since here, ϕ is given by the map that sends X to X^3.

7.4 Optimizing $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$

The proof for the relation $\mathcal{R}_{\mathsf{1Many}}$ requires first executing the proof of well-formedness of sn, captured by $\mathcal{R}_{\mathsf{sn}}$, before executing the proof of well-formedness of tg, captured by $\mathcal{R}_{\mathsf{tg}}$, to guarantee that Construction 6.1 satisfies exculpability. However, $\mathcal{R}_{\mathsf{sn}}$ and $\mathcal{R}_{\mathsf{tg}}$ share common prefixes, such as proving commitment to the same index $j \in [\mathfrak{n}]$. Moreover, when these relations are merged with the commitment underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, one must prove that $\mathcal{R}_{\mathsf{sn}}$ and $\mathcal{R}_{\mathsf{tg}}$ use the same committed witness. Overall, this results in proof sizes that are roughly twice as large as they would be if $\mathcal{R}_{\mathsf{sn}}$ and $\mathcal{R}_{\mathsf{tg}}$ could be proven in tandem.

Here, we show that in the lattice-based setting, this overhead can be reduced by modifying how tg is computed during the execution of Show, and how identities are recovered by the identification protocol. Specifically, let $\mathbf{G}_{\mathfrak{b}}$ be the base-$\mathfrak{b}$ gadget matrix, for $\mathfrak{b} \geq 2$. Suppose users sample their keys uniformly at random from $\mathcal{B}_{\mathfrak{b}-1}^{n\lceil \log_{\mathfrak{b}} q \rceil}$. Then Construction 6.1 can be adapted as follows:

- During the execution of Show:
  - Replace line 7 with $a := \mathsf{nonce}$. ▶ *Here, a no longer needs to be the output of a hash function. Instead,* $\mathsf{nonce} \in \mathfrak{N}$*, where $\mathfrak{N}$ is the nonce space described above.*
  - Replace line 8 with $\mathsf{tg} \leftarrow \mathbf{G}_{\mathfrak{b}} \cdot \mathsf{sk}_U + a \cdot \left\lfloor \mathbf{X}'_j \cdot \mathbf{k} \right\rfloor_q$. ▶ *That is, instead of using $\mathsf{pk}_U = \mathbf{A}_s \cdot \mathsf{sk}_U$, we use $\mathbf{G}_{\mathfrak{b}} \cdot \mathsf{sk}_U$.*
- During the execution of $\mathsf{Identify}$:
  - Compute $\tilde{\mathsf{sk}}_U \leftarrow \mathsf{tg}_0 - \frac{a_0}{a_0 - a_1} \cdot (\mathsf{tg}_0 - \mathsf{tg}_1)$.
  - Return $\mathbf{A}_s \cdot \mathbf{G}_{\mathfrak{b}}^{-1}(\tilde{\mathsf{sk}}_U)$ as the recovered identity.

These changes do not affect anonymity, soundness, or linkability. The proofs of these properties do not rely on the nonces being random, except through the facts that the difference of two distinct nonces should be invertible and that distinct showings should carry distinct nonces, and both properties continue to hold in this modified construction. The exculpability proof, however, must be adapted. Previously, in the proof of Theorem 6.4, we relied on Construction 6.1 to satisfy anonymity and on the nonces to be randomly sampled from a ν-annihilator-coset evasive set. With the above modification, it suffices instead to assume anonymity and the hardness of $\mathsf{M\text{-}SIS}_{n,m,q,\sqrt{(\mathfrak{b}-1)^2 n\lceil \log_{\mathfrak{b}} q \rceil d+1}}$ to be hard, where $m = n\lceil \log_{\mathfrak{b}} q \rceil + Q_{\mathsf{NewUser}}$, and Q_{NewUser} is an upper bound on the number of users that can join.

Indeed, during the exculpability game, suppose an adversary causes $\mathsf{Identify}$ to output the public key of an honest user. By replacing that user's public key with a uniformly random vector $\mathbf{t} \in R_q^n$, which is valid by the anonymity property, it follows that $\mathbf{G}^{-1}(\tilde{\mathsf{sk}}_U)$, where $\tilde{\mathsf{sk}}_U$ is the value recovered by $\mathsf{Identify}$ from the adversary's input, is a short pre-image of $\mathbf{t}$ with respect to the matrix $\mathbf{A}_s$. This gives a solution to the corresponding $\mathsf{M\text{-}SIS}$ instance.

With these changes, and still omitting the component corresponding to the commitment scheme underlying $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, we can rewrite $\mathcal{R}_{\mathsf{1Many}}$ as follows:

$$\mathcal{R}_{\mathsf{1Many}} := \left\{ \begin{array}{l} \mathbb{x} := \left(\mathsf{sn}, \mathsf{tg}, a, \{\mathbf{X}_i, \mathbf{X}'_i\}_{i \in [\mathfrak{n}]}\right), \mathbb{w} := (\mathsf{sk}_U, \mathbf{k}, \tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}}, j) : \\ j \in [\mathfrak{n}] \wedge \tilde{\mathbf{e}}_{\mathsf{sn}} \in \mathcal{B}_1^{n\lceil \log q_0 \rceil} \wedge \tilde{\mathbf{e}}_{\mathsf{tg}} \in \mathcal{B}_1^{n\lceil \log q_0 \rceil} \\ \wedge \left(q_0 \cdot \mathsf{sn} = [\mathbf{X}_j \mid -\tilde{\mathbf{G}}_\gamma] \cdot [\mathbf{k}^\top \mid \tilde{\mathbf{e}}_{\mathsf{sn}}^\top]^\top\right) \\ \wedge \left(q_0 \cdot \mathsf{tg} = [q_0 \mathbf{G}_{\mathfrak{b}} \mid a q_0 \cdot \mathbf{X}'_j \mid -a \cdot \tilde{\mathbf{G}}_\gamma] \cdot [\mathsf{sk}_U^\top \mid \mathbf{k}^\top \mid \tilde{\mathbf{e}}_{\mathsf{tg}}^\top]^\top\right) \end{array} \right\} \tag{7.6}$$

With this formulation, we no longer need to commit separately to j, since the well-formedness of sn and tg is proven in tandem. As before, the constraints $\tilde{\mathbf{e}}_{\mathsf{sn}}, \tilde{\mathbf{e}}_{\mathsf{tg}} \in \mathcal{B}_1^{n\lceil \log q_0 \rceil}$ can be folded into $\mathcal{R}_{M,\Sigma}$, defined in equation (7.3), assuming that $\mathcal{R}_{M,\Sigma}$ also commits these vectors. The dependence on the hidden index j can then be handled, as in the previous section, using a disjunctive proof à la Lyubashevsky [Lyu12], compressed with our PBVC (see Section 5).

