Post-Quantum Single Secret Leader Election (SSLE) From Publicly Re-randomizable Commitments

Dan Boneh, Aditi Partap, and Lior Rotem

Stanford University
{dabo,aditi712,lrotem}@cs.stanford.edu

Abstract

A *Single Secret Leader Election (SSLE)* enables a group of parties to randomly choose exactly one leader from the group with the restriction that the identity of the leader will be known to the chosen leader and nobody else. At a later time, the elected leader should be able to publicly reveal her identity and prove that she is the elected leader. The election process itself should work properly even if many registered users are passive and do not send any messages. SSLE is used to strengthen the security of proof-of-stake consensus protocols by ensuring that the identity of the block proposer remains unknown until the proposer publishes a block. Boneh, Eskandarian, Hanzlik, and Greco (AFT'20) defined the concept of an SSLE and gave several constructions. Their most efficient construction is based on the difficulty of the Decision Diffie-Hellman problem in a cyclic group.

In this work we construct the first efficient SSLE protocols based on the standard Learning With Errors (LWE) problem on integer lattices, as well as the Ring-LWE problem. Both are believed to be post-quantum secure. Our constructions generalize the paradigm of Boneh et al. by introducing the concept of a re-randomizable commitment (RRC). We then construct several post-quantum RRC schemes from lattice assumptions and prove the security of the derived SSLE protocols. Constructing a lattice-based RRC scheme is non-trivial, and may be of independent interest.

1 Introduction

Leader election is a core component of many consensus protocols used in practice. In proof-of-work systems such as [51], the identity of the leader remains hidden until the moment that the leader publishes a proposed block. In contrast, in many proof-of-stake systems, the identity of the leader is known in advance, long before the leader publishes a proposed block. This opens up the leader to certain attacks, including denial of service, that may prevent the chosen leader from publishing the newly created block. This in turn, can lead to a liveness failure for the chain.

In response, several works have studied *secret* leader election, where the identity of a randomly chosen leader remains secret until she publishes the new block and reveals herself as the leader [45, 40, 9]. Some works [9, 38, 11] even keep the identity of the leader hidden *after* the new block is published. The added secrecy protects the leader from attacks that may prevent her from publishing the new block. However, existing proposals for secret leader election work by electing a few potential leaders *in expectation*, and describing a run-off procedure so that exactly one of the potential leaders is recognized as the final leader once all potential leaders have revealed themselves. The possibility of several potential leaders, however, can lead to wasted effort and may even cause a safety violation in case of an attack on the run-off procedure.

This issue motivates the need for a different type of leader election protocol, called a *Single Secret Leader Election*, or SSLE [24] (see also [30]). An SSLE protocol is comprised of two phases.

- In the first phase, parties may register to participate in leader elections. This step involves publishing some public information on a public bulletin board, while keeping some secret information associated with it private.
- In the second phase, elections are held using a protocol that is executed by the participating parties. The election protocol uses a randomness beacon and the public information on the bulletin to choose a leader among the parties. At a later time, the leader can declare themselves as such by providing a proof that they were selected as the leader.

Informally, an SSLE protocol needs to satisfy three security properties. **Uniqueness** asserts that at most a single party can prove that they were elected as leader. **Fairness** requires that all participating parties have the same probability of being elected as leader, even if some parties are malicious. **Unpredictability** means that until the leader reveals itself, its identity should remain essentially hidden from the other parties, even if a subset of them colludes. It was recently shown that relying on SSLE leads to more efficient consensus protocols than relying on a secret leader election protocol that elects few leaders in expectation [8].

The concept of SSLE was formalized by Boneh, Eskandarian, Hanzlik, and Greco [24] who also presented a number of constructions. Their most efficient construction is based on the Decision Diffie-Hellman problem (DDH) in cyclic groups. We refer to this SSLE protocol as the **BEHG protocol**. The Ethereum Foundation optimized BEHG to obtain *Whisk* [44], which is the current proposal for SSLE to be used in Ethereum consensus. Since then, additional works have suggested alternative SSLE constructions with various security and efficiency tradeoffs (see, for example, [34, 59, 27, 10, 28, 36]).

Due to the potential long-term risk of a large scale quantum computer [60] there is a desire to also develop a *post-quantum* secure SSLE. One approach, already in [24], is an SSLE protocol based on fully homomorphic encryption (FHE). A further optimized FHE-based construction was recently proposed by Freitas et al. [36]. However, the complexity of these proposals is far greater than the simple DDH-based scheme. Another elegant approach to post-quantum SSLE was proposed by Sanso [59], who showed how to adapt Whisk to use an isogeny-based assumption, which is believed to be post-quantum secure. Finally, Drake [34] proposed an SSLE protocol that can be made post-quantum secure, but the proposal inherently relies on the availability of an anonymous broadcast channel (e.g., ToR).

Our results. In this paper we construct the first practical post-quantum SSLE protocols based on the Learning With Errors (LWE) problem [57] and Ring-LWE problem [50]. We do so by generalizing the BEHG protocol using a new concept we call a re-randomizable commitment (RRC). We show that an RRC together with a shuffle protocol gives an SSLE. We then construct a number of RRC schemes from lattices. The next section gives a detailed overview of the construction and explains the technical challenges in building an RRC from lattices.

1.1 Technical Overview

We briefly sketch the main ideas behind our construction. We begin with an abstract view of the BEHG protocol. Then, we present the notion of re-randomizable commitments (RRC) used by this protocol. Finally, we present our new lattice-based post-quantum RRCs for instantiating the abstract BEHG protocol.

The BEHG approach. The BEHG protocol employs a commit-and-shuffle approach. The following is a generalized and abstract view of the protocol.

- When party i registers for elections, it chooses some secret key k_i, computes a commitment c_i to k_i, and publishes c_i. We will define what is needed of this commitment in a minute. To avoid duplicity of secrets, each party also publishes a deterministic hash of k_i.
- At election time, participating parties run a protocol to shuffle and rerandomize the commitments. For simplicity of presentation in this overview, let us assume that the shuffle protocol works as follows: in each round, one of the parties locally permutes the entire list of commitments and then rerandomizes each of the commitments. It then publishes the new list of commitments, and proves in zero-knowledge that this new list is well-formed (i.e., it is obtained from the previous list by permuting and rerandomizing the commitments). Once the shuffle protocol is done, the parties obtain a list of commitments $\tilde{c}_1, \ldots, \tilde{c}_N$, where each $\tilde{c}_i$ is a rerandomization of $c_{\pi(i)}$ for some unknown permutation π on $\{1, \ldots, N\}$. They then let the randomness beacon choose an index $i^* \xleftarrow{\$} \{1, \ldots, N\}$, and party $j^* = \pi(i^*)$ is the chosen leader. In due time, party j^* can prove that it was elected by publishing k_{j^*} and the other parties can check this value against the commitment $\tilde{c}_{i^*}$.

Re-randomizable commitments. We identify several properties that the commitment scheme being used must satisfy for the resulting SSLE protocol to be correct and secure. First, the commitments have to be **re-randomizable** in a very specific sense. Given a commitment c to some value k, one should be able to re-randomize c without knowledge of k or the randomness used to generate c. Moreover, given a value k and a (potentially re-randomized) commitment $\tilde{c}$, one should be able to efficiently test whether $\tilde{c}$ is a commitment to k. In particular, this test should not require the randomness used for re-randomization. In the BEHG protocol, this means that the original committer to $\tilde{c}_{i^*}$ can: (i) recognize itself as the winner of the elections (by checking if $\tilde{c}_{i^*}$ is a commitment to k_{j^*}); and (ii) prove that it won by publishing k_{j^*}.

The commitment scheme should also satisfy the standard notion of **binding**. This means that it should be infeasible to produce a commitment c alongside two *distinct* values k and k', such that c passes both as a commitment to k and as a commitment to k'. In the context of the BEHG protocol, this means that there is only a single party that can prove ownership of the chosen commitment $\tilde{c}_{i^*}$ by publishing k_{j^*}.

Finally, commitments should also be **unlinkable**. This means that given two commitments c_0 and c_1 to two random values, and a re-randomization $\tilde{c}$ for one of them, it should be infeasible to determine if $\tilde{c}$ is a re-randomization of c_0 or of c_1. This is essential for the BEHG protocol to achieve unpredictability: an adversary should not be able to link the chosen commitment $\tilde{c}_{i^*}$ to the original commitment c_{j^*} and therefore identify party j^* as the leader. Looking ahead, the use of re-randomizable commitments in the generalized BEHG SSLE protocol actually requires a stronger notion of unlinkability. We postpone the discussion on this matter and will revisit it shortly.

The DDH-based construction of re-randomizable commitments (RRCs) suggested by BEHG is as follows. Let $\mathbb{G}$ be a cyclic group of order p generated by $g \in \mathbb{G}$. A commitment c to a random value $k \xleftarrow{\$} \mathbb{Z}_p$ is a pair (g^r, g^{rk}) where $r \xleftarrow{\$} \mathbb{Z}_p$. To check if a commitment $c = (c_1, c_2)$ is a consistent with a value k, once can simply check if $c_2 = c_1^k$. To re-randomize, one chooses a random $r' \xleftarrow{\$} \mathbb{Z}_p$ and outputs $\tilde{c} = (c_1^{r'}, c_2^{r'})$. The scheme is perfectly binding, and unlinkability easily follows from the DDH assumption.

It should be noted that previous works also considered other variants of re-randomizable commitments (see, for example, [5, 29]). However, in these works, opening a re-randomized commitment requires knowledge of the randomness used for re-randomization (or a function thereof). Such commitments are much simpler to construct, and indeed, many long-standing algebraic and lattice-based constructions can be easily

re-randomized according to this weaker definition. Unfortunately, as discussed above, such commitments are insufficient for instantiating the BEHG protocol.

RRCs from LWE: A first attempt. Consider the following (flawed) RRC scheme. The secret key space is $\mathbb{Z}_q^n$, where q is a prime and $n \approx \lambda$ is the LWE hardness parameter. To commit to a random $\boldsymbol{k} \in \mathbb{Z}_q^n$ the *Commit* algorithm samples a uniformly random $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and outputs $(\boldsymbol{A}, \boldsymbol{u}) = (\boldsymbol{A}, \boldsymbol{A} \cdot \boldsymbol{k} + \boldsymbol{e})$, where $\boldsymbol{e}$ is an LWE noise vector and $m > n$. To test whether a key $\boldsymbol{k}$ is tied to a commitment $c = (\boldsymbol{A}, \boldsymbol{u})$, we can check whether $\boldsymbol{A} \cdot \boldsymbol{k}$ is close (say, in Euclidean distance) to $\boldsymbol{u}$. We accept $\boldsymbol{k}$ if this is the case and reject otherwise. If $\boldsymbol{A}$ is chosen randomly and $m \approx n \log n$ ($\boldsymbol{A}$ is a "tall" matrix), a standard argument shows that with high probability over the choice of $\boldsymbol{A}$, there are no $\boldsymbol{k}, \boldsymbol{k}'$ and $\boldsymbol{u}$ such that $\boldsymbol{A} \cdot \boldsymbol{k} \approx \boldsymbol{u}$ and $\boldsymbol{A} \cdot \boldsymbol{k}' \approx \boldsymbol{u}$.

To re-randomize, the rerandomization algorithm samples a low-norm m-by-m matrix $\boldsymbol{R}$ and computes $c' = (\boldsymbol{A}', \boldsymbol{u}') = (\boldsymbol{R} \cdot \boldsymbol{A}, \boldsymbol{R} \cdot \boldsymbol{u})$. Since $\boldsymbol{R}$ is of low norm $\boldsymbol{Re}$ may only be slightly longer than $\boldsymbol{e}$. Hence, $\boldsymbol{Re}$ is also short and we have

$$\boldsymbol{A}' \cdot \boldsymbol{k} = \boldsymbol{R} \cdot \boldsymbol{A} \cdot \boldsymbol{k} \approx \boldsymbol{R} \cdot \boldsymbol{A} \cdot \boldsymbol{k} + \boldsymbol{R} \cdot \boldsymbol{e} \approx \boldsymbol{R} \cdot \boldsymbol{u} = \boldsymbol{u}'.$$

The noise does grow a bit with each re-randomization, which is why the scheme only supports a bounded number of re-randomizations (the LWE parameters can be chosen according to the number of re-randomizations required by the SSLE shuffle protocol). In terms of unlinkability, note that assuming LWE is hard, a fresh commitment $c = (\boldsymbol{A}, \boldsymbol{u})$ is just a pseudorandom matrix-vector pair. Moreover, if m is sufficiently greater than n and each row of $\boldsymbol{R}$ has high min-entropy, the leftover hash lemma [39, 43] shows that c' is also pseudorandom, which implies that the scheme is unlinkable.

The problem. Unfortunately, the above analysis is flawed. It is true that the scheme is binding when the matrix $\boldsymbol{A}$ is chosen uniformly at random from $\mathbb{Z}_q^{m \times n}$. But since $\boldsymbol{A}$ is part of the commitment c, the adversary may choose it from some other skewed distribution, thus breaking the binding argument. This is not just an issue of reworking the proof. The scheme is in fact insecure: fix any $\boldsymbol{k}$ and $\boldsymbol{k}'$ and it is easy to come up with a matrix $\boldsymbol{A}$ for which $\boldsymbol{A} \cdot \boldsymbol{k} \approx \boldsymbol{A} \cdot \boldsymbol{k}'$. To fix this issue, one might be tempted to choose the matrix $\boldsymbol{A}$ as part of the public parameters, or to force committers to choose $\boldsymbol{A}$ as the output of a hash function modeled as a random oracle. Indeed, this would make the scheme binding, but then it becomes unclear how to re-randomize the commitments.

The key observation. Let us revisit the naive "proof" of binding for the above construction. If $\boldsymbol{A}$ is indeed chosen uniformly at random, then with overwhelming probability there are no $\boldsymbol{k}$ and $\boldsymbol{k}'$ such that $\boldsymbol{A} \cdot \boldsymbol{k} \approx \boldsymbol{A} \cdot \boldsymbol{k}'$. In particular, this would suggest that for random $\boldsymbol{A}$, $\boldsymbol{k}$ and $\boldsymbol{k}'$ it holds that $\boldsymbol{A} \cdot \boldsymbol{k}$ and $\boldsymbol{A} \cdot \boldsymbol{k}'$ are almost surely far apart. Put differently, for a uniform $\boldsymbol{k}$ and $\boldsymbol{k}'$, there are *very few* matrices $\boldsymbol{A}$ for which $\boldsymbol{Ak} \approx \boldsymbol{Ak}'$. So what if instead of choosing a single $\boldsymbol{k}$, we make the *Commit* algorithm sample the commitment key k as a pair $(\boldsymbol{k}_1, \boldsymbol{k}_2)$ of independent and uniformly-random vectors? One could expect that for two such random pairs $(\boldsymbol{k}_1, \boldsymbol{k}_2)$ and $(\boldsymbol{k}'_1, \boldsymbol{k}'_2)$, the set of matrices $\boldsymbol{A}$ for which $\boldsymbol{A} \cdot \boldsymbol{k}_1 \approx \boldsymbol{A} \cdot \boldsymbol{k}'_1$ *and* $\boldsymbol{A} \cdot \boldsymbol{k}_2 \approx \boldsymbol{A} \cdot \boldsymbol{k}'_2$ is *even smaller*. Indeed, we show that for $\ell \approx n$, if one samples two ℓ-tuples $(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell)$ and $(\boldsymbol{k}'_1, \ldots, \boldsymbol{k}'_\ell)$ of vectors uniformly at random, then with very high probability a matrix $\boldsymbol{A}$ for which $\boldsymbol{A} \cdot \boldsymbol{k}_i \approx \boldsymbol{A} \cdot \boldsymbol{k}'_i$ for every i *simply does not exist*.

Alas, the proposed commitment scheme is binding for keys that are *random* tuples of vectors, but the binding security game allows the adversary to choose the "colliding" keys $(\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell)$ and $(\boldsymbol{k}'_1, \ldots, \boldsymbol{k}'_\ell)$ as it pleases — they need not be uniformly random. On the face of it, it might seem that we are back to square one. Fortunately, this is not the case. The final observation is that for this construction, we *can* make

the commitment algorithm choose the vectors $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell$ as the output of a cryptographic hash function H, without hampering re-randomization. That is, to commit, one samples a matrix $\boldsymbol{A}$ and a key $k \xleftarrow{\$} \{0,1\}^\lambda$, computes $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell \leftarrow \mathsf{H}(k)$ and outputs the commitment $c \leftarrow (\boldsymbol{A}, \{\boldsymbol{A} \cdot \boldsymbol{k}_i + \boldsymbol{e}_i\}_i)$ where all the $\boldsymbol{e}_i$s are independent LWE noise vectors. To test a key k against a commitment $c = (\boldsymbol{A}, \{\boldsymbol{u}_i\}_i)$, the *Test* algorithm simply recomputes $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_\ell$ from k and checks that $\boldsymbol{A} \cdot \boldsymbol{k}_i \approx \boldsymbol{u}_i$ for every $i = 1, \ldots, \ell$.

Adversarial re-randomizations. The construction that we just saw indeed satisfies the notion of unlinkability sketched above. Unfortunately, as we already mentioned, this notion is insufficient for the resulting SSLE protocol to achieve unpredictability. This reason is this: unlinkability only guarantees that if honestly-generated commitments $c_1, \ldots, c_n$ are honestly re-randomized and shuffled, an adversary cannot trace the re-randomized commitments to the original ones. In the SSLE protocol above, an honest re-randomization might follow an adversarial one. So we need to require unlinkability of commitments even after adversarial re-randomizations. We call this *strong unlinkability*.

In the DDH-based construction of BEHG, strong unlinkability comes "for free". Unfortunately, this is not the case with our LWE-based RRC scheme. For example, consider an adversary that given a commitment $c = (\boldsymbol{A}, \{\boldsymbol{A} \cdot \boldsymbol{k}_i + \boldsymbol{e}_i\}_i)$, finds a matrix $\boldsymbol{R}$ such that $\boldsymbol{R} \cdot \boldsymbol{A}$ has short columns. The adversary then uses this $\boldsymbol{R}$ to re-randomize c into $\tilde{c} \leftarrow (\boldsymbol{R} \cdot \boldsymbol{A}, \{\boldsymbol{R} \cdot \boldsymbol{A} \cdot \boldsymbol{k}_i + \boldsymbol{e}_i\}_i)$. Now, even if we honestly re-randomize $\tilde{c}$, we will almost surely end up with a commitment $\hat{c}$ whose first coordinate is still a short-columns matrix. Hence, the adversary can easily trace $\hat{c}$ back to c.

We present several methods to thwart such attacks. In this overview, we focus on what we view as the simplest and most practical one. Ahead of time, all parties commit to the matrices $\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots$ they are going to use for re-randomization using a standard additively homomorphic commitment scheme. When a party now has to carry out its ith re-randomization, it does so using the matrix $\boldsymbol{R}_i + \boldsymbol{R}'_i$, where $\boldsymbol{R}'_i$ is a low norm matrix outputted by a public randomness beacon. Such a beacon can be external or implemented in various standard ways. Using the homomorphic properties of the commitment scheme, everyone can now compute a commitment to $\boldsymbol{R}_i + \boldsymbol{R}'_i$. The re-randomizer can hence prove that this is the matrix it used. Informally, since $\boldsymbol{R}_i$ was committed to ahead of time, it is independent of $\boldsymbol{R}'_i$. Hence, the re-randomizer is forced to use a high-entropy matrix for re-randomization, which guarantees the resulting commitment is from the appropriate distribution. Since $\boldsymbol{R}_i$ is always hidden, $\boldsymbol{R}_i + \boldsymbol{R}'_i$ has high min-entropy even given $\boldsymbol{R}'_i$, and we can still rely on the leftover hash lemma to argue that subsequent honest re-randomizations provide unlinkability.

Extending the scheme to Ring LWE. We extend our LWE-based RRC scheme to the ring setting, relying on the Ring Learning with Errors (Ring-LWE) assumption. As we discuss in Section 5 in detail, moving to the ring setting offers several gains in efficiency. Specifically, we work in a polynomial ring $\mathcal{R}$ modulo a cyclotomic polynomial f, which factors into a constant number of irreducible polynomials over $\mathbb{Z}_q$. Concretely, we choose $q = 3 \bmod 8$ so that f has exactly two irreducible factors f_1, f_2 over $\mathbb{Z}_q$ (but other choices are possible).

The construction follows the same template as our LWE-based construction, but the matrix $\boldsymbol{A}$ is now replaced with a vector of ring elements. To commit, one samples $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q^m$, and a key $k \xleftarrow{\$} \{0,1\}^\lambda$, computes ℓ ring elements as $k_1, \ldots, k_\ell \leftarrow \mathsf{H}(k)$ and the commitment is given by $c \leftarrow (\boldsymbol{a}, \{\boldsymbol{a} \cdot k_i + \boldsymbol{e}_i\}_i)$ where all $\boldsymbol{e}_i$s are independent RLWE noise vectors in $\mathcal{R}_q^m$. Re-randomization is done by sampling a low-norm matrix $\boldsymbol{R} \xleftarrow{\$} \mathcal{R}^{m \times m}$, and computing $c' = (\boldsymbol{R} \cdot \boldsymbol{a}, \{\boldsymbol{R} \cdot \boldsymbol{u}_i\}_i)$. To test a commitment $c = (\boldsymbol{a}, \boldsymbol{u})$ against a key k, one computes $k_1, \ldots, k_\ell \leftarrow \mathsf{H}(k)$ and check that $\boldsymbol{a} \cdot k_i \approx \boldsymbol{u}_i$ for all i. Correctness and unlinkability are proven similarly to the integer case, with one exception: instead of relying on the leftover hash lemma,

we rely on the regularity lemma of [61].

Two main observations make our ring-based scheme more efficient than our integer-based one:

- We can choose ℓ to be smaller than in the integer case, and still make the binding argument go through. Intuitively, the reason is that each entry of $\boldsymbol{a} \cdot k_i$ is now a polynomial in the ring $\mathcal{R}$ and not an integer. Thus, we may hope that it has more than $\log q$ bits of min-entropy (roughly the entropy of a random integer in $\mathbb{Z}_q$). If this is indeed the case, then the probability that $\boldsymbol{a} \cdot k \approx \boldsymbol{a} \cdot k'$, over the choice of random $\boldsymbol{a}, k, k'$, is much smaller than the probability that $\boldsymbol{a}^T \cdot \boldsymbol{k} \approx \boldsymbol{a}^T \cdot \boldsymbol{k'}$ in the integer case for random $\boldsymbol{a}, \boldsymbol{k}, \boldsymbol{k'} \xleftarrow{\$} \mathbb{Z}_q^n$. This would imply that we can choose ℓ to be smaller, resulting in smaller commitments. To argue that $\boldsymbol{a} \cdot k_i$ indeed has high min-entropy, we rely on the particular structure of the ring $\mathcal{R}$. If $k \neq k'$, it means that the polynomials must be distinct modulo f_1 or modulo f_2. Assume with loss of generality that they are distinct modulo f_1. Since f_1 is irreducible mod q, $a \cdot (k - k')$ is uniformly random in $\mathbb{Z}_q[x]/f_1$, and hence it has at least $\approx \deg(f_1) \cdot \log q$ bits of min entropy. This analysis is inspired by the statistically-binding commitments of Benhamouda, Krenn, Lyubashevsky, and Pietrzak [21].

- The second observation is that our use of the leftover hash lemma in the LWE setting incurred an overhead that can be avoided in the Ring LWE setting. To explain this point, we need to revisit our LWE unlinkability argument in more detail. Recall that we wanted to argue that if we have a commitment $c = (\boldsymbol{A}, \boldsymbol{U})$ and we re-randomize it to $c' = (\boldsymbol{R} \cdot \boldsymbol{A}, \boldsymbol{R} \cdot \boldsymbol{U})$, then the commitment c' we end up with is pseudorandom. The first step was to argue that c is pseudorandom, thanks to the LWE assumption. This step remains essentially unchanged here, relying on the Ring-LWE assumption instead. The second step was to rely on the leftover has lemma; this step required each row of $\boldsymbol{R}$ to have more than $\Omega((n + \ell) \cdot \log q)$ bits of min-entropy. This implied that m had to be set to be at least $(n + \ell) \cdot \log q$. In the ring setting, however, since each coordinate of $\boldsymbol{R}$ can have $\Omega(n)$ bits of min-entropy, m can be reduced to roughly $\log q$. This results in much "shorter" matrices $\boldsymbol{A}, \boldsymbol{U}$ making up the commitment.

Reducing communication. Catalano, Fiore, and Giuta [28] observed that when instantiating the BEHG protocol with a DDH-based RRC of the form $c = (g^r, g^{rk})$, the commitments of all parties can share the same first coordinate $h = g^r$, which is part of the public parameters. Then, to re-randomize N commitments $(h^r, g^{rk_1}, \ldots, g^{rk_N})$, a shuffler can sample a single $r' \xleftarrow{\$} \mathbb{Z}_q$ and raise all the elements to the r'. This optimization cuts storage and communication by about half. It is tempting to implement this optimization using our lattice-based commitments; have all commitments share the first coordinate $\boldsymbol{A}$ (or $\boldsymbol{a}$ in the ring setting) and use a single re-randomization matrix $\boldsymbol{R}$ to re-randomize all commitments. The problem is that to retain unlinkability, the dimensions of $\boldsymbol{R}$ need to grow as a function of the number of commitments N, which may eliminate the gains of sharing $\boldsymbol{A}$ across all commitments. We discuss this further in Section 7 where we consider settings where this can still lead to some savings.

Post-quantum proof of shuffle. Recall that in the BEHG protocol, after each shuffle, the shuffling party has to prove that it indeed performed a valid shuffle; that is, it applied the *Randomize* algorithm of the RRC scheme to each commitment and then permuted the resulting commitments. This can be done by using any general-purpose non-interactive argument of knowledge, proving that the shuffler knows random coins for *Randomize* and a permutation that together yield the resulting list of re-randomized commitments (for such argument systems based on post-quantum secure assumptions, see for example [20, 18, 19, 4, 23, 41, 12, 6, 49, 2, 53] and the references therein).

When using our RLWE-based RRC commitments, we also show how we can change the recent lattice-based proof-of-shuffle protocol of [31] to work with our commitments. This is a simple protocol that may provide better concrete efficiency. This is discussed in more detail in Section 8.

2 Preliminaries

In this section, we present the basic notions and cryptographic primitives that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow\!\$\, X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$, we denote by $x \leftarrow\!\$\, \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For a pair X, Y of distributions defined over the same domain Ω, we denote by $\mathsf{SD}(X, Y)$ the *statistical distance* between them, defined as $\mathsf{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$.

We denote matrices by boldface capital letters, e.g. $\boldsymbol{A}$, and vectors in boldface lower-case letters, e.g. $\boldsymbol{v}$. We may use a non-bold capital letter, e.g. A or V, to describe a matrix or a vector, when we wish to emphasize that this matrix or vector is being treated as a random variable. As standard, we identify $\mathbb{Z}_q$ for a prime q with the set $(-q/2, \ldots, q/2]$, and we define the absolute value of an element $x \in \mathbb{Z}_q$ as $|x| = \{\min |y| : y \in \mathbb{Z}, y = x \pmod q\}$.

For $n, p \in \mathbb{N}$ where p is prime, we define the rings $\mathcal{R} = \mathbb{Z}[x]/f(x)$ and $\mathcal{R}_p = \mathcal{R}/\langle p \rangle$, where $f(x)$ is monic and of degree n. That is, $\mathcal{R}_p$ is the ring of polynomials modulo $f(x)$ with integer coefficients in $\mathbb{Z}_p$. We define the norm of elements in these rings to be the norm of their coefficient vector in $\mathbb{Z}^n$, which is also called the coefficient embedding. For any $g(x) = \sum_{i \in 0 \cup [n-1]} \alpha_i x^i \in \mathcal{R}$, we use $\mathsf{coeff}(g)$ to denote the vector $\{\alpha_0, \ldots, \alpha_{n-1}\}$, i.e. the coefficient embedding of $g(x)$, and the norm is defined as follows:

$$||g||_1 = \sum \alpha_i \quad ||g||_2 = (\sum \alpha_i^2)^{1/2} \quad ||g||_\infty = \max|\alpha_i|$$

For a vector $\boldsymbol{v}$ over $\mathcal{R}$, we define $||\boldsymbol{v}|| = (\sum_i ||\boldsymbol{v}_i||^2)^{1/2}$.

2.1 Lattice Assumption

The paper makes use of two basic and standard lattice-based assumptions, the learning with errors (LWE) assumption and the short integer solution (SIS) assumption, both of which over integer lattices. We briefly recall these assumptions here. For a more detailed survey of these assumptions and their hardness, see, for example, [54] and the many references therein.

The LWE assumption. We rely on the following formulation of the learning with errors (LWE) problem, introduced by Regev [58]. The problem is parameterized by a prime modulus q, a vector length n which typically corresponds to the security parameter λ, and a noise distribution χ. For our needs, the important thing is that χ is highly concentrated on low-norm vectors such that with overwhelming probability $\|\boldsymbol{x}\|_2 \leq \delta$ for $\boldsymbol{x} \xleftarrow{\$} \chi$ for some $\delta = \delta(\lambda)$ (one typically takes χ to be a discrete Gaussian with appropriate parameters)

Definition 2.1. *Let $q = q(\lambda)$ be a prime, $n = n(\lambda)$ be an integer, and $\chi = \chi(\lambda)$ be a distribution over $\mathbb{Z}_q$, all public functions of the security parameter $\lambda \in \mathbb{N}$. The (q, n, χ)-LWE assumption states that for every probabilistic polynomial time algorithm $\mathcal{A}$ and for all polynomially-bounded functions $m = m(\lambda)$ there exists a negligible function $\nu(\cdot)$ such that*

$$\mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\boldsymbol{A}, \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}) = 1] - \Pr[\mathcal{A}(\boldsymbol{A}, \boldsymbol{v}) = 1]| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $\boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $e \xleftarrow{\$} \chi^m$, and $\boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_q^m$.

2.2 Ring Lattice Assumption

We will also use the ring-based variant of the LWE assumption, introduced by [50].

The RLWE assumption. This problem is also parameterized by the prime modulus q, degree of the modulus polynomial n, and a noise distribution χ. We focus on a special case of the Ring-LWE problem where $f(x) = x^n + 1$, and n is a power of two. Similar to LWE, χ is highly concentrated on low-norm polynomials such that with overwhelming probability $||x||_2 \leq \delta$ for $x \leftarrow\!\!\$\, \chi$ for some $\delta = \delta(\lambda)$. χ is usually taken to be a discrete gaussian in the coefficient embedding of $\mathcal{R}$.

Definition 2.2. *Let $q = q(\lambda)$ be a prime, $n = n(\lambda)$ be an integer, and $\chi = \chi(\lambda)$ be a distribution over $\mathcal{R}$, all public functions of the security parameter $\lambda \in \mathbb{N}$. The (q, n, χ)-RLWE assumption states that for every probabilistic polynomial time algorithm $\mathcal{A}$ and for all polynomially-bounded functions $m = m(\lambda)$, there exists a negligible function $\nu(\cdot)$ such that*

$$\mathsf{Adv}^{\mathrm{rlwe}}_{\mathcal{A}}(\lambda) := \left|\Pr\left[\mathcal{A}(\boldsymbol{a}, \boldsymbol{b}) = 1\right] - \Pr\left[\mathcal{A}(\boldsymbol{a}, \boldsymbol{v}) = 1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}_q^m, s \xleftarrow{\$} \mathcal{R}_q, \boldsymbol{e} \xleftarrow{\$} \chi^m, \boldsymbol{b}_i = \boldsymbol{a}_i \cdot s + \boldsymbol{e}_i \,\forall\, i \in [m]$, and $\boldsymbol{v} \xleftarrow{\$} \mathcal{R}_q^m$.

2.3 Randomness Extraction

We will use the following lemma from the work of Gentry, Peikert, and Vaikuntanathan [39]. The lemma follows from the leftover hash lemma [43].

Lemma 2.1 ([39, 43])**.** *Let q be a prime and let m, n be integers. Let R, A and B be random variables distributed uniformly in $\{-1, 1\}^{m \times m}$, $\mathbb{Z}_q^{m \times n}$, and $\mathbb{Z}_q^{m \times n}$, respectively. Then, it holds that*

$$\mathsf{SD}\left((A, R \cdot A), (A, B)\right) \leq \frac{m}{2} \cdot \sqrt{2^{-m + n \log q}}.$$

When working over polynomial rings, we will not be able to use the leftover hash lemma. Instead, we will use the regularity lemma defined over rings [61].

Lemma 2.2 (Generalization of Theorem 3.2, [61])**.** *Let $\mathbb{F}$ be a finite field and $f \in \mathbb{F}[x]$ be monic and of degree $n > 0$. Let $\mathcal{R}$ be the ring $\mathbb{F}[x]/f$ and $m > 0$. For every $i, j \in [m]$ and $k \in [n]$, let $D_{i,j,k} \subseteq \mathbb{F}$, with $|D_{i,j,k}| = d$. Let A, B be random variables distributed uniformly in $\mathcal{R}^{m \times \ell}$. Let $R \in \mathcal{R}^{m \times m}$ be a matrix of polynomials, wherein the kth coefficient of $R_{i,j}$ is chosen uniformly randomly and independently from $D_{i,j,k}$, for all $i, j \in [m]$ and $k \in [n]$. Then, it holds that,*

$$\mathsf{SD}\left((A, RA), (A, B)\right) \leq \frac{m}{2} \sqrt{\prod_{i \in [t]} \left(1 + \left(\frac{|\mathbb{F}|}{d^m}\right)^{\mathsf{deg}(f_i)}\right)^{\ell} - 1}$$

where $f = \prod_{i \in [t]} f_i$ is the factorization of f over $\mathbb{F}[x]$, and $\mathsf{deg}(f_i)$ is the degree of the polynomial f_i.