Reintroducing the commitment component underlying the $\mathsf{BLNS}_{\mathsf{SEP}}$ instantiation of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ amounts to adding one linear relationm, independent of the index j, to the definition of $\mathcal{R}_{\mathsf{1Many}}$ in equation (7.6). This is because the $\mathsf{BLNS}_{\mathsf{SEP}}$ instantiation relies on LNP, which uses Ajtai's commitment [Ajt96] to commit to low-norm vectors.

Remark 7.2. Finally, we note that the above construction can be further optimized by observing that the first and last two messages of LNP corresponds to a Σ-protocol with respect to LNP commitment scheme. Therefore, instead of instantiation $\mathcal{R}_{\mathsf{1Many}}$ separately, we can instantiate it on top of $\mathsf{LNP}/\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$. However, we use the instantion described above instead of this one to simplify the exposition.

7.5 Concrete Parameter Instantiation

In this section, we propose concrete parameters for our optimized lattice-based instantiation of Construction 6.1. The parameters are chosen to target 128-bit security against classical attacks. To evaluate the hardness of the M-LWE/M-LWR assumption, we use the lattice estimator of Albrecht, Player and Scott [APS15]. To estimate the hardness of the M-SIS assumption, we follow the methodology of Gama and Nguyen [GN08]. In both cases, we target a root Hermite factor $\delta < 1.0045$, corresponding to at least 128 bits of security.

First, we set the rate limit per context to $\mathfrak{n} = 256$ and the total limit of token per users to $\mathfrak{t} = 2^{70}$. We set the limit on the number of issue query per users to be $Q_{\mathsf{Iss}} = 2^{32}$. We set out context space to $\mathfrak{C} = \{0,1\}^{32}$.

PRF parameters. To instantiate our weak PRF, we set the output space of our hash function to be $R_{\hat{q}}^{1\times 14}$, where $R_{\hat{q}} = \mathbb{Z}_{\hat{q}}[X]/(X^{128}+1)$ and $\hat{q} = q_0 \cdot q$, with q_0 and q both prime and congruent to $5 \bmod 8$. Specifically, we set $q_0 = 61 \approx 2^6$, $q = 1818663989 \approx 2^{31}$. For the keys, we set the distribution to be $\chi = D_{R,\mathfrak{s}}^{14}$, where $\mathfrak{s} \approx 10$. By the results of Micciancio and Peikert [MP13, Theorem 3.3], this ensures that $\chi * \chi = D_{R,\mathfrak{s}\sqrt{2}}^{14}$. In addition, from equation 7.4, the PRF evaluation produces error polynomials in $R_{\hat{q}}$ whose coefficients lie in $\{0,\dots,q_0-1\}$. Pluggin these parameters in the lattice estimator yields a root Hermite factor of 1.0030, which correspond to approximately 168 bits of security and is enough to account for the loss incurred by Lemma 3.1.

Users' keys parameters. Users' public keys are polynomials over the ring $\mathbb{Z}_q[X]/(X^{128}+1)$, where q is the same prime used for PRFs. To compute tgs as described in Section 7.4, we use the base-18 gadget vector $\mathbf{g}_{18} = [1, 18, \dots, 18^{\lceil \log_{18} q \rceil}]$. Users' secret keys are sampled uniformly at random from $\mathcal{B}_{17}^8$ over $\mathbb{Z}_q[X]/(X^{128}+1)$. With these parameters, we obtain a root Hermite factor of 1.0036, which translates to approximately 132 bits of security againts the M-SIS problem described in Section 7.4.

$\mathsf{BLNS}_{\mathsf{SEP}}$ parameters. We follow the optimistic instantiation approach of Bootle, Lyubashevsky, Nguyen, and Sorniotti [BLNS23b], which heuristically assumes that $\mathsf{Int\text{-}NTRU\text{-}ISIS}_f$ for f_{bin} is as hard as M-SIS with respect to the matrix $[h \mid 1 \mid \mathbf{c}_0 \mid \mathbf{c}_1]$. That is, in Construction 7.1, the vector $\mathbf{g}$ is set to $\mathbf{0}$. Since, in our setting, a forger against $\mathsf{BLNS}_{\mathsf{SEP}}$ is only bounded in expected polynomial time (see the proof of Theorem 6.2), we assume the hardness of M-SIS against expected polynomial time adversaries. We use a prime $q_{\mathsf{SEP}} = 27581 \approx 2^{15}$ that is congruent to $5 \bmod 8$, the dimension of the ring is $d_{\mathsf{SEP}} = 1024$, the domain of f_{bin} is $N = 2^{256}$. For NTRU sampling, we use $\varepsilon_{\mathsf{SEP}} = 2^{-256}/(4d_{\mathsf{SEP}})$, which results in

a standard deviation $\mathfrak{s}_{\mathsf{SEP}} = 1.17\sqrt{q_{\mathsf{SEP}}} \cdot \tilde{\eta}_{\varepsilon_{\mathsf{SEP}}}(\mathbb{Z}) = 597 \approx 2^{10}$. For the commitment, we take $\mathbf{c}_0$ of length 2 and $\mathbf{c}_1$ of length 3. The commitment randomness is sampled from $\mathcal{B}_1^2$ over $\mathbb{Z}_{q_{\mathsf{SEP}}}[X]/(X^{d_{\mathsf{SEP}}}+1)$, which gives at least 194 bits of security for the M-LWE assumption. Using the automorphism between $\mathbb{Z}[X]/(X^{dk}+1)$ and $\mathbb{Z}[X]/(X^k+1)$ described by Lyubashevsky et al. [LNPS21, Section 2.8], the vector $[\mathbf{c}_0, \mathbf{c}_1]$ can be used to commit to vectors of length 24 over $\mathbb{Z}[X]/(X^{128}+1)$. This is sufficient to commit to both the users secret keys and PRF keys. Finally, given our PRF parameters, user-key parameters, and NTRU sampling parameters above, we obtain an M-SIS bound $B \approx 27037$ against the instance $[h \mid 1 \mid \mathbf{c}_0 \mid \mathbf{c}_1]$, which results in a root Hermite factor of 1.0024.

$\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ parameters. As previously mentioned, the $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$ of BLNS relies on LNP. In addition to proving the relation induced by $\mathsf{BLNS}_{\mathsf{SEP}}$ (see equation (7.3)), we augment $\mathcal{R}_{M,\Sigma}$ to prove that the binary decompositions of the PRF error vectors are well formed. Thus, during the execution of LNP, we need to commit to the user's secret key, the PRF key, the $\mathsf{BLNS}_{\mathsf{SEP}}$ signature, and the binary error vectors generated by the computation of sn and tg. We set $q_{\mathsf{LNP}} = \hat{q} \cdot q_{\mathsf{SEP}} \approx 2^{52}$ which allows statements over both $\mathbb{Z}_{\hat{q}}$ and $\mathbb{Z}_{q_{\mathsf{SEP}}}$ to be embedded into $\mathbb{Z}_{q_{\mathsf{LNP}}}$. We set our ring dimension of LNP to be $d_{\mathsf{LNP}} = 128$. Again, using the results of Lyubashevsky et al. [LNPS21, Section 2.8], statements over rings of dimension 1024 to the LNP ring. For the LNP commitment matrices, we use dimensions 10×68 for commitments to low-norm elements and $(10+14) \times 54$ for commitments to unconstrained elements. The commitment randomness is sampled uniformy from $\mathcal{B}_1^{54}$ over the LNP ring. With these parameters, the resulting proof size is approximately 48KB.

$\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{\mathsf{1Many}}}$ parameters. Upon augmenting our optimized R_{1Many} relation from 7.6 with the LNP commitment produced by the $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, we obtain a disjunctive clause of 256 literals. Each literals correspond to a matrix-vector pair, where the matrix has dimension $(10+2) \times (68+54)$ over $\mathbb{Z}_{q_{\mathsf{LNP}}}[X]/(X^{d_{\mathsf{LNP}}}+1)$. To instantiate our 1-out-of-2^8 partial binding vector commitment, we set $q_{\mathsf{pbvc}} \approx 2^{60}$ and $d_{\mathsf{pbvc}} = 1024$. To apply GGHK's compiler 5.1, we need to commit to each first messages $\mathbf{w}_i \in (\mathbb{Z}_{q_{\mathsf{LNP}}}[X]/(X^{d_{\mathsf{LNP}}}+1))^{12}$, with $i \in [2^8]$. By using a gadget vector decomposition of base 2^{14}, we can set the PBVC dimensions to $n_{\mathsf{pbvc}} = 2$ and $m_{\mathsf{pbvc}} = 6$. Doing so, we can embedded the decomposition of vectors $\mathbf{w}_i$ in the larger pbvc ring. With these parameters we obtain an M-SIS bound $B_{\mathsf{pbvc}} \approx 2^{56}$. The resulting disjunctive proof has size 851KB.

Combining this with the proof size of $\Pi_{\mathsf{NIZK}}^{\mathcal{R}_{M,\Sigma}}$, we obtain a total proof size of 876KB, which lead to tokens of size 899KB. For comparison, using the asymptotic estimates of Albrecht and Gür [AG24], proving the correct execution of the efficient PRF of Banerjee and Peikert [BP14] directly with LNP, without relying on LaBRADOR, already yields a proof size of at least 3MB. This estimate does not account for the additional norm checks that would need to be performed as part of LNP.

8 Anonymous Counting Tokens in the ROM

In this section, we show how to adapt our generic ARLT Construction 6.1 to obtain an anonymous counting token.

- UKeyGen as follows: $\mathsf{UKeyGen}(\mathsf{crs})$: Given the crs, proceed as follows:
  1. Compute $k_U \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda, \mathsf{aux} := \mathfrak{c})$.
  2. Compute $(c_U, o_U) \leftarrow \Pi_{\mathsf{Com}}^{\mathsf{SEP}}.\mathsf{Commit}(\mathsf{crs}, k)$.
  3. Return $\mathsf{pk}_U := c_U$ and $\mathsf{sk}_U := (k_U, o_U)$.

- $\mathsf{Issue}.U_1(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I, \mathsf{msg})$: Given the crs, $(k_U, o_U) := \mathsf{sk}_U$, pk_I and msg, proceed as follows:
  1. Compute $\hat{k}_{\mathsf{msg}} \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Gen}(1^\lambda)$.
  2. Compute $\mathsf{tg}_{\mathsf{Iss}} \leftarrow \Pi_{\mathsf{PRF}}.\mathsf{Eval}^{\mathsf{H}}(k_U, \mathsf{msg})$.
  3. Compute $\bar{\mathsf{msg}} \leftarrow \mathsf{H}(\mathsf{msg})$.
  4. Compute $(c_{\mathsf{SEP}}, o_{\mathsf{SEP}}) \leftarrow \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, (\bar{\mathsf{msg}}, \hat{k}_{\mathsf{msg}}))$.
  5. Compute $\pi_{\mathsf{Eval}} \leftarrow \Pi^{\mathcal{R}_{\mathsf{Eval},\mathsf{PRF}}}_{\mathsf{NIZK}}.\mathsf{Prove}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}, \mathsf{tg}_{\mathsf{Iss}}), \mathbb{w} := (k_U, o_U, o_{\mathsf{SEP}}, \bar{\mathsf{msg}}))$.
  6. Return $\mathsf{st}_U := (\mathsf{pk}_I, c_{\mathsf{SEP}}, \hat{k}_{\mathsf{msg}}, o_{\mathsf{SEP}}, \mathsf{msg}, \mathsf{sk}_U)$ and $\mathsf{umsg} := (c_{\mathsf{SEP}}, \mathsf{tg}_{\mathsf{Iss}}, \pi_{\mathsf{Eval}})$.
- $\mathsf{Issue}.I_1(\mathsf{crs}, \mathsf{sk}_I, \mathsf{st}_I, \mathsf{pk}_U, \mathsf{umsg})$: Given the crs, sk_I, $\mathsf{Map} := \mathsf{st}_I$, pk_U, and $(c_{\mathsf{SEP}}, \mathsf{tg}, \pi_{\mathsf{Eval}}) := \mathsf{umsg}$, proceed as follows:
  1. If $\Pi^{\mathcal{R}_{\mathsf{Eval},\mathsf{PRF}}}_{\mathsf{NIZK}}.\mathsf{Verify}^{\mathsf{H}}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_U, c_{\mathsf{SEP}}, \mathsf{tg}_{\mathsf{Iss}}), \pi_{\mathsf{Eval}}) = 0$, **abort**.
  2. If $\mathsf{Map}[\mathsf{pk}_U, \mathsf{tg}_{\mathsf{Iss}}] = 1$, **abort**. ▶ *the client has already requested counting tokens for this* msg.
  3. Compute $\tilde{k} \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^\lambda)$.
  4. Compute $c'_{\mathsf{SEP}} \leftarrow c_{\mathsf{SEP}} + \Pi^{\mathsf{SEP}}_{\mathsf{Com}}.\mathsf{Commit}(\mathsf{crs}, (\mathbf{0}, \tilde{k}); \perp)$.
  5. Compute $\sigma'_{\mathsf{SEP}} \leftarrow \Pi_{\mathsf{SEP}}.\mathsf{SignCom}(\mathsf{crs}, \mathsf{sk}_I, c'_{\mathsf{SEP}})$.
  6. Set $\mathsf{Map}[\mathsf{pk}_U, \mathsf{tg}_{\mathsf{Iss}}] \leftarrow 1$.
  7. Return $\mathsf{imsg} := (\tilde{k}, \sigma'_{\mathsf{SEP}})$.
- During Show, proceeds as follows and return $(\mathsf{msg}, \mathsf{sn}, \mathsf{tg})$, where $\mathsf{H}(\mathsf{msg}) = \bar{\mathsf{msg}}$.