Specifically, we will choose $\mathbb{F} = \mathbb{Z}_q$ and $D_{i,j,k} = \{-1, 1\} \,\forall\, i, j \in [m], k \in [n]$.

We will also rely on the following definition for the norm of a matrix and a related lemma from Agrawal, Boneh, and Boyen [1] (a similar lemma appears in [3]), which states that a random Bernoulli matrix has low norm with overwhelming probability.

Definition 2.3. *Let $\boldsymbol{R}$ be an $m \times m$ matrix over $\mathbb{Z}$. Let $\boldsymbol{B}_m := \{\boldsymbol{x} \in \mathbb{R}^m : \|\boldsymbol{x}\|_2 = 1\}$ be the unit ball in $\mathbb{R}^m$. Define the norm of the matrix $\boldsymbol{R} \in \mathbb{Z}^{m \times m}$ as*

$$\|\boldsymbol{R}\| := \max_{\boldsymbol{x} \in \boldsymbol{B}_m} \|\boldsymbol{R} \cdot x\|_2 .$$

The norm for a matrix in $\mathcal{R}_q^{m \times m}$ is defined similarly. The following two lemmas bound the norm of random matrices where all entries are sampled i.i.d. from a distribution concentrated around 0.

Lemma 2.3 ([1, 3])**.** *Let q be a prime and let m be an integer. Let R be a random variable uniformly sampled from $\{-1, 1\}^{m \times m}$. Then, there is a universal constant $C > 0$ such that*

$$\Pr\left[\|R\| \geq C \cdot \sqrt{m}\right] < e^{-2m}.$$

Lemma 2.4. *Let q be a prime and let m, n be integers. Let $R \in \mathcal{R}_q^{m \times m}$ be a random variable, such that for all $i, j \in [m]$, the coefficient vector of $R_{i,j}$ is sampled uniformly at random from $\{-1, 1\}^n$. Then,*

$$\Pr\left[\|R\| \geq m\sqrt{mn} \cdot \omega(\sqrt{logn})\right] < negl(n)$$

Proof. Consider any vector $\boldsymbol{u} \in \mathcal{R}_q^m$ such that $\|\boldsymbol{u}\|_2 = 1$. We first look at $R_{i,j} \cdot \boldsymbol{u}_j$ for some $i, j \in [m]$. As observed in [46], the coefficient embedding of $R_{i,j} \cdot \boldsymbol{u}_j$ can be written as the multiplication of $\mathsf{coeff}(\boldsymbol{u}_j)$ with a nega-circulant matrix whose entries are derived from the coefficients of $R_{i,j}$. Specifically, we have,

$$\mathsf{coeff}(R_{i,j} \cdot \boldsymbol{u}_j) = \begin{bmatrix} \mathsf{coeff}(R_{i,j})_0 & -\mathsf{coeff}(R_{i,j})_{n-1} & \dots & -\mathsf{coeff}(R_{i,j})_1 \\ \mathsf{coeff}(R_{i,j})_1 & \mathsf{coeff}(R_{i,j})_0 & \dots & -\mathsf{coeff}(R_{i,j})_2 \\ \vdots & \vdots & \vdots & \vdots \\ \mathsf{coeff}(R_{i,j})_{n-1} & \mathsf{coeff}(R_{i,j})_{n-2} & \dots & \mathsf{coeff}(R_{i,j})_0 \end{bmatrix} \mathsf{coeff}(\boldsymbol{u}_j)$$

Without loss of generality, let us consider the last coefficient of the product, i.e. $\mathsf{coeff}(R_{i,j} \cdot \boldsymbol{u}_j)_{n-1} = \sum_{k=0}^{n-1} \mathsf{coeff}(R_{i,j})_{n-1-k} \cdot \mathsf{coeff}(\boldsymbol{u}_j)_k$. Since each coefficient of $R_{i,j}$ is sampled uniformly randomly from $\{-1, 1\}$, this sum can be re-written as $\sum_{k=0}^{n-1} x_k$, where $x_k = \mathsf{coeff}(R_{i,j})_{n-1-k} \cdot \mathsf{coeff}(\boldsymbol{u}_j)_k$. Since $E(x_k) = 0$ and $x_k \in \{\mathsf{coeff}(\boldsymbol{u}_j)_k, -\mathsf{coeff}(\boldsymbol{u}_j)_k\}$ for all k, applying the Hoeffding bound (similar to Lemma 16 proof in [1]), we get:

$$\Pr\left[\mathsf{coeff}(R_{i,j} \cdot \boldsymbol{u}_j)_{n-1} > \|\mathsf{coeff}(\boldsymbol{u}_j)\| \cdot \omega(\sqrt{\log n})\right] < \mathrm{negl}(n)$$

Next, we note that $\|\boldsymbol{u}\| = 1$ implies that $\|\boldsymbol{u}_j\| \leq 1$ for all j. Hence, each coefficient of $R_{i,j} \cdot \boldsymbol{u}_j$ is bounded by $\omega(\sqrt{\log n})$ with high probability. This implies that , for all $i, j \in \{0, \dots, m-1\}$,

$$\Pr\left[\|R_{i,j} \cdot \boldsymbol{u}_j\| > \sqrt{n} \cdot \omega(\sqrt{\log n})\right] < \mathrm{negl}(n) \tag{1}$$

since a vector whose entries are all less than some bound B, has L_2 norm less than $\sqrt{n}B$.

We now want to show that

$$\Pr\left[\sum_{j=0}^{m-1} R_{i,j}\boldsymbol{u}_j > m\sqrt{n} \cdot \omega(\sqrt{\log n})\right] < \mathrm{negl}(n) \tag{2}$$

This follows directly by combining Eq. 1 with the fact that $\|\boldsymbol{v}_1 + \boldsymbol{v}_2\| \leq \|\boldsymbol{v}_1\| + \|\boldsymbol{v}_2\|$.

Eq. 2 gives us a bound on the norm of every element in $R\boldsymbol{u}$. The lemma now follows since a vector whose entries are all bounded by $B = m\sqrt{n} \cdot \omega(\sqrt{\log n})$ with high probability, has L_2 norm less than $\sqrt{m}B$. □

3 Re-randomizable Commitments

Informally, a re-randomizable commitment (RRC, for short) is a scheme that allows one to commit to random keys.[1] Moreover, an RRC scheme supports re-randomizations of commitments: given a commitment c to a key k, one should be able to re-randomize to commitment to produce a new commitment c' for k. Importantly, knowledge of c suffices for such re-randomization, and no additional secrets are needed. In particular, the re-randomizing entity is not required to know the key k nor the randomness used to create c.

We first present the syntax for RRC schemes and the associated correctness requirement. Then, we discuss two security notions that such schemes should satisfy.

3.1 Syntax & Correctness

An RRC scheme R is a tuple of four algorithms:

- *Setup*$(1^\lambda) \to pp$: outputs public parameters pp,
- *Commit*$(pp) \to (c, k)$: outputs a commitment string c and a key k,
- *Randomize*$(pp, c) \to c'$: randomize the commitment,
- *Test*$(pp, c, k) \to \{0, 1\}$: outputs 1 if k is a valid key for c.

The first three are probabilistic polynomial time (PPT) and the fourth is deterministic polynomial time.

In terms of correctness, we require that *Test*(pp, c, k) outputs 1 for (c, k) outputted by *Commit*(pp). Moreover, *Test*(pp, c', k) should output 1 if c' was obtained from c via at most B consecutive rerandomizations, where B is a parameter. We call this correctness requirement *B-randomizability*. If a scheme is B-randomizable for all B, we call it *fully randomizable*.

Definition 3.1. *An RRC scheme is B-**randomizable** if there exists a negligible function $\nu(\cdot)$ such that the following holds for every $\lambda \in \mathbb{N}$:*
let $pp \xleftarrow{\$}$ Setup(1^λ), $(c_0, k) \xleftarrow{\$}$ Commit(pp), and $c_i \xleftarrow{\$}$ Randomize(pp, c_{i-1}) for $i = 1, 2, \ldots$, then

$$\Pr\Big[\textit{Test}(pp, c_i, k) = 1 \quad \text{for } i = 0, 1, 2, \ldots, B\Big] \geq 1 - \nu(\lambda).$$

An RRC scheme that is B-randomizable for all $B \in \mathbb{N}$ is said to be **fully randomizable**. □

For the notion of RRC schemes to be non-trivial, we require that the key k generated by *Commit* to have high min-entropy.

Definition 3.2. *An RRC scheme is B-**randomizable** is **non-trivial** if there exists a negligible function $\nu(\cdot)$ such that the following holds for every $\lambda \in \mathbb{N}$: let $pp \xleftarrow{\$}$ Setup(1^λ), $(c_0, k_0) \xleftarrow{\$}$ Commit(pp) and $(c_1, k_1) \xleftarrow{\$}$ Commit(pp), then*

$$\Pr[k_0 = k_1] \leq \nu(\lambda).$$

□

3.2 Notions of Security

An RRC scheme should satisfy two security properties: Binding and Unlinkability.

[1] Committing to random keys is sufficient for the main application we consider, which is SSLE protocols. Observe, however, that such a scheme can be easily converted into a scheme that allows one to commit to arbitrary messages via a one-time pad.

Game $\mathbf{G}_{\mathcal{A},\mathsf{R}}(\lambda, B)$

$b \xleftarrow{\$} \{0,1\}$
$pp \xleftarrow{\$} \mathsf{R}.\mathit{Setup}(1^\lambda)$
$(c_0, k_0) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp), \quad (c_1, k_1) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp)$
$(\mathsf{state}, i_0, i_1) \xleftarrow{\$} \mathcal{A}(pp, c_0, c_1)$
if $(i_0 > B)$ OR $(i_1 > B)$: **abort**
if $(i_0 = 0)$ OR $(i_1 = 0)$: **abort**
$c \leftarrow c_b$
for t **in** $\{1, \ldots, i_b\}$: $c \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, c)$
$b' \xleftarrow{\$} \mathcal{A}(c, \mathsf{state})$
return $b = b'$

Figure 1: The security game for an adversary $\mathcal{A}$ attacking the unlinkability of an RRC scheme R

Binding. Similarly to standard commitment schemes, we require that a commitment can be tied to at most one key.

Definition 3.3. *An RRC scheme is* **perfectly binding** *if for every $\lambda \in \mathbb{N}$ and for all c, k, k' we have*

$$\Pr_{pp \xleftarrow{\$} \mathit{Setup}(1^\lambda)}\left[k \neq k' \text{ AND } \mathit{Test}(pp, c, k) = \mathit{Test}(pp, c, k') = 1\right] = 0. \tag{3}$$

□

Condition (3) ensures that a commitment c will *never* be accepted by two distinct keys. As we will later discuss, this is satisfied by the previous DDH-based construction of Boneh et al. [24]. For our lattice-based construction, we need to weaken this condition a bit and only require that (3) holds computationally. This leads to the following definition.

Definition 3.4. *We say that an RRC scheme is* **computationally binding** *if for all* PPT *adversaries $\mathcal{A}$ the following function is negligible.*

$$\Pr\left[k \neq k' \text{ AND } \mathit{Test}(pp, c, k) = \mathit{Test}(pp, c, k') = 1 \ : \ \begin{array}{c} pp \xleftarrow{\$} \mathit{Setup}(1^\lambda) \\ (c, k, k') \xleftarrow{\$} \mathcal{A}(pp) \end{array}\right] \tag{4}$$

□

Unlinkability. An RRC scheme R is unlinkable if a PPT adversary is unable to distinguish the i-th re-randomization of a commitment c_0 from the j-th re-randomization of another commitment c_1. This is captured in the security game $\mathbf{G}_{\mathcal{A},\mathsf{R}}(\lambda, B)$ in Figure 1. As usual, we define the adversary's advantage in this game as

$$\mathsf{Adv}^{\mathrm{rrc}}_{\mathcal{A},\mathsf{R},B}(\lambda) := \left|2\Pr[\mathbf{G}_{\mathcal{A},\mathsf{R}}(\lambda, B) = 1] - 1\right|.$$

Definition 3.5. *A B-randomizable RRC scheme is* **unlinkable** *if for all* PPT *adversaries $\mathcal{A}$ the function $\mathsf{Adv}^{\mathrm{rrc}}_{\mathcal{A},\mathsf{R},B}(\lambda)$ is negligible.*

We make two remarks on the unlinkability definition:

- Looking ahead, for some applications, we might want the scheme to remain unlinkable even if adversarial re-randomizations were applied to it at some point. We present such a definition in Section 6. We also discuss ways to augment our basic LWE-based and Ring-LWE-based constructions to accommodate this stronger security definition. Since the stronger unlinkability definition is much more complicated than the one in Fig. 1, we first focus on this weaker notion.

- An unlinkable RRC scheme is, in particular, *hiding*. Meaning, that a commitment c leaks no information (in a computational sense) regarding the committed key k. Intuitively, an adversary that can distinguish between a commitment to a key k and a commitment to a different key k' can trivially link a commitment c to either a commitment c_0 to k_0 or to a commitment c_1 to k_1 by outputting the bit b such that c is a commitment to k_b.

3.3 An RRC scheme based on DDH

Equipped with the above definitions, we can briefly recall the DDH-based RRC scheme used in [24]. The scheme, called R_{ddh}, is defined by:

- *Setup*(1^λ): choose a finite cyclic group $\mathbb{G}$ with generator $g \in \mathbb{G}$ and output $pp := (\mathbb{G}, g)$.
- *Commit*(pp): choose random $u \xleftarrow{\$} \mathbb{G}$ and $k \xleftarrow{\$} \mathbb{Z}_q$, set $c \leftarrow (u, u^k)$, and output (c, k).
- *Randomize*(pp, c): parse $c = (u, v)$, choose a random $\rho \xleftarrow{\$} \mathbb{Z}_q$, and output $c' := (u^\rho, v^\rho)$.
- *Test*(pp, c, k): parse $c = (u, v)$ and output 1 iff $u^k = v$, otherwise output 0.

Theorem 3.1 ([24])**.** *If the DDH assumption holds in $\mathbb{G}$ then R_{ddh} is a perfectly-binding, unlinkable, and fully randomizable RRC.*

The fact that the scheme is fully randomizable and perfectly binding is easy to observe. The proof of unlinkability is a direct application of DDH. In the next section, we construct an RRC scheme that is post-quantum secure based on the LWE assumption.

4 A Construction from Learning with Errors

In this section, we present a construction of an RRC scheme from the LWE assumption [58] (see Section 2). An informal overview of the construction is presented in Section 1.1.

4.1 The Construction

Our construction of an RRC scheme from LWE, denoted $\mathsf{R}_{\mathsf{lwe}}$ is presented in Fig. 2. The construction is parameterized by an integer B, which serves as a bound on the number of rerandomizations that can be applied to a commitment. In the construction, we use Δ to denote $(C \cdot \sqrt{m})^B \cdot \delta$, where m is a parameter of the scheme determined by the analysis (think of $m = O(\lambda)$), C is the universal constant from Lemma 2.3 and δ is a bound on the ℓ_2 norm of the LWE noise vectors used in the construction.

- *Setup*(1^λ):

  1 : Let $n := \lambda$, choose a prime q, and choose $m = m(n, q)$ and $\ell = \ell(n, q)$.

  // we will explain how to choose m and ℓ in the analysis

  2 : Let χ be the LWE noise distribution over $\mathbb{Z}_q$.

  // if $\boldsymbol{e} \xleftarrow{s} \chi^m$, and we lift $\boldsymbol{e}$ to $\mathbb{Z}^m$, then with high probability, $\|\boldsymbol{e}\|_2 \leq \delta$ for some $\delta \ll q$

  3 : **return** $pp \leftarrow (\lambda, q, n, m, \ell, \chi)$

- *Commit*(pp):

  1 : $\boldsymbol{A} \xleftarrow{s} \mathbb{Z}_q^{m \times n}$ // choose a random matrix $\boldsymbol{A}$

  2 : $k \xleftarrow{s} \{0,1\}^{1^\lambda}$ // choose a random λ-bit string

  3 : $\boldsymbol{V} \leftarrow \mathsf{H}(k) \in \mathbb{Z}_q^{n \times \ell}$ // hash k to an n-by-ℓ matrix

  4 : sample $\boldsymbol{E} \in \mathbb{Z}_q^{m \times \ell}$ from the LWE noise distribution $\chi^{m \times \ell}$

  // then for each column $\boldsymbol{e}$ of $\boldsymbol{E}$, $\|\boldsymbol{e}\|_2 \leq \delta$ w.h.p when $\boldsymbol{e}$ is lifted to $\mathbb{Z}^m$

  5 : $\boldsymbol{U} \leftarrow \boldsymbol{A} \cdot \boldsymbol{V} + \boldsymbol{E} \in \mathbb{Z}_q^{m \times \ell}$

  6 : $c \leftarrow (\boldsymbol{A}, \boldsymbol{U})$

  7 : **return** (c, k)

- *Randomize*(pp, c): parse $c = (\boldsymbol{A}, \boldsymbol{U})$ and do

  1 : sample a random matrix $\boldsymbol{R} \xleftarrow{s} \{-1, 1\}^{m \times m}$ // $\boldsymbol{R}$ is a low-norm matrix

  2 : $c' \leftarrow (\boldsymbol{R} \cdot \boldsymbol{A},\ \boldsymbol{R} \cdot \boldsymbol{U}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^{m \times \ell}$

  3 : **return** c'

- *Test*(pp, c, k): parse $c = (\boldsymbol{A}, \boldsymbol{U})$ and do

  1 : $\boldsymbol{V} \leftarrow \boldsymbol{A} \cdot \mathsf{H}(k) - \boldsymbol{U} \in \mathbb{Z}_q^{m \times \ell}$

  2 : **return** 1 iff for each column $\boldsymbol{v}$ of $\boldsymbol{V}$, $\|\boldsymbol{v}\|_2 \leq \Delta$ when $\boldsymbol{v}$ is lifted to $\mathbb{Z}^m$
  Otherwise, **return** 0

Figure 2: $\mathsf{R}_{\mathsf{lwe}}$ – A B-randomizable RRC scheme based on the learning with errors (LWE) problem

Correctness. First, note that prior to any randomization being preformed, for an honestly-generated commitment $c = (\boldsymbol{A}, \boldsymbol{U})$ it holds that $\boldsymbol{A} \cdot \mathsf{H}(k) - \boldsymbol{U}$ is equal to the noise matrix $\boldsymbol{E}$ sampled according to $\chi^{m \times \ell}$ during the generation of the commitment. Hence, the matrix computed by the $Test$ algorithm is simply $\boldsymbol{E}$, and each of its columns has norm at most δ. Now, after $t \leq B$ applications of $Randomize$ to c using matrices $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_t$, the commitment we get is of the form

$$(\boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{A}, \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{U}) = (\boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{A}, \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{A} \cdot \mathsf{H}(k) + \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{E}).$$

Hence, the matrix computed by the *Test* algorithm is $\boldsymbol{E}' = \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{E}$. Since $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_t$ are sampled independently from $\{-1, 1\}^{m \times m}$, Lemma 2.3 guarantees that with overwhelming probability, each column of $\boldsymbol{E}'$ has norm at most $(C \cdot \sqrt{m})^t \cdot \delta \leq (C \cdot \sqrt{m})^B \cdot \delta = \Delta$.

4.2 Binding

Theorem 4.1. *The above scheme is computationally binding when* H *is modeled as a random oracle. Concretely, for every adversary* $\mathcal{A}$ *making at most* Q *queries to* H *it holds that*

$$\Pr\left[\begin{array}{c} k \neq k' \text{ AND} \\ Test(pp, c, k) = Test(pp, c, k') = 1 \end{array} : \begin{array}{c} pp \xleftarrow{\$} Setup(1^\lambda) \\ (c, k, k') \xleftarrow{\$} \mathcal{A}(pp) \end{array}\right] \leq Q^2 \cdot q^n \cdot \left(\frac{4\Delta + 1}{q}\right)^\ell$$

The proof of Theorem 4.1 makes use of the following lemma.

Lemma 4.2. *For all positive integers* $n, m, \ell \in \mathbb{Z}$*, prime* q*, and bound* $\beta < q$ *it holds that*

$$\Pr\left[\exists \boldsymbol{A} \in \mathbb{Z}_q^{m \times n} \setminus \{\boldsymbol{0}\} \textit{ s.t. } \forall i \in [\ell],\ \|\boldsymbol{A} \cdot \boldsymbol{v}_i\|_2 \leq \beta\right] \leq q^n \cdot \left(\frac{2\beta + 1}{q}\right)^\ell$$

where the probability is taken over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell \xleftarrow{\$} \mathbb{Z}_q^n$.

Proof. Fix integers n, m and ℓ, a prime q, and a bound $\beta < q$. We first prove that for every non-zero vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ it holds that

$$\Pr\left[\forall i \in [\ell],\ |\boldsymbol{a}^T \cdot \boldsymbol{v}_i| \leq \beta\right] \leq \left(\frac{2\beta + 1}{q}\right)^\ell \tag{5}$$

where the probability is taken over $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell \xleftarrow{\$} \mathbb{Z}_q^n$. Fix a non-zero vector $\boldsymbol{a} \in \mathbb{Z}_q^n$. For every $i \in [\ell]$, the scalar $\boldsymbol{a}^T \cdot \boldsymbol{v}_i$ is uniformly distributed in $\mathbb{Z}_q$. In particular the probability that $\boldsymbol{a}^T \cdot \boldsymbol{v}_i \in \{-\lceil\beta\rceil, \ldots, 0, \ldots, \lfloor\beta\rfloor\}$ is at most $(2\beta + 1)/q$. Eq. (5) then follows from the fact that $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell$ are statistically independent, and hence so are the events $\left\{\left|\boldsymbol{a}^T \cdot \boldsymbol{v}_i\right| \leq \beta\right\}_i$.

Taking a union bound over all vectors $\boldsymbol{a} \in \mathbb{Z}_q^n$, we observe that

$$\Pr\left[\exists \boldsymbol{a} \in \mathbb{Z}_q^n \setminus \{\boldsymbol{0}\} \text{ s.t. } \forall i \in [\ell],\ |\boldsymbol{a}^T \cdot \boldsymbol{v}_i| \leq \beta\right] \leq q^n \cdot \left(\frac{2\beta + 1}{q}\right)^\ell. \tag{6}$$

We now wish to argue that

$$\Pr\left[\exists \boldsymbol{A} \in \mathbb{Z}_q^{m \times n} \setminus \{\boldsymbol{0}\} \text{ s.t. } \forall i \in [\ell],\ \|\boldsymbol{A} \cdot \boldsymbol{v}_i\|_\infty \leq \beta\right] \leq q^n \cdot \left(\frac{2\beta + 1}{q}\right)^\ell. \tag{7}$$

To see why that is, fix a non-zero matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$. Since $\boldsymbol{A}$ is non-zero, it has at least one non-zero row. Let $\boldsymbol{a}^T$ denote the first such row. If for some $i \in [\ell]$ it holds that $|\boldsymbol{a}^T \cdot \boldsymbol{v}_i| > \beta$ then $\|\boldsymbol{A} \cdot \boldsymbol{v}_i\|_\infty > \beta$.

Hence, the event $\{\forall i \in [\ell],\ \|\boldsymbol{A} \cdot \boldsymbol{v}_i\|_\infty \leq \beta\}$ is contained in the the event $\left\{\forall i \in [\ell],\ |\boldsymbol{a}^T \cdot \boldsymbol{v}_i| \leq \beta\right\}$. Since this holds for every $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$, the event $\left\{\exists \boldsymbol{A} \in \mathbb{Z}_q^{m \times n} \text{ s.t. } \forall i \in [\ell],\ \|\boldsymbol{A} \cdot \boldsymbol{v}_i\|_\infty \leq \beta\right\}$ is contained in the event $\left\{\exists \boldsymbol{a} \in \mathbb{Z}_q^n \text{ s.t. } \forall i \in [\ell],\ |\boldsymbol{a}^T \cdot \boldsymbol{v}_i| \leq \beta\right\}$. Hence, Eq. (7) follows from Eq. (6).

Finally, $\|\boldsymbol{x}\|_2 \geq \|\boldsymbol{x}\|_\infty$ for all $\boldsymbol{x}$, Eq. (7) implies in particular that

$$\Pr\left[\exists \boldsymbol{A} \in \mathbb{Z}_q^{m \times n} \text{ s.t. } \forall i \in [\ell],\ \|\boldsymbol{A} \cdot \boldsymbol{v}_i\|_2 \leq \beta\right] \leq q^n \cdot \left(\frac{2\beta+1}{q}\right)^\ell. \tag{8}$$

This concludes the proof of the lemma. □

We can now prove Theorem 4.1.

Proof of Theorem 4.1. Let $\mathcal{A}$ be an adversary making at most Q queries to the random oracle. Assume without loss of generality that all $\mathcal{A}$'s queries to H are distinct and that before outputting a triple of the form (c, k, k'), $\mathcal{A}$ queries the oracle on k and k'. Fix a pair $(k, k') \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ such that $k \neq k'$. We wish to bound the probability that there exists a $c = (\boldsymbol{A}, \boldsymbol{U})$ such that $Test(pp, c, k) = Test(pp, c, k') = 1$. Recall that this event is defined as all columns of $\boldsymbol{A} \cdot \mathsf{H}(k) - \boldsymbol{U}$ and of $\boldsymbol{A} \cdot \mathsf{H}(k') - \boldsymbol{U}$ having ℓ_2-norm at most Δ. By the triangle inequality, this implies that all columns of $\boldsymbol{A} \cdot (\mathsf{H}(k) - \mathsf{H}(k')) = \boldsymbol{A} \cdot \mathsf{H}(k) - \boldsymbol{A} \cdot \mathsf{H}(k)$ have norm at most 2Δ.

For every $k, k' \in \{0,1\}^\lambda$, consider the matrix $\boldsymbol{V}_{k,k'} = \mathsf{H}(k) - \mathsf{H}(k')$. Since H is modeled as a random oracle, $\boldsymbol{V}_{k,k'}$ is uniformly distributed in $\mathbb{Z}_q^{n \times \ell}$. Hence, by Lemma 4.2, the probability that there exists a non-zero matrix $\boldsymbol{A}$ such that the columns of $\boldsymbol{A} \cdot \boldsymbol{V}_{k,k'}$ all have ℓ_2-norm at most 2Δ is bounded by $q^n \cdot \left(\frac{4\Delta+1}{q}\right)^\ell$. Taking a union bound over all pairs of queries (k, k') made by $\mathcal{A}$ to H we obtain that the probability that there is a pair k, k' of distinct queries to H for which there exists a non-zero matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m \times n}$ such that the columns of $\boldsymbol{A} \cdot \boldsymbol{V}_{k,k'}$ are all shorter than 2Δ, is bounded by $Q^2 \cdot q^n \cdot \left(\frac{4\Delta+1}{q}\right)^\ell$. In particular, this implies that

$$\Pr\left[\exists k, k' \in S_{pp} \text{ s.t. } \begin{array}{c} k \neq k' \text{ AND} \\ \exists c = (\boldsymbol{A}, \boldsymbol{U}) \text{ s.t. } Test(pp, c, k) = Test(pp, c, k') = 1 \end{array}\right] \leq Q^2 \cdot q^n \cdot \left(\frac{4\Delta+1}{q}\right)^\ell$$

where S_{pp} is the random variable corresponding to the set of all $\mathcal{A}$'s queries to H on input pp, and the probability is taken over $pp \leftarrow Setup(1^\lambda)$, the random coins of $\mathcal{A}$, and the random choice of H.

Since we assumed that $\mathcal{A}$ queries H on the k, k' that it outputs, this concludes the proof of Theorem 4.1. □

4.3 Unlinkability

Theorem 4.3. *The above construction is unlinkable, assuming the learning with errors assumption. In particular, for every* PPT *adversary $\mathcal{A}$ making at most $Q = Q(\lambda)$ queries to* H*, there exists a* PPT *adversary $\mathcal{B}$ such that for all $\lambda \in \mathbb{N}$ it holds that:*

$$\mathsf{Adv}^{\mathrm{rrc}}_{\mathcal{A},\mathsf{R}_{\mathsf{LWE}},B}(\lambda) \leq \frac{2Q}{2^\lambda - Q} + 2\ell \cdot \mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}}(\lambda) + \frac{B \cdot m}{2} \cdot \sqrt{2^{-m+(n+\ell)\cdot \log q}}.$$

Proof. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm, and let $\mathbf{G}_0$ denote the original security game from Fig. 1. Consider a modified game $\mathbf{G}_1$ obtained from $\mathbf{G}_0$ as follows. For each $i \in \{0,1\}$, instead of setting c_i as in $\mathbf{G}_0$ (that is, $c_i \leftarrow (\boldsymbol{A}_i, \boldsymbol{A}_i \cdot \mathsf{H}(k_i) + \boldsymbol{E}_i)$), the challenger sets $c_i \leftarrow (\boldsymbol{A}_i, \boldsymbol{U}_i) = (\boldsymbol{A}_i, \boldsymbol{A}_i \cdot \boldsymbol{V}_i +$

$\boldsymbol{E}_i$) where $\boldsymbol{V}_i$ is sampled uniformly at random from $\mathbb{Z}_q^{n\times\ell}$ independently of k_i. For a game $\mathbf{G}$, we denote by $\mathbf{G}_{\mathcal{A}}(\lambda)$ the random variable corresponding to the output of $\mathbf{G}$ ran with $\mathcal{A}$ on security parameter 1^λ. With this notation, the following claim shows that moving from $\mathbf{G}_0$ to $\mathbf{G}_1$ should not noticeably increase the winning chances of $\mathcal{A}$.

Claim 4.4. *For all $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[\mathbf{G}_{0,\mathcal{A}}(\lambda)=1\right] \leq \Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\right] + \frac{2Q}{2^\lambda - Q}.$$

Proof. For $j \in \{0,1\}$ let bad_j denote the event in which the adversary $\mathcal{A}$ queries H on either k_0 or k_1 in $\mathbf{G}_j$. Note that the view of $\mathcal{A}$ in $\mathbf{G}_0$ conditioned on the complementing event $\overline{\mathsf{bad}_0}$ is distributed identically to $\mathcal{A}$'s view in $\mathbf{G}_1$ conditioned on $\overline{\mathsf{bad}_1}$. Hence, $\Pr[\mathsf{bad}_0] = \Pr[\mathsf{bad}_1]$ (where the probability on the left-hand side is over the random choices of $\mathcal{A}$ and the challenger in $\mathbf{G}_0$, and the probability on the right-hand side is over the random choices of $\mathcal{A}$ and the challenger in $\mathbf{G}_1$). Moreover,

$$\begin{aligned}
\Pr\left[\mathbf{G}_{0,\mathcal{A}}(\lambda)=1\right] &\leq \Pr\left[\mathbf{G}_{0,\mathcal{A}}(\lambda)=1 \middle| \overline{\mathsf{bad}_0}\right] + \Pr\left[\mathsf{bad}_0\right] \\
&= \Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1 \middle| \overline{\mathsf{bad}_1}\right] + \Pr\left[\mathsf{bad}_1\right] \\
&= \Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\right] + \Pr\left[\mathsf{bad}_1\right] \qquad (9)\\
&\leq \Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\right] + \frac{2Q}{2^\lambda - Q}. \qquad (10)
\end{aligned}$$

Eq. (9) above follows from the fact that the view of $\mathcal{A}$ in $\mathbf{G}_1$ is independent of the choice k_0, k_1. Hence, the event in which $\mathbf{G}_{1,\mathcal{A}}(\lambda)=1$ is independent of the event bad_1. Eq. 10 follows from the fact that as long as $\mathcal{A}$ has not queried k_0 or k_1, the two values are uniformly distributed in $\{0,1\}^\lambda$ conditioned on $\mathcal{A}$'s view. □

Now consider the game $\mathbf{G}_2$ which is obtained from $\mathbf{G}_1$ by the following modification. For each $i \in \{0,1\}$, the challenger now sets $c_i \leftarrow (\boldsymbol{A}_i, \boldsymbol{U}_i)$ where $\boldsymbol{U}_i$ is sampled uniformly at random from $\mathbb{Z}_q^{m\times\ell}$ (instead of being computed as $\boldsymbol{U}_i \leftarrow \boldsymbol{A}_i \cdot \boldsymbol{V}_i + \boldsymbol{E}_i$ as in $\mathbf{G}_1$).

Claim 4.5. *There exists a PPT adversary $\mathcal{B}$ such that for all $\lambda \in \mathbb{N}$ it holds that*

$$\Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\right] \leq \Pr\left[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1\right] + 2\ell \cdot \mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}}(\lambda).$$

Proof. The proof is by a hybrid argument. Consider a sequence of games $\mathbf{G}_{1,i}$ for $i = 0,\ldots,\ell$. For each $i \in \{0,\ldots,\ell\}$, the game $\mathbf{G}_{1,i}$ is obtained from $\mathbf{G}_1$ by sampling the first $\ell - i$ columns of $\boldsymbol{U}_0$ according to $\boldsymbol{U}_0 \leftarrow \boldsymbol{A}_0 \cdot \boldsymbol{V}_0 + \boldsymbol{E}_0$, but sampling the remaining i columns of $\boldsymbol{U}_0$ uniformly at random.