In lattice-based setting, we can commit to the output of H by using binary decomposition in the case of Ajtai Commitment [Ajt96], or by using Baum et al. [BDL+18] commitment, which allows to commit to unrestricted ring elements. In addition, using a argument similar to Agrawal et al. [AKSY22], committing to the random-oracle evaluation of msg does not introduce additional complications, since msg is reveal during presentations of the token. The one-more unforgeability, and anonymity/unlinkability of this construction follows from the soundness and anonymity of construction 6.1.

Acknowledgments

Victor Youdom Kemmoe and Anna Lysyanskaya were supported by NSF Grants 2312241, 2154170, and 2247305 as well as the Ethereum Foundation.

References

[ABF+25] Gennaro Avitabile, Vincenzo Botta, Daniele Friolo, Daniele Venturi, and Ivan Visconti. Compact proofs of partial knowledge for overlapping CNF formulae. J. Cryptol., 38(1):7, 2025.

[ABFV22] Gennaro Avitabile, Vincenzo Botta, Daniele Friolo, and Ivan Visconti. Efficient proofs of knowledge for threshold relations. In ESORICS (3), volume 13556 of Lecture Notes in Computer Science, pages 42–62. Springer, 2022.

[AG24] Martin R. Albrecht and Kamil Doruk Gür. Verifiable oblivious pseudorandom functions from lattices: Practical-ish and thresholdisable. In ASIACRYPT (4), Lecture Notes in Computer Science, pages 205–237. Springer, 2024.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In CCS, pages 2087–2104. ACM, 2017.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In EUROCRYPT, Lecture Notes in Computer Science, pages 483–501. Springer, 2012.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In STOC, pages 99–108. ACM, 1996.

[AKSY22] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In CCS, pages 39–53. ACM, 2022.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. J. Math. Cryptol., 9(3):169–203, 2015.

[BBL+14] Abhishek Banerjee, Hai Brenner, Gaëtan Leurent, Chris Peikert, and Alon Rosen. SPRING: fast pseudorandom functions from rounded ring products. In FSE, volume 8540 of Lecture Notes in Computer Science, pages 38–57. Springer, 2014.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable vrfs revisited. In Pairing, volume 5671 of Lecture Notes in Computer Science, pages 114–131. Springer, 2009.

[BDL+18] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In SCN, Lecture Notes in Computer Science, pages 368–385. Springer, 2018.

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In ASIACRYPT (2), volume 12492 of Lecture Notes in Computer Science, pages 464–492. Springer, 2020.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In CRYPTO (1), volume 8042 of Lecture Notes in Computer Science, pages 410–428. Springer, 2013.

[BLNS23a] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In CRYPTO (2), volume 14082 of Lecture Notes in Computer Science, pages 384–417. Springer, 2023.

[BLNS23b] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. IACR Cryptol. ePrint Arch., page 560, 2023.

[BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In CRYPTO (1), volume 8616 of Lecture Notes in Computer Science, pages 353–370. Springer, 2014.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In EUROCRYPT, volume 7237 of Lecture Notes in Computer Science, pages 719–737. Springer, 2012.

[BPS19] Florian Bourse, David Pointcheval, and Olivier Sanders. Divisible e-cash from constrained pseudo-random functions. In ASIACRYPT (1), volume 11921 of Lecture Notes in Computer Science, pages 679–708. Springer, 2019.

[BRS23] Fabrice Benhamouda, Mariana Raykova, and Karn Seth. Anonymous counting tokens. In ASIACRYPT (2), volume 14439 of Lecture Notes in Computer Science, pages 245–278. Springer, 2023.

[BS23] Ward Beullens and Gregor Seiler. Labrador: Compact proofs for R1CS from module-sis. In CRYPTO (5), volume 14085 of Lecture Notes in Computer Science, pages 518–548. Springer, 2023.

[CDLT25a] Rutchathon Chairattana-Apirom, Nico Döttling, Anna Lysyanskaya, and Stefano Tessaro. Everlasting anonymous rate-limited tokens. In ASIACRYPT (6), volume 16250 of Lecture Notes in Computer Science, pages 435–468. Springer, 2025.

[CDLT25b] Rutchathon Chairattana-Apirom, Nico Döttling, Anna Lysyanskaya, and Stefano Tessaro. Everlasting anonymous rate-limited tokens. IACR Cryptol. ePrint Arch., page 1030, 2025.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In CRYPTO, volume 839 of Lecture Notes in Computer Science, pages 174–187. Springer, 1994.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, page 201–210. Association for Computing Machinery, 2006.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In EUROCRYPT, volume 6110 of Lecture Notes in Computer Science, pages 523–552. Springer, 2010.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In EUROCRYPT, volume 3494 of Lecture Notes in Computer Science, pages 302–321. Springer, 2005.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In SCN, volume 2576 of Lecture Notes in Computer Science, pages 268–289. Springer, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In CRYPTO, volume 3152 of Lecture Notes in Computer Science, pages 56–72. Springer, 2004.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In ASIACRYPT, Lecture Notes in Computer Science, pages 125–142. Springer, 2002.

[DFPS23] Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of fiat-shamir with aborts. In CRYPTO (5), Lecture Notes in Computer Science, pages 327–357. Springer, 2023.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. Proc. Priv. Enhancing Technol., 2018(3):164–180, 2018.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2018(1):238–268, 2018.

[DLNS20] Amit Deo, Benoît Libert, Khoa Nguyen, and Olivier Sanders. Lattice-based e-cash, revisited. In ASIACRYPT (2), volume 12492 of Lecture Notes in Computer Science, pages 318–348. Springer, 2020.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In ASIACRYPT (2), volume 8874 of Lecture Notes in Computer Science, pages 22–41. Springer, 2014.

[DM14] Léo Ducas and Daniele Micciancio. Improved short lattice signatures in the standard model. IACR Cryptol. ePrint Arch., page 495, 2014.

[DN12] Léo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In ASIACRYPT, volume 7658 of Lecture Notes in Computer Science, pages 415–432. Springer, 2012.

[dPKPR24] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In CRYPTO (1), volume 14920 of Lecture Notes in Computer Science, pages 409–444. Springer, 2024.

[dPLNS17] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In CCS, pages 1565–1581. ACM, 2017.

[EG14] Alex Escala and Jens Groth. Fine-tuning groth-sahai proofs. In Public Key Cryptography, volume 8383 of Lecture Notes in Computer Science, pages 630–649. Springer, 2014.

[ESLR23] Muhammed F. Esgin, Ron Steinfeld, Dongxi Liu, and Sushmita Ruj. Efficient hybrid exact/relaxed lattice proofs and applications to rounding and vrfs. In CRYPTO (5), volume 14085 of Lecture Notes in Computer Science, pages 484–517. Springer, 2023.

[FMN24] Giacomo Fenzi, Hossein Moghaddas, and Ngoc Khanh Nguyen. Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. J. Cryptol., 37(3):31, 2024.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In CRYPTO, volume 263 of Lecture Notes in Computer Science, pages 186–194. Springer, 1986.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. Electron. Colloquium Comput. Complex., TR96-042, 1996.

[GGHK22] Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Stacking sigmas: A framework to compose σ-protocols for disjunctions. In EUROCRYPT (2), volume 13276 of Lecture Notes in Computer Science, pages 458–487. Springer, 2022.

[GHKS23] Aarushi Goel, Mathias Hall-Andersen, Gabriel Kaptchuk, and Nicholas Spooner. Speed-stacking: Fast sublinear zero-knowledge proofs for disjunctions. In EUROCRYPT (2), Lecture Notes in Computer Science, pages 347–378. Springer, 2023.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In EUROCRYPT, Lecture Notes in Computer Science, pages 31–51. Springer, 2008.

[GPV07] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. IACR Cryptol. ePrint Arch., 2007:432, 2007.

[JRS23] Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. Lattice signature with efficient protocols, application to anonymous credentials. In CRYPTO (2), volume 14082 of Lecture Notes in Computer Science, pages 351–383. Springer, 2023.

[Kat21] Shuichi Katsumata. A new simple technique to bootstrap various lattice zero-knowledge proofs to QROM secure nizks. In CRYPTO (2), volume 12826 of Lecture Notes in Computer Science, pages 580–610. Springer, 2021.

[Kim20] Sam Kim. Key-homomorphic pseudorandom functions from lwe with small modulus. In Advances in Cryptology – EUROCRYPT 2020, pages 576–607. Springer, 2020.

[KLN25] Victor Youdom Kemmoe, Anna Lysyanskaya, and Ngoc Khanh Nguyen. Lattice-based accumulator and application to anonymous credential revocation. IACR Cryptol. ePrint Arch., page 1099, 2025.

[KLOR20] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In CRYPTO (1), Lecture Notes in Computer Science, pages 308–336. Springer, 2020.

[LLM+16] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In ASIACRYPT (2), volume 10032 of Lecture Notes in Computer Science, pages 373–403, 2016.

[LLNW17a] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based prfs and applications to e-cash. In ASIACRYPT (3), volume 10626 of Lecture Notes in Computer Science, pages 304–335. Springer, 2017.

[LLNW17b] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based prfs and applications to e-cash. IACR Cryptol. ePrint Arch., page 856, 2017.

[LMPY16] Benoît Libert, Fabrice Mouhartem, Thomas Peters, and Moti Yung. Practical "signatures with efficient protocols" from simple assumptions. In AsiaCCS, pages 511–522. ACM, 2016.

[LNP22] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In CRYPTO (2). Springer, 2022.

[LNPS21] Vadim Lyubashevsky, Ngoc Khanh Nguyen, Maxime Plançon, and Gregor Seiler. Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In ASIACRYPT (4), Lecture Notes in Computer Science, pages 218–248. Springer, 2021.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. IACR Cryptol. ePrint Arch., page 293, 2013.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. Des. Codes Cryptogr., 75(3):565–599, 2015.

[LS18] Vadim Lyubashevsky and Gregor Seiler. Short, invertible elements in partially splitting cyclotomic rings and applications to lattice-based zero-knowledge proofs. In EUROCRYPT (1), Lecture Notes in Computer Science, pages 204–224. Springer, 2018.

[LW20] Feng-Hao Liu and Zhedong Wang. Rounding in the rings. In CRYPTO (2), volume 12171 of Lecture Notes in Computer Science, pages 296–326. Springer, 2020.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In EUROCRYPT, volume 7237 of Lecture Notes in Computer Science, pages 738–755. Springer, 2012.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In EUROCRYPT, volume 7237 of Lecture Notes in Computer Science, pages 700–718. Springer, 2012.

[MP13] Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In CRYPTO (1), Lecture Notes in Computer Science, pages 21–39. Springer, 2013.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput., 37(1):267–302, 2007.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In EUROCRYPT, Lecture Notes in Computer Science, pages 327–346. Springer, 1999.

[Ped91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In CRYPTO, volume 576 of Lecture Notes in Computer Science, pages 129–140. Springer, 1991.

[Pei07] Chris Peikert. Limits on the hardness of lattice problems in ℓ_p norms. In CCC, pages 333–346. IEEE Computer Society, 2007.

[PW11] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. SIAM J. Comput., 40(6):1803–1844, 2011.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In CRYPTO, volume 435 of Lecture Notes in Computer Science, pages 239–252. Springer, 1989.

[SS11] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In EUROCRYPT 2011, pages 27–47. Springer, 2011.

[TW24] Martin Thomson and Christopher A. Wood. Oblivious HTTP. RFC 9458, January 2024.

[ZLH+22] Gongxian Zeng, Junzuo Lai, Zhengan Huang, Yu Wang, and Zhiming Zheng. Dag-Σ: A dag-based sigma protocol for relations in CNF. In ASIACRYPT (2), Lecture Notes in Computer Science, pages 340–370. Springer, 2022.