Consider two consecutive games $\mathbf{G}_{1,i}$ and $\mathbf{G}_{1,i+1}$. We claim that there exists a probabilistic polynomial-time algorithm $\mathcal{B}_i$ such that $\Pr[\mathbf{G}_{1,i,\mathcal{A}}(\lambda)=1] \leq \Pr[\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)=1] + \mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}_i}(\lambda)$. The algorithm $\mathcal{B}_i$ receives an LWE instance $(\boldsymbol{A}, \boldsymbol{u})$ and decides on its output as follows. It invokes $\mathcal{A}$ and simulates to it either $\mathbf{G}_{1,i,\mathcal{A}}(\lambda)$ or $\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)$. It uses $\boldsymbol{A}$ to generate to commitment c_0: It samples $\boldsymbol{V} \xleftarrow{\$} \mathbb{Z}_q^{n\times(\ell-i)}$ and $\boldsymbol{E} \xleftarrow{\$} \chi^{m\times(\ell-i)}$, and computes $\boldsymbol{U}_0' \leftarrow \boldsymbol{A}\cdot\boldsymbol{V} + \boldsymbol{E}$; the result will serve as the left $\ell - i$ columns of $\boldsymbol{U}_0$. It then samples $\boldsymbol{U}_0'' \xleftarrow{\$} \mathbb{Z}_q^{m\times(i-1)}$; the result will serve as the right $i-1$ columns of $\boldsymbol{U}_0$. Finally, it sets $\boldsymbol{U}_0 \leftarrow [\boldsymbol{U}_0', \boldsymbol{u}, \boldsymbol{U}_0'']$ and $c_0 \leftarrow (\boldsymbol{A}, \boldsymbol{U}_0)$. Then, $\mathcal{B}_i$ samples c_1 honestly, and passes the LWE parameters and the commitments c_0 and c_1 to $\mathcal{A}$. When $\mathcal{A}$ outputs integers i_0 and i_1, $\mathcal{B}$ samples a random bit $b \xleftarrow{\$} \{0,1\}$ and computes the randomized commitment c by iteratively applying the $Randomize$ algorithm i_b times to c_b. Finally, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}_i$ outputs 1 iff $b' = b$.

Observe that if $\boldsymbol{u}$ given as input to $\mathcal{B}_i$ is a noisy product of the form $\boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$, then $\mathcal{B}_i$ perfectly simulates $\mathbf{G}_{1,i}$ to $\mathcal{A}$. If $\boldsymbol{u}$ is a truly random vector, then $\mathcal{B}_i$ perfectly simulates $\mathbf{G}_{1,i+1}$ to $\mathcal{A}$. Hence,

$$\begin{aligned}\Pr\left[\mathbf{G}_{1,i,\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)=1\right] &= \Pr\left[\mathcal{B}_i(\boldsymbol{A},\boldsymbol{A}\cdot\boldsymbol{s}+\boldsymbol{e})\right]-\Pr\left[\mathcal{B}_i(\boldsymbol{A},\boldsymbol{u})\right]\\ &\leq \left|\Pr\left[\mathcal{B}_i(\boldsymbol{A},\boldsymbol{A}\cdot\boldsymbol{s}+\boldsymbol{e})\right]-\Pr\left[\mathcal{B}_i(\boldsymbol{A},\boldsymbol{u})\right]\right|\\ &= \mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}_i}(\lambda).\end{aligned}$$

Now consider a sequence of games $\mathbf{G}_{1,i}$ for $i=\ell+1,\ldots,2\ell$. For each $i\in\{\ell+1,\ldots,2\ell\}$, the game $\mathbf{G}_{1,i}$ is obtained from $\mathbf{G}_{1,\ell}$ by sampling the first $2\ell-i$ columns of $\boldsymbol{U}_1$ according to $\boldsymbol{U}_1 \leftarrow \boldsymbol{A}_1\cdot\boldsymbol{V}_1+\boldsymbol{E}_1$, but sampling the remaining $i-\ell$ columns of $\boldsymbol{U}_1$ uniformly at random. A similar argument to the one above shows that for every two consecutive games $\mathbf{G}_{1,i}$ and $\mathbf{G}_{1,i+1}$ (for $i\in\{\ell+1,\ldots,2\ell-1\}$) there exists an adversary $\mathcal{B}_i$ such that $\Pr\left[\mathbf{G}_{1,i,\mathcal{A}}(\lambda)=1\right]\leq\Pr\left[\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)=1\right]+\mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}_i}(\lambda)$.

Observe that $\mathbf{G}_{1,0}=\mathbf{G}_1$ and $\mathbf{G}_{1,2\ell}=\mathbf{G}_2$. Therefore, we have that

$$\Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1\right]\leq\sum_{i=0}^{2\ell-1}\mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}_i}(\lambda). \tag{11}$$

Let $\mathcal{B}$ be the LWE algorithm that samples a random $i^*\xleftarrow{\$}\{0,\ldots,2\ell-1\}$ and invokes $\mathcal{B}_{i^*}$ to decide on its output. For this algorithm it holds that

$$\Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\right]\leq\Pr\left[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1\right]+2\ell\cdot\mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}}(\lambda).$$

This concludes the proof of the claim. □

Finally, consider a final game $\mathbf{G}_3$ obtained from $\mathbf{G}_2$ by sampling the randomized commitment c that is returned to the adversary in line 9 as a pair of uniformly random matrices. That is, $c\leftarrow(\boldsymbol{A}',\boldsymbol{U}')$ where $\boldsymbol{A}'\leftarrow\$\,\mathbb{Z}_q^{m\times n}$ and $\boldsymbol{U}'\leftarrow\$\,\mathbb{Z}_q^{m\times\ell}$.

Claim 4.6. *For all $\lambda\in\mathbb{N}$ it holds that*

$$\Pr\left[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1\right]\leq\Pr\left[\mathbf{G}_{3,\mathcal{A}}(\lambda)=1\right]+B\cdot\frac{m}{2}\cdot\sqrt{2^{-m+(n+\ell)\cdot\log q}}.$$

Proof. Consider a sequence of games $\mathbf{G}_{2,i}$ for $i=0,\ldots,B$. Each $\mathbf{G}_{2,i}$ is obtained from $\mathbf{G}_2$ by replacing the output of the first j calls to $Randomize$ with a pair of uniformly random matrices $(\boldsymbol{A}_b,\boldsymbol{U}_b)\xleftarrow{\$}\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^{m\times\ell}$. By Lemma 2.1, it holds that $\Pr\left[\mathbf{G}_{2,i,\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathbf{G}_{2,i+1,\mathcal{A}}(\lambda)=1\right]\leq\frac{m}{2}\cdot\sqrt{2^{-m+(n+\ell)\cdot\log q}}$. Noting that $\mathbf{G}_{2,0}$ and $\mathbf{G}_{2,B}=\mathbf{G}_3$, and summing over all calls to $Randomize$, we have that

$$\begin{aligned}\Pr\left[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathbf{G}_{3,\mathcal{A}}(\lambda)=1\right] &= \sum_{i=0}^{B-1}\Pr\left[\mathbf{G}_{2,i,\mathcal{A}}(\lambda)=1\right]-\Pr\left[\mathbf{G}_{2,i+1,\mathcal{A}}(\lambda)=1\right]\\ &\leq B\cdot\frac{m}{2}\cdot\sqrt{2^{-m+(n+\ell)\cdot\log q}},\end{aligned}$$

concluding the proof of the claim. □

The theorem is then proven by combining all the claims and noting that the final randomized commitment c in $\mathbf{G}_3$ is independent of the bit b. Hence, the view of $\mathcal{A}$ is independent of b and therefore $\Pr\left[\mathbf{G}_{3,\mathcal{A}}(\lambda)\right]=1/2$.

□

5 A Construction from Ring LWE

In this section we present our RRC construction from the Ring LWE assumption [50] (see Section 2). Our construction, denoted $\mathsf{R}_{\mathsf{rlwe}}$ is presented in Fig. 3. The construction works in a polynomial ring $\mathcal{R}$ modulo a cyclotomic polynomial f that has exactly two irreducible factors f_1, f_2 over $\mathbb{Z}_q$.

Improvements over $\mathsf{R}_{\mathsf{LWE}}$. Compared to the integer-based scheme, the ring-based scheme accommodates more efficient parameter choices. For concreteness, the ensuing discussion focuses on the regime in which $q = \Omega(\Delta^2)$. In this regime, for the ring-based scheme to be binding, we only need ℓ to be $\Omega(\log(q) + \lambda/n)$, where λ is the security parameter. This is a factor of $\Omega(n)$ smaller than the LWE case. Secondly, m only needs to be of order $\Omega(\log(q) + (\ell + \kappa)/n)$ where κ is a statistical security parameter (we want the re-randomized commitments to be distributed $1/2^\kappa$ close to a uniform distribution). This also turns out to be a factor of $\Omega(n)$ smaller than the integer case. Combining these together, each ring-based RRC commitment and each re-randomization matrix is $\Omega(n)$-times smaller than the integer-based commitment and matrix, respectively (this already takes into account the fact that representing each ring element takes n-times the representation length of a $\mathbb{Z}_q$ element).

We now prove the correctness, binding, and unlinkability for our ring-based RRC scheme.

Correctness. We first note that, prior to any rerandomization, for an honestly generated commitment $c = (\boldsymbol{a}, \boldsymbol{U})$, it holds that $\boldsymbol{U} - \boldsymbol{a} \cdot \mathsf{H}(k)^T$ is equal to the noise matrix $\boldsymbol{E}$ sampled at the generation of the commitments. Hence, the matrix $\boldsymbol{V}$ computed by the *Test* algorithm is just $\boldsymbol{E}$, and each of its columns has norm at most δ, since $\boldsymbol{E}$ was sampled according to $\chi^{m \times \ell}$. Now, after $t \leq B$ applications of *Randomize* to the commitment c using matrices $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_t$, the commitment we get is of the form

$$(\boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{a}, \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{U}) = (\boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{a}, \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{a} \cdot \mathsf{H}(k)^T + \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{E}).$$

Hence, the matrix computed by the $Test$ algorithm is $\boldsymbol{E}' = \boldsymbol{R}_t \cdots \boldsymbol{R}_1 \cdot \boldsymbol{E}$. Lemma 2.4 guarantees that with a high probability, each column of $\boldsymbol{E}'$ has norm at most

$$(m\sqrt{mn} \cdot \omega(\sqrt{\log n}))^t \cdot \delta \leq (m\sqrt{mn} \cdot \omega(\sqrt{\log n}))^B \cdot \delta = \Delta$$

where Δ is an upper bound on the expression $\delta \cdot (m\sqrt{mn} \cdot \omega(\sqrt{\log n}))^B$.

5.1 Binding

Theorem 5.1. *The above scheme is computationally binding when* H *is modeled as a random oracle. Concretely, for every adversary* $\mathcal{A}$ *making at most* Q *queries to* H *it holds that*

$$\Pr\left[\begin{array}{c} k \neq k' \text{ AND} \\ Test(pp, c, k) = Test(pp, c, k') = 1 \end{array} : \begin{array}{c} pp \xleftarrow{\$} Setup(1^\lambda) \\ (c, k, k') \xleftarrow{\$} \mathcal{A}(pp) \end{array}\right] \leq Q^2 \cdot q^n \cdot \left(\frac{4\Delta + 1}{\sqrt{q}}\right)^{n\ell}$$

The proof of Theorem 5.1 makes use of the following lemma.

Lemma 5.2. *For every integer* $n, \ell, m \in \mathbb{Z}$*, prime* q*, and bound* $\beta < q$ *it holds that*

$$\Pr\left[\exists \boldsymbol{a} \in \mathcal{R}_q^m \setminus \{\boldsymbol{0}\} \text{ s.t. } \forall i \in [\ell],\ \|\boldsymbol{a} \cdot \boldsymbol{v}_i\|_2 \leq \beta\right] \leq \frac{q^n \cdot (2\beta + 1)^{n\ell}}{q^{n\ell/2}}$$

where the probability is taken over $\boldsymbol{v} \xleftarrow{\$} \mathcal{R}_q^\ell$.

- *Setup*(1^λ):

  1 : Let $n := 2^r$ where $r = r(\lambda)$, and let $f(x) = x^n + 1$ and $\mathcal{R} = \mathbb{Z}[x]/f(x)$.
  Choose a prime q and let $\mathcal{R}_q = \mathbb{Z}_q[x]/f(x)$.
  // r, q are chosen such that f factors into two irreducible polynomials over $\mathbb{Z}_q$

  2 : Let $m = m(n, q)$ and $\ell = \ell(n, q)$.
  // we will explain how to choose m and ℓ in the analysis

  3 : Let χ be the RLWE noise distribution over $\mathcal{R}_q$.
  // if we sample a vector $\boldsymbol{e} \xleftarrow{s} \chi^m$, then with high probability, $\|\boldsymbol{e}\|_2 \leq \delta$ for some $\delta \ll q$

  4 : **return** $pp \leftarrow (\lambda, q, n, m, \ell, \chi)$

- *Commit*(pp):

  1 : $\boldsymbol{a} \xleftarrow{s} \mathcal{R}_q^m$ // choose a random vector $\boldsymbol{a}$

  2 : $k \xleftarrow{s} \{0,1\}^\lambda$ // choose a random 1^λ-bit string

  3 : $\boldsymbol{v} \leftarrow \mathsf{H}(k) \in \mathcal{R}_q^\ell$ // hash k to a vector of length ℓ

  4 : Sample $\boldsymbol{E} \in \mathcal{R}_q^{m \times \ell}$ from the RLWE noise distribution $\chi^{m \times \ell}$
  // then for each column $\boldsymbol{e}$ of $\boldsymbol{E}$, $\|\boldsymbol{e}\|_2 \leq \delta$ w.h.p

  5 : $\boldsymbol{U} \leftarrow \boldsymbol{a} \cdot \boldsymbol{v}^T + \boldsymbol{E} \in \mathcal{R}_q^{m \times \ell}$

  6 : $c \leftarrow (\boldsymbol{a}, \boldsymbol{U})$

  7 : **return** (c, k)

- *Randomize*(pp, c): parse $c = (\boldsymbol{a}, \boldsymbol{U})$ and do

  1 : Sample $\boldsymbol{R} \in \mathcal{R}^{m \times m}$:
  $\forall i, j \in [m]$, sample the coefficients of $\boldsymbol{R}_{i,j}$ uniformly and independently from $\{-1, 1\}$
  // $\boldsymbol{R}$ is a low-norm matrix

  2 : $c' \leftarrow (\boldsymbol{R} \cdot \boldsymbol{a},\ \boldsymbol{R} \cdot \boldsymbol{U}) \in \mathcal{R}_q^m \times \mathcal{R}_q^{m \times \ell}$

  3 : **return** c'

- *Test*(pp, c, k): parse $c = (\boldsymbol{a}, \boldsymbol{U})$ and do

  1 : $\boldsymbol{V} \leftarrow \boldsymbol{U} - \boldsymbol{a} \cdot \mathsf{H}(k)^T \in \mathcal{R}_q^{m \times \ell}$

  2 : **return** 1 iff for each column $\boldsymbol{v}$ of $\boldsymbol{V}$, $\|\boldsymbol{v}\|_2 \leq \Delta$, and **return** 0 otherwise

Figure 3: $\mathsf{R}_{\mathsf{rlwe}}$ – A B-randomizable RRC scheme based on the learning with errors over rings (RLWE) problem

Proof. Fix integers n, ℓ and a bound $\beta < q$. Let f_1, f_2 denote the two irreducible polynomials of degree $n/2$ such that $f = f_1 \cdot f_2$. Then, as stated in [21], for any non-zero $a \in \mathcal{R}_q$, either $a \neq 0 \bmod f_1(x)$ or $a \neq 0 \bmod f_2(x)$. Without loss of generality, let us assume that $a \neq 0 \bmod f_1(x)$.

Claim 5.3. *For any non-zero $a \in \mathcal{R}_q$ such that $a \neq 0$ mod f_1, and any $\boldsymbol{v}_i$ that is chosen uniformly randomly from $\mathcal{R}_q$, $a\boldsymbol{v}_i \mod f_1$ is uniformly random in $\mathbb{Z}_q[x]/f_1$.*

Proof. For any $b \in \mathbb{Z}_q[x]/f_1(x)$, let $A_b = \{v \in \mathcal{R}_q : av = b \bmod f_1\}$. Note that the probability that $a\boldsymbol{v}_i = b \bmod f_1$ (over the random choice of $\boldsymbol{v}_i$) equals $|A_b|/q^n$.

Let $b = av \bmod f_1$ for an arbitrary v. We claim that $|A_b| = |A_0|$. Observe that $v' \in A_b$ if and only if $v' - v \in A_0$. Since $v' \to v' - v$ is a bijection between A_b and A_0, it follows that $|A_b| = |A_0|$. This proves that all $b \in \mathbb{Z}_q[x]/f_1$ have the same probability, thus proving the claim. □

Next, consider an arbitrary $e \in \mathcal{R}_q$. Then,

$$\begin{aligned} \Pr\left[av = e : v \xleftarrow{\$} \mathcal{R}_q\right] &= \Pr\left[(av = e \bmod f_1 \wedge av = e \bmod f_2) : v \xleftarrow{\$} \mathcal{R}_q\right] \\ &\leq \Pr\left[av = e \bmod f_1 : v \xleftarrow{\$} \mathcal{R}_q\right] \\ &= \frac{1}{q^{n/2}} \end{aligned} \tag{12}$$

Here, the first equality follows from the Chinese remainder theorem, and Equation 12 follows from Claim 5.3.

This directly implies that for any arbitrary $e_1, \ldots e_\ell \in \mathcal{R}_q$, the following holds:

$$\Pr\left[a\boldsymbol{v}_i = e_i \forall i \in [\ell] : \boldsymbol{v} \xleftarrow{\$} \mathcal{R}_q^\ell\right] \leq \left(\frac{1}{q^{n/2}}\right)^\ell$$

Next, taking a union bound over all possible values of $a \in \mathcal{R}_q$, and all $e_i \in \mathcal{R}_q$ such that $||e_i||_\infty \leq \beta \, \forall \, i \in [\ell]$, we get :

$$\Pr\left[\exists a \in \mathcal{R}_q \setminus 0 \text{ s.t. } \forall i \in [\ell], ||a \cdot \boldsymbol{v}_i||_\infty \leq \beta : \boldsymbol{v} \xleftarrow{\$} \mathcal{R}_q^\ell\right] \leq \frac{q^n \cdot (2\beta+1)^{n\ell}}{q^{n\ell/2}} \tag{13}$$

We now wish to argue that

$$\Pr\left[\exists \boldsymbol{a} \in \mathcal{R}_q^m \setminus \{\boldsymbol{0}\} \text{ s.t. } \forall i \in [\ell], \ \|\boldsymbol{a} \cdot \boldsymbol{v}_i\|_\infty \leq \beta\right] \leq \frac{q^n \cdot (2\beta+1)^{n\ell}}{q^{n\ell/2}}. \tag{14}$$

To see that, fix a non-zero vector $\boldsymbol{a} \in \mathcal{R}_q^m$. Since $\boldsymbol{a}$ is non-zero, at least one of its elements is non-zero. Let a denote the first such element. If for some $i \in [\ell]$ it holds that $|a \cdot \boldsymbol{v}_i| > \beta$ then $||\boldsymbol{a} \cdot \boldsymbol{v}_i||_\infty > \beta$. Hence, the event $\{\forall i \in [\ell], ||\boldsymbol{a} \cdot \boldsymbol{v}_i||_\infty \leq \beta\}$ is contained in the event $\{\forall i \in [\ell], |a \cdot \boldsymbol{v}_i| \leq \beta\}$. Since this holds for every $\boldsymbol{a} \in \mathcal{R}_q^m$, the event $\left\{\exists \boldsymbol{a} \in \mathcal{R}_q^m \text{ s.t. } \forall i \in [\ell], \ \|\boldsymbol{a} \cdot \boldsymbol{v}_i\|_\infty \leq \beta\right\}$ is contained in the event $\{\exists a \in \mathcal{R}_q \text{ s.t. } \forall i \in [\ell], \ |a \cdot \boldsymbol{v}_i| \leq \beta\}$. Hence, Eq. (14) follows from Eq. (13).

Finally, $\|\boldsymbol{x}\|_2 \geq \|\boldsymbol{x}\|_\infty$ for all $\boldsymbol{x}$, Eq. (14) implies in particular that

$$\Pr\left[\exists \boldsymbol{a} \in \mathcal{R}_q^m \text{ s.t. } \forall i \in [\ell], \ \|\boldsymbol{a} \cdot \boldsymbol{v}_i\|_2 \leq \beta\right] \leq \frac{q^n \cdot (2\beta+1)^{n\ell}}{q^{n\ell/2}}. \tag{15}$$

This concludes the proof of the lemma.

□

Proof of Theorem 5.1. Let $\mathcal{A}$ be an adversary making at most Q queries to the random oracle. Assume without loss of generality that all $\mathcal{A}$'s queries to H are distinct and that before outputting a triple of the form (c, k, k'), $\mathcal{A}$ queries the oracle on k and k'. Fix a pair $(k, k') \in \{0,1\}^\lambda \times \{0,1\}^\lambda$ such that $k \neq k'$. We wish to bound the probability that there exists a $c = (\boldsymbol{a}, \boldsymbol{U})$ such that $Test(pp, c, k) = Test(pp, c, k') = 1$. Recall that this event is defined as all columns of $\boldsymbol{a} \cdot \mathsf{H}(k)^T - \boldsymbol{U}$ and of $\boldsymbol{a} \cdot \mathsf{H}(k')^T - \boldsymbol{U}$ having ℓ_2-norm at most Δ. By the triangle inequality, this implies that all columns of $\boldsymbol{a} \cdot (\mathsf{H}(k) - \mathsf{H}(k'))^T = \boldsymbol{a} \cdot \mathsf{H}(k)^T - \boldsymbol{a} \cdot \mathsf{H}(k)^T$ have norm at most 2Δ.

For every $k, k' \in \{0,1\}^\lambda$, consider the matrix $\boldsymbol{v}_{k,k'} = \mathsf{H}(k) - \mathsf{H}(k')$. Since H is modeled as a random oracle, $\boldsymbol{v}_{k,k'}$ is uniformly distributed in $\mathcal{R}_q^\ell$. Hence, by Lemma 5.2, the probability that there exists a non-zero vector $\boldsymbol{a}$ such that the columns of $\boldsymbol{a} \cdot \boldsymbol{v}_{k,k'}$ all have ℓ_2-norm at most 2Δ is bounded by $\frac{q^n \cdot (4\Delta+1)^{n\ell}}{q^{n\ell/2}}$. Taking a union bound over all pairs of queries (k, k') made by $\mathcal{A}$ to H we obtain that the probability that there is a pair k, k' of distinct queries to H for which there exists a non-zero vector $\boldsymbol{a} \in \mathcal{R}_q^m$ such that the columns of $\boldsymbol{a} \cdot \boldsymbol{v}_{k,k'}$ are all shorter than 2Δ, is bounded by $Q^2 \cdot q^n \cdot \left(\frac{4\Delta+1}{\sqrt{q}}\right)^{n\ell}$. In particular, this implies that

$$\Pr\left[\exists k, k' \in S_{pp} \text{ s.t. } \begin{array}{c} k \neq k' \text{ AND} \\ \exists c = (\boldsymbol{a}, \boldsymbol{U}) \text{ s.t. } Test(pp, c, k) = Test(pp, c, k') = 1 \end{array}\right] \leq Q^2 \cdot q^n \cdot \left(\frac{4\Delta+1}{\sqrt{q}}\right)^{n\ell}$$

where S_{pp} is the random variable corresponding to the set of all $\mathcal{A}$'s queries to H on input pp, and the probability is taken over $pp \leftarrow Setup(1^\lambda)$, the random coins of $\mathcal{A}$, and the random choice of H.

Since we assumed that $\mathcal{A}$ queries H on the k, k' that it outputs, this concludes the proof of Theorem 5.1. □

5.2 Unlinkability

Theorem 5.4. *The above construction is unlinkable, assuming the ring learning with errors assumption. In particular, for every* PPT *adversary* $\mathcal{A}$ *making at most* $Q = Q(\lambda)$ *queries to* H*, there exists a* PPT *adversary* $\mathcal{B}$ *such that for all* $\lambda \in \mathbb{N}$ *it holds that:*

$$\mathsf{Adv}^{\mathrm{rrc}}_{\mathcal{A},\mathsf{R}_{\mathsf{RLWE}},B}(\lambda) \leq \frac{2Q}{2^\lambda - Q} + (2\ell) \cdot \mathsf{Adv}^{\mathrm{rlwe}}_{\mathcal{B}}(\lambda) + B \cdot \frac{m}{2}\sqrt{\left(1 + \left(\frac{q}{2^m}\right)^{n/2}\right)^{2(\ell+1)} - 1}.$$

Proof. Let $\mathcal{A}$ be a probabilistic polynomial-time algorithm, and let $\mathbf{G}_0$ denote the original security game from Fig. 1. Consider a modified game $\mathbf{G}_1$ obtained from $\mathbf{G}_0$ as follows. For each $i \in \{0,1\}$, instead of setting c_i as in $\mathbf{G}_0$ (that is, $c_i \leftarrow (\boldsymbol{a}_i, \boldsymbol{a}_i \cdot \mathsf{H}(k_i)^T + \boldsymbol{E}_i)$), the challenger sets $c_i \leftarrow (\boldsymbol{a}_i, \boldsymbol{U}_i) = (\boldsymbol{a}_i, \boldsymbol{a}_i \cdot \boldsymbol{v}_i^T + \boldsymbol{E}_i)$ where $\boldsymbol{v}_i$ is sampled uniformly at random from $\mathcal{R}_q^\ell$ independently of k_i. For a game $\mathbf{G}$, we denote by $\mathbf{G}_{\mathcal{A}}(\lambda)$ the random variable corresponding to the output of $\mathbf{G}$ ran with $\mathcal{A}$ on security parameter 1^λ. With this notation, the following claim shows that moving from $\mathbf{G}_0$ to $\mathbf{G}_1$ should not noticeably increase the winning chances of $\mathcal{A}$.

Claim 5.5. *For all* $\lambda \in \mathbb{N}$ *it holds that*

$$\Pr\left[\mathbf{G}_{0,\mathcal{A}}(\lambda) = 1\right] \leq \Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda) = 1\right] + \frac{2Q}{2^\lambda - Q}.$$

Proof. The proof strategy is similar to the one for the integer case. For $j \in \{0,1\}$ let bad_j denote the event in which the adversary $\mathcal{A}$ queries H on either k_0 or k_1 in $\mathbf{G}_j$. Note that the view of $\mathcal{A}$ in $\mathbf{G}_0$ conditioned

on the complementing event $\overline{\mathsf{bad}_0}$ is distributed identically to $\mathcal{A}$'s view in $\mathbf{G}_1$ conditioned on $\overline{\mathsf{bad}_1}$. Hence, $\Pr\left[\mathbf{G}_{0,\mathcal{A}}(\lambda)=1\middle|\overline{\mathsf{bad}_0}\right]=\Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\middle|\overline{\mathsf{bad}_1}\right]$. Moreover, $\Pr[\mathsf{bad}_0]=\Pr[\mathsf{bad}_1]$ (where the probability on the left hand side is over the random choices of $\mathcal{A}$ and the challenger in $\mathbf{G}_0$, and the probability on the right hand side is over the random choices of $\mathcal{A}$ and the challenger in $\mathbf{G}_1$). We get,

$$\begin{aligned}
\Pr[\mathbf{G}_{0,\mathcal{A}}(\lambda)=1] &\leq \Pr\left[\mathbf{G}_{0,\mathcal{A}}(\lambda)=1\middle|\overline{\mathsf{bad}_0}\right]+\Pr[\mathsf{bad}_0] \\
&= \Pr\left[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1\middle|\overline{\mathsf{bad}_1}\right]+\Pr[\mathsf{bad}_1] \\
&= \Pr[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1]+\Pr[\mathsf{bad}_1] \qquad (16)\\
&\leq \Pr[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1]+\frac{2Q}{2^\lambda-Q}. \qquad (17)
\end{aligned}$$

Eq. (16) above follows from the fact that the view of $\mathcal{A}$ in $\mathbf{G}_1$ is independent of the choice k_0, k_1. Hence, the event in which $\mathbf{G}_{1,\mathcal{A}}(\lambda)=1$ is independent of the event bad_1. Eq. 17 follows from the fact that as long as $\mathcal{A}$ has not queried k_0 or k_1, the two values are uniformly distributed in $\{0,1\}^\lambda$ conditioned on $\mathcal{A}$'s view. □

Now consider the game $\mathbf{G}_2$ which is obtained from $\mathbf{G}_1$ by the following modification. For each $i\in\{0,1\}$, the challenger now sets $c_i\leftarrow(\boldsymbol{a}_i,\boldsymbol{U}_i)$ where $\boldsymbol{U}_i$ is sampled uniformly at random from $\mathcal{R}_q^{m\times\ell}$ (instead of being computed as $\boldsymbol{U}_i\leftarrow\boldsymbol{a}_i\cdot\boldsymbol{v}_i^T+\boldsymbol{E}_i$ as in $\mathbf{G}_1$).

Claim 5.6. *There exists a* PPT *adversary $\mathcal{B}$ such that for all $\lambda\in\mathbb{N}$ it holds that*

$$\Pr[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1]\leq\Pr[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1]+(2\ell)\cdot\mathsf{Adv}^{\mathrm{rlwe}}_{\mathcal{B}}(\lambda).$$

Proof. Consider a sequence of games $\mathbf{G}_{1,i}$ for $i=0,\ldots,\ell$. For each $i\in\{0,\ldots,\ell\}$, the game $\mathbf{G}_{1,i}$ is obtained from $\mathbf{G}_1$ by sampling the first $\ell-i$ columns of $\boldsymbol{U}_0$ according to $\boldsymbol{U}_0\leftarrow\boldsymbol{a}_0\cdot\boldsymbol{v}_0^T+\boldsymbol{E}_0$, but sampling the remaining i columns of $\boldsymbol{U}_0$ uniformly at random.

Consider two consecutive games $\mathbf{G}_{1,i}$ and $\mathbf{G}_{1,i+1}$. We claim that there exists a probabilistic polynomial-time algorithm $\mathcal{B}_i$ such that $\Pr[\mathbf{G}_{1,i,\mathcal{A}}(\lambda)=1]\leq\Pr[\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)=1]+\mathsf{Adv}^{\mathrm{rlwe}}_{\mathcal{B}_i}(\lambda)$. The algorithm $\mathcal{B}_i$ receives an RLWE instance $(\boldsymbol{a},\boldsymbol{u})$ and decides on its output as follows. It invokes $\mathcal{A}$ and simulates to it either $\mathbf{G}_{1,i,\mathcal{A}}(\lambda)$ or $\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)$. It uses $\boldsymbol{a}$ to generate commitment c_0: It samples $\boldsymbol{v}\xleftarrow{\$}\mathcal{R}_q^{\ell-i-1}$ and $\boldsymbol{E}\xleftarrow{\$}\chi^{m\times(\ell-i-1)}$, and computes $\boldsymbol{U}_0'\leftarrow\boldsymbol{a}\cdot\boldsymbol{v}^T+\boldsymbol{E}$; the result will serve as the left $\ell-i-1$ columns of $\boldsymbol{U}_0$. It then samples $\boldsymbol{U}_0''\xleftarrow{\$}\mathcal{R}_q^{m\times i}$; the result will serve as the right $\ell-i-1$ columns of $\boldsymbol{U}_0$. Finally, it sets $\boldsymbol{U}_0\leftarrow[\boldsymbol{U}_0',\boldsymbol{u},\boldsymbol{U}_0'']$ and $c_0\leftarrow(\boldsymbol{a},\boldsymbol{U}_0)$. Then, $\mathcal{B}_i$ samples c_1 honestly, and passes the LWE parameters and the commitments c_0 and c_1 to $\mathcal{A}$. When $\mathcal{A}$ outputs integers i_0 and i_1, $\mathcal{B}$ samples a random bit $b\xleftarrow{\$}\{0,1\}$ and computes the randomized commitment c by iteratively applying the $Randomize$ algorithm i_b times to c_b. Finally, when $\mathcal{A}$ outputs a bit b', $\mathcal{B}_i$ outputs 1 iff $b'=b$.