A Extra Preliminaries

Definition A.1 (Commitment Scheme)**.** *A commitment scheme for a domain $\mathcal{M}$ is a tuple of* ppt *algorithms* $\Pi_{\mathsf{Com}} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}_{\mathsf{Com}}$*: Given a security parameter $\lambda \in \mathbb{N}$ in unary, the* Setup *algorithm outputs a common reference string* $\mathsf{crs}_{\mathsf{Com}}$.

- $\mathsf{Commit}(\mathsf{crs}_{\mathsf{Com}}, m) \to (c, o)$*: Given a common reference string* $\mathsf{crs}_{\mathsf{Com}}$ *and a value* $m \in \mathcal{M}$*, the* Commit *algorithm outputs a commitment* c *and opening* o*.*
- $\mathsf{Open}(\mathsf{crs}_{\mathsf{Com}}, c, m, o) \to 1/0$*: Given a common reference string* $\mathsf{crs}_{\mathsf{Com}}$*, a commitment* c*, a value* $m \in \mathcal{M}$*, and a opening* o*, the deterministic* Open *algorithm outputs* 1 *if* o *is a valid opening of the value* m *with respect to* c*. ELse, it outputs* 0*.*

In addition, Π_{Com} *should satisfies the following properties:*

- ***Correctness****: For any* $\lambda \in \mathbb{N}$*,* $m \in \mathcal{M}$*,* $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(c, o) \leftarrow \mathsf{Commit}(\mathsf{crs}_{\mathsf{Com}}, m)$*,*
$$\Pr[\mathsf{Open}(\mathsf{crs}_{\mathsf{Com}}, c, m, o) = 1] \geq 1 - \mathsf{negl}(\lambda).$$
- ***Hiding****: For any* ppt *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$*,* $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $b \leftarrow\!\!\$\ \{0,1\}$*,*
$$\Pr\left[\begin{array}{r}(m_0, m_1, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}_{\mathsf{Com}}), \\ (c_b, o_b) \leftarrow \mathsf{Commit}(\mathsf{crs}_{\mathsf{Com}}, m_b) \\ b' \leftarrow \mathcal{A}(\mathsf{st}_{\mathcal{A}}, c_b) : \\ b' = b\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$
- ***Binding****: For any* ppt *adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$*,* $\mathsf{crs}_{\mathsf{Com}} \leftarrow \mathsf{Setup}(1^\lambda)$
$$\Pr\left[\begin{array}{r}(c, m_0, o_0, m_1, o_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}_{\mathsf{Com}}) : \\ \mathsf{Open}(\mathsf{crs}_{\mathsf{Com}}, c, m_0, o_0) = 1 \wedge \\ \mathsf{Open}(\mathsf{crs}_{\mathsf{Com}}, c, m_1, o_1) = 1 \wedge \\ m_0 \neq m_1\end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition A.2 (Digital Signature)**.** *A digital signature for a domain* $\mathcal{M}$ *is a tuple of* ppt *algorithms* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: Given a security parameter* $\lambda \in \mathbb{N}$ *in unary, the* Gen *algorithm outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$*: Given a secret key* sk*, a message* $m \in \mathcal{M}$*, the* Sign *algorithm outputs a signature* σ*.*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to 0/1$*: Given a public key* pk*, a message* $m \in \mathcal{M}$*, and a signature* σ*, the deterministic* Verify *algorithm outputs* 1 *if* σ *is a signature on* m *generated by the secret key associated to* pk*. Else, it outputs* 0*.*

In addition, Σ *should satisfy the following properties:*

- ***Correctness****: For any* $\lambda \in \mathbb{N}$*,* $m \in \mathcal{M}$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*,* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$*,*
$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1\right] \geq 1 - \mathsf{negl}(\lambda)$$
- ***Existential Unforgeability against Chosen Message Attack (EUF-CMA)****: For any* ppt *adversary* $\mathcal{A}$ *with oracle access to* Sign*, there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for any* $\lambda \in \mathbb{N}$*,*
$$\Pr\left[\begin{array}{r}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{pk}) : \\ \mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1 \wedge m^* \notin Q_{\mathsf{Sign}}\end{array}\right] \leq \mathsf{negl}(\lambda)$$
where Q_{Sign} *represents the set of signature queries that* $\mathcal{A}$ *sent to its oracle.*

A.1 Anonymous Counting Tokens

In this section, we recall the definiton of Anonymous Counting Tokens proposed by Benhamouda, Raykova, and Seth [BRS23].

Definition A.3. *An anonymous counting token with message space $\mathcal{M}$ and rate limiting at issuance is a tuple of* ppt *algorithms* $\Pi_{\mathsf{ACT}} = (\mathsf{Setup}, \mathsf{IKeyGen}, \mathsf{UKeyGen}, \mathsf{Issue}, \mathsf{Show}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda, \mathfrak{n}, \mathfrak{c}) \to \mathsf{crs}$: *Given a security parameter $\lambda \in \mathbb{N}$ and a rate limit $\mathfrak{n} \in \mathbb{N}$ such that $\mathfrak{n} = \mathsf{poly}(\lambda)$, the* Setup *algorithm outputs a common reference string* crs.

- $\mathsf{IKeyGen}(\mathsf{crs}) \to (\mathsf{pk}_I, \mathsf{sk}_I, \mathsf{st}_I)$: *Given the* crs*, the issuer key generation algorithm* IKeyGen *outputs a public-secret key pair $(\mathsf{pk}_I, \mathsf{sk}_I)$ and a state st_I.*

- $\mathsf{UKeyGen}(\mathsf{crs}) \to (\mathsf{pk}_U, \mathsf{sk}_U)$: *Given the* crs*, the user key generation algorithm* UKeyGen *outputs a public-secret key pair $(\mathsf{pk}_U, \mathsf{sk}_U)$.*

- $\mathsf{Issue}\left(I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{st}_I, \mathsf{pk}_U) \rightleftharpoons U(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I, \mathsf{msg})\right) \to (\mathsf{st}'_I; \mathcal{T}_{\mathsf{msg}})$: *This an interactive protocol between an issuer I with inputs $(\mathsf{crs}, \mathsf{sk}_I, \mathsf{st}_I, \mathsf{pk}_U)$ and a user U with inputs $(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I, \mathsf{msg})$, where $\mathsf{msg} \in \mathcal{M}$. At the end of the interaction, I outputs an updated state st'_I and U outputs a token dispenser $\mathcal{T}_{\mathsf{msg}}$ for the message* msg.