Observe that if $\boldsymbol{u}$ given as input to $\mathcal{B}_i$ is a noisy product of the form $\{\boldsymbol{a}_j\cdot\boldsymbol{s}+\boldsymbol{e}_j\}$, then $\mathcal{B}_i$ perfectly simulates $\mathbf{G}_{1,i}$ to $\mathcal{A}$. If $\boldsymbol{u}$ is a truly random vector, then $\mathcal{B}_i$ perfectly simulates $\mathbf{G}_{1,i+1}$ to $\mathcal{A}$. Hence,

$$\begin{aligned}
\Pr[\mathbf{G}_{1,i,\mathcal{A}}(\lambda)=1]-\Pr[\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)=1] &= \Pr[\mathcal{B}_i(\boldsymbol{a},\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e})]-\Pr[\mathcal{B}_i(\boldsymbol{a},\boldsymbol{u})] \\
&\leq |\Pr[\mathcal{B}_i(\boldsymbol{a},\boldsymbol{a}\cdot\boldsymbol{s}+\boldsymbol{e})]-\Pr[\mathcal{B}_i(\boldsymbol{a},\boldsymbol{u})]| \\
&= \mathsf{Adv}^{\mathrm{rlwe}}_{\mathcal{B}_i}(\lambda).
\end{aligned}$$

Now consider a sequence of games $\mathbf{G}_{1,i}$ for $i=\ell+1,\ldots,2\ell$. For each $i\in\{\ell+1,\ldots,2\ell\}$, the game $\mathbf{G}_{1,i}$ is obtained from $\mathbf{G}_{1,\ell}$ by sampling the first $2\ell-i$ columns of $\boldsymbol{U}_1$ according to $\boldsymbol{U}_1\leftarrow\boldsymbol{a}_1\cdot\boldsymbol{v}_1^T+\boldsymbol{E}_1$, but sampling the remaining $i-\ell$ columns of $\boldsymbol{U}_1$ uniformly at random. A similar argument to the one above

shows that for every two consecutive games $\mathbf{G}_{1,i}$ and $\mathbf{G}_{1,i+1}$ (for $i \in \{\ell+1,\ldots,2\ell-1\}$) there exists an adversary $\mathcal{B}_i$ such that $\Pr[\mathbf{G}_{1,i,\mathcal{A}}(\lambda)=1] \leq \Pr[\mathbf{G}_{1,i+1,\mathcal{A}}(\lambda)=1] + \mathsf{Adv}^{\mathrm{rlwe}}_{\mathcal{B}_i}(\lambda)$.

Observe that $\mathbf{G}_{1,0} = \mathbf{G}_1$ and $\mathbf{G}_{1,2\ell} = \mathbf{G}_2$. Therefore, we have that

$$\Pr[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1] - \Pr[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1] \leq \sum_{i=0}^{2\ell-1} \mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}_i}(\lambda). \tag{18}$$

Let $\mathcal{B}$ be the RLWE algorithm that samples a random $i^* \xleftarrow{\$} \{0,\ldots,2\ell-1\}$ and invokes $\mathcal{B}_{i^*}$ to decide on its output. For this algorithm it holds that

$$\Pr[\mathbf{G}_{1,\mathcal{A}}(\lambda)=1] \leq \Pr[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1] + 2\ell \cdot \mathsf{Adv}^{\mathrm{lwe}}_{\mathcal{B}}(\lambda).$$

This concludes the proof of the claim. □

Finally, consider a final game $\mathbf{G}_3$ obtained from $\mathbf{G}_2$ by sampling the randomized commitment c that is returned to the adversary in line 9 as a pair of uniformly random vector and matrix. That is, $c \leftarrow (\boldsymbol{a}', \boldsymbol{U}')$ where $\boldsymbol{a}' \leftarrow\!\$\, \mathcal{R}_q^m$ and $\boldsymbol{U}' \leftarrow\!\$\, \mathcal{R}_q^{m\times\ell}$.

Claim 5.7. *For all $\lambda \in \mathbb{N}$ it holds that*

$$\Pr[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1] \leq \Pr[\mathbf{G}_{3,\mathcal{A}}(\lambda)=1] + B \cdot \frac{m}{2}\sqrt{\left(1+\left(\frac{q}{2^m}\right)^{n/2}\right)^{2(\ell+1)} - 1}.$$

Proof. Consider a sequence of games $\mathbf{G}_{2,i}$ for $i = 0,\ldots,B$. Each $\mathbf{G}_{2,i}$ is obtained from $\mathbf{G}_2$ by replacing the output of the first i calls to $Randomize$ with a pair of uniformly random matrices $(\boldsymbol{a}_b, \boldsymbol{U}_b) \xleftarrow{\$} \mathcal{R}_q^m \times \mathcal{R}_q^{m\times\ell}$. By lemma 2.2 and the fact that we are using a polynomial modulus with two factors in $\mathbb{Z}_q$, it holds that $\Pr[\mathbf{G}_{2,i,\mathcal{A}}(\lambda)=1] - \Pr[\mathbf{G}_{2,i+1,\mathcal{A}}(\lambda)=1] \leq \frac{m}{2}\sqrt{\left(1+\left(\frac{q}{2^m}\right)^{n/2}\right)^{2(\ell+1)} - 1}$. Noting that $\mathbf{G}_{2,0} = \mathbf{G}_2$ and $\mathbf{G}_{2,B} = \mathbf{G}_3$, and summing over all calls to $Randomize$, we have that

$$\begin{aligned}\Pr[\mathbf{G}_{2,\mathcal{A}}(\lambda)=1] - \Pr[\mathbf{G}_{3,\mathcal{A}}(\lambda)=1] &= \sum_{i=0}^{B-1} \Pr[\mathbf{G}_{2,i,\mathcal{A}}(\lambda)=1] - \Pr[\mathbf{G}_{2,i+1,\mathcal{A}}(\lambda)=1] \\ &\leq B \cdot \frac{m}{2}\sqrt{\left(1+\left(\frac{q}{2^m}\right)^{n/2}\right)^{2(\ell+1)} - 1},\end{aligned}$$

concluding the proof of the claim. □

The theorem is then proven by combining all the claims and noting that the final randomized commitment c in $\mathbf{G}_3$ is independent of the bit b. Hence, the view of $\mathcal{A}$ is independent of b and therefore $\Pr[\mathbf{G}_{3,\mathcal{A}}(\lambda)] = 1/2$. □

6 Handling Adversarially-Randomized Commitments

In this section, we present a stronger notion of unlinkability, called *strong unlinkability* for RRC schemes, and then present different approaches to augment our basic schemes from Sections 4 and 5 to satisfy this definition.

Loosely speaking, strong unlinkability requires that re-randomization should result in unlinkable commitments, even if they were previously re-randomized by the adversary. This trivially holds for the DDH-based construction of Boneh et al. [24] thanks to two properties of the scheme:

- Suppose the adversary receives a commitment c for which k is a valid key, and outputs a randomized commitment c'. As long as $Test(pp, c', k) = 1$, there exists some randomness r such that $c' = Randomize(pp, c; r)$.
- Re-randomization using $Randomize$ is a commutative operation. Hence, in conjunction with the observation above, any knowledge the adversary could gain by re-randomizing a commitment before an honest re-randomization, it could also gain by re-randomizing it afterwards (which the adversary can already do in the security game from Fig. 1).

Alas, this is not the case for our lattice-based constructions. The main issue is that matrix multiplication is not commutative. Hence a "bad" re-randomization (even one that does not invalidate the honest commitment key) can have a long lasting effect on a commitment even after many subsequent honest re-randomizations have taken place. Concretely, on input $c = (\boldsymbol{A}, \boldsymbol{U})$, the adversary may output $c' = (\boldsymbol{A}', \boldsymbol{U}')$, such that $\boldsymbol{A}'$ is "bad" in the sense that the distribution $R \cdot \boldsymbol{A}'$ for a random $R \xleftarrow{s} \{-1, 1, \}^{m \times m}$ is very far from the uniform distribution over $\mathbb{Z}_q^{m \times n}$. As a hypothetical example, suppose that the adversary can find a matrix $\boldsymbol{R} \in \{-1, 1, \}^{m \times m}$ such that $\boldsymbol{A}' = \boldsymbol{R} \cdot \boldsymbol{A}$ is a low-norm matrix. Then, the distribution $R \cdot \boldsymbol{A}'$ will be concentrated on low-norm matrices as well, enabling the adversary to distinguish between this distribution and the uniform distribution over $\mathbb{Z}_q^{m \times n}$, which is concentrated on high-norm matrices.

6.1 A Stronger Unlinkability Definition

We first need to define what it means for an RRC scheme to be unlinkable in the face of adversarial re-randomizations. To do this, we augment the security game of RRC schemes by letting the adversary re-randomize the commitments at points in time of its choosing. To avoid trivial attacks, we require that the adversary justifies its outputs by providing the randomness it used for re-randomization.

To this end, and to facilitate our constructions, we introduce several new notions for RRC schemes:

- We augment an RRC scheme with a corresponding *beacon distribution* D. This distribution is used to model a randomness beacon, and will be used by one of our constructions of a strongly-unlinkable RRC scheme. In practice, the beacon may be assumed as an outside resource or implemented using known techniques [55].
- We introduce two new algorithms $\mathsf{R}.\mathit{Precommit}$ and $\mathsf{R}.\mathit{Extract}$ to an RRC scheme R. $\mathsf{R}.\mathit{Precommit}$ is a randomized algorithm that takes in the public parameters *pp* and outputs some "precommitment" pcom, whose role will become apparent in a minute. $\mathsf{R}.\mathit{Extract}$ is a (potentially randomized) algorithm that takes in *pp*, the randomness $r \in \{0,1\}^*$ used by $\mathsf{R}.\mathit{Precommit}$ to generate pcom, and a sample rand from D, and outputs some randomness r' to be used by $\mathsf{R}.\mathit{Randomize}$. Throughout this section, we will denote the number of random coins used by $\mathsf{R}.\mathit{Precommit}$ by $\rho = \rho(\lambda)$.
- An RRC scheme R is now also parameterized by a class $\mathcal{G}$ of *admissible random strings* , and only members of $\mathcal{G}$ can be used as randomness for $\mathsf{R}.\mathit{Randomize}$. This is checked by the security game for randomness used by the adversary. A natural selection for $\mathcal{G}$ is the entire support of the randomness used by the honest *Randomize* algorithm; for example, in our (integer) LWE-based construction, this corresponds to $\mathcal{G} = \{-1, 1\}^{m \times m}$, but one might also consider strict supersets or subsets of this set. We allow $\mathcal{G}$ to depend on a precommitment pcom, the randomness $r \in \{0,1\}^*$ used by $\mathsf{R}.\mathit{Precommit}$ to generate pcom, and a sample rand from D. We denote this by $\mathcal{A}(\mathsf{pcom}, r, \mathsf{rand})$. The set $\mathcal{G}$ may also depend on the public parameters *pp*, but we do not note this explicitly, since the public parameters typically remain fixed.

To recap, an RRC scheme R now consists of six algorithms ($\mathsf{R}.\mathit{Setup}$, $\mathsf{R}.\mathit{Commit}$, $\mathsf{R}.\mathit{Randomize}$, $\mathsf{R}.\mathit{Test}$, and now also $\mathsf{R}.\mathit{Precommit}$ and $\mathsf{R}.\mathit{Extract}$), a distribution D, and a set $\mathcal{G} = \mathcal{G}(\mathsf{pcom}, r, \mathsf{rand})$.

Correctness and unlinkability. For correctness, we now require that the scheme is B-rerandomizable (Definition 3.1), where the randomness for rerandomization is generated by *Precommit*, D, and *Extract*. We additionally require that honestly generated randomness for *Randomize* is indeed admissible.

Definition 6.1. *Let R be an RRC scheme such that $\mathsf{R}.\mathit{Precommit}$ takes $\rho = \rho(\lambda)$ random coins. We say R is* **B-randomizable** *if there exists a negligible function $\nu(\cdot)$ such that the following conditions hold for every $\lambda \in \mathbb{N}$:*

1. *Let $pp \xleftarrow{\$} \mathsf{R}.\mathit{Setup}(1^\lambda)$, $(c_0, k) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp)$, $r_i \xleftarrow{\$} \{0,1\}^\rho$, $\mathsf{rand}_i \xleftarrow{\$} D$, $r'_i \xleftarrow{\$} \mathsf{R}.\mathit{Extract}(pp, r_i, \mathsf{rand}_i)$, $c_i \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, c_{i-1}; r'_i)$ for $i \in [B]$, then*
$$\Pr\Big[\mathsf{R}.\mathit{Test}(pp, c_i, k) = 1 \quad \text{for } i = 0, 1, 2, \ldots, B\Big] \geq 1 - \nu(\lambda).$$

2. *Let $pp \xleftarrow{\$} \mathsf{R}.\mathit{Setup}(1^\lambda)$, $r \xleftarrow{\$} \{0,1\}^\rho$, $\mathsf{pcom} \leftarrow \mathsf{R}.\mathit{Precommit}(pp; r)$, $\mathsf{rand} \xleftarrow{\$} D$, and $r' \xleftarrow{\$} \mathsf{R}.\mathit{Extract}(pp, r, \mathsf{rand})$, then*
$$\Pr\left[r' \in \mathcal{G}(\mathsf{pcom}, r, \mathsf{rand})\right] \geq 1 - \nu(\lambda).$$

An RRC scheme that is B-randomizable for all $B \in \mathbb{N}$ is said to be **fully randomizable**. □

The new strong-unlinkability game is defined in Figure 4. It uses the following abbreviated writing: we write $(\mathsf{rand}, c') \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, r, c)$ as a shorthand for the process of (1) sampling $\mathsf{rand} \xleftarrow{\$} D$, (3) sampling $r' \xleftarrow{\$} \mathsf{R}.\mathit{Extract}(pp, r, \mathsf{rand})$, (4) computing $c' \leftarrow \mathsf{R}.\mathit{Randomize}(pp, c; r')$, and (5) outputting (rand, c'). The new game is obtained from the old unlinkability security game (Figure 1) by the following modifications:

1. At the onset of the game, the challenger samples precommitments $\{\mathsf{pcom}\}$ to be used for the honest re-randomizations it performs. The adversary then also outputs a precommitment $\{\mathsf{pcom}\}$ for its own future re-randomizations. For each re-randomization, the set $\mathcal{G}$ will depend on the corresponding precommitment. Looking ahead, in a couple of our constructions, the precommitments will serve as commitments for randomness to be used in the future re-randomizations.

2. The challenger samples $\{\mathsf{rand}\}$ values from the beacon distribution D. These serve as the beacon values for each re-randomization (adversarial or honest). The adversary receives the corresponding rand value before each adversarial re-randomization, and together with each honest re-randomization.

3. Each time the adversary $\mathcal{A}$ outputs re-randomized commitments, it also outputs the associated randomness used to generate the associated precommitment and the randomness used for re-randomization. The challenger then checks that this randomness is indeed admissible.

As before, we define the adversary's advantage as
$$\mathsf{Adv}^{\text{strong-rrc}}_{\mathcal{A},\mathsf{R}}(\lambda) := \left|2\Pr[\mathbf{G}^{\text{strong}}_{\mathcal{A},\mathsf{R}}(\lambda) = 1] - 1\right|.$$

Definition 6.2. *An RRC scheme R is* **strongly-unlinkable** *if for all PPT adversaries $\mathcal{A}$ the function $\mathsf{Adv}^{\text{strong-rrc}}_{\mathcal{A},\mathsf{R}}(\lambda)$ is negligible.*

Game $\boldsymbol{G}^{\text{strong}}_{\mathcal{A},\mathsf{R}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$

$pp \xleftarrow{\$} \mathsf{R}.\mathit{Setup}(1^\lambda)$

$(T, i_0^{(1)}, i_1^{(1)} \ldots, i_0^{(T)}, i_1^{(T)}, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(pp)$

$\overrightarrow{\mathsf{pcom}} \leftarrow ()$ // initialize an empty vector

for t **in** $\{1, \ldots, T\}$:

  for j **in** $\{1, \ldots, i_0^{(t)}\}$: $r_{t,0,j} \xleftarrow{\$} \{0,1\}^\rho,\ \mathsf{pcom}_{t,0,j} \leftarrow \mathsf{R}.\mathit{Precommit}(pp; r_{t,0,j}),\ \overrightarrow{\mathsf{pcom}} \leftarrow \overrightarrow{\mathsf{pcom}} \| \mathsf{pcom}_{t,0,j}$

  for j **in** $\{1, \ldots, i_1^{(t)}\}$: $r_{t,1,j} \xleftarrow{\$} \{0,1\}^\rho,\ \mathsf{pcom}_{t,1,j} \leftarrow \mathsf{R}.\mathit{Precommit}(pp; r_{t,1,j}),\ \overrightarrow{\mathsf{pcom}} \leftarrow \overrightarrow{\mathsf{pcom}} \| \mathsf{pcom}_{t,1,j}$

    // ρ denotes the number of random coins used by $\mathsf{R}.\mathit{Precommit}$

$(\mathsf{pcom}'_{1,0}, \mathsf{pcom}'_{1,1}, \ldots, \mathsf{pcom}'_{T,0}, \mathsf{pcom}'_{T,1}, \mathsf{state}) \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \overrightarrow{\mathsf{pcom}})$

$\mathsf{rand}_{1,0}, \mathsf{rand}_{1,1} \ldots, \mathsf{rand}_{T,0}, \mathsf{rand}_{T,1} \xleftarrow{\$} D$

$(c_0^{(0)}, k_0) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp), \quad (c_1^{(0)}, k_1) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp)$

$\mathsf{aux}_0 \leftarrow c_0^{(0)} \| c_1^{(0)}$

for t **in** $\{1, \ldots, T\}$:

  $(\mathsf{state}, c_0^{(t)}, c_1^{(t)}, r_0, r_1, r'_0, r'_1) \xleftarrow{\$} \mathcal{A}(\mathsf{state}, \mathsf{aux}_{t-1}, \mathsf{rand}_{t,1}, \mathsf{rand}_{t,0})$

    // r_0 and r_1 are the random coins $\mathcal{A}$ claims to have used to generate $\mathsf{pcom}_{t,0}$ and $\mathsf{pcom}_{t,1}$

    // r'_0 and r'_1 are the random coins $\mathcal{A}$ claims to have used for re-randomization

  if $(c_0^{(t)} \neq \mathsf{R}.\mathit{Randomize}(pp, c_0^{(t-1)}; r'_0))$ OR $(c_1^{(t)} \neq \mathsf{R}.\mathit{Randomize}(pp, c_1^{(t-1)}; r'_1))$: **abort**

    // check that r'_0 and r'_1 were used by $\mathcal{A}$ for re-randomization

  for d **in** $\{0,1\}$: $\mathcal{G}_d \leftarrow \mathcal{G}(\mathsf{pcom}'_{t,d}, r_d, \mathsf{rand}_{t,d})$

  $\mathsf{aux}_t \leftarrow ()$ // initialize an empty vector

  for j **in** $\{1, \ldots, i_0^{(t)}\}$: $(\mathsf{rand}^t_{0,j}, c_0^{(t)}) \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, r_{t,0,j}, c_0^{(t)}),\ \mathsf{aux}_t \leftarrow \mathsf{aux}_t \| (\mathsf{rand}^t_{0,j}, c_0^{(t)})$

  for j **in** $\{1, \ldots, i_1^{(t)}\}$: $(\mathsf{rand}^t_{1,j}, c_1^{(t)}) \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, r_{t,1,j}, c_1^{(t)}),\ \mathsf{aux}_t \leftarrow \mathsf{aux}_t \| (\mathsf{rand}^t_{1,j}, c_1^{(t)})$

    // the notation $(\mathsf{rand}, c') \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, r, c)$ is defined above

  if $r'_0 \notin \mathcal{G}_0$ OR $r'_1 \notin \mathcal{G}_1$: $c_0^{(t)} \leftarrow c_0^{(t-1)},\ c_1^{(t)} \leftarrow c_1^{(t-1)}$

$(\mathsf{rand}_0, c_0) \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, c_0^{(T)}),\ (\mathsf{rand}_1, c_1) \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, c_1^{(T)})$

$b' \xleftarrow{\$} \mathcal{A}(c_b, c_{1-b}, \mathsf{rand}_0, \mathsf{rand}_1, \mathsf{state})$

return $b = b'$

Figure 4: The strong unlinkability security game for an adversary $\mathcal{A}$ and an RRC scheme R

Game $\mathbf{G}^{\text{pr}}_{\mathcal{A},\mathsf{R}}(\lambda)$

$b \xleftarrow{\$} \{0,1\}$
$pp \xleftarrow{\$} \mathsf{R}.\mathit{Setup}(1^\lambda)$
$r \xleftarrow{\$} \{0,1\}^\rho,\ \mathsf{pcom} \leftarrow \mathsf{R}.\mathit{Precommit}(pp; r)$
$(\mathsf{pcom}', \mathsf{state}) \xleftarrow{\$} \mathcal{A}(pp, \mathsf{pcom})$
$\mathsf{rand} \xleftarrow{\$} D$
$(c, k) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp),\ (c'_0, k') \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp)$
$(c'', r', r'', \mathsf{state}) \xleftarrow{\$} \mathcal{A}(\mathsf{state}, c, \mathsf{rand})$
if $c'' \neq \mathsf{R}.\mathit{Randomize}(pp, c; r'')$: **abort**
if $r'' \notin \mathcal{G}(\mathsf{pcom}', r', \mathsf{rand})$: **abort**
$(\mathsf{rand}', c_0) \xleftarrow{\$} \mathsf{R}.\mathit{Randomize}(pp, r, c'')$
$c_1, c'_1 \xleftarrow{\$} \mathcal{C}_\lambda$
$b' \xleftarrow{\$} \mathcal{A}(c_b, c'_b, \mathsf{rand}', \mathsf{state})$
return $b = b'$

Figure 5: The security game for an adversary $\mathcal{A}$ attacking the strong pseudorandomness of R

How to put the strong unlinkability definition to use. In the strengthened security game from Fig. 4, whenever the adversary re-randomizes, it also sends to the challenger the randomness that went into this re-randomization process (that is, the randomness that went into *Precommit* and into *Randomize*). This means that whenever using a strongly-unlinkable RRC scheme within a larger protocol, one should require that re-randomizers provide a argument of knowledge for such randomness (and potentially of additional secrets that are related to the larger super-protocol). Then, a security reduction that tries to break the security of the RRC scheme can use the knowledge extractor of the proof system to extract the randomness and output it in the RRC security game. Our SSLE protocol in Section 7 provides an example of how to use RRC schemes within a larger protocol. In Section 8 we discuss specific ways to construct the necessary arguments of knowledge for our lattice-based RRC schemes.

Strongly-pseudorandom RRC schemes. We present the notion of strong pseudorandomness for RRC schemes. Roughly speaking, an RRC scheme enjoys strong pseudorandomness, if honestly rerandomized commitments are pseudorandom. That is, it is indistinguishable from a uniformly-random member of the domain $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ of commitments. Moreover, honest re-randomization should output pseudorandom commitments even on commitments that were previously re-randomized by the adversary (using admissible randomness). This is captured by the security game in Fig. 5.

As before, we define the adversary's advantage as

$$\mathsf{Adv}^{\text{pr-rrc}}_{\mathcal{A},\mathsf{R}}(\lambda) := \left|2\Pr[\mathbf{G}^{\text{pr}}_{\mathcal{A},\mathsf{R}}(\lambda) = 1] - 1\right|.$$

Definition 6.3. *A B-randomizable* R *scheme is* **strongly-pseudorandom** *if for all* PPT *adversaries* $\mathcal{A}$ *the function* $\mathsf{Adv}^{\text{pr-rrc}}_{\mathcal{A},\mathsf{R}}(\lambda)$ *is negligible.*

A simple hybrid argument shows that an RRC scheme that is strongly-pseudorandom is also strongly-unlinkable.

Proposition 6.1. *If an RRC scheme* R *is strongly-pseudorandom then it is also strongly-unlinkable.*

We now turn to present several ways to augment our basic RRC schemes so that they achieve strong-pseudorandomness, and hence strong unlinkability.

6.2 Constructing Strongly-Pseudorandom RRCs

We now present a way to turn our lattice-based constructions of RRC schemes to ones that provide strong pseudorandomness, and hence strong unlinkability. We start by describing such a mechanism for our LWE-based scheme, and then discuss how the same ideas can also be applied to our Ring-LWE-based scheme.

Immunizing our LWE-based RRC scheme $\mathsf{R}_{\mathsf{LWE}}$ against adversarial re-randomizations per the Definition 6.2 amounts to defining the beacon distribution D, the algorithms $\mathsf{R}_{\mathsf{LWE}}.\mathit{Precommit}$ and $\mathsf{R}_{\mathsf{LWE}}.\mathit{Extract}$, and the set $\mathcal{G}$ of admissible random strings. We do so as follows:

- D is the uniform distribution over $\{-1,1\}^{m\times m}$.
- $\mathsf{R}_{\mathsf{LWE}}.\mathit{Precommit}(pp;r)$: the randomness r to the algorithm is parsed as a tuple $(\boldsymbol{R},r')$ of a uniformly-random matrix $\boldsymbol{R}$ in $\{-1,1\}^{m\times m}$ and randomness r' to a standard (not re-randomizable) statistically-binding non-interactive commitment scheme $\mathsf{C}=(\mathsf{C}.\mathit{Setup},\mathsf{C}.\mathit{Commit})$ (for definitions of standard commitment schemes, see for example [25] and also Section 8). The algorithm then commits to $\boldsymbol{R}$ using C: it computes $\mathsf{pcom}\leftarrow\mathsf{C}.\mathit{Commit}(pp_{\mathsf{C}},\boldsymbol{R};r')$ and outputs pcom (the public parameters pp_{C} for C are sampled by the $\mathsf{C}.\mathit{Setup}$ algorithm during the operation of $\mathsf{R}_{\mathsf{LWE}}.\mathit{Setup}$ and are included as part of the public parameters of $\mathsf{R}_{\mathsf{LWE}}$).
- $\mathsf{R}_{\mathsf{LWE}}.\mathit{Extract}(pp,r,\mathsf{rand})$ parses r as $(\boldsymbol{R},r')$ and treats rand as a matrix $\boldsymbol{R}'$ in $\{-1,1\}^{m\times m}$. It outputs $\boldsymbol{R}''\leftarrow\boldsymbol{R}+\boldsymbol{R}'\in\mathbb{Z}_q^{m\times m}$.
- The set $\mathcal{G}=\mathcal{G}(\mathsf{pcom},r=(\boldsymbol{R},r'),\mathsf{rand}=\boldsymbol{R}')$ is then the singleton set $\{\boldsymbol{R}+\boldsymbol{R}'\}$ if $\mathsf{pcom}=\mathsf{C}.\mathit{Commit}(pp_{\mathsf{C}},\boldsymbol{R};r')$. Otherwise, if $\mathsf{pcom}\neq\mathsf{C}.\mathit{Commit}(pp_{\mathsf{C}},\boldsymbol{R};r')$ then $\mathcal{G}=\emptyset$ and there is no admissible randomness. That is, $\mathcal{G}$ "checks" if pcom is a valid commitment to $\boldsymbol{R}$ given the randomness used to generate it, and if so, the only admissible randomness for $\mathsf{R}_{\mathsf{LWE}}.\mathit{Randomize}$ is the sum of $\boldsymbol{R}+\boldsymbol{R}'$.

We denote the RRC scheme obtained by these augmentations by $\mathsf{R}^{+}_{\mathsf{LWE}}$. We first argue that the scheme is correct per Definition 6.1. Condition 2 of the definition holds trivially. To see why Condition 1 holds, observe that honest r_is used for re-randomization are now m-by-m matrices, whose coordinates are independently sampled from a distribution which attains 0 with probability $1/2$, and -2 or 2 with probability $1/4$ each. A straightforward adaptation of the proof of Lemma 2.3 shows that it still applies (with a slightly worse constant C) and hence the previous proof of correctness still goes through.

As for security, the following theorem proves that $\mathsf{R}^{+}_{\mathsf{LWE}}$ satisfies strong pseudorandomness. In conjunction with Proposition 6.1, this implies that it is also strongly-unlinkable.

Theorem 6.2. *The scheme* $\mathsf{R}^{+}_{\mathsf{LWE}}$ *is a strongly-pseudorandom RRC scheme.*

Proof. The proof follows a similar template to that of Theorem 4.3. Let $\mathcal{A}$ be an adversary participating in the strong pseudorandomness security game, and let $\mathbf{G}_0=\mathbf{G}^{\mathrm{pr}}_{\mathcal{A},\mathsf{R}}(\lambda)$. Consider a hybrid game $\mathbf{G}_1$ obtained from $\mathbf{G}_0$ by changing the precommitment pcom sampled by the challenger to be a commitment to the all zero matrix in $\mathbb{Z}_q^{m\times m}$. The games $\mathbf{G}_0$ and $\mathbf{G}_1$ are indistinguishable by the hiding property of the commitment scheme C.

Now, consider a hybrid game $\mathbf{G}_2$ which is obtained from $\mathbf{G}_1$ by sampling the commitments c and c'_0 as independent pairs of uniformly-random and independent matrices. The games $\mathbf{G}_1$ and $\mathbf{G}_2$ are indistinguishable by the LWE assumption. Now fix any matrix $\boldsymbol{R} \in \mathbb{Z}_q^{m \times m}$ outputted by $\mathcal{A}$ as part of the randomness used in Line 4, and consider the family of functions $\{f_{\boldsymbol{R},\boldsymbol{A},\boldsymbol{U}}(\boldsymbol{R}') = (\boldsymbol{R}+\boldsymbol{R}') \cdot [\boldsymbol{A},\boldsymbol{U}]\}_{\boldsymbol{A},\boldsymbol{U}}$. This family is universal over the choices of $\boldsymbol{A}$ and $\boldsymbol{U}$. Hence, since $\boldsymbol{A}$ and $\boldsymbol{U}$ making up the commitment c in $\mathbf{G}_1$ are sampled after $\boldsymbol{R}$ is determined, the leftover hash lemma [43] guarantees that c_0 is statistically close to uniform. Hence, $\mathbf{G}_2$ is statistically indistinguishable from a game $\mathbf{G}_3$ in which c_0 and c'_0 are sampled uniformly at random from $\mathbb{Z}_q^{m \times (n+\ell)}$. In $\mathbf{G}_3$, the view of $\mathcal{A}$ is independent of b and hence its advantage is 0. □

Strong unlinkability from the Ring LWE assumption. We can use a similar technique in order to augment our Ring-LWE-based RRC scheme with strong unlinkability. The only difference is that now $\boldsymbol{R}$ and $\boldsymbol{R}'$ are sampled as matrices of "short" polynomials. That is, the distribution D samples a matrix $\boldsymbol{R}'$ as follows: Each coordinate is an independent polynomial, whose coefficients are sampled independently and uniformly from $\{-1, 1\}$. *Precommit* samples a commitment to a matrix $\boldsymbol{R}$ sampled from the same distribution, and *Extract* outputs $\boldsymbol{R} + \boldsymbol{R}'$. Finally, the set $\mathcal{G}(\mathsf{pcom}, r, \mathsf{rand}) = \{\boldsymbol{R} + \boldsymbol{R}'\}$ as before if the precommitment pcom is consistent with r and $\emptyset$ otherwise. Correctness follows similarly as in the LWE case, replacing the use of Lemma 2.3 with Lemma 2.4. For strong pseudorandomness, we replace the use of the leftover hash lemma [43] with Lemma 2.2.[2]

6.3 Strong Pseudorandomness without A Randomness Beacon

The above approach requires a randomness beacon, which is a very reasonable assumption in the context of SSLE protocols. However, there might be other scenarios in which one might want to use RRCs without assuming the availability of such a beacon. This is formally captured by the above definitions by fixing D to be the constant distribution outputting $\perp$ with probability 1. In Appendix A, we present three different approaches to augment our schemes to provide strong unlinkability without assuming a randomness beacon.

7 From Rerandomizable Commitments to Single Secret Leader Election

In this section, we present our main application of post-quantum RRCs: constructing post-quantum secure protocols for *single secret leader election (SSLE)*. We begin this section by presenting definitions for SSLE protocols, and then show how an RRC scheme can be used generically to construct such a protocol. As described in the introduction, our construction generalizes the efficient DDH-based construction by Boneh, Eskandarian, Hanzlik, and Greco [24].

7.1 SSLE: Syntax and Security Notions

We adapt the definitions of Boneh et al. [24]. On the one hand, they presented a broad syntax that allowed them to capture various constructions in addition to their DDH-based one. Since we are primarily focused on our construction from RRC schemes, we can simplify their definitions. On the other hand, we also generalize their definitions to allow for interactive election protocols, as is the case for the *whisk* implementation of the BEHG protocol [44].

Formally, an SSLE protocol ssle is a tuple of three algorithms:

[2] Technically speaking, we require a generalization of Lemma 2.4, in which the coefficients of each entry of $\boldsymbol{R}$ may be chosen from different (but small) sets. Fortunately, the proof of 2.4 readily extends to this setting.

- *Setup*$(1^\lambda, N) \to pp$: The setup algorithm takes in the security parameter 1^λ and an integer bound N and outputs public parameters *pp*. Here, N serves as an upper bound on the number of participants in an election.

- *Elect*$(pp, \mathcal{J}, i) \to (\mathsf{pst}, b, \pi)$: This is the interactive (stateful) algorithm that each party runs during the election protocol. Its initial input is *pp*, a set $\mathcal{J}$ of size at most N of registered participating parties, and the executing party's index i. Its final output is a public state pst, a bit $b \in \{0,1\}$ indicating whether i was elected as leader, and a proof π asserting that this is the case ($\pi = \perp$ if $b = 0$). The public state pst is used to verify the proof π and will end up being the same across all parties.

  For concreteness, we assume here that each party is associated with a unique id $\in [N]$, and hence $\mathcal{J} \subseteq [N]$. We elaborate on how the election protocol works below.

- *Verify*$(pp, i, \mathsf{pst}, \pi) \to b$: The winner verification algorithm takes in the public parameters *pp*, an index i for the claimed winner, the public state pst, and a proof π and outputs a bit $b \in \{0,1\}$, where 1 implies acceptance and 0 rejection.

The election protocol's syntax in depth. The election protocol is specified by the interactive (stateful) algorithm *Elect*, which the participating parties run locally. The protocol is potentially interactive and proceeds in rounds. The initial input to the *Elect* algorithm, executed by party i, is the public parameters *pp*, the set $\mathcal{J}$ of the parties participating in the protocol, and its own index i. On this input, *Elect* outputs the outgoing message $m_i^{(1)}$ from the ith party in the first round and some local state st. We assume that all messages are broadcast to all other parties over a public broadcast channel. In a subsequent round $k > 1$, the input to *Elect* are the incoming messages $\{m_j^{(k-1)}\}_{j \in \mathcal{I} \setminus \{i\}}$, the local state st, and potentially additional randomness r_k generated by a randomness beacon. The value r_k is sampled only *after* the kth round has been completed. On this input, *Elect* again outputs an outgoing message $m_i^{(k)}$ from party i and an updated local state st'. Finally, in the last round, *Elect* outputs a public state pst, a bit $b \in \{0,1\}$ – indicating whether i was elected or not – and a proof π. If $b = 1$, π asserts that i was indeed elected, and if $b = 0$ then $\pi = \perp$. Recall that in an execution of the protocol, the final public state pst will be the same across all parties.

Security. Following Boneh et al. [24], we require that an SSLE protocol satisfies three security properties: *uniqueness*, *unpredictability*, and *fairness*. We now define each of them.

Uniqueness. Informally, uniqueness means that after an election has taken place, there is at most one participant that can prove that they have been chosen as leader. Formally, for an SSLE protocol $\mathsf{ssle} = (\mathit{Setup}, \mathit{Elect}, \mathit{Verify})$, and adversary $\mathcal{A}$, and a security parameter $\lambda \in \mathbb{N}$, uniqueness is defined via the following security experiment, denoted $\mathsf{UNIQUE}_{\mathcal{A},\mathsf{ssle}}(\lambda)$:

1. **Setup stage:**
   (a) On input 1^λ, the adversary $\mathcal{A}$ outputs an integer N and a local state st.
   (b) $\mathsf{ssle}.\mathit{Setup}(1^\lambda, N)$ is executed and outputs *pp*.
2. **Election stage:**

(a) On input (st, pp), the adversary $\mathcal{A}$ outputs a subset $\mathcal{J} \subseteq [N]$ of parties to participate in the election. $\mathcal{A}$ also outputs an additional subset $\mathcal{I} \subseteq \mathcal{J}$; this is the set of corrupted users during the protocol.[3]

(b) The *Elect* protocol is executed among the parties in $\mathcal{J}$, where $\mathcal{A}$ plays the roles of the parties in $\mathcal{I}$. The initial input of party $j \in \mathcal{J} \setminus \mathcal{I}$ is $(pp, \mathcal{J}, j)$. At the end of the protocol, each honest party $j \in \mathcal{J} \setminus \mathcal{I}$ has an output $(\mathsf{pst}_j, b_j, \pi_j)$. The output of $\mathcal{A}$ is an updated local state st.

3. **Output stage:**

(a) On input st, $\mathcal{A}$ outputs two indices $i, j \in \mathcal{J}$ and two proofs π_i, π_j.

(b) The experiment's output is 1 if there is an honest party $k \in \mathcal{J} \setminus \mathcal{I}$ such that $\mathit{Verify}(pp, i, \mathsf{pst}_k, \pi_i) = \mathit{Verify}(pp, j, \mathsf{pst}_k, \pi_j) = 1$. Otherwise, the experiment outputs 0.

The advantage of an adversary $\mathcal{A}$ in breaking the unlinkability of an SSLE protocol ssle is defined by:

$$\mathsf{Adv}^{\mathrm{unique}}_{\mathcal{A},\mathsf{ssle}}(\lambda) := \Pr\left[\mathsf{UNIQUE}_{\mathcal{A},\mathsf{ssle}}(\lambda) = 1\right].$$

Definition 7.1. *An SSLE protocol* ssle *satisfies* uniqueness *if for any probabilistic polynomial-time adversary* $\mathcal{A}$ *the function* $\mathsf{Adv}^{\mathrm{unique}}_{\mathcal{A},\mathsf{ssle}}(\lambda)$ *is negligible.*

Unpredictability. Intuitively, unpredictability requires that after an election has taken place, the identity of the chosen leader remains hidden from all other parties (until she reveals herself). In more detail, for an SSLE protocol $\mathsf{ssle} = (\mathit{Setup}, \mathit{Elect}, \mathit{Verify})$, and adversary $\mathcal{A}$, integers n and c such that $c < n$, and a security parameter $\lambda \in \mathbb{N}$, unpredictability is defined via the following security experiment, denoted $\mathsf{UNPRED}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda)$:

1. The **Setup stage** and **Election stage** are defined as in $\mathsf{UNIQUE}_{\mathcal{A},\mathsf{ssle}}(\lambda)$, with the restriction that the subset $\mathcal{J}$ of participating parties is of size n,[4] and number $|\mathcal{I}|$ of corrupted parties is at most c.

2. **Output stage:**

(a) On input st, $\mathcal{A}$ outputs an index $i \in \mathcal{J}$.

(b) If $i \in \mathcal{I}$, then the experiment outputs 0. Otherwise, the experiment outputs 1 if and only if $b_i = 1$.

For parameters n, c such that $n > c$, the advantage of an adversary $\mathcal{A}$ in breaking the unpredictability of an SSLE protocol ssle is defined by:

$$\mathsf{Adv}^{\mathrm{unpred}}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda) := \Pr\left[\mathsf{UNPRED}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda) = 1 \mid i \in \mathcal{J} \setminus \mathcal{I}\right] - \frac{1}{n-c}.$$

Definition 7.2. *Let* $n, c \in \mathbb{N}$ *such that* $n > c$. *An SSLE protocol* ssle *satisfies* (n, c)-unpredictability *if for any probabilistic polynomial-time adversary* $\mathcal{A}$ *it holds that* $\mathsf{Adv}^{\mathrm{unique}}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda)$ *is negligible.*

[3] As we discuss below, one can strengthen the adversarial model to allow the adversary to adaptively corrupt the subset $\mathcal{I}$.

[4] In this section, n is used to denote the number of parties participating in the election protocol. Recall that in previous sections, n was used as a parameter for the LWE and Ring-LWE assumptions. Since the use of n is always clear from context, we allow ourselves this overloading of notation.

Fairness. An SSLE protocol is said to be fair if all parties have essentially the same probability of being chosen as leaders. Formally, for an SSLE protocol $\mathsf{ssle} = (\mathit{Setup}, \mathit{Elect}, \mathit{Verify})$, an adversary $\mathcal{A}$, integers n, c such that $c < n$, and a security parameter $\lambda \in \mathbb{N}$, unpredictability is defined via the following security experiment, denoted $\mathsf{FAIR}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda)$:

1. The **Setup stage** and the **Election stage** are defined as in $\mathsf{UNPRED}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda)$.
2. **Output stage:**
   (a) The experiment outputs 0 if there is a party $i \in \mathcal{J} \setminus \mathcal{I}$ such that for all $k \in \mathcal{J} \setminus \mathcal{I}$ it holds that $\mathit{Verify}(pp, i, \mathsf{pst}_k, \pi_i) = 1$. Otherwise, the output of the experiment is 1.

For parameters n, c such that $n > c$, the advantage of an adversary $\mathcal{A}$ in breaking the unpredictability of an SSLE protocol ssle is defined by:

$$\mathsf{Adv}^{\mathrm{fair}}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda) := \Pr\left[\mathsf{FAIR}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda) = 1 \mid i \in \mathcal{J} \setminus \mathcal{I}\right] - \frac{c}{n}.$$

Definition 7.3. *Let $n, c \in \mathbb{N}$ such that $n > c$. An SSLE protocol* ssle *satisfies* (n, c)-unpredictability *if for any probabilistic polynomial-time adversary $\mathcal{A}$ it holds that* $\mathsf{Adv}^{\mathrm{unique}}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda)$ *is negligible.*

7.2 The Commit-and-Shuffle Protocol

We now present our construction of an SSLE protocol from re-randomizable commitments. The construction follows the commit-and-shuffle approach of the DDH-based SSLE protocol presented by Boneh et al. [24]. However, it replaces the reliance on DDH by a generic reliance on RRCs, thus allowing us to instantiate the protocol based on lattice-based assumptions.

The protocol relies on the following building blocks:

1. An RRC scheme $\mathsf{R} = (\mathsf{R}.\mathit{Setup}, \mathsf{R}.\mathit{Commit}, \mathsf{R}.\mathit{Randomize}, \mathsf{R}.\mathit{Test}, \mathsf{R}.\mathit{Precommit}, \mathsf{R}.\mathit{Extract}, D, \mathcal{R})$. We assume that R is non-trivial, B-randomizable, binding, and strongly unlinkable (recall Sections 3 and 6). Looking ahead, in each round of the SSLE protocol, we will view the output of the randomness beacon as a tuple of n samples from the distribution D. The SSLE protocol we construct supports a bound of $N = B$ on the number of parties participating in an election. However, as we discuss below, this bound can be improved using different shuffle patterns.
2. A hash function H mapping keys k outputted by $\mathsf{R}.\mathit{Commit}$ to strings in $\{0,1\}^\lambda$.
3. A non-interactive zero-knowledge argument of knowledge $\mathsf{NIZK} = (\mathsf{NIZK}.\mathit{Setup}, \mathsf{NIZK}.P, \mathsf{NIZK}.V)$ for proving knowledge of a permutation σ and *admissible* randomness (recall Section 6) $r_1, \ldots, r_n$ used to shuffle one vector of RRC commitments $(c_1, \ldots, c_n)$ to another vector $(c'_1, \ldots, c'_n)$. That is, $c'_{\sigma(i)}$ is a re-randomization of c_i using randomness r_i to $\mathsf{R}.\mathit{Randomize}$. Moreover, we require that the NIZK is simulation sound; that is, knowledge soundness is preserved even against a malicious prover who observed simulated proofs. More formally, NIZK_R is an argument of knowledge for the relation $\mathcal{W} = \{\mathcal{W}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{W}_\lambda$ is defined as:

$$\left\{ \left(\begin{array}{c} pp, (c_1, \ldots, c_n) \\ (c'_1, \ldots, c'_n) \\ (\mathsf{rand}_1, \ldots, \mathsf{rand}_n) \\ (\mathsf{pcom}_1, \ldots, \mathsf{pcom}_n) \end{array} ; \begin{array}{c} \sigma, (r_1, \ldots, r_n) \\ (r'_1, \ldots, r'_n) \\ (r''_1, \ldots, r''_n) \end{array} \right) : \begin{array}{c} \sigma \text{ is a permutation on } [n] \\ \forall i \in [n] \ : \ r''_i = \mathsf{R}.\mathit{Extract}(pp, r_i, \mathsf{rand}_i; r'_i) \\ \wedge\ r''_i \in \mathcal{R}(\mathsf{pcom}_i, r_i, \mathsf{rand}_i) \\ \wedge\ c'_{\sigma(i)} = \mathsf{R}.\mathit{Randomize}(pp, c_i; r''_i) \end{array} \right\}.$$

   See Section 8 for a formal definition of such a proof system.

Equipped with these building blocks, the SSLE protocol is defined in Fig. 6. The protocol follows the commit-and-shuffle approach described informally in the introduction.

Theorem 7.1. *Let n and c be any integers such that $c < n$. If H is modeled as a random oracle, then the protocol* ssle *defined in Fig. 6 satisfies uniqueness, (n, c)-fairness, and (n, c)-unpredictability.*

The proof of the theorem can be found in Appendix C.1.

General shuffle patterns. The protocol in Fig. 6 relies on the "complete shuffle": *All* of the parties take turns, and each party shuffles *all* of the commitments when it is her turn. This is a simple and easy-to-present shuffle, and it also provides optimal unpredictability guarantees. However, this is not without cost, as this means that the proofs of correct shuffle that parties compute are for large statements. Moreover, when instantiating it with our lattice-based RRC schemes, one has to set the modulus q to be large enough to support N re-randomizations.

To mitigate these costs, one can employ other shuffles rather than the complete shuffle. Boneh et al. [24] consider dividing the parties to equally-sized buckets and having each party shuffle only the commitments in her bucket. This reduces the size of its shuffle and also the number of re-randomizations each commitment undergoes. However, it weakens the unpredictability guarantees. Whisk [44] employs a more sophisticated Feistel-based approach. Recently, Larsen, Obremski, and Simkin [47] proposed a randomized approach for distributed shuffling in which each party permutes k commitments and the number of rounds is $\widetilde{O}(n/k + c)$, where c is the number of corruptions.

Reducing storage and communication. The commit-and-shuffle protocol may be modified such that in each shuffle, the shuffler uses the same randomness to re-randomize all commitments, as proposed in [28] for DDH commitments. When using our lattice-based commitments, this comes at a cost: in order to use a single re-randomization matrix $\boldsymbol{R}$, one needs to increase the parameter m (the number of rows in each commitment), otherwise the scheme is insecure – it is no longer unlinkable. This cost is considerable in the integer setting but is more manageable in the ring setting. In the ring setting, to use a single re-randomization matrix $\boldsymbol{R}$ across all commitments, m needs to scale by a factor of N/n to maintain unlinkability, where N is the number of commitments being re-randomized and n is the degree parameter of the ring-LWE assumption. In contrast, in the integer case, m would need to grow by a factor of N to retain unlinkability. As a concrete example, for $N = \Omega(n)$, the ring-based RRC commitments and the re-randomization matrix would only grow by a constant factor, while in the integer case, they would grow linearly in N.[5]

This modification opens up the possibility of also using the same vector $\boldsymbol{a}$ across all commitments. We briefly describe this change in the more efficient ring setting:

- An initial vector $\boldsymbol{a}$ is sampled as part of the public parameters of the protocol. Each initial commitment is then of the form $\boldsymbol{U} \leftarrow \boldsymbol{a} \cdot \boldsymbol{v}^T + \boldsymbol{E} \in \mathcal{R}_q^{m \times \ell}$, where $\boldsymbol{v} = \mathsf{H}(k)$ and k is the committed random value.
- When re-randomizing commitments $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_N$, the shuffler samples a single matrix $\boldsymbol{R}$, and broadcasts a single $\boldsymbol{a}' \leftarrow \boldsymbol{R} \cdot \boldsymbol{a}$ and $\boldsymbol{U}'_i \leftarrow \boldsymbol{R} \cdot \boldsymbol{U}_i$ for $i = 1, \ldots, N$.

Applying this modification, in each shuffle, we only need to transmit and store a single re-randomized vector $\boldsymbol{a}' \in \mathcal{R}_q^m$ instead of N such vectors. However, in our setting, this change does not immediately lead to savings in total storage and communication for a large number of commitments due to enlarging m.

[5] While the re-randomization matrix would typically not be transmitted in the clear, its size does affect the complexity of the proof of shuffle (recall our discussion in the introduction and also see Section 8).

- *Setup*$(1^\lambda, N)$:
  1. Sample $pp_{\mathsf{NIZK}} \xleftarrow{\$} \mathsf{NIZK}.\mathit{Setup}(1^\lambda)$ and $pp_{\mathsf{R}} \xleftarrow{\$} \mathsf{R}.\mathit{Setup}(1^\lambda)$.
  2. Return $pp = (pp_{\mathsf{NIZK}}, pp_{\mathsf{R}})$.
- The election protocol *Elect* for subset $\mathcal{J} = \{1, \ldots, n\}$:
  1. Party $i \in \mathcal{J}$ starts with a local input $pp, \mathcal{J}, i$. The protocol proceeds in three stages.
  2. **Commit stage:**
     (a) Party i samples $(c_i, k_i) \xleftarrow{\$} \mathsf{R}.\mathit{Commit}(pp_{\mathsf{R}})$, and computes $h_i \leftarrow \mathsf{H}(k_i)$.
     (b) For $j = 1, \ldots, n$: Party i samples $r_{i,1}, \ldots, r_{i,n} \xleftarrow{\$} \{0,1\}^\rho$ and $\mathsf{pcom}_{i,j} \leftarrow \mathsf{R}.\mathit{Precommit}(pp_{\mathsf{R}}; r_{i,j})$ for each $j \in [n]$.
     (c) Party i broadcasts $\mathsf{pcom}_{i,1}, \ldots, \mathsf{pcom}_{i,n}$ to all other users in $\mathcal{J}$.
     (d) Upon receiving all precommitments, party i broadcasts c_i, h_i to all other users in $\mathcal{J}$.
     (e) Upon receiving $c_1, \ldots, c_n, h_1, \ldots, h_n$ each party i sets $c_{i,j}^{(0)} \leftarrow c_j$ for all $j \in [n]$.
  3. **Shuffle stage:** The shuffle stage proceeds in n rounds. Let $\mathsf{rand}_{i,1}, \ldots, \mathsf{rand}_{i,n}$ be the output of the randomness beacon in round $i \in [n]$.
     In round $i \in [n]$, party i does:
     (a) Compute $r''_{i,j} \xleftarrow{\$} \mathsf{R}.\mathit{Extract}(pp, r_{i,j}, \mathsf{rand}_{i,j})$ for each $j \in [n]$.
     (b) Compute $c'_j \leftarrow \mathsf{R}.\mathit{Randomize}(pp, c_{i,j}^{(i-1)}; r''_{i,j})$ for each $j \in [n]$.
     (c) Sample a random permutation σ on $[n]$ and set $c_{i,j}^{(i)} \leftarrow c'_{\sigma(j)}$ for each $j \in [n]$.
     (d) Compute a proof pf_i by invoking $\mathsf{NIZK}.P$ on input pp_{NIZK}, the instance composed of pp_{R}, $(c_{i,1}^{(i-1)}, \ldots, c_{i,n}^{(i-1)})$, $(c_{i,1}^{(i)}, \ldots, c_{i,n}^{(i)})$, $(\mathsf{rand}_{i,1}, \ldots, \mathsf{rand}_{i,n})$ and $(\mathsf{pcom}_{i,1}, \ldots, \mathsf{pcom}_{i,n})$, and the witness composed of σ, $(r_{i,1}, \ldots, r_{i,n})$, $(r'_{i,1}, \ldots, r'_{i,n})$, and $(r''_{i,1}, \ldots, r''_{i,n})$, where $r'_{i,1}, \ldots, r'_{i,n}$ is the randomness used for $\mathsf{R}.\mathit{Extract}$ above.
     (e) Broadcast $(c_{i,1}^{(i)}, \ldots, c_{i,n}^{(i)})$ and pf_i to all other users in $\mathcal{J}$.

     In round $j \neq i$, Upon receiving $(c_{j,1}^{(j)}, \ldots, c_{j,n}^{(j)})$ and pf_j from party j, party i does:
     (a) Verify the proof pf_j by invoking $\mathsf{NIZK}.V$ on input pp_{NIZK}, the instance composed of pp_{R}, $(c_{i,1}^{(j-1)}, \ldots, c_{i,n}^{(j-1)})$, $(c_{j,1}^{(j)}, \ldots, c_{j,n}^{(j)})$, $(\mathsf{rand}_{j,1}, \ldots, \mathsf{rand}_{j,n})$ and $(\mathsf{pcom}_{j,1}, \ldots, \mathsf{pcom}_{j,n})$.
     (b) If verification passes, set $(c_{i,1}^{(j)}, \ldots, c_{i,n}^{(j)}) \leftarrow (c_{j,1}^{(j)}, \ldots, c_{j,n}^{(j)})$. Otherwise, set $(c_{i,1}^{(i)}, \ldots, c_{i,n}^{(i)}) \leftarrow (c_{i,1}^{(j-1)}, \ldots, c_{i,n}^{(j-1)})$.
  4. **Selection stage:** Let $\mathcal{S} = \{j \in [n] : \exists j' \neq j \text{ s.t. } h_j = h_{j'}\}$. After the selection stage, the randomness beacon outputs a value we treat as an integer $i^* \xleftarrow{\$} [n] \setminus \mathcal{S}$. Each party $i \in [n]$ does:
     (a) Set $\mathsf{pst}_i \leftarrow (\mathcal{S}, h_1, \ldots, h_n, c_{i,i^*}^{(n)})$.
     (b) Check if $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{i,i^*}^{(n)}, k_i) = 1$. If so, let $b_i \leftarrow 1$ and $\pi_i \leftarrow k_i$. Otherwise, let $b_i \leftarrow 0$ and $\pi_i \leftarrow \perp$.
     (c) Output $(\mathsf{pst}_i, b_i, \pi_i)$
- *Verify*$(pp, i, \mathsf{pst}, \pi)$: parse pst as $(\mathcal{S}, h_1, \ldots, h_n, c)$ and π as k and do:
  1. If $i \notin \mathcal{S}$, $h_i = \mathsf{H}(k)$ and $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c, k) = 1$ then output 1. Otherwise, output 0.

Figure 6: ssle – A commit-and-shuffle SSLE protocol

Fortunately, when using the precommitment to randomness approach from Section 6, we can further reduce the size of commitments by setting the parameter ℓ to be just 1. The commitments will still be binding; a similar analysis to the proof of Theorem 6.2 shows that the vector $\boldsymbol{a}'$ remains close to uniformly random throughout the execution of the protocol.

We stress that the discussion above only pertains to the "complete shuffle" pattern used by the commit-and-shuffle protocol in Fig. 6. We cannot use a single $\boldsymbol{a}$ vector if using a shuffle pattern in which different commitments are randomized by different shufflers, as in [44, 47].

Adaptive adversaries. In the security experiments above, we considered a static adversary, that decides on the corrupted parties $\mathcal{I}$ in the beginning of the election protocol. We do this for simplicity, as the distributed shuffle itself is not the focus of this paper. However, the proof of Theorem 7.1 readily captures an adaptive adversary that may decide, in the beginning of each round, which parties to corrupt in this round. Then, c is the total number of corruptions the adversary is allowed. See, for example, [44, 47].

8 Proof of Well-Formed Shuffle

Our SSLE application of RRC schemes required that the shuffler in each round proves that they performed a "well-formed" shuffle; that is, the new list of commitments is obtained from the old one by re-randomizing and permuting. Moreover, the randomness used for re-randomization has to be consistent with the transcript of the protocol so far, including the outputs of the randomness beacon; in the language of Section 6 the shuffler has to prove that it used *admissible* randomness. For example, if $\mathsf{R}^{+}_{\mathsf{LWE}}$ is used, the shufflers need to prove that when re-randomizing, they used the sum of the matrix that they pre-committed to and the matrix sampled from the beacon. In this section, we present various approaches for realizing such a proof of a well-formed shuffle for our lattice-based RRC schemes.

The outline of this section is as follows. We start by presenting formal definitions for non-interactive statistically-binding commitment schemes, who will play an instrumental role in this section. Then, we discuss how to instantiate the proof of shuffle using generic proof systems. Finally, we present an adaptation of the recent protocol of Costa, Martínez, and Morillo [31] (following the classic Bayer and Groth protocol [16], a variant of which is used in Ethereum's Whisk protocol [64]).

8.1 Commitment Scheme

A non-interactive commitment scheme is defined by the following three algorithms:

- $\mathit{Setup}(1^\lambda) \to \mathit{pp}$: outputs public parameters pp. Implicit in these parameters is a message space $\mathcal{M}$. Typically, this will be $\mathbb{Z}_q$ and $\mathcal{R}_q$ for LWE-based and Ring-LWE based commitments, respectively. We add the subscripts $\mathsf{LWE}, \mathsf{RLWE}$ when referring to parameters for the integer and the ring-based schemes, e.g. $\mathcal{M}_{\mathsf{LWE}} = \mathbb{Z}_q$ and $\mathcal{M}_{\mathsf{RLWE}} = \mathcal{R}_q$,
- $\mathit{Commit}(\mathit{pp}, m; r) \to c$: this is a randomized poly-time algorithm that takes as input the public parameters pp, a message $m \in \mathcal{M}$, and randomness r, and returns a commitment c to m. We also define the notation $\mathit{Commit}(\mathit{pp}, m) \to (c, r)$ to denote the process of randomly sampling r and returning $\mathit{Commit}(\mathit{pp}, m; r)$ as the commitment and r as the opening,
- $\mathit{Verify}(\mathit{pp}, m, r, c) \to 0/1$: takes in pp, a message m, randomness r, and a commitment c, and outputs 0 or 1. The output 1 implies that c is a valid commitment on m with opening r.

The standard properties a commitment scheme should satisfy are correctness, hiding, and binding.

Correctness. A scheme is correct if *Verify* outputs 1 whenever the commitment and opening are computed by an honest party. More formally,

$$\Pr[\textit{Verify}(pp, m, r, c) = 1 : pp \leftarrow\$ \textit{Setup}(1^\lambda), m \leftarrow\$ \mathcal{M}, (c, r) \leftarrow\$ \textit{Commit}(pp, m)] = 1$$

Hiding. A commitment scheme is hiding if the commitment does not reveal anything about the committed value. More formally, we say that a scheme is computationally hiding if for all probabilistic polynomial time adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, there is a negligible function negl such that :

$$\Pr\left[b = b' \ : \ \begin{array}{c} pp \leftarrow\$ \textit{Setup}(1^\lambda), (m_0, m_1, \text{aux}) \leftarrow\$ \mathcal{A}_1(pp), b \leftarrow\$ \{0,1\}, \\ (c, r) \leftarrow\$ \textit{Commit}(pp, m_b), b' \leftarrow\$ \mathcal{A}_2(c, \text{aux}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Binding. A commitment scheme is binding if a commitment cannot be opened to different messages. A scheme is called statistically binding if this holds unconditionally, i.e. with overwhelming probability over the choice of the public parameters $pp \leftarrow\$ \textit{Setup}(1^\lambda)$, we have that, for all $m, m' \in \mathcal{M}$, for all r, r', and for all commitments c,

$$\textit{Verify}(pp, m, r, c) = \textit{Verify}(pp, m', r', c) = 1 \implies m = m'$$

In the following sections, we will use $\mathsf{C}_{\mathsf{LWE}}$ and $\mathsf{C}_{\mathsf{RLWE}}$ to refer to a commitment scheme with the above properties, with the message space being $\mathbb{Z}_q$ and $\mathcal{R}_q$ respectively.

8.2 The Relation That Needs Proving

Equipped with the above definition of commitment schemes, we can present the relation that the proof of shuffle needs to prove, when using our lattice-based RRC schemes. We begin by presenting the relation for our LWE-based scheme, and then turn to the ring setting.

Let us use $\{c_{r,ijk}\}_{i\in[t],j\in[m],k\in[m]}$ to denote the pre-commitment to the randomness sampled by the prover. For any $i \in [t], j, k \in [m]$, $c_{r,ijk}$ is the commitment to $\boldsymbol{R}_{i,jk}$. Let us use $\{(\boldsymbol{A}_1, \boldsymbol{U}_1), \ldots, (\boldsymbol{A}_t, \boldsymbol{U}_t)\}$ to denote the original list of commitments, and $\{(\boldsymbol{A}'_1, \boldsymbol{U}'_1), \ldots, (\boldsymbol{A}'_t, \boldsymbol{U}'_t)\}$ to denote the shuffled list. Let $\{\boldsymbol{R}'_i\}_{i\in[t]}$ be the randomness sampled from the beacon at the time of shuffle. The randomizer $\mathcal{P}$ wants to prove that they re-randomized and shuffled the list correctly using $\{\boldsymbol{R}_i + \boldsymbol{R}'_i\}_{i\in[t]}$. More formally, $\mathcal{P}$ holds a list of matrices $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_t$, the randomness $\{r_{i,jk}\}$ for the pre-commitments and a permutation π as witness, and wants to prove that

1. $\boldsymbol{A}'_i = (\boldsymbol{R}_i + \boldsymbol{R}'_i) \cdot \boldsymbol{A}_{\pi(i)}$ and $\boldsymbol{U}'_i = (\boldsymbol{R}_i + \boldsymbol{R}'_i) \cdot \boldsymbol{U}_{\pi(i)}$ for all $i \in [t]$.
2. $\mathsf{C}_{\mathsf{LWE}}.\textit{Verify}(pp_{\mathsf{LWE}}, \boldsymbol{R}_{i,jk}, r_{i,jk}, c_{r,ijk}) = 1$ for all $i \in [t], j, k \in [m]$

We formalize this with the following relation:

$$R_{\mathsf{LWE}} = \left\{ \left(\begin{array}{c} \{c_{r,ijk}\}_{i\in[t],j\in[m],k\in[m]}, \{(\boldsymbol{A}_i, \boldsymbol{U}_i)\}_{i\in[t]}, \\ \{(\boldsymbol{A}'_i, \boldsymbol{U}'_i)\}_{i\in[t]}, \{\boldsymbol{R}'_i\}_{i\in[t]} \\ (\{\boldsymbol{R}_i\}_{i\in[t]}, \{r_{i,jk}\}_{i\in[t],j\in[m],k\in[m]}, \pi) \end{array} \right), \ : \ \begin{array}{c} \boldsymbol{A}'_i = (\boldsymbol{R}_i + \boldsymbol{R}'_i) \cdot \boldsymbol{A}_{\pi(i)} \forall i \in [t] \\ \wedge \boldsymbol{U}'_i = (\boldsymbol{R}_i + \boldsymbol{R}'_i) \cdot \boldsymbol{U}_{\pi(i)} \forall i \in [t] \\ \wedge \forall i \in [t], j, k \in [m] : \\ \mathsf{C}_{\mathsf{LWE}}.\textit{Verify}(pp_{\mathsf{LWE}}, \boldsymbol{R}_{i,jk}, r_{i,jk}, c_{r,ijk}) = 1 \end{array} \right\}$$

We can write a similar relation for the Ring-LWE version. The main difference is that, assuming $t = cn$ for some constant $c > 1$ and $m = O(c \cdot (\log(q) + \frac{\lambda}{n}) + \frac{\kappa}{n})$, a single matrix $\boldsymbol{R}$ can be used to re-randomize all the commitments. We define the following relation to formalize this.

$$R_{\mathsf{RLWE}} = \left\{ \begin{pmatrix} \{c_{r,jk}\}, \{(\boldsymbol{a}_i, \boldsymbol{U}_i)\}_{i\in[t]}, \\ \{(\boldsymbol{a}'_i, \boldsymbol{U}'_i)\}_{i\in[t]}, \boldsymbol{R}' \\ (\boldsymbol{R}, \{r_{jk}\}_{j\in[m],k\in[m]}, \pi) \end{pmatrix}, \quad : \begin{array}{c} \boldsymbol{a}'_i = (\boldsymbol{R} + \boldsymbol{R}') \cdot \boldsymbol{a}_{\pi(i)} \forall i \in [t] \\ \wedge \boldsymbol{U}'_i = (\boldsymbol{R} + \boldsymbol{R}') \cdot \boldsymbol{U}_{\pi(i)} \forall i \in [t] \\ \wedge \forall j, k \in [m] : \\ \mathsf{C}_{\mathsf{RLWE}}.\mathit{Verify}(pp_{\mathsf{RLWE}}, \boldsymbol{R}_{jk}, r_{jk}, c_{r,jk}) = 1 \end{array} \right\}$$

Note that for correctness to hold, one also needs to check that $\boldsymbol{R}$ is low-norm. This can be formalized as another constraint in the relations. However, this can also be achieved by having each party checking after each shuffle that her commitment still appears on the list, and so we do not include it in the descriptions of the relations.

8.3 Constructing Proof of Shuffle

There are a few methods that can be used to prove the above relations.

8.3.1 Generic Post-Quantum Proof Systems

One option to construct a proof of shuffle for the relations above is to rely on generic non-interactive proof systems. Recent years have seen tremendous improvements in the succinctness and computational efficiency of such proof systems. This includes, in particular, proof systems that rely on assumptions that are assumed to be post-quantum secure. Examples include hash-based proof systems (e.g., [20, 18, 19, 41] and the references therein), lattice-based proof systems (e.g., [2, 53, 49, 6, 12] and the references therein), and proof systems that rely on MPC in the head techniques (e.g., [4, 23] and the references therein).

Such generic proof systems can be applicable to our needs, since the above relations can be expressed as simple systems of algebraic constraints. For completeness, we now sketch one way to convert these relations to algebraic constraints. We emphasize, however, that this presentation is merely meant to exemplify the simplicity of the constraints, and is by no means optimized for any specific proof system.