- $\mathsf{Show}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{msg}, \mathcal{T}_{\mathsf{msg}}) \to (\tau, \mathcal{T}'_{\mathsf{msg}})$: *Given the* crs*, an issuer public key pk_I, a message $\mathsf{msg} \in \mathcal{M}$, and a token dispenser $\mathcal{T}_{\mathsf{msg}}$, the* Show *algorithm outputs a token τ and an updated token dispenser $\mathcal{T}'_{\mathsf{msg}}$.*

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{msg}, \tau) \to 1/0$*: Given the* crs*, an issuer public key pk_I, a message* msg*, and a token τ, the deterministic* Verify *algorithm outputs 1 if τ is a valid token on* msg *with respect to pk_I. Otherwise, it outputs 0.*

In addition, Π_{ACT} should satisfy the following properties:

- **Correctness***: For any $\lambda \in \mathbb{N}$, $\mathfrak{n} = \mathsf{poly}(\lambda)$, and $\mathsf{msg} \in \mathcal{M}$,*

$$\Pr\left[\begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathfrak{n}); \\ (\mathsf{pk}_I, \mathsf{sk}_I, \mathsf{st}_I) \leftarrow \mathsf{IKeyGen}(\mathsf{crs}); \\ (\mathsf{pk}_U, \mathsf{sk}_U) \leftarrow \mathsf{UKeyGen}(\mathsf{crs}); \\ (\mathsf{st}'_I; \mathcal{T}_{\mathsf{msg},0}) \leftarrow \mathsf{Issue}\begin{pmatrix} I(\mathsf{crs}, \mathsf{sk}_I, \mathsf{st}_I, \mathsf{pk}_U) \rightleftharpoons \\ U(\mathsf{crs}, \mathsf{sk}_U, \mathsf{pk}_I, \mathsf{msg}) \end{pmatrix}; \\ (\tau_i; \mathcal{T}_{\mathsf{msg},i}) \leftarrow \mathsf{Show}\left(\mathsf{crs}, \mathsf{pk}_I, \mathsf{msg}, \mathcal{T}_{\mathsf{msg},i-1}\right) \forall i \in [\mathfrak{n}] : \\ \exists j \in [\mathfrak{n}], \mathsf{Verify}(\mathsf{crs}, \mathsf{pk}_I, \mathsf{msg}, \tau_j) = 0 \end{array}\right] \leq \mathsf{negl}(\lambda)$$

- **One-more unforgeability***: Given a rate limit $\mathfrak{n} \in \mathbb{N}$, it should be hard for an adversary controlling P users and performing a total of Q issue queries to either produce $\mathfrak{n}Q + 1$ distinct message-token pairs, or generate more than $\mathfrak{n}P$ tokens for a specific message $\mathsf{msg} \in \mathcal{M}$. More formally, for any* ppt *adversary $\mathcal{A}$ with access to an oracle $\mathcal{O}_{\mathsf{Issue}}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for any $\lambda \in \mathbb{N}$ and rate limit $\mathfrak{n} = \mathsf{poly}(\lambda)$,*

$$\Pr[\mathsf{ACT\text{-}OMU\text{-}Exp}^{\mathcal{A}}(1^\lambda, \mathfrak{n}) = 1] \leq \mathsf{negl}(\lambda)$$

where ACT-OMU-Exp *is defined in Fig. 5*

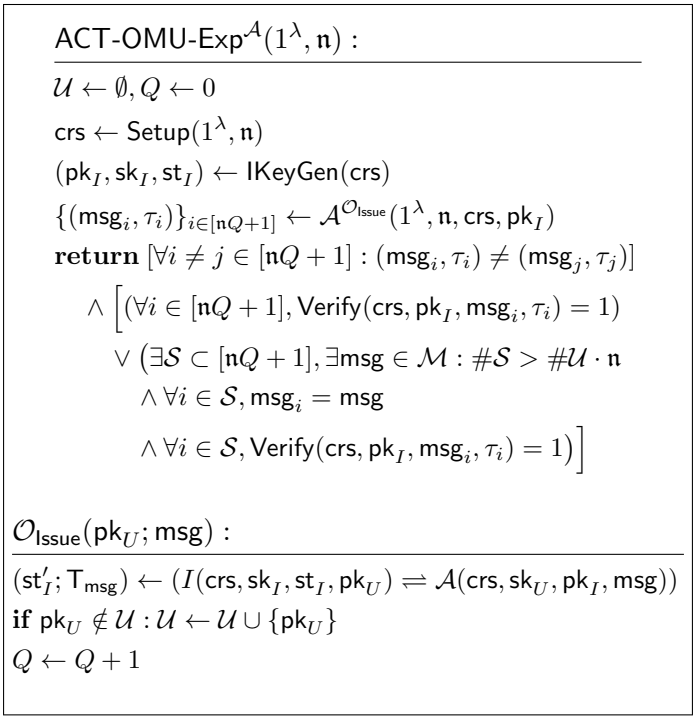

Figure 5: Π_{ACT}'s one-more unforgeability Experiment

- **Anonymity**: *As defined in Definition 4.1.*

B 1-out-of-2^k Partial Binding Vector Commitment

Construction B.1 (1-out-of-2^k Binding Commitment)**.** Let $\Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Equiv})$ be the 1-out-of-2 binding commitment from Construction 5.1. For any $\lambda \in \mathbb{N}$, let $k = k(\lambda)$ be a vector length parameter, $\beta = \beta(\lambda)$ be a norm bound. We construct a 1-out-of-2^k binding commitment for the message space $\mathcal{M}^{2^k} = (R_q^n)^{2^k}$ as follows:

- $\mathsf{Setup}(1^\lambda)$: Given a security parameter $\lambda \in \mathbb{N}$, execute $\mathsf{pp} \leftarrow \Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}}.\mathsf{Setup}(1^\lambda)$, and then, return pp.

- $\mathsf{Gen}(\mathsf{pp}, i)$: Given pp and $i \in \mathbb{Z}_{2^k}$, proceed as follows:

  1. Compute $u \leftarrow \mathsf{BitDecomp}(i)$.
  2. For $j \in \{0, \ldots, k-1\}$, compute $(\tilde{\mathsf{ck}}_j, \tilde{\mathsf{ek}}_j) \leftarrow \Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}}.\mathsf{Gen}(\mathsf{pp}, u[j])$.
  3. Return $\mathsf{ck} := (\tilde{\mathsf{ck}}_0, \ldots, \tilde{\mathsf{ck}}_{k-1})$ and $\mathsf{ek} := (\tilde{\mathsf{ek}}_0, \ldots, \tilde{\mathsf{ek}}_{k-1}, i)$.