The constraint system. We describe how to convert the relation R_{LWE} to a constraint system and then briefly discuss how to extend this to R_{RLWE}. We can describe the permutation π as a list of t^2 boolean variables $\{p_{1,1}, \ldots, p_{t,t}\}$, where $p_{i,j}$ denotes whether $\pi(i)$ is j. Then, we can express $\boldsymbol{A}_{\pi(i)}$ as the following sum: $\boldsymbol{A}_{\pi(i)} = \Sigma_{j\in[t]} \boldsymbol{A}_j \cdot p_{i,j}$. To ensure that these variables are indeed boolean, we add the following constraint for all $i, j \in [t]$, $p_{i,j}^2 = p_{i,j}$. Next, to ensure that they represent a valid permutation, we add two sets of constraints: (i) For every $i \in [t]$, $\Sigma_{j\in[t]} p_{i,j} = 1$ and (ii) for every $j \in [t]$, $\Sigma_{i\in[t]} p_{i,j} = 1$. We now describe the relation with $\{p_{i,j}\}_{i\in[t],j\in[t]}$ included in the witness instead of π.

$$\forall i \in [t], j \in [m], k \in [m], \boldsymbol{A}'_{i,jk} = \Sigma_{l\in[m]}(\boldsymbol{R}_{i,jl} + \boldsymbol{R}'_{i,jl}) \cdot (\Sigma_{y\in[t]} \boldsymbol{A}_{y,lk} \cdot p_{i,y}) \tag{19}$$

$$\forall i \in [t], j \in [m], k \in [m], \boldsymbol{U}'_{i,jk} = \Sigma_{l\in[m]}(\boldsymbol{R}_{i,jl} + \boldsymbol{R}'_{i,jl}) \cdot (\Sigma_{y\in[t]} \boldsymbol{U}_{y,lk} \cdot p_{i,y}) \tag{20}$$

$$\forall i \in [t], j \in [m], k \in [m], \mathsf{C}_{\mathsf{LWE}}.\mathit{Verify}(pp_{\mathsf{LWE}}, \boldsymbol{R}_{i,jk}, r_{i,jk}, c_{r,ijk}) = 1 \tag{21}$$

$$\forall i \in [t], p_{i,j}^2 - p_{i,j} = 0 \tag{22}$$

$$\forall i \in [t], \Sigma_{j\in[t]} p_{i,j} = 1 \tag{23}$$

$$\forall j \in [t], \Sigma_{i\in[t]} p_{i,j} = 1 \tag{24}$$

The constraints in Eq. 21 still appear in their generic form, but they too translate to simple algebraic constraints when the commitment scheme $\mathsf{C}_{\mathsf{LWE}}$ is instantiated using a lattice-based commitment scheme with an algebraically simple verification procedure (e.g., [22, 15, 13]; see also Section 8.4 for an example of such a commitment scheme in the ring setting).

We can similarly construct a family of constraints for the Ring-LWE based scheme, wherein the main difference is that the same matrix $\boldsymbol{R}$ is used to re-randomize all the commitments.

8.4 A Bayer-Groth-Like Shuffle Argument for Ring-LWE

Another option is to construct a protocol that is specifically tailored to prove the above relations. In the context of shuffling discrete-log-based commitments or encryptions, there has been a long line of research on constructing shuffle proofs (see, for example, [52, 37, 42, 16, 26], and the many references therein). Recent works have extended the ideas from the discrete-log setting to the lattice-based setting [62, 31].

In particular, a recent work by Costa, Martínez, and Morillo [31] constructed a proof of shuffle for Ring-LWE-based *homomorphic* encryptions, by non-trivially importing ideas from the work of Bayer and Groth [16] in the discrete-log setting. It cannot be directly used to prove shuffle for our RRC commitments, since they are not homomorphic. We show that, in the case where we use a single large matrix to re-randomize the list of ring-based RRC commitments, we can tweak their shuffle proof to be compatible with our scheme. Recall that the prover uses $\mathsf{C}_{\mathsf{RLWE}}$ to commit to its witnesses for this proof. Similarly to the protocol of Costa et al. our protocol requires that in addition to the standard notions of correctness, hiding, and binding, the commitment scheme is also *additively homomorphic*.[6] A commitment scheme is called additively homomorphic in both the message and the randomness, if there are binary group operations $+, \otimes$ such that the following is true for all $\lambda \in \mathbb{N}$, and for all $pp \leftarrow\!\$\, \mathit{Setup}(1^\lambda)$, for all messages $m, m' \in \mathcal{M}$, for all randomness r, r':

$$\mathit{Commit}(pp, m; r) + \mathit{Commit}(pp, m'; r') = \mathit{Commit}(pp, m \otimes m'; r \otimes r')$$

We define $\otimes$ as ring addition, and $+$ is defined as vector addition. From this point on, we will use $+$ to refer to both these operations.[7]

We also require that $\mathsf{C}_{\mathsf{RLWE}}$ is equipped with two special types of zero-knowledge proof systems: A *proof of linear relations*, and a *proof of product relations*:

1. **Proof of Linear Relations:** We use $\mathsf{LR}_{\mathsf{RLWE}}$ to refer to a zero-knowledge proof of knowledge of openings $\{(m_i, r_i)\}_{i\in[L]}$ to commitments $\{c_i\}_{i\in[L]}$, where the messages $m_i \in \mathcal{M}$ additionally satisfy a linear relation, $\Sigma_{i\in[L]} \alpha_i m_i = S$. More formally, $\mathsf{LR}_{\mathsf{RLWE}}$ is a zero-knowledge proof of knowledge for the following relation:

[6] Costa et al. considered a specific commitment scheme that satisfies this property.

[7] Lattice-based commitment schemes are typically homomorphic only up to a bounded number of additions, determined by the parameters of the scheme. We abstract this fact out to simplify the presentation.

$$R_{\mathsf{L,RLWE}} = \left\{ \begin{array}{c} (pp, \{c_i\}_{i\in[L]}, \{\alpha_i\}_{i\in[L]}, S), \\ (\{m_i, r_i\}_{i\in[L]}) \end{array} : \begin{array}{c} \mathit{Verify}(pp, m_i, r_i, c_i) = 1 \; \forall \; i \in [L] \\ \wedge \Sigma_{i\in[L]} \alpha_i m_i = S \end{array} \right\}$$

We characterize $\mathsf{LR_{RLWE}}$ with the following functions :

- $\mathsf{LR_{RLWE}}.\mathit{Prove}((pp, \{c_i\}_{i\in[L]}, \{\alpha_i\}_{i\in[L]}, S), (\{m_i, r_i\}_{i\in[L]})) \rightarrow \Pi$: a randomized algorithm that returns a zero-knowledge proof for the relation,
- $\mathsf{LR_{RLWE}}.\mathit{Verify}((pp, \{c_i\}_{i\in[L]}, \{\alpha_i\}_{i\in[L]}, S), \Pi) \rightarrow \{0, 1\}$: outputs 1 if Π is a valid proof for the relation.

2. **Proof of Product Relations:** We use $\mathsf{PR_{RLWE}}$ to refer to a zero-knowledge proof of knowledge of openings $\{(m_i, r_i)\}_{i\in[L]}$ to commitments $\{c_i\}_{i\in[L]}$, where the messages m_i additionally satisfy a multiplicative relation, $\Pi_{i\in[L]} m_i = M$. More formally, $\mathsf{PR_{RLWE}}$ is a zero-knowledge proof of knowledge for the following relation:

$$R_{\mathsf{P,RLWE}} = \left\{ \begin{array}{c} (pp, \{c_i\}_{i\in[L]}, P), \\ (\{m_i, r_i\}_{i\in[L]}) \end{array} : \begin{array}{c} \mathit{Verify}(pp, m_i, r_i, c_i) = 1 \; \forall \; i \in [L] \\ \wedge \Pi_{i\in[L]} m_i = P \end{array} \right\}$$

Similar to $\mathsf{LR_{RLWE}}$, we characterize $\mathsf{PR_{RLWE}}$ with the following two functions:

- $\mathsf{PR_{RLWE}}.\mathit{Prove}((pp, \{c_i\}_{i\in[L]}, P), (\{m_i, r_i\}_{i\in[L]})) \rightarrow \Pi$: a randomized algorithm that returns a zero-knowledge proof for the product relation,
- $\mathsf{PR_{RLWE}}.\mathit{Verify}((pp, \{c_i\}_{i\in[L]}, P), \Pi) \rightarrow \{0, 1\}$: outputs 1 if Π is a valid proof for the relation.

Overview of the protocol. We start with a high level overview of [31], and explain where we deviate from it. In [31], the prover first sends a commitment to the permutation π. Specifically, it sends a list of t commitments :

$$\{c_{\pi(i)} \leftarrow\!\$ \; \mathsf{C_{RLWE}}.\mathit{Commit}(pp_{\mathsf{RLWE}}, \pi(i))\}_{i\in[t]}.$$

It then uses techniques from [32, 14] to generate a zero-knowledge proof Π_1 to show that the prover knows $\{m_i\}_{i\in[t]}$, that are valid openings for the commitments $\{c_{\pi(i)}\}_{i\in[t]}$, and for all i, m_i is a polynomial with degree $< n/2$. These techniques can be used because the commitment scheme $\mathsf{C_{RLWE}}$ is additively homomorphic and hiding, and hence can be characterized as a homomorphic one-way function. We use $\mathsf{SS_{RLWE}}$ to refer to such an argument system, and characterize it with two algorithms: (a) *Prove* which takes as input the public statement and the prover's witness and outputs a proof, and (b) *Verify* which takes as input the public statement and a proof, and outputs whether the proof is valid. Note that the *Prove* algorithm is actually a constant-round public-coin interactive protocol, but we refer to it as a function for simplicity in this overview (since the protocol is constant-round public-coin protocol, one can apply the Fiat-Shamir heuristic [35] to make it non-interactive).

Next, the verifier sends a random challenge $\alpha \leftarrow\!\$ \; \mathcal{S}$, where $\mathcal{S} \subset \mathcal{R}_q$ is the set of polynomials with degree $< \frac{n}{2}$. The prover responds with another list of commitments:

$$\{c_{\alpha^{\pi(i)}} \leftarrow\!\$ \; \mathsf{C_{RLWE}}.\mathit{Commit}(pp_{\mathsf{RLWE}}, \alpha^{\pi(i)})\}_{i\in[t]}$$

The prover then gets two more random challenges $\beta, \gamma \xleftarrow{\$} \mathcal{S}$ from the verifier, and generates a proof Π_2 that, for every $i \in [t]$, it knows valid openings $m_i \in \mathbb{Z}_q, m_{\alpha,i} \in \mathcal{R}_q$ for the commitments $c_{\pi(i)}, c_{\alpha^{\pi(i)}}$ that satisfy the following relation:

$$\Pi_{i\in[t]}(\beta m_i + m_{\alpha,i} - \gamma) = \Pi_{i\in[t]}(\beta i + \alpha^i - \gamma)$$

The prover can use the linear relations proof $\mathsf{LR}_{\mathsf{RLWE}}$ and the product relations proof $\mathsf{PR}_{\mathsf{RLWE}}$ to generate such a proof. As stated in [31], this proof would convince the verifier, that with overwhelming probability, $\{c_{\alpha^{\pi(i)}}\}_{i\in[t]}$ are indeed commitments to α with exponents 1 to t permuted in an order that was fixed by $\{c_{\pi(i)}\}_{i\in[t]}$ before α was chosen.

Next, the prover wants to prove that the new list was generated using the randomness consistent with the pre-commitments. In [31], they exploit the fact that the Ring-LWE encryptions (and re-encryptions) they are shuffling are homomorphic, allowing them to express the constraints as a single linear relation and simply use $\mathsf{LR}_{\mathsf{RLWE}}$ to generate a proof.

Unfortunately, since our RRC commitments are not homomorphic, we cannot express this constraint directly as linear or product relations. To solve this issue, we extend a technique from [64] by exploiting the fact that we use the same matrix $\boldsymbol{R}$ to re-randomize all our commitments. Let us use $\{c_{r,ij} \leftarrow\$ \mathsf{C}_{\mathsf{RLWE}}.\mathit{Commit}(\boldsymbol{R}_{ij})\}_{i,j\in[m]}$ to refer to the commitment to the re-randomization matrix $\boldsymbol{R}$, that the prover sends before the beginning of the shuffle protocol. Let us denote $\boldsymbol{a} = \Sigma_{i\in[t]}\alpha^i \cdot \boldsymbol{a}_i$ and $\boldsymbol{U} = \Sigma_{i\in[t]}\alpha^i \cdot \boldsymbol{U}_i$. The prover starts with sending values $\boldsymbol{a}' = (\boldsymbol{R} + \boldsymbol{R}') \cdot \boldsymbol{a}$ and $\boldsymbol{U}' = (\boldsymbol{R} + \boldsymbol{R}') \cdot \boldsymbol{U}$ to the verifier. It then uses $\mathsf{LR}_{\mathsf{RLWE}}$ to prove that it knows openings $\{m_{r,ij}\}_{i,j\in[m]}$ for commitments $\{c_{r,ij}\}_{i,j\in[m]}$ that satisfy the following linear relations (Π_3):

$$\forall i \in [m], \boldsymbol{a}'_i = \Sigma_{j\in[m]}(m_{r,ij} + \boldsymbol{R}'_{ij}) \cdot \boldsymbol{a}_j$$

$$\forall i \in [m], j \in [\ell], \boldsymbol{U}'_{ij} = \Sigma_{k\in[m]}(m_{r,ik} + \boldsymbol{R}'_{ik}) \cdot \boldsymbol{U}_{kj}$$

The idea of using the same randomness across all re-randomizations is inspired by the "proof of same-exponent relation" in [64].

Next, the prover again uses the linear relation proof $\mathsf{LR}_{\mathsf{RLWE}}$ to prove that it knows openings $\{m_{\alpha,i}\}_{i\in[t]}$ for commitments $\{c_{\alpha^{\pi(i)}}\}_{i\in[t]}$ which satisfy the following relations (Π_4):

$$\forall i \in [m], \boldsymbol{a}'_i = \Sigma_{j\in[t]} m_{\alpha,j} \cdot \boldsymbol{a}'_{j,i}$$

$$\forall i \in [m], j \in [\ell], \boldsymbol{U}'_{ij} = \Sigma_{k\in[t]} m_{\alpha,k} \cdot \boldsymbol{U}'_{k,ij}$$

The proofs Π_3, Π_4 together convince the verifier, that with overwhelming probability, $\{c_{r,ij}\}_{i,j\in[m]}$ are indeed commitments to the matrix $\boldsymbol{R}$, such that $\boldsymbol{a}'_i = (\boldsymbol{R} + \boldsymbol{R}') \cdot \boldsymbol{a}_{\pi(i)}$ and $\boldsymbol{U}' = (\boldsymbol{R} + \boldsymbol{R}') \cdot \boldsymbol{U}_{\pi(i)}$ for all i in the list $[t]$.

Note that we can use this proof with our LWE scheme as well, by using the same re-randomization matrix $\boldsymbol{R} + \boldsymbol{R}'$ for each RRC commitment. This, however, comes at the cost of scaling all the RRC commitments by a factor of t (the size of the shuffle) to make the unlinkability analysis go through (as mentioned in Section 7).

The protocol in detail and its analysis. We now formalize the full interactive proof Π_{shuffle} in Figure 7.

We analyse the security of our protocol when instantiated with the commitment scheme from [22]. To prove that the commitments $\{c_{\pi(i)}\}_{i\in[t]}$ are commitments to polynomials with degrees $< \frac{n}{2}$, we use the cut-and-choose based protocol $\mathsf{SS}_{\mathsf{RLWE}}$ from [14]. Specifically, the public parameters for the commitment scheme contain $\boldsymbol{a}, \boldsymbol{b}$ which are vectors of length k, uniformly randomly sampled from $\mathcal{R}_q^k$. To commit to a

$\underline{\mathcal{P}(\boldsymbol{R}, \pi)}$ $\underline{\mathcal{V}(pp_{\mathsf{RLWE}}, \{\boldsymbol{a}_i, \boldsymbol{U}_i\}_{i\in[t]}, \{\boldsymbol{a}'_i, \boldsymbol{U}'_i\}_{i\in[t]}, \boldsymbol{R}')}$

$\forall i \in [m], j \in [m]:$

$r_{r,ij} \xleftarrow{\$} \mathsf{R}_{\mathsf{RLWE}}, c_{r,ij} \leftarrow \mathit{Commit}(pp_{\mathsf{RLWE}}, \boldsymbol{R}_{ij}; r_{r,ij})$

$\xrightarrow{\{c_{r,ij}\}_{i\in[m],j\in[m]}}$

$\forall i \in \{1, \ldots, t\}:$

$r_i \xleftarrow{\$} \mathsf{R}_{\mathsf{RLWE}}, c_{\pi(i)} \leftarrow \mathit{Commit}(pp_{\mathsf{C},\mathsf{RLWE}}, \pi(i); r_i)$

$\Pi_1 \xleftarrow{\$} \mathsf{SS}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{\pi(i)}\}_{i\in[t]}), (\{m_i, r_i\}_{i\in[t]}))$

$\xrightarrow{\{c_{\pi(i)}\}_{i\in[t]}, \Pi_1}$

$\alpha \xleftarrow{\$} \mathcal{S}$

$\xleftarrow{\alpha}$

$\boldsymbol{a} \leftarrow \Sigma_{i\in[t]}\alpha^i \boldsymbol{a}_i, \boldsymbol{U} \leftarrow \Sigma_{i\in[t]}\alpha^i \boldsymbol{U}_i$

$\boldsymbol{a}' = (\boldsymbol{R} + \boldsymbol{R}')\boldsymbol{a}, \boldsymbol{U}' = (\boldsymbol{R} + \boldsymbol{R}')\boldsymbol{U}$

$\forall i \in \{1, \ldots, t\}:$

$r_{\alpha,i} \xleftarrow{\$} \mathsf{R}_{\mathsf{RLWE}}, c_{\alpha^{\pi(i)}} \leftarrow \mathit{Commit}(pp_{\mathsf{RLWE}}, \alpha^{\pi(i)}; r_{\alpha,i})$

$\xrightarrow{\boldsymbol{a}', \boldsymbol{U}', \{c_{\alpha^{\pi(i)}}\}_{i\in[t]}}$

$\beta, \gamma \xleftarrow{\$} \mathcal{S}$

$\xleftarrow{\beta, \gamma}$

$P \leftarrow \Pi_{i\in[t]}(\beta i + \alpha^i - \gamma)$

$\forall i \in [t]:$

$m_{i,L} \leftarrow \beta\pi(i) + \alpha^{\pi(i)} - \gamma$

$r_{i,L} \xleftarrow{\$} \mathsf{R}_{\mathsf{RLWE}}, c_{i,L} \leftarrow \mathit{Commit}(pp_{\mathsf{RLWE}}, m_{i,L}; r_{i,L})$

$\Pi_{2,i,L} \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{\pi(i)}, c_{\alpha^{\pi(i)}}, c_{i,L}\}, \{\beta, 1, -1\}, \gamma), (\{(m_i, r_i), (m_{\alpha,i}, r_{\alpha,i}), (m_{i,L}, r_{i,L})\}))$

$\Pi_{2,P} \xleftarrow{\$} \mathsf{PR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{i,L}\}_{i\in[t]}, P), (\{(m_{i,L}, r_{i,L})\}_{i\in[t]}))$

$\xrightarrow{\{(c''_i, \Pi_{2,i,L})\}_{i\in[t]}, \Pi_{2,P}}$

$\forall i \in [m]:$

$\Pi_{3,a,i} \leftarrow\!\$\, \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{r,ij}\}_{j\in[m]}, \{\boldsymbol{a}_j\}_{j\in[m]}, \boldsymbol{a}'_i), (\{(m_{r,ij}, r_{r,ij})\}_{j\in[m]}))$

$\forall i \in [m], j \in [\ell]:$

$\Pi_{3,u,ij} \leftarrow\!\$\, \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{r,ik}\}_{k\in[m]}, \{\boldsymbol{U}_{kj}\}_{k\in[m]}, \boldsymbol{U}'_{ij}), (\{(m_{r,ik}, r_{r,ik})\}_{k\in[m]}))$

$\xrightarrow{\{\Pi_{3,a,i}\}_{i\in[m]}, \{\Pi_{3,u,ij}\}_{i\in[m]j\in[\ell]}}$

$\forall i \in [m]:$

$\Pi_{4,a,i} \leftarrow\!\$\, \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{\alpha^{\pi(j)}}\}_{j\in[t]}, \{\boldsymbol{a}'_{ji}\}_{j\in[t]}, \boldsymbol{a}'_i), (\{(m'_j, r'_j)\}_{j\in[t]}))$

$\forall i \in [m], j \in [\ell]:$

$\Pi_{4,u,ij} \leftarrow\!\$\, \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{\alpha^{\pi(k)}}\}_{k\in[t]}, \{\boldsymbol{U}'_{k,ij}\}_{k\in[t]}, \boldsymbol{U}'_{ij}), (\{(m'_k, r'_k)\}_{k\in[t]}))$

$\xrightarrow{\{\Pi_{4,a,i}\}_{i\in[m]}, \{\Pi_{4,u,ij}\}_{i\in[m].j\in[\ell]}}$

If $\mathsf{SS}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{\pi(i)}\}_{i\in[t]}), \Pi_1) = 1$

$\wedge\ \forall i \in [t]\ \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{\pi(i)}, c_{\alpha^{\pi(i)}}, c_{i,L}\}, \{\beta, 1, -1\}, \gamma)\Pi_{2,i,L}) = 1$

$\wedge\ \mathsf{PR}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{i,L}\}_{i\in[t]}, P), \Pi_{2,P}) = 1$

$\wedge\ \forall i \in [m]\mathsf{LR}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{r,ij}\}_{j\in[m]}, \{\boldsymbol{a}_j\}_{j\in[m]}, \boldsymbol{a}'_i), \Pi_{3,a,i}) = 1$

$\wedge\ \forall i \in [m], j \in [\ell], \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{r,ik}\}_{k\in[m]}, \{\boldsymbol{U}_{kj}\}_{k\in[m]}, \boldsymbol{U}'_{ij}), \Pi_{3,u,ij}) = 1$

$\wedge\ \forall i \in [m], \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{\alpha^{\pi(j)}}\}_{j\in[t]}, \{\boldsymbol{a}'_{ji}\}_{j\in[t]}, \boldsymbol{a}'_i), \Pi_{4,a,i}) = 1$

$\wedge\ \forall i \in [m], j \in [\ell], \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Verify}((pp_{\mathsf{RLWE}}, \{c_{\alpha^{\pi(k)}}\}_{k\in[t]}, \{\boldsymbol{U}'_{k,ij}\}_{k\in[t]}, \boldsymbol{U}'_{ij}), \Pi_{4,u,ij}) = 1$

then, accept.

Figure 7: The proof of shuffle protocol Π_{shuffle}.

message $m \in \mathcal{R}_q$, we sample $r \xleftarrow{\$} \mathcal{R}_q$ and $\boldsymbol{e} \xleftarrow{\$} D^k_{\sigma_e}$ conditioned on $||\boldsymbol{e}||_\infty \leq n$, where D_{σ_e} is a discrete gaussian distribution over $\mathcal{R}_q$ with zero-mean and standard deviation σ_e. Then the commitment to m is $\boldsymbol{a}m + \boldsymbol{b}r + \boldsymbol{e}$ and the opening is $(m, r, \boldsymbol{e}, 1)$. The verify protocol accepts an opening $(m', r', \boldsymbol{e}', f')$ for a commitment $\boldsymbol{c}$ only if :

$$\boldsymbol{c}' = \boldsymbol{a}m' + \boldsymbol{b}r' + f'^{-1}\boldsymbol{e}' \ \wedge \ ||\boldsymbol{e}||_\infty \leq \lfloor \frac{n^X}{2} \rfloor \ \wedge \ ||f'||_\infty \leq 1 \ \wedge \ \mathsf{deg}(f') < \frac{n}{2}$$

Here, $X > 2$ is a parameter, such that $q \geq n^\gamma$ and $\gamma = 2X, k = 12$. We discuss how these parameters can be set in more detail later.

The paper [22] also presents protocols $\mathsf{LR}_{\mathsf{RLWE}}, \mathsf{PR}_{\mathsf{RLWE}}$ for proof of linear and product relations respectively, over messages committed using this scheme. Both these protocols are 3-round Sigma protocols, and use the same challenge space $\mathcal{C}$. They can be characterized by the following 3-move form:

- The prover runs $(\mathsf{st}, t) \xleftarrow{\$} \mathit{Message}_1(pp, x, w)$, where pp, x and w denote the public parameters, the statement and the witness of the relation. The output is a message t that the prover sends to the verifier, and st, which represents the prover's state.
- The verifier then samples a challenge $d \xleftarrow{\$} \mathcal{C}$ and sends d to the prover.
- The prover first runs rejection sampling, i.e. $b \xleftarrow{\$} \mathsf{Rej}(d, \mathsf{st})$. If $b = 1$, then the prover aborts. Otherwise, the prover runs $z \xleftarrow{\$} \mathit{Message}_3(d, \mathsf{st})$, and sends z to the verifier.
- The verifier runs $\mathit{Verify}(pp, x, (t, d, z))$ and returns 1 if it accepts the proof.

For simplicity, we use $\mathit{Prove}(pp, x, w)$ to refer to the first three steps together, and we use $\Pi \leftarrow (t, d, z)$ to refer to the full transcript as the proof. The paper describes both the protocols for proving linear and product relations over three messages, but they can be easily generalized to relations over L messages. We also set the norm bound in the *Verify* function to be $\lfloor \frac{n^X}{4} \rfloor$, where $X > 2$ is a parameter.

These protocols as well as $\mathsf{SS}_{\mathsf{RLWE}}$ require an auxiliary commitment scheme, which we instantiate using the scheme above. Note that the scheme can similarly be instantiated using the commitment scheme from [15] and $\mathsf{SS}_{\mathsf{RLWE}}$ from [32]. We leave this as another future direction.

Reducing the number of rounds in Π_{shuffle} to a constant. We described our protocol above as a sequential composition of $\mathsf{poly}(t, m, l)$ number of $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ proofs for simplicity. However, since these are Sigma protocols, we can invoke the standard transformation for the conjunction of relations: the prover would execute all the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ protocols in parallel. Moreover, since the challenge space is the same for $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$, the verifier can use the same challenge for all the proofs. We now describe the 3-round protocol combining all the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ proofs in Π_{shuffle}:

- The prover computes the first message for all the proofs, i.e.

$$(\mathsf{st}_{2,i,L}, t_{2,i,L}) \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_1 \left(\begin{array}{c} (pp_{\mathsf{RLWE}}, \{c_{\pi(i)}, c_{\alpha^{\pi(i)}}, c_{i,L}\}, \{\beta, 1, -1\}, \gamma), \\ (\{(m_i, r_i), (m_{\alpha,i}, r_{\alpha,i}), (m_{i,L}, r_{i,L})\}) \end{array} \right) \forall i \in [t]$$

$$(\mathsf{st}_{2,P}, t_{2,P}) \xleftarrow{\$} \mathsf{PR}_{\mathsf{RLWE}}.\mathit{Message}_1((pp_{\mathsf{RLWE}}, \{c_{i,L}\}_{i\in[t]}, P), (\{(m_{i,L}, r_{i,L})\}_{i\in[t]}))$$

$$(\mathsf{st}_{3,a,i}, t_{3,a,i}) \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_1((pp_{\mathsf{RLWE}}, \{c_{r,ij}\}_{j\in[m]}, \{\boldsymbol{a}_j\}_{j\in[m]}, \boldsymbol{a}'_i), (\{(m_{r,ij}, r_{r,ij})\}_{j\in[m]})) \forall i \in [m]$$

$$(\mathsf{st}_{3,u,ij}, t_{3,u,ij}) \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_1 \left(\begin{array}{c} (pp_{\mathsf{RLWE}}, \{c_{r,ik}\}_{k\in[m]}, \{\boldsymbol{U}_{kj}\}_{k\in[m]}, \boldsymbol{U}'_{ij}), \\ (\{(m_{r,ik}, r_{r,ik})\}_{k\in[m]}) \end{array} \right) \forall i \in [m], j \in [\ell]$$

$$(\mathsf{st}_{4,a,i}, t_{4,a,i}) \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_1((pp_{\mathsf{RLWE}}, \{c_{\alpha^{\pi(j)}}\}_{j\in[t]}, \{\boldsymbol{a}'_{ji}\}_{j\in[t]}, \boldsymbol{a}'_i), (\{(m'_j, r'_j)\}_{j\in[t]}))\forall i \in [m]$$

$$(\mathsf{st}_{4,u,ij}, t_{4,u,ij}) \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Prove}((pp_{\mathsf{RLWE}}, \{c_{\alpha^{\pi(k)}}\}_{k\in[t]}, \{\boldsymbol{U}'_{k,ij}\}_{k\in[t]}, \boldsymbol{U}'_{ij}), (\{(m'_k, r'_k)\}_{k\in[t]}))\forall i \in [m], j \in [\ell]$$

The prover then sends all the messages to the verifier, i.e. $\{t_{2,i,L}\}_{i\in[t]}, t_{2,P}, \{t_{3,a,i}\}_{i\in[m]}, \{t_{3,u,ij}\}_{i\in[m],j\in\ell}$, $\{t_{4,a,i}\}_{i\in[m]}, \{t_{4,u,ij}\}_{i\in[m],j\in\ell}$.

- The verifier samples a random challenge $d \xleftarrow{\$} \mathcal{C}$, and sends it to the prover.
- The prover then runs rejection sampling for all the protocols. Specifically,

$$b \xleftarrow{\$} \vee_{i\in[t]} \mathsf{LR}_{\mathsf{RLWE}}.\mathsf{Rej}(d, \mathsf{st}_{2,i,L}) \vee \mathsf{PR}_{\mathsf{RLWE}}.\mathsf{Rej}(d, \mathsf{st}_{2,P}) \tag{25}$$

$$\vee_{i\in[m]} \mathsf{LR}_{\mathsf{RLWE}}.\mathsf{Rej}(d, \mathsf{st}_{3,a,i}) \tag{26}$$

$$\vee_{i\in[m],j\in[\ell]} \mathsf{LR}_{\mathsf{RLWE}}.\mathsf{Rej}(d, \mathsf{st}_{3,u,ij}) \tag{27}$$

$$\vee_{i\in[m]} \mathsf{LR}_{\mathsf{RLWE}}.\mathsf{Rej}(d, \mathsf{st}_{4,a,i}) \tag{28}$$

$$\vee_{i\in[m],j\in[\ell]} \mathsf{LR}_{\mathsf{RLWE}}.\mathsf{Rej}(d, \mathsf{st}_{4,u,ij}) \tag{29}$$

If $b = 1$ then the prover aborts. Otherwise, the prover computes the final message for all the protocols as follows:

$$z_{2,i,L} \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_3(d, \mathsf{st}_{2,i,L})\forall i \in [t]$$

$$z_{2,P} \xleftarrow{\$} \mathsf{PR}_{\mathsf{RLWE}}.\mathit{Message}_3(d, \mathsf{st}_{2,P})$$

$$z_{3,a,i} \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_3(d, \mathsf{st}_{3,a,i})\forall i \in [m]$$

$$z_{3,u,ij} \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_3(d, \mathsf{st}_{3,u,ij})\forall i \in [m], j \in [\ell]$$

$$z_{4,a,i} \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_3(d, \mathsf{st}_{4,a,i})\forall i \in [m]$$

$$z_{4,u,ij} \xleftarrow{\$} \mathsf{LR}_{\mathsf{RLWE}}.\mathit{Message}_3(d, \mathsf{st}_{4,u,ij})\forall i \in [m], j \in [\ell]$$

The prover then sends all these messages $\{z_{2,i,L}\}_{i\in[t]}, z_{2,P}, \{z_{3,a,i}\}_{i\in[m]}, \{z_{3,u,ij}\}_{i\in[m],j\in\ell}, \{z_{4,a,i}\}_{i\in[m]}$, $\{z_{4,u,ij}\}_{i\in[m],j\in\ell}$ to the verifier.

By combining the constant-round protocol $\mathsf{SS}_{\mathsf{RLWE}}$ and the above 3-round sigma protocol, we get that Π_{shuffle} is a constant-round protocol. Hence, it can be made non-interactive using the Fiat-Shamir heuristic [35, 33].

Theorem 8.1 below proves completeness, soundness and honest-verifier zero-knowledge for our protocol with the aforementioned commitment scheme.

Theorem 8.1. *The proof of shuffle protocol $\Pi_{shuffle}$ is an honest verifier zero-knowledge proof of knowledge for the relation R_{RLWE}.*

We prove the theorem using the following claims, which prove the security of the underlying protocols, i.e. $\mathsf{SS}_{\mathsf{RLWE}}, \mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$.

Claim 8.2 (Theorem 2, [14])**.** *Let k' be a statistical security parameter, such that $t > k' \log(k')$. Then the $\mathsf{SS}_{\mathsf{RLWE}}$ protocol is an interactive honest-verifier zero-knowledge proof of the following relation with knowledge error $2^{-k'+1}$:*

$$R_{\mathsf{SS}} = \left\{ \begin{array}{c} (\{\boldsymbol{c}_i\}_{i\in[t]}, pp_{\mathsf{RLWE}} = \{\boldsymbol{a}, \boldsymbol{b}\}) \\ \{(m_i, r_i, \boldsymbol{e}_i)\}_{i\in[t]} \end{array} \quad : \quad \begin{array}{c} \boldsymbol{c}_i = \boldsymbol{a}m_i + \boldsymbol{b}r_i + \boldsymbol{e}_i \; \forall \; i \in [t] \\ \mathsf{deg}(m_i) < \frac{n}{2} \; \forall \; i \in [t] \\ ||\boldsymbol{e}_i|| \leq \lfloor \frac{n^X}{2} \rfloor \; \forall \; i \in [t] \end{array} \right\}$$

More specifically, $\mathsf{SS}_{\mathsf{RLWE}}$ has the following properties:

Completeness. *If $\mathcal{P}, \mathcal{V}$ are honest, then the protocol succeeds with probability at least $1 - 2^{O(k')}$.*

Knowledge Soundness. *For every deterministic prover $\hat{\mathcal{P}}$ that makes the verifier accept with probability $p > 2^{-k'+1}$, there exists an extractor $\mathcal{E}_{\mathsf{SS}}$ that outputs t values $(m'_i, r'_i, \boldsymbol{e}'_i)$ such that Verify$(pp_{\mathsf{RLWE}}, m'_i, r'_i, \boldsymbol{e}'_i, 1, \boldsymbol{c}_i) = 1$, $\mathsf{deg}(m'_i) < \frac{n}{2}$ and $||\boldsymbol{e}'_i|| \leq O((2k'+1)^{\log(k')/2} \cdot t \cdot (k) \cdot n) \leq \lfloor \frac{n^X}{2} \rfloor$ for all $i \in [t]$, except with probability $2^{-O(k')}$. The expected time of the extractor is $\mathsf{poly}(t, k, n, \log(q), k')/p$.*

Zero-Knowledge. *There exists a probabilistic polynomial time algorithm $\mathsf{Sim}_{\mathsf{SS,RLWE}}$, which takes as input the public statement $((\{\boldsymbol{c}_i\}_{i\in[t]}, pp_{\mathsf{RLWE}} = \{\boldsymbol{a}, \boldsymbol{b}\})$, and outputs a transcript that is computationally indistinguishable from a transcript generated by a real protocol execution by honest $\mathcal{P}, \mathcal{V}$.*

Claim 8.3 (Theorem 4.5, [22])**.** *The $\mathsf{LR}_{\mathsf{RLWE}}$ protocol is an honest-verifier zero-knowledge proof of knowledge with knowledge error $\frac{1}{\binom{n/2}{k''}}$ for the following relation*[8]*.*

$$R_{\mathsf{L,RLWE}} = \left\{ \begin{array}{c} (\{\boldsymbol{a}, \boldsymbol{b}\}, \{\boldsymbol{c}_i\}_{i\in[L]}, \{\alpha_i\}_{i\in[L]}, S) \\ (\{(m_i, r_i, \boldsymbol{e}_i, f_i)\}_{i\in[L]}) \end{array} \quad : \quad \begin{array}{c} \boldsymbol{c}_i = \boldsymbol{a}m_i + \boldsymbol{b}r_i + f_i^{-1}\boldsymbol{e}_i \wedge ||f_i||_\infty \leq 1 \forall i \in [L] \\ ||\boldsymbol{e}_i||_\infty \leq \lfloor \frac{n^X}{2} \rfloor \wedge \mathsf{deg}(f_i) < \frac{n}{2} \forall i \in [L] \\ \Sigma_{i\in[L]} \alpha_i m_i = S \end{array} \right\}$$

Specifically, the protocol has the following properties:

Completeness. *An honest prover $\mathcal{P}$ responds with probability $\frac{1}{M^L}$, where M is a constant from rejection sampling. When it does respond, an honest $\mathcal{V}$ accepts the proof with overwhelming probability.*

Special Soundness. *There exists a PPT algorithm $\mathcal{E}_{\mathsf{LR}}$, which takes as input two accepting transcripts with different challenges, and outputs $\{m'_i, r'_i, \boldsymbol{e}'_i, f'_i\}_{i\in[L]}$, such that Verify$(\{\boldsymbol{a}, \boldsymbol{b}\}, m'_i, r'_i, \boldsymbol{e}'_i, f'_i, \boldsymbol{c}_i) = 1 \forall$ $i \in [L]$, and $\Sigma_{i\in[L]} \alpha_i m'_i = S$.*

Honest-verifier Zero-knowledge. *There exists a PPT algorithm $\mathsf{Sim}_{\mathsf{LR,RLWE}}$, which takes as input the public statement i.e. $\{\boldsymbol{a}, \boldsymbol{b}\}, \{\boldsymbol{c}_i\}_{i\in[L]}, \{\alpha_i\}_{i\in[L]}, S$, and outputs a transcript that is computationally indistinguishable from a transcript generated by a real protocol execution by honest $\mathcal{P}, \mathcal{V}$.*

Claim 8.4 (Theorem 4.6, [22])**.** *The $\mathsf{PR}_{\mathsf{RLWE}}$ protocol is an honest-verifier zero-knowledge proof of knowledge with knowledge error $\frac{2}{\binom{n/2}{k''}}$ for the following relation.*

$$R_{\mathsf{P,RLWE}} = \left\{ \begin{array}{c} (\{\boldsymbol{a}, \boldsymbol{b}\}, \{\boldsymbol{c}_i\}_{i\in[L]}, P) \\ (\{(m_i, r_i, \boldsymbol{e}_i, f_i)\}_{i\in[L]}) \end{array} \quad : \quad \begin{array}{c} \boldsymbol{c}_i = \boldsymbol{a}m_i + \boldsymbol{b}r_i + f_i^{-1}\boldsymbol{e}_i \wedge ||f_i||_\infty \leq 1 \forall i \in [L] \\ ||\boldsymbol{e}_i||_\infty \leq \lfloor \frac{n^X}{2} \rfloor \wedge \mathsf{deg}(f_i) < \frac{n}{2} \forall i \in [L] \\ \Pi_{i\in[L]} m_i = P \end{array} \right\}$$

Specifically, the protocol has the following properties:

Completeness. *An honest prover $\mathcal{P}$ responds with probability $\frac{1}{M^{2L}}$, where M is a constant from rejection sampling. When it does respond, an honest $\mathcal{V}$ accepts the proof with overwhelming probability.*

Special Soundness. *There exists a PPT algorithm $\mathcal{E}_{\mathsf{PR}}$, which takes as input two accepting transcripts with different challenges, and outputs $\{m'_i, r'_i, \boldsymbol{e}'_i, f'_i\}_{i\in[L]}$, such that Verify$(\{\boldsymbol{a}, \boldsymbol{b}\}, m'_i, r'_i, \boldsymbol{e}'_i, f'_i, \boldsymbol{c}_i) = 1 \forall$ $i \in [L]$, and $\Pi_{i\in[L]} m'_i = P$.*

Honest-verifier Zero-knowledge. *There exists a PPT algorithm $\mathsf{Sim}_{\mathsf{PR,RLWE}}$, which takes as input the public statement i.e. $\{\boldsymbol{a}, \boldsymbol{b}\}, \{\boldsymbol{c}_i\}_{i\in[L]}, P$, and outputs a transcript that is computationally indistinguishable from a transcript generated by a real protocol execution by honest $\mathcal{P}, \mathcal{V}$.*

[8] Here, k'' is a parameter that characterizes the challenge space $\mathcal{C}$

We can now prove Theorem 8.1.

Proof of Theorem 8.1. **Completeness.** In the case that the honest prover does not abort, an honest verifier will a) accept the proof Π_1 by completeness of $\mathsf{SS}_{\mathsf{RLWE}}$, b) accept $\Pi_{2,P}$ by completeness of $\mathsf{PR}_{\mathsf{RLWE}}$, and lastly, accept the proofs $\{\Pi_{2,i,L}\}_{i\in[t]}, \{\Pi_{3,a,i}\}_{i\in[m]}, \{\Pi_{3,u,ij}\}_{i\in[m],j\in[\ell]}, \{\Pi_{4,a,i}\}_{i\in[m]}, \{\Pi_{4,u,ij}\}_{i\in[m],j\in[\ell]}$ because $\mathsf{LR}_{\mathsf{RLWE}}$ is complete. Hence, the verifier will accept the proof of shuffle with overwhelming probability, implying that Π_{shuffle} is complete.

Bounding the abort probability. The protocols $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ from [22] use rejection sampling.

We now analyse the probability that the prover does not abort:

$$\left(1-\frac{1}{2^{O(k')}}\right)\cdot\left(\left(\frac{1}{M}\right)^3\right)^t\cdot\left(\frac{1}{M}\right)^t\cdot\left(\left(\frac{1}{M}\right)^m\right)^m\cdot\left(\left(\frac{1}{M}\right)^m\right)^{m\ell}\cdot\left(\left(\frac{1}{M}\right)^t\right)^m\cdot\left(\left(\frac{1}{M}\right)^t\right)^{m\ell}$$

Here, M is a rejection sampling parameter used within all the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ protocols. As stated in [48], $M = e^{\frac{12}{\alpha}+\frac{1}{2\alpha^2}}$, where α is a function of σ_η, a parameter controlling the standard deviation of the messages output by the prover. Specifically, $\alpha = \frac{\sigma_\eta}{nk''\sqrt{k}}$. To ensure that our protocol succeeds with a constant probability, we set $\sigma_\eta = 11(4t + m^2 + m^2\ell + mt + m\ell t)\cdot nk''\sqrt{k}$, and choose any X such that $n^X \geq 4\sigma_\eta\sqrt{2k}$. Note that this implies that our commitment scheme has a larger multiplicative overhead in size as compared to the parameters in [22]. This is not unique to our protocol, but is in fact a standard phenomenon that arises when composing multiple sub-protocols, each of which uses rejection sampling (such as the protocols in [31, 7]).

Honest-verifier zero-knowledge. We build a simulator $\mathsf{Sim}_{\text{shuffle}}$ for Π_{shuffle}. Specifically, given the public statement $pp_{\mathsf{RLWE}}, \{\boldsymbol{a}_i, \boldsymbol{U}_i\}_{i\in[t]}, \{\boldsymbol{a}'_i, \boldsymbol{U}'_i\}_{i\in[t]}$, the simulator does the following.

1. Sample the challenges $\hat{\alpha}, \hat{\beta}, \hat{\gamma} \leftarrow\!\!\$\, \mathcal{S}$. Observe that α, β, γ are sampled randomly from $\mathcal{S}$ in the real transcript as well, meaning, $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ is distributed identically in the real and the simulated transcript.

2. For all $i \in [m], j \in [m]$, set $\hat{c}_{r,ij}$ to be a randomized commitment to zero. Specifically, we set $\hat{c}_{r,ij} \xleftarrow{\$} \mathit{Commit}(pp_{\mathsf{RLWE}}, 0)$. Since the commitment scheme is computationally hiding, $\{\hat{c}_{r,ij}\}_{i\in[m],j\in[m]} \approx_c \{c_{r,ij}\}_{i\in[m],j\in[m]}$.

3. For all $i \in [t]$, set $\hat{c}_{\pi(i)}$, $\hat{c}_{\alpha^{\pi(i)}}$ and $\hat{c}_{i,L}$ to be random commitments to zero, i.e. $\hat{c}_{\pi(i)}, \hat{c}_{\alpha^{\pi(i)}}, \hat{c}_{i,L} \xleftarrow{\$} \mathit{Commit}(pp_{\mathsf{RLWE}}, 0)$. Again, since the commitment scheme is computationally hiding, $\{\hat{c}_{\pi(i)}, \hat{c}_{\alpha^{\pi(i)}}, \hat{c}_{i,L}\}_{i\in[t]} \approx_c \{c_{\pi(i)}, c_{\alpha^{pi(i)}}, c_{i,L}\}_{i\in[t]}$

4. Run $\mathsf{Sim}_{\mathsf{SS},\mathsf{RLWE}}(pp_{\mathsf{RLWE}}, \{\hat{\mathsf{c}}_{\pi(\mathsf{i})}\}_{\mathsf{i}\in[\mathsf{t}]})$ to get a simulated $\hat{\Pi}_1$. As per Claim 8.2, the $\mathsf{SS}_{\mathsf{RLWE}}$ protocol is computational zero-knowledge, hence we get $\hat{\Pi}_1 \approx_c \Pi_1$.

5. For every $i \in [t]$, run $\mathsf{Sim}_{\mathsf{LR},\mathsf{RLWE}}$ to simulate $\hat{\Pi}_{2,i,L}$. Run $\mathsf{Sim}_{\mathsf{PR},\mathsf{RLWE}}$ to simulate $\hat{\Pi}_{2,P}$. As stated in Claims 8.3 and 8.4, the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ protocols are both computational zero-knowledge, hence no efficient adversary can distinguish between the simulated and the real proofs. Specifically, $\{\hat{\Pi}_{2,i,L}\}_{i\in[t]} \approx_c \{\Pi_{2,i,L}\}_{i\in[t]}$ and $\hat{\Pi}_{2,P} \approx_c \Pi_{2,P}$. We note that the simulators for these protocols, specifically $\mathsf{Sim}_{\mathsf{LR},\mathsf{RLWE}}$ and $\mathsf{Sim}_{\mathsf{PR},\mathsf{RLWE}}$ all generate a valid transcript even if the statement is false.

6. Compute $\hat{\boldsymbol{a}}, \hat{\boldsymbol{U}}$ similar to the original protocol, i.e. $\hat{\boldsymbol{a}} = \Sigma_{i\in[t]}\hat{\alpha}^i \boldsymbol{a}_i$ and $\hat{\boldsymbol{U}} = \Sigma_{i\in[t]}\hat{\alpha}^i \boldsymbol{U}_i$. Sample $\hat{\boldsymbol{a}}' \xleftarrow{s} \mathcal{R}_q^m$ and $\hat{\boldsymbol{U}}' \xleftarrow{s} \mathcal{R}_q^{m\times l}$ uniformly randomly. By Lemma 2.2, we get that the statistical distance between $(\boldsymbol{a}', \boldsymbol{U}')$ (computed as in the real protocol execution) and uniformly random variables $(\hat{\boldsymbol{a}}', \hat{\boldsymbol{U}}')$ is bounded by $\frac{m}{2}\sqrt{\left(1+\left(\frac{q}{2^m}\right)^{n/2}\right)^{2(t\ell+1)} - 1}$. Since we set $\ell = O(\log(q) + \frac{\lambda}{n})$ and $m = O(\frac{t}{n}\cdot(\log(q)+\frac{\lambda}{n}))$, this distance is negligible. In other words, $(\hat{\boldsymbol{a}}', \hat{\boldsymbol{U}}') \approx_s (\boldsymbol{a}', \boldsymbol{U}')$.

7. Lastly, for all $i \in [m], j \in [\ell]$, we use the simulator for linear relations proof $\mathsf{Sim}_{\mathsf{LR,RLWE}}$ to simulate $\hat{\Pi}_{3,a,i}, \hat{\Pi}_{4,a,i}, \hat{\Pi}_{3,u,ij}$ and $\hat{\Pi}_{4,u,ij}$. Since $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ are both computational zero-knowledge protocols, we get that $\{\hat{\Pi}_{3,a,i}, \hat{\Pi}_{4,a,i}\}_{i\in[m]} \approx_c \{\Pi_{3,a,i}, \Pi_{4,a,i}\}_{i\in[m]}$ and $\{\hat{\Pi}_{3,u,ij}, \hat{\Pi}_{4,u,ij}\}_{i\in[m],j\in[\ell]} \approx_c \{\Pi_{3,u,ij}, \Pi_{4,u,ij}\}_{i\in[m],j\in[\ell]}$.

The final transcript $(\{\hat{c}_{r,ij}\}_{i\in[m],j\in[m]}, \{\hat{c}_{\pi(i)}\}_{i\in[t]}, \hat{\Pi}_1, \hat{\alpha}, \{\hat{c}_{\alpha^{\pi(i)}}\}_{i\in[t]}, \hat{\beta}, \hat{\gamma}, \{\hat{c}_{i,L}, \hat{\Pi}_{2,i,L}\}_{i\in[t]}, \hat{\Pi}_{2,P}, \hat{\boldsymbol{a}}', \hat{\boldsymbol{U}}', \{\hat{\Pi}_{3,a,i}\}_{i\in[m]}, \{\hat{\Pi}_{3,u,ij}\}_{i\in[m],j\in[\ell]}, \{\hat{\Pi}_{4,a,i}\}_{i\in[m]}, \{\hat{\Pi}_{4,u,ij}\}_{i\in[m],j\in[\ell]})$ is computationally indistinguishable from a real transcript, meaning that Π_{shuffle} is computational honest-verifier zero-knowledge.

Knowledge Soundness. Let us consider a malicious prover $\mathcal{P}^*$ that convinces the verifier with probability $p > 2^{-k'+1}$. We construct an extractor $\mathcal{E}_{\text{shuffle}}$ as follows.

- We first construct a malicious prover for the $\mathsf{SS}_{\mathsf{RLWE}}$ protocol $\mathcal{P}^*_{\mathsf{SS}}$ from $\mathcal{P}^*$. This prover simply runs $\mathcal{P}^*$ and simulates all the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ protocols by sampling a random challenge.
- We then run $\mathcal{E}_{\mathsf{SS}}$ with $\mathcal{P}^*_{\mathsf{SS}}$, which will extract pre-images $\{(m'_i, r'_i, \boldsymbol{e}'_i)\}$ for $\{c_{\pi(i)}\}_{i\in[t]}$. By Claim 8.2, for all $i \in [t]$, $\mathsf{deg}(m'_i) < \frac{n}{2}$, $\mathit{Verify}(pp_{\mathsf{RLWE}}, m'_i, r'_i, \boldsymbol{e}'_i, 1, \boldsymbol{c}_{\pi(i)}) = 1$ and $||\boldsymbol{e}'_i|| \leq O((2k'+1)^{\log(k')/2} \cdot t \cdot k \cdot n)$.
- Next, we run $\mathcal{P}^*$ once with randomly sampled challenge. Then, we rewind the prover to just after its first message, sample fresh challenges and re-run $\mathcal{P}^*$ with these challenges. The probability that we get two full transcripts, with a different challenge for the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ proofs (recall that we are sampling just one challenge for all of these protocol instances) is $p^2 - \frac{1}{|\mathcal{C}|}$ [17].
- If we do indeed get two transcripts where the challenge for all the $\mathsf{LR}_{\mathsf{RLWE}}$ and $\mathsf{PR}_{\mathsf{RLWE}}$ proofs is different, we can run the extractor $\mathcal{E}_{\mathsf{LR}}$ on transcripts for $\{\Pi_{2,i,L}\}_{i\in[t]}$ to extract pre-images $\{(m''_i, r''_i, \boldsymbol{e}''_i, f''_i), (m'_{\alpha,i}, r'_{\alpha,i}, \boldsymbol{e}'_{\alpha,i}, f'_{\alpha,i}), (m'_{i,L}, r'_{i,L}, \boldsymbol{e}'_{i,L}, f'_{i,L})\}$ for $\{c_{\pi(i)}, c_{\alpha^{\pi(i)}}, c_{i,L}\}_{i\in[t]}$, such that for all $i \in [t]$,

$$\mathit{Verify}(pp_{\mathsf{RLWE}}, m''_i, r''_i, \boldsymbol{e}''_i, f''_i, c_{\pi(i)}) = 1$$
$$\mathit{Verify}(pp_{\mathsf{RLWE}}, m'_{\alpha,i}, r'_{\alpha,i}, \boldsymbol{e}'_{\alpha,i}, f'_{\alpha,i}, c_{\alpha^{\pi(i)}}) = 1$$
$$\mathit{Verify}(pp_{\mathsf{RLWE}}, m'_{i,L}, r'_{i,L}, \boldsymbol{e}'_{i,L}, f'_{i,L}, c_{i,L}) = 1$$
$$m'_{i,L} = \beta m'_i + m'_{\alpha,i} - \gamma$$

  Note that since the commitment scheme is perfectly binding, $m'_i = m''_i \,\forall\, i \in [t]$.
- Similarly we can run $\mathcal{E}_{\mathsf{PR}}$ on the two transcripts for $\Pi_{2,P}$ to extract $\{(m''_{i,L}, r''_{i,L}, \boldsymbol{e}''_{i,L}, f''_{i,L})\}_{i\in[t]}$ such that,

$$\mathit{Verify}(pp_{\mathsf{RLWE}}, m''_{i,L}, r''_{i,L}, \boldsymbol{e}''_{i,L}, f''_{i,L}, c_{i,L}) = 1$$
$$\Pi_{i\in[t]} m'_{i,L} = P = \Pi_{i\in[t]}(\beta i + \alpha^i - \gamma)$$

Since the commitment scheme is perfectly binding (with overwhelming probability over its setup algorithm), the pre-images for $\{c_{i,L}\}_{i\in[t]}$ found in this and the last step are equal, i.e. $m'_{i,L} = m''_{i,L}$ for all $i \in [t]$.

As discussed in [31], the above equations combined with the fact that $\forall\, i \in [t], \mathsf{deg}(m'_i) < \frac{n}{2}$, imply that $m'_i = \pi'(i)$ and $m'_{\alpha,i} = \alpha^{\pi'(i)}$ for some permutation π' that the prover committed to, before the verifier chose α.

- We can similarly run $\mathcal{E}_{\mathsf{LR}}$ on the transcripts to extract witnesses for the proofs $\{\Pi_{3,a,i}\}_{i\in[m]}$. Specifically, we would get $\{(m'_{r,ij}, r'_{r,ij}, \boldsymbol{e}'_{r,ij}, f'_{r,ij}\}_{i,j\in[m]}$ such that :

$$\textit{Verify}(pp_{\mathsf{RLWE}}, m'_{r,ij}, r'_{r,ij}, \boldsymbol{e}'_{r,ij}, f'_{r,ij}, c_{r,ij})) = 1$$
$$\Sigma_{j\in[m]} m'_{r,ij}\boldsymbol{a}_j = \boldsymbol{a}'_i \forall i \in [m]$$

- Using the two transcripts for the proof $\Pi_{3,u,ij}$ for any $i, j \in [m]$, we can extract witnesses $\{(m''_{r,ik,(j)}, r''_{r,ik,(j)}, \boldsymbol{e}''_{r,ik,(j)}, f''_{r,ik,(j)})\}_{k\in[m]}$ such that :

$$\textit{Verify}(pp_{\mathsf{RLWE}}, m''_{r,ik,(j)}, r''_{r,ik,(j)}, \boldsymbol{e}''_{r,ik,(j)}, f''_{r,ik,(j)}, c_{r,ik}) = 1$$
$$\Sigma_{k\in[m]} m''_{r,ik,(j)}\boldsymbol{U}_{kj} = \boldsymbol{U}'_{ij}$$

  By perfect binding of the commitment scheme, $m'_{r,ik} = m''_{r,ik,(j)} \forall i, j, k \in [m]$. We have now extracted a matrix $\hat{\boldsymbol{R}}$ with $\hat{\boldsymbol{R}}_{ik} = m'_{r,ik} \forall i, k \in [m]$, that the prover committed to in advance, such that $\boldsymbol{a}' = (\hat{\boldsymbol{R}} + \boldsymbol{R}')\boldsymbol{a}$ and $\boldsymbol{U}' = (\hat{\boldsymbol{R}} + \boldsymbol{R}')\boldsymbol{U}$.

- For every $i \in [m]$, we run the extractor $\mathcal{E}_{\mathsf{LR}}$ on two transcripts for $\Pi_{4,a,i}$, to get $\{m''_{\alpha,j,(i)}, r''_{\alpha,j,(i)}, e''_{\alpha,j,(i)}, f''_{\alpha,j,(i)}\}_{j\in[t]}$ such that,

$$\textit{Verify}(pp_{\mathsf{RLWE}}, m''_{\alpha,j,(i)}, r''_{\alpha,j,(i)}, e''_{\alpha,j,(i)}, f''_{\alpha,j,(i)}, c_{\alpha^{\pi(i)}}) = 1 \forall j \in [t]$$
$$\boldsymbol{a}'_i = \Sigma_{j\in[t]} m''_{\alpha,j,(i)}\boldsymbol{a}'_{ji}$$

  The above equations hold for all $i \in [m]$. By perfect binding of the commitment scheme, we get that for all $j \in [t]$, $m'_{\alpha,j} = \alpha^{\pi'(j)} = m''_{\alpha,j,(i)} \forall i \in [m]$.

- Lastly, for every $i \in [m], j \in [l]$, we can run the extractor $\mathcal{E}_{\mathsf{LR}}$ on two transcripts for $\Pi_{4,u,ij}$ to get $\{m''_{\alpha,k,(ij)}, r''_{\alpha,k,(ij)}, e''_{\alpha,k,(ij)}, f''_{\alpha,k,(ij)}\}_{k\in[t]}$ such that,

$$\textit{Verify}(pp_{\mathsf{RLWE}}, m''_{\alpha,k,(ij)}, r''_{\alpha,k,(ij)}, e''_{\alpha,k,(ij)}, f''_{\alpha,k,(ij)}, c_{\alpha^{\pi(i)}}) = 1 \forall j \in [t]$$
$$\boldsymbol{U}'_{ij} = \Sigma_{k\in[t]} m''_{\alpha,k,(ij)}\boldsymbol{U}'_{ij}$$

  Again since the commitment scheme is perfectly binding, we have, for all $k \in [t]$, $m'_{\alpha,k} = m''_{\alpha,k,(ij)} = \alpha^{\pi'(i)} \forall i \in [m], j \in [l]$. We get that $\boldsymbol{a}' = \Sigma_{i\in[t]} \alpha^{\pi'(i)}\boldsymbol{a}'_i$ and $\boldsymbol{U}' = \Sigma_{i\in[t]} \alpha^{\pi'(i)}\boldsymbol{U}'_i$.

Combining the above equations, we get the following:

$$\begin{aligned}\Sigma_{i\in[t]}\alpha^{\pi'(i)}\boldsymbol{a}'_i &= \boldsymbol{a}' \\ &= (\hat{\boldsymbol{R}}+\boldsymbol{R}')\boldsymbol{a} \\ &= (\hat{\boldsymbol{R}}+\boldsymbol{R}')\Sigma_{i\in[t]}\alpha^i\boldsymbol{a}_i \\ &= \Sigma_{i\in[t]}\alpha^i(\hat{\boldsymbol{R}}+\boldsymbol{R}')\boldsymbol{a}_i\end{aligned}$$

Consider the polynomial $\Sigma_{i\in[t]}x^{\pi'(i)}\cdot((\hat{\boldsymbol{R}}+\boldsymbol{R}')\boldsymbol{a}_{\pi'(i)}-\boldsymbol{a}'_i)$. The prover committed to this polynomial before receiving anything from the verifier, and then the verifier queries it at a random point α. By the Generalized Schwartz zippel lemma from [31], we get that with probability $1-\frac{t}{|\mathcal{S}|}$, this polynomial is the zero polynomial. This implies that $\boldsymbol{a}'_i=(\hat{\boldsymbol{R}}+\boldsymbol{R}')\boldsymbol{a}_{\pi'(i)}\forall i$.

So we were able to extract a permutation π' and a re-randomization matrix $\hat{\boldsymbol{R}}$ such that the relation R_{RLWE} is satisfied. The overall success probability of our extractor is $\left(1-\frac{1}{2^{O(k')}}\right)\cdot(p^2-\frac{1}{|\mathcal{C}|})$, and the expected run-time is $\mathsf{poly}(t,k,n,\log(q),k')/p$. This proves that Π_{shuffle} is knowledge-sound.

□

Acknowledgments. This work was funded by NSF, DARPA, the Simons Foundation, UBRI, and NTT Research. Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA. Lior Rotem is supported by a research grant from Protocol Labs.

References

[1] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (h)ibe in the standard model. In *Advances in Cryptology – EUROCRYPT 2010*, pages 553–572, 2010.

[2] M. R. Albrecht, V. Cini, R. W. F. Lai, G. Malavolta, and S. A. Thyagarajan. Lattice-based snarks: Publicly verifiable, preprocessing, and recursively composable. In *Advances in Cryptology – CRYPTO 2022*, pages 102–132, 2022.

[3] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theory of Computing Systems*, 48:535–553, 2011.

[4] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 2017*, pages 2087–2104, Dallas, TX, USA, Oct. 31 – Nov. 2, 2017. ACM Press.

[5] P. Ananth, A. Deshpande, Y. T. Kalai, and A. Lysyanskaya. Fully homomorphic NIZK and NIWI proofs. In D. Hofheinz and A. Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 356–385, Nuremberg, Germany, Dec. 1–5, 2019. Springer, Heidelberg, Germany.

[6] T. Attema, R. Cramer, and L. Kohl. A compressed Σ-protocol theory for lattices. In *Advances in Cryptology – CRYPTO 2021*, pages 549–579, 2021.

[7] T. Attema, V. Lyubashevsky, and G. Seiler. Practical product proofs for lattice commitments. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 470–499, Santa Barbara, CA, USA, Aug. 17–21, 2020. Springer, Heidelberg, Germany.

[8] S. Azouvi and D. Cappelletti. Private attacks in longest chain proof-of-stake protocols with single secret leader elections. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, AFT '21, page 170–182, 2021.

[9] S. Azouvi, P. McCorry, and S. Meiklejohn. Betting on blockchain consensus with fantomette, 2018.

[10] M. Backes, P. Berrang, L. Hanzlik, and I. Pryvalov. A framework for constructing single secret leader election from MPC. In V. Atluri, R. Di Pietro, C. D. Jensen, and W. Meng, editors, *ESORICS 2022, Part II*, volume 13555 of *LNCS*, pages 672–691, Copenhagen, Denmark, Sept. 26–30, 2022. Springer, Heidelberg, Germany.

[11] F. Baldimtsi, V. Madathil, A. Scafuro, and L. Zhou. Anonymous lottery in the proof-of-stake setting. In L. Jia and R. Küsters, editors, *CSF 2020 Computer Security Foundations Symposium*, pages 318–333, Boston, MA, USA, June 22–26, 2020. IEEE Computer Society Press.

[12] C. Baum, J. Bootle, A. Cerulli, R. del Pino, J. Groth, and V. Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In *Advances in Cryptology – CRYPTO 2018*, pages 669–699, 2018.

[13] C. Baum, J. Bootle, A. Cerulli, R. del Pino, J. Groth, and V. Lyubashevsky. Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 669–699, Santa Barbara, CA, USA, Aug. 19–23, 2018. Springer, Heidelberg, Germany.

[14] C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In M. Robshaw and J. Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 478–498, Santa Barbara, CA, USA, Aug. 14–18, 2016. Springer, Heidelberg, Germany.

[15] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert. More efficient commitments from structured lattice assumptions. In D. Catalano and R. De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 368–385, Amalfi, Italy, Sept. 5–7, 2018. Springer, Heidelberg, Germany.

[16] S. Bayer and J. Groth. Efficient zero-knowledge argument for correctness of a shuffle. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280, Cambridge, UK, Apr. 15–19, 2012. Springer, Heidelberg, Germany.

[17] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399, Alexandria, Virginia, USA, Oct. 30 – Nov. 3, 2006. ACM Press.

[18] E. Ben-Sasson, I. Bentov, Y. Horesh, , and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

[19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for R1CS. In *Advances in Cryptology – EUROCRYPT 2019*, pages 103–128, 2019.

[20] E. Ben-Sasson, A. Chiesa, and N. Spooner. Interactive oracle proofs. In *Theory of Cryptography*, pages 31–60, 2016.

[21] F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. Cryptology ePrint Archive, Report 2014/889, 2014. https://eprint.iacr.org/2014/889.

[22] F. Benhamouda, S. Krenn, V. Lyubashevsky, and K. Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In G. Pernul, P. Y. A. Ryan, and E. R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 305–325, Vienna, Austria, Sept. 21–25, 2015. Springer, Heidelberg, Germany.

[23] R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang. Ligero++: A new optimized sublinear IOP. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *ACM CCS 2020*, pages 2025–2038, Virtual Event, USA, Nov. 9–13, 2020. ACM Press.

[24] D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco. Single secret leader election. In *AFT '20*, pages 12–24. ACM, 2020. Available online at eprint/2020/025.

[25] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography, Draft 0.6*. Cambridge University Press, 2023.

[26] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334, San Francisco, CA, USA, May 21–23, 2018. IEEE Computer Society Press.

[27] D. Catalano, D. Fiore, and E. Giunta. Efficient and universally composable single secret leader election from pairings. Cryptology ePrint Archive, Report 2021/344, 2021. https://eprint.iacr.org/2021/344.

[28] D. Catalano, D. Fiore, and E. Giunta. Adaptively secure single secret leader election from ddh. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, PODC'22, page 430–439, 2022.

[29] R. Chairattana-Apirom and A. Lysyanskaya. Compact cut-and-choose: Boosting the security of blind signature schemes, compactly. Cryptology ePrint Archive, Paper 2022/003, 2022. https://eprint.iacr.org/2022/003.

[30] M. Christ, V. Nikolaenko, and J. Bonneau. Leader election from randomness beacons and other strategies, 2022. https://a16zcrypto.com/posts/article/leader-election-from-randomness-beacons-and-other-strategies.

[31] N. Costa, R. Martínez, and P. Morillo. Lattice-based proof of a shuffle. In *FC 2019: Financial Cryptography and Data Security*, pages 330–346, 2019.

[32] R. del Pino and V. Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 365–394, Santa Barbara, CA, USA, Aug. 20–24, 2017. Springer, Heidelberg, Germany.

[33] J. Don, S. Fehr, and C. Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In *Advances in Cryptology – CRYPTO 2020*, pages 602–631, 2020.

[34] J. Drake. Low-overhead secret single-leader election, 2019. https://ethresear.ch/t/low-overhead-secret-single-leader-election/5994.

[35] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, Aug. 1987. Springer, Heidelberg, Germany.

[36] L. Freitas, A. Tonkikh, A.-A. Bendoukha, S. Tucci-Piergiovanni, R. Sirdey, O. Stan, and P. Kuznetsov. Homomorphic sortition – single secret leader election for pos blockchains. Cryptology ePrint Archive, Paper 2023/113, 2023. https://eprint.iacr.org/2023/113.

[37] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 368–387, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer, Heidelberg, Germany.

[38] C. Ganesh, C. Orlandi, and D. Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 690–719, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[39] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 197–206, 2008.

[40] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. https://eprint.iacr.org/2017/454.