- $\mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v})$: Given pp, $(\tilde{\mathsf{ck}}_0, \ldots, \tilde{\mathsf{ck}}_{k-1}) := \mathsf{ck}$, and $\mathbf{v} \in \mathcal{M}^{2^k}$, proceed as follows:

  1. Initialize an empty list L and set $L \leftarrow \mathbf{v}$.
  2. For $\mathtt{i} \in \{0, \ldots, k-1\}$, do the following:
     (a) Initialize an empty list L' and set $\mathtt{j} := 0$.
     (b) Sample $\mathbf{r}_\mathtt{i} \leftarrow D^m_{R,\mathfrak{s}}$.

(c) While $\mathtt{j} < 2^k/2^{\mathtt{i}}$, do the following:
- Compute $\tilde{\mathsf{cmt}} \leftarrow \Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}}.\mathsf{Commit}(\mathsf{pp}, \tilde{\mathsf{ck}}_{\mathtt{i}}, (L[\mathtt{j}], L[\mathtt{j}+1]); \mathbf{r}_{\mathtt{i}})$.
- Append $\tilde{\mathsf{cmt}}$ to L' and set $\mathtt{j} \leftarrow \mathtt{j} + 2$.

(d) Set $L \leftarrow L'$.

3. Return $\mathsf{cmt} := L[0]$ and $o := (\mathbf{r}_0, \ldots, \mathbf{r}_{k-1})$.

- $\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}, o)$: Given pp, ck, $\mathbf{t} := \mathsf{cmt}$, $\mathbf{v} \in \mathcal{M}^{2k}$, and $(\mathbf{r}_0, \ldots, \mathbf{r}_{k-1}) := o$, proceed as follows:

1. Compute $\mathbf{t}' \leftarrow \mathsf{Commit}(\mathsf{pp}, \mathsf{ck}, \mathbf{v}; \mathbf{r}_0, \ldots, \mathbf{r}_{k-1})$.
2. Return $(\mathbf{t}' = \mathbf{t}) \wedge \left(\bigwedge_{\mathtt{i} \in \{0,\ldots,k-1\}} \|\mathbf{r}_{\mathtt{i}}\| \leq \beta\right)$

- $\mathsf{Equiv}(\mathsf{pp}, \mathsf{ek}, \mathbf{v}, \mathsf{cmt}, o, \mathbf{v}')$: Given pp, $(\tilde{\mathsf{ek}}_0, \ldots, \tilde{\mathsf{ek}}_{k-1}, i) := \mathsf{ek}$, $\mathbf{v} \in \mathcal{M}^{2^k}$, cmt, $(\mathbf{r}_0, \ldots, \mathbf{r}_{k-1}) := o$, $\mathbf{v}' \in \mathcal{M}^{2^k}$, proceed as follows:

1. Extract $\mathsf{ck} := (\tilde{\mathsf{ck}}_0, \ldots, \tilde{\mathsf{ck}}_{k-1})$ from ek.
2. If $(\mathbf{v}[i] \neq \mathbf{v}'[i]) \vee (\mathsf{Open}(\mathsf{pp}, \mathsf{ck}, \mathsf{cmt}, \mathbf{v}, o) = 0)$, **abort**.
3. Initialize empty lists L and $\tilde{L}$, and then, set $L \leftarrow \mathbf{v}$ and $\tilde{L} \leftarrow \mathbf{v}'$.
4. For $\mathtt{i} \in \{0, \ldots, k-1\}$, do the following:
   (a) Initialize empty lists L' and $\tilde{L}'$.
   (b) Set $\mathtt{j}_1 := 0$ and $\mathtt{j}_2 := 0$.
   (c) While $\mathtt{j}_1 < 2^k/2^{\mathtt{i}}$, do the following:
   - Compute $\tilde{\mathsf{cmt}} \leftarrow \Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}}.\mathsf{Commit}(\mathsf{pp}, \tilde{\mathsf{ck}}_{\mathtt{i}}, (L[\mathtt{j}], L[\mathtt{j}+1]); \mathbf{r}_{\mathtt{i}})$.
   - Append $\tilde{\mathsf{cmt}}$ to L' and set $\mathtt{j}_1 \leftarrow \mathtt{j}_1 + 2$.

   (d) If $\lfloor i/2^{\mathtt{i}} \rfloor \equiv 0 \bmod 2$, set $\mathtt{s} \leftarrow \lfloor i/2^{\mathtt{i}} \rfloor$. Otherwise, set $\mathtt{s} \leftarrow \lfloor i/2^{\mathtt{i}} \rfloor - 1$.
   (e) $\mathbf{r}'_{\mathtt{i}} \leftarrow \Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}}.\mathsf{Equiv}(\mathsf{pp}, \tilde{\mathsf{ek}}_{\mathtt{i}}, (L[\mathtt{s}, \mathtt{s}+1]), L'[\lfloor \mathtt{s}/2 \rfloor], \hat{o} := \mathbf{r}_{\mathtt{i}}, (\tilde{L}[\mathtt{s}, \mathtt{s}+1]))$.
   (f) While $\mathtt{j}_2 < 2^k/2^{\mathtt{i}}$, do the following:
   - If $\mathtt{j}_2 = \mathtt{s}$, set $\mathtt{j}_2 \leftarrow \mathtt{s} + 2$ and go to line 1.(f).
   - Compute $\tilde{\mathsf{cmt}} \leftarrow \Pi^{\mathsf{1outOf2}}_{\mathsf{PBVC}}.\mathsf{Commit}(\mathsf{pp}, \tilde{\mathsf{ck}}_{\mathtt{i}}, (\tilde{L}[\mathtt{j}], \tilde{L}[\mathtt{j}+1]); \mathbf{r}'_{\mathtt{i}})$.
   - Append $\tilde{\mathsf{cmt}}$ to $\tilde{L}'$ and set $\mathtt{j}_2 \leftarrow \mathtt{j}_2 + 2$.

   (g) Insert $L'[\lfloor \mathtt{s}/2 \rfloor]$ into $\tilde{L}'$ at position $\lfloor \mathtt{s}/2 \rfloor$.
   (h) Set $L \leftarrow L'$ and $\tilde{L} \leftarrow \tilde{L}'$.
5. Return $o' := (\mathbf{r}'_0, \ldots, \mathbf{r}'_{k-1})$.