[41] A. Golovnev, J. Lee, S. Setty, J. Thaler, , and R. S. Wahby. Brakedown: Linear-time and post-quantum snarks for R1CS. Cryptology ePrint Archive, Paper 2021/1043, 2021. https://eprint.iacr.org/2021/1043.

[42] J. Groth and Y. Ishai. Sub-linear zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology – EUROCRYPT 2008*, pages 379–396, 2008.

[43] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[44] G. Kadianakis. Whisk: A practical shuffle-based ssle protocol for ethereum, 2022. X.

[45] T. Kerber, A. Kiayias, M. Kohlweiss, and V. Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press.

[46] A. Langlois and D. Stehle. Worst-case to average-case reductions for module lattices. Cryptology ePrint Archive, Paper 2012/090, 2012. https://eprint.iacr.org/2012/090.

[47] K. G. Larsen, M. Obremski, and M. Simkin. Distributed shuffling in adversarial environments. Cryptology ePrint Archive, Paper 2022/560, 2023. Appeared in Information-Theoretic Cryptography (ITC) 2023. Available at https://eprint.iacr.org/2022/560.

[48] V. Lyubashevsky. Lattice signatures without trapdoors. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755, Cambridge, UK, Apr. 15–19, 2012. Springer, Heidelberg, Germany.

[49] V. Lyubashevsky, N. K. Nguyen, and M. Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In *Advances in Cryptology – CRYPTO 2022*, pages 71–101, 2022.

[50] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[51] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[52] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In M. K. Reiter and P. Samarati, editors, *ACM CCS 2001*, pages 116–125, Philadelphia, PA, USA, Nov. 5–8, 2001. ACM Press.

[53] N. K. Nguyen and G. Seiler. Practical sublinear proofs for r1cs from lattices. In *Advances in Cryptology – CRYPTO 2022*, pages 133–162, 2022.

[54] C. Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 10(4):283–424, 2016. Available online at eprint/2015/939.

[55] M. Raikwar and D. Gligoroski. Sok: Decentralized randomness beacon protocols. In *Australasian Conference on Information Security and Privacy*, pages 420–446. Springer, 2022. available here.

[56] A. Rao. An exposition of bourgain's 2-source extractor. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 14. Citeseer, 2007.

[57] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *37th ACM STOC*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[58] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. Available online here.

[59] A. Sanso. Towards practical post quantum single secret leader election (ssle) - part 1, 2022. X.

[60] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134, Santa Fe, NM, USA, Nov. 20–22, 1994. IEEE Computer Society Press.

[61] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa. Efficient public key encryption based on ideal lattices. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635, Tokyo, Japan, Dec. 6–10, 2009. Springer, Heidelberg, Germany.

[62] M. Strand. A verifiable shuffle for the GSW cryptosystem. In A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, and M. Sala, editors, *FC 2018 Workshops*, volume 10958 of *LNCS*, pages 165–180, Nieuwpoort, Curaçao, Mar. 2, 2019. Springer, Heidelberg, Germany.

[63] T. Tao and V. Vu. On the singularity probability of random bernoulli matrices. *Journal of the American Mathematical Society*, 20(3):603–628, 2007.

[64] E. F. R. Team. A shuffle argument protocol specification, 2020. X.

[65] K. Tikhomirov. Singularity of random bernoulli matrices. *Annals of Mathematics*, 191(2):593–634, 2020.

A Strong Unlinkability without A Randomness Beacon

In this section, we sketch three we present three different approaches to augment our lattice-based RRC schemes to provide strong unlinkability without assuming a randomness beacon (recall Section 6).

Method I: Enforcing true randomness. One method for augmenting $\mathsf{R}_{\mathsf{LWE}}$ to be secure according to the above-strengthened notion is to have the *Randomize* algorithm choose the matrix $\boldsymbol{R}$ used for re-randomization as $\boldsymbol{R} \leftarrow \mathsf{H}_{\mathsf{R}}(r)$ where $r \xleftarrow{\$} \{0,1\}^{\lambda}$ and H_{R} is a cryptographic hash function mapping λ-bit strings to matrices in $\{-1,1\}^{m \times m}$. This is formalized as follows. The *Precommit* does nothing and outputs $\perp$. The *Extract* algorithm gets no input other than *pp*, samples a random string $r \xleftarrow{\$} \{0,1\}^{\lambda}$, and outputs $\boldsymbol{R} \leftarrow \mathsf{H}_{\mathsf{R}}(r)$. Intuitively, the set $\mathcal{G}$ of admissible randomness for *Randomize* is the set of all matrices $\boldsymbol{R}$ for which the re-randomizer "knows" a pre-image r under H_{R}. Formalizing this requires an extension of the above definitions in the random oracle model, in which $\mathcal{G}$ is set to be the set of all outputs of the random oracle observed by the adversary. When using this approach with a concrete hash function H_{R}, the randomizer has to prove (as discussed above) that it used randomness for which it knows a pre-image.

We argue that this method results in a scheme that is strongly pseudorandom and hence strongly unlinkable, when H_{R} is modeled as a random oracle. This is true even for unbounded adversaries (in the random oracle model). The argument is a standard one, but we sketch it here for completeness. Consider an adversary $\mathcal{A}$ that makes at most a polynomial number $Q = Q(\lambda)$ of queries to H_{R}. Consider a modified game, denoted $\boldsymbol{G}'^{\mathrm{pr}}_{\mathcal{A},\mathsf{R}}(\lambda)$ in which instead of letting $\mathcal{A}$ choose the randomness r used to produce c', we sample r honestly. By the proof of Theorem 4.3, we know that there exists a negligible function $\nu(\cdot)$ such that $\mathcal{A}$'s advantage in $\boldsymbol{G}'^{\mathrm{pr}}_{\mathcal{A},\mathsf{R}}(\lambda)$ is bounded by $\nu(\lambda)$. We say that randomness r is *bad* if $\mathcal{A}$'s advantage in $\boldsymbol{G}'^{\mathrm{pr}}_{\mathcal{A},\mathsf{R}}(\lambda)$ conditioned on the challenger choosing r as the randomness for randomization is at least $\sqrt{\nu(\lambda)}$. Total probability implies that the probability that r is bad is bounded by $\sqrt{\nu(\lambda)}$. Hence, by sampling Q random possible values for r, the overall probability that one of them is bad is at most $Q \cdot \sqrt{\nu(\lambda)}$ and the $\mathcal{A}$'s total advantage in $\boldsymbol{G}^{\mathrm{pr}}_{\mathcal{A},\mathsf{R}}(\lambda)$ is at most $(Q+1) \cdot \sqrt{\nu(\lambda)}$.

The same technique works for augmenting our ring-based scheme, $\mathsf{R}_{\mathsf{RLWE}}$, with strong pseudorandomness in the same manner.

Method II: Pre-committing to well-structured randomness. Our second method to achieve strong unlinkability using our LWE-based RRC scheme, without a randomness beacon, is to have the randomizer precommit to a *well-structured* $\boldsymbol{R}$ in advance. By well-structured, we mean that $\boldsymbol{R}$ is full-rank. Formally, *Precommit*$(pp; r)$ parses $r = (\boldsymbol{R}, r')$ and commits to $\boldsymbol{R}$ using a statistically-binding non-interactive commitment scheme C using randomness r'. Given the randomness $r = (\boldsymbol{R}, r')$ to the precommitment pcom, the *Extract* algorithm simply outputs $\boldsymbol{R}$. The set $\mathcal{G}(\mathsf{pcom}, r = (\boldsymbol{R}, r'), \perp)$ of admissible randomness is the equal to $\{\boldsymbol{R}\}$ if $\boldsymbol{R}$ is a full-rank matrix and $\mathsf{pcom} = \mathsf{C}.\mathit{Commit}(pp_{\mathsf{C}}, \boldsymbol{R}; r')$, and to the empty set $\emptyset$ otherwise.

To see why this approach provides strong pseudorandomness, first observe that since the adversary commits to $\boldsymbol{R}$ in advance in the strong pseudorandomness game, it is statistically independent of the commitment c. Next, observe that for any *fixed* full-rank matrix $\boldsymbol{R} \in \{-1,1\}^{m \times m}$, the distribution $(\boldsymbol{R} \cdot A, \boldsymbol{R} \cdot U)$ when

$A \xleftarrow{\$} \mathbb{Z}_q^{m\times n}$ and $U \xleftarrow{\$} \mathbb{Z}_q^{m\times \ell}$, is uniformly random in $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times \ell}$. This is the case since the function $f_{\boldsymbol{R}}(\boldsymbol{x}) = \boldsymbol{R}\cdot\boldsymbol{x}$ is a permutation on $\mathbb{Z}_q^m$. Hence, since c is indistinguishable from pairs of independent uniformly-random matrices, this is also the case for c'. Therefore, Lemma 2.1 guarantees that the subsequent honest call to *Randomize* assures that the commitment c_b given to the adversary in line 12 of the strong-pseudorandomness security game is indistinguishable from a pair of uniformly random matrices, which is independent of the bit b (the proof is identical to the proof of Claim 4.6).

To retain the correctness of the scheme, we need that a uniformly-random matrix in $\{-1,1\}^{m\times m}$ to be full-rank with overwhelming probability. To achieve that, we need to set the modulus q to be super-polynomial in the security parameter λ. In particular, if we assume that $q > m^{m/2}$ we may leverage the recent result of Tikhomirov [65] showing that a random $m \times m$ Bernoulli matrix is singular *over* $\mathbb{R}$ with probability $(1/2 + o(1))^m$. We can do so, since the determinant of a matrix $\{-1,1\}^{m\times m}$ is at most $m^{m/2}$; this can be seen, for example, by recalling that the magnitude of the determinant is equal to the volume of the parallelopiped spanned by the rows of the matrix (see also [63]).

A similar approach may be pursued for our Ring-LWE based RRC scheme as well, by replacing the notion of "full-rank" with an analogous notion in the ring setting that is sufficient for the above analysis to go through. We leave this as an interesting question for future work.

Method III: Assuming bad randomness is hard to find Our third approach is the simplest one: It only requires that the randomizer uses a matrix $\boldsymbol{R}$ with entries in $\{-1,1\}$ (that is, it constitutes "honest randomness" for *Randomize*) for re-randomization. That is, *Precommit* does nothing and outputs $\perp$, and *Extract* samples a uniform matrix in $\{-1,1\}^{m\times m}$. The set $\mathcal{G}$ of admissible random strings is the entire set $\{-1,1\}^{m\times m}$.

We introduce a lattice-based hardness assumption, under which restricting the randomizer to such randomness is sufficient for guaranteeing unlinkability. The rough idea is this: the leftover hash lemma guarantees that (A, RA, RU) is close to a tuple of three random matrices when A, R and U are chosen at random. A standard argument shows that this implies that for a *fixed* matrix $[\boldsymbol{A},\boldsymbol{U}]$, the distribution $R\cdot[\boldsymbol{A},\boldsymbol{U}]$ is (slightly less) close to a uniformly random matrix. Hence, the set of "bad" matrices, i.e., a matrix $[\boldsymbol{A},\boldsymbol{U}]$ such that $R\cdot[\boldsymbol{A},\boldsymbol{U}]$ is far from uniform, is a relatively small set.

Now recall the task of the adversary. The adversary is given a commitment, $c = (\boldsymbol{A},\boldsymbol{U})$, which is just a pair of pseudorandom matrices. The adversary comes up with $\boldsymbol{R} \in \{-1,1\}^{m\times m}$ and produces $c' = (\boldsymbol{A}',\boldsymbol{U}') = (\boldsymbol{RA},\boldsymbol{RU})$. Then, we wish to claim that $c_0 = (\boldsymbol{R}'\boldsymbol{A}',\boldsymbol{R}'\boldsymbol{U}')$ for $\boldsymbol{R}' \xleftarrow{\$} \{-1,1\}^{m\times m}$ is pseudorandom. Now, if $[\boldsymbol{A}',\boldsymbol{U}']$ is not a bad matrix, then c_0 is statistically close to a uniformly-random pair of matrices, and we are done. This means that to have a meaningful advantage, $[\boldsymbol{A}',\boldsymbol{U}']$ should be a bad matrix. This, in turn, means that to be successful, the adversary needs to find a *low-norm matrix* $\boldsymbol{R}$ such that $\boldsymbol{R}\cdot[\boldsymbol{A},\boldsymbol{U}]$ is in the *small set* of bad matrices. We can make the assumption that this is hard when $\boldsymbol{A}$ and $\boldsymbol{U}$ are uniformly-random matrices.

In Appendix B we formally present this assumption (called the SMS-SIS assumption), show that the set of "bad matrices" the adversary has to hit is indeed of negligible density, and prove that this assumption is sufficient to obtain strong pseudorandomness of our RRC schemes. We focus on the LWE setting, but a similar assumption and analysis can be made for the Ring-LWE case as well.

B The SMS-SIS Assumption

In this appendix we define the SMS-SIS assumption, which essentially states that given a random matrix $\boldsymbol{A}$ it should be hard to find a matrix $\boldsymbol{R}$ such that $\boldsymbol{R}\cdot\boldsymbol{A}$ hits some fixed sparse subset of matrices. We then

show that the subset of "bad" matrices the adversary in our RRC scheme has to hit is indeed very sparse, and conclude by reducing the active security of the scheme to this new assumption.

We focus the discussion on our LWE-based RRC, but it can be extended to the ring setting as well.

The SMS-SIS assumption. We prove the security of the scheme based on a new assumption we put forth, which we call the *small matrix set short integer solution* problem, or SMS-SIS for short.

Definition B.1. *Let $q = q(\lambda)$ be a prime, $n = n(\lambda)$ and $m = m(\lambda)$ be integers, and $\beta = \beta(\lambda)$ be a real number, all public functions of the security parameter $\lambda \in \mathbb{N}$. Let $\mathcal{S} = \{\mathcal{S}_\lambda\}_\lambda$ be a set of matrices in $\mathbb{Z}_q^{n \times m}$. The $(n, q, \beta, m, \mathcal{S})$-SMS-SIS assumption states that for every probabilistic polynomial time algorithm $\mathcal{A}$ there exists an negligible function $\nu(\cdot)$ such that*

$$\mathsf{Adv}^{\text{sms-sis}}_{\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c} \boldsymbol{A} \cdot \boldsymbol{R} \in \mathcal{S} \text{ AND} \\ \textit{for each row } \boldsymbol{x} \textit{ of } \boldsymbol{R}, \|\boldsymbol{x}\|_2 \leq \beta \end{array} \middle| \begin{array}{c} \boldsymbol{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \\ \boldsymbol{R} \xleftarrow{\$} \mathcal{A}(\boldsymbol{A}) \end{array}\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

The reduction. We now explain how an adversary breaking the strong pseudorandomness of the commitment scheme can be used to break the SMS-SIS assumption with respect to a small subset $\mathcal{S}$ of matrices. First, we define a set of "bad" matrices and show that it is small. Concretely, we show that the parameters can be set such that the size of this set is less than $2^{c\lambda}$ for some constant $c \leq 1$. Then, we claim that an adversary which breaks the unlinkability of our scheme essentially has to break SMS-SIS with respect to this set.

Bad matrices. For a value $\epsilon \in [0, 1]$, we say that a (fixed) matrix $\boldsymbol{A} \in \mathbb{Z}_q^{m \times (n+\ell)}$ is ϵ-*bad* if

$$\Pr\left[\mathsf{SD}(R \cdot \boldsymbol{A}, U) > \epsilon\right],$$

where the probability is over $R \xleftarrow{\$} \{-1, 1\}^{m \times m}$ and $U \xleftarrow{\$} \mathbb{Z}_q^{m \times (n+\ell)}$. Intuitively, a matrix $\boldsymbol{A}$ is bad if it does not act like a good extractor (with respect to multiplication). Let $\mathcal{S}_{\epsilon,m,n,\ell}$ denote the set of ϵ-bad $m \times (n + \ell)$-matrices.

We first prove a useful lemma, bounding the number of bad matrices.

Lemma B.1. *For any $m, n \in \mathbb{N}$ and $\epsilon \in [0, 1]$ it holds that*

$$|\mathcal{S}_{\epsilon,m,n,\ell}| \leq \frac{m \cdot \sqrt{2^{-m+(n+\ell)\log q - 2}} \cdot q^{m \cdot (n+\ell)}}{\epsilon}$$

Proof. The proof follows a, by now standard, observation by Rao (see [56] for example). Let $R \xleftarrow{\$} \{-1, 1\}^{m \times m}$, $A \xleftarrow{\$} \mathbb{Z}_q^{m \times (n+\ell)}$ and $U \xleftarrow{\$} \mathbb{Z}_q^{m \times (n+\ell)}$ be random variables. Assume towards contradiction that

$$|\mathcal{S}_{\epsilon,m,n,\ell}| > \frac{m \cdot \sqrt{2^{-m+(n+\ell)\log q - 2}} \cdot q^{m \cdot (n+\ell)}}{\epsilon}.$$

Then, it holds that

$$\begin{aligned}
\mathsf{SD}((A, R\cdot A),(A,U)) &= \frac{1}{2}\sum_{\boldsymbol{A},\boldsymbol{B}}\left|\Pr_{A,R}[A=\boldsymbol{A}\wedge RA=\boldsymbol{B}] - q^{-2\cdot m\cdot(n+\ell)}\right| \\
&= \frac{1}{2}\sum_{\boldsymbol{A},\boldsymbol{B}}\left|\Pr_{A}[A=\boldsymbol{A}]\cdot\Pr_{R}[R\cdot\boldsymbol{A}=\boldsymbol{B}] - q^{-2\cdot m\cdot(n+\ell)}\right| \\
&= q^{-m\cdot(n+\ell)}\cdot\sum_{\boldsymbol{A}}\sum_{\boldsymbol{B}}\frac{1}{2}\left|\Pr_{R}[R\cdot\boldsymbol{A}=\boldsymbol{B}] - q^{-m\cdot(n+\ell)}\right| \\
&= q^{-m\cdot(n+\ell)}\cdot\sum_{\boldsymbol{A}}\mathsf{SD}(R\cdot\boldsymbol{A},U) \\
&\geq q^{-m\cdot(n+\ell)}\cdot\sum_{\boldsymbol{A}\in\mathcal{S}_{\epsilon,m,n,\ell}}\mathsf{SD}(R\cdot\boldsymbol{A},U) \\
&\geq q^{-m\cdot(n+\ell)}\cdot|\mathcal{S}_{\epsilon,m,n,\ell}|\cdot\epsilon \qquad (30)\\
&> m\cdot\sqrt{2^{-m+(n+\ell)\log q-2}}, \qquad (31)
\end{aligned}$$

where unless otherwise specified, summation is over $\mathbb{Z}_q^{m\times(n+\ell)}$, Eq. (30) follows from the definition of $\mathcal{S}_{\epsilon,m,n,\ell}$ and Eq. (31) follows from our assumption.

But we have arrived at a contradiction, since by Lemma 2.1, we know that $\mathsf{SD}((A, R\cdot A),(A,U)) \leq m\cdot\sqrt{2^{-m+(n+\ell)\log q-2}}$. This concludes the proof of the lemma. □

Corollary B.2. *For any $c\in[0,1]$, if λ and ϵ are such that $\lambda \geq 2\cdot m\cdot(n+\ell)\cdot\log q + \log(1/\epsilon)$, then $|\mathcal{S}_{\epsilon,m,n,\ell}| \leq 2^{c\lambda}$.*

As a concrete example, think of setting m, n, ℓ and $\log q$ such that all of them are at most $\lambda^{1/3}/3$, and setting $1/\epsilon = o(2^{\lambda})$ but still super-polynomial.

From breaking strong pseudorandomness to breaking SMS-SIS. Consider an adversary $\mathcal{A}$ trying to break strong pesudorandomness. The reduction algorithm $\mathcal{B}$ gets as input a matrix $\boldsymbol{B} = [\boldsymbol{A}, \boldsymbol{U}] \in \mathbb{Z}_q^{m\times(n+\ell)}$ and sends it to $\mathcal{A}$. $\mathcal{A}$ then produces $c' = (\boldsymbol{A}', \boldsymbol{U}')$ alongside randomness $\boldsymbol{R}$. If $\boldsymbol{R}$ is invalid, the reduction aborts. Otherwise, $\mathcal{B}$ outputs $\boldsymbol{R}$.

To see why the reduction is successful, consider the case where $[\boldsymbol{A}', \boldsymbol{U}']$ is outside of $\mathcal{S}_{\epsilon,m,n,\ell}$. In this case, $\boldsymbol{R}'\cdot[\boldsymbol{A}', \boldsymbol{U}']$ is close to uniform for an honest choice of $\boldsymbol{R}'$. Hence, the adversary can only have a negligible advantage in guessing b. Hence, $\mathcal{A}$ may have noticeable advantage only if $[\boldsymbol{A}', \boldsymbol{U}'] = \boldsymbol{R}\cdot[\boldsymbol{A}, \boldsymbol{U}] \in \mathcal{S}_{\epsilon,m,n,\ell}$. Conditioned on this being the case, $\mathcal{B}$ solves the SMS-SIS problem with respect to $\mathcal{S}_{\epsilon,m,n,\ell}$ with probability 1.

C Deferred Proofs

C.1 Proof of Theorem 7.1

We prove uniqueness, fairness, and unpredictability separately.

Uniqueness. Suppose that there exists an adversary $\mathcal{A}$ breaking the uniqueness of the scheme. We construct an adversary $\mathcal{B}$ that breaks the binding property of R. On input pp_{R}, $\mathcal{B}$ simulates the $\mathsf{UNIQUE}_{\mathcal{A},\mathsf{ssle}}(\lambda)$ to $\mathcal{A}$. It does so by sampling public parameters pp_{NIZK} for NIZK and forwarding $pp = (pp_{\mathsf{NIZK}}, pp_{\mathsf{R}})$ to $\mathcal{A}$. It then plays the honest parties, and the randomness beacon D in the election protocol ssle.*Elect*. Let $(\mathsf{pst}_i, b_i, \pi_i)$ denote the output of the honest user i (emulated by $\mathcal{B}$) at the end of the election protocol, and let $\mathsf{pst}_i = (h_i^*, c_i^*)$. Then, to win the uniqueness experiment, $\mathcal{A}$ needs to come up with $\pi_{j_1} = k_{j_1}$ and $\pi_{j_2} = k_{j_2}$ such that for some honest i, it holds that $\mathsf{ssle}.\mathit{Verify}(pp, j_1, \mathsf{pst}_i, \pi_{j_1}) = \mathsf{ssle}.\mathit{Verify}(pp, j_2, \mathsf{pst}_i, \pi_{j_2}) = 1$. By definition, this implies that $h_{j_1} \neq h_{j_2}$, which means that $k_{j_1} \neq k_{j_2}$. But for both proofs to go through verification by party i, it must be the case that $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_i^*, k_{j_1}) = \mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_i^*, k_{j_2}) = 1$. This means that $\mathcal{B}$ can output the commitment c^* and the keys k_{j_1} and k_{j_2} to break binding.

Fairness. The fairness of the protocol follows from the re-randomizability of R and the soundness of NIZK. Let $c_1, \ldots, c_n$ be the commitments broadcast by parties in the commitment stage of the protocol, and let $c_{i,1}^{(n)}, \ldots, c_{i,n}^{(n)}$ denote the commitments held by party i at the end of the shuffle stage. By the soundness of NIZK, with all but negligible probability, there exists a permutation σ on $[n]$ such that for every $i, j \in [n] \setminus \mathcal{I}$, $c_{i,\sigma(j)}^{(n)}$ is obtained by a sequence of at most n re-randomizations of c_j using admissible randomness. Hence, by the re-randomizability of R, $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{i,\sigma(j)}^{(n)}, k_j) = 1$ with all but negligible probability.

Additionally, we argue that with overwhelming probability, the set $\mathcal{S}$ contains no honest parties. For any two honest parties j_1, j_2, it holds that $\mathsf{H}(k_1) = \mathsf{H}(k_2)$ with negligible probability. This is the case since by assumption, $k_{j_1} = k_{j_2}$ with negligible probability, and conditioned on $k_1 \neq k_2$, it holds that $\mathsf{H}(k_{j_1}) = \mathsf{H}(k_{j_2})$ with probability $2^{-\lambda}$ since H is modeled as a random oracle.

Let Bad denote the event in which $\mathcal{S} \cap ([n] \setminus \mathcal{I}) \neq \emptyset$ (that is, there is an honest party in $\mathcal{S}$), *or* there exist honest parties $i, j \in [n] \setminus \mathcal{I}$ such that for all $\ell \in [n]$ it holds that $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{i,\ell}^{(n)}, k_j) = 0$. The above analysis shows that the probability of Bad is negligible. Let $i \in [n] \setminus \mathcal{I}$. Conditioned on the complementing event $\overline{\mathsf{Bad}}$, the probability that an honest party $j \in [n] \setminus \mathcal{I}$ such that $\mathsf{ssle}.\mathit{Verify}(pp, j, \mathsf{pst}_i, k_j) = 1$ is exactly $1 - |\mathcal{I}|/n$, concluding the proof.

Unpredictability. Let $\mathcal{A}$ be an adversary in the unpredictability experiment $\mathsf{UNPRED}_{\mathcal{A},\mathsf{ssle},n,c}(\lambda)$. We construct an adversary $\mathcal{B}$ participating in the strong unlinkability game of R. The adversary $\mathcal{B}$ gets pp_{R}, samples $pp_{\mathsf{NIZK}} \xleftarrow{\$} \mathsf{NIZK}.\mathit{Setup}(1^\lambda)$ and forwards $pp = (pp_{\mathsf{NIZK}}, pp_{\mathsf{R}})$ to $\mathcal{A}$. Then, $\mathcal{B}$ simulates the election protocol to $\mathcal{A}$, playing the role of the honest parties $[n] \setminus \mathcal{I}$ as follows:

1. On each query k to the random oracle by $\mathcal{A}$: if the value $\mathsf{H}(k)$ was previously defined, $\mathcal{B}$ replies consistently. Otherwise, it samples a fresh uniformly random answer from $\{0,1\}^\lambda$.

2. It gets the subset $\mathcal{I}$ of corrupted parties from $\mathcal{A}$. Let $\mathcal{S}_1, \ldots, \mathcal{S}_T$ denote the maximally-sized of consecutive honest parties. That is, the sets are defined iteratively: $\mathcal{S}_1 = \{i_1, \ldots, j_1\}$, where i_1 is the smallest index not in $\mathcal{I}$ and j_1 is the smallest index not in $\mathcal{I}$ for which $j_1 + 1$ is in $\mathcal{I}$; $S_2 = \{i_2, \ldots j_2\}$, where i_2 is the smallest index not in $\mathcal{I} \cup \mathcal{S}_2$ and j_2 is the smallest index not in $\mathcal{I} \cup \mathcal{S}_2$ for which $j_2 + 1 \in \mathcal{I}$; and so forth. $\mathcal{B}$ outputs $T, i_0^1 = i_1^{(1)} = |\mathcal{S}_1|, \ldots, i_0^{(T)} = i_1^{(T)} = |\mathcal{S}_T|$.

3. $\mathcal{B}$ guesses two indices $j_0, j_1 \xleftarrow{\$} [n] \setminus \mathcal{I}$. It then generates the precommitments for the honest parties: It sets the precommitments of parties j_0, j_1 to be the precommitments received from the challenger in the strong unlinkability game, and honestly samples all of the other precommitments. It sends all precommitments to $\mathcal{A}$.

4. Upon receiving the precommitments of the corrupted parties from $\mathcal{A}$, $\mathcal{B}$ generates all commitments $c_1, c_2, \ldots$ and hash values $h_1, h_2, \ldots$ for the honest parties: it sets c_{j_0} and c_{j_1} to be the fresh commitments received from the challenger in the unlinkability game. All other commitments $\{c_i\}$ are generated honestly, together with the corresponding keys $\{k_i\}$. The hash values $h_1, h_2, \ldots$ are sampled uniformly at random from $\{0,1\}^\lambda$. If for some $i \in [n] \setminus (\mathcal{I} \cup \{j_0, j_1\})$, it holds that $\mathcal{A}$ has previously queried H on k_i, then $\mathcal{B}$ outputs a random bit $b' \xleftarrow{\$} \{0,1\}$ and terminates.

5. $\mathcal{B}$ simulates the shuffle stage of the protocol in the following manner:

   - When it is the turn of a corrupted party $i \in \mathcal{I}$ to shuffle: $\mathcal{B}$ samples the randomness beacon values for this round and sends them to $\mathcal{A}$. It then receives the vector of re-randomized and shuffled commitments from $\mathcal{A}$, together with a proof of shuffle pf. It uses the extractor guaranteed by the NIZK to extract the permutation σ, the randomness $r_1, \ldots, r_n$ to $\mathsf{R}.\mathit{Precommit}$, and the randomness $r'_1, \ldots, r'_n$ used by $\mathcal{A}$ to perform the shuffle. It sends the r and r' values corresponding to j_1 and j_2 together with the corresponding re-randomized commitments to the challenger in the unlinkability game.
   - When it is the turn of an honest party $i \in [n] \setminus \mathcal{I}$ to shuffle: $\mathcal{B}$ obtains the re-randomized commitments for parties j_0 and j_1 from the challenger, together with the corresponding random beacon values. It re-randomizes honestly the commitments of all other parties, sampling the associated randomness beacon values on its own. It then samples a permutation σ on $[n]$ and applies it to the re-randomized commitments. Let $c_1, \ldots, c_n$ denote the vector of commitments as stored by party i before $\mathcal{B}$ applied the re-randomization and permutation, and let $c'_1, \ldots, c'_n$ denote the re-randomized and shuffled commitments. Let $\mathsf{rand}_1, \ldots, \mathsf{rand}_n$ denote the corresponding outputs of the beacon (two of which obtained from the challenger, and the rest sampled by $\mathcal{B}$), and let $\mathsf{pcom}_1, \ldots, \mathsf{pcom}_n$ be the corresponding precommitments (two of which obtained from the challenger, and the rest sampled by $\mathcal{B}$ in the beginning of the execution). Using the simulator for NIZK, $\mathcal{B}$ computes a proof pf for the instance

     $$(pp_{\mathsf{R}}, (c_1, \ldots, c_n), (c'_1, \ldots, c'_n), (\mathsf{rand}_1, \ldots, \mathsf{rand}_n), (\mathsf{pcom}_1, \ldots, \mathsf{pcom}_n)).$$

     $\mathcal{B}$ sends the re-randomized and shuffled commitments $c'_1, \ldots, c'_n$, the randomness beacon values $\mathsf{rand}_1, \ldots, \mathsf{rand}_n$, and the proof pf to $\mathcal{A}$.

     In the final pair of re-randomized commitments that $\mathcal{B}$ obtains from the challenger, it does not know whether their order was swapped or not. $\mathcal{B}$ continues as if they were not. Note that this does not affect the distribution over the view of $\mathcal{A}$ since $\mathcal{B}$ applies a random permutation onto the n commitments.

6. $\mathcal{B}$ computes the set $\mathcal{S}$ according to the hash values $\{h_i\}$ and samples a random index $i^* \xleftarrow{\$} [n] \setminus \mathcal{S}$. Let i_0, i_1 denote the possible locations of the commitments corresponding to parties j_0 and j_1, respectively, assuming that $b = 0$ in the unlinkability experiment (note if $b = 1$, it holds that i_0, i_1 correspond to parties j_1 and j_0, respectively). If $i^* \notin \{i_0, i_1\}$, then $\mathcal{B}$ outputs a random $b' \xleftarrow{\$} \{0,1\}$ and terminates. Otherwise, let d be the bit such that $i^* = i_d$. $\mathcal{B}$ sends i^* to $\mathcal{A}$, who responds with some index $\ell \in [n]$.

7. Let $c_1^{(n)}, \ldots, c_n^{(n)}$ be the final vector of commitments. Let $\mathcal{K}$ be the set of H-queries issued by $\mathcal{A}$. If there is a query $k \in \mathcal{Q}$ such that $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{j_0}, k) = 1$, then $\mathcal{B}$ outputs 0 (and terminates) if $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{i_0}, k) = 1$ and 1 if $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{i_1}, k) = 1$. Similarly, if there is a query $k \in \mathcal{Q}$ such that $\mathsf{R}.\mathit{Test}(pp_{\mathsf{R}}, c_{j_1}, k) = 1$, then $\mathcal{B}$ decides on its output symmetrically and terminates.

8. If $\ell = j_d$, then $\mathcal{B}$ outputs $b' = 0$. If $\ell = j_{1-d}$ then $\mathcal{B}$ outputs $b' = 1$. Otherwise, it outputs a uniformly random bit $b' \xleftarrow{\$} \{0,1\}$.

Let E_1 denote the event in which $\mathcal{B}$ terminates in Step 4 of the simulation. Since $\mathcal{A}$ makes at most a polynomial number of queries to $\mathcal{A}$, it holds that E_1 occurs with negligible probability. Let E_2 denote the event in which $\mathcal{B}$ terminates in Step 6 of the simulation. Conditioned on $\overline{\mathsf{E}}_1$, E_2 occurs with probability at most $(n-2)/n$. Moreover, conditioned on E_2 occurring, the probability that $\mathcal{B}$ guesses b correctly is $1/2$. Let E_3 be the event in which $\mathcal{B}$ terminates in Step 7. Conditioned on E_3, $\mathcal{B}$ guesses the bit b with probability negligibly close to 1. Hence, if the probability of E_3 is non-negligible, we are done. So for the remainder of the analysis, we assume E_3 occurs with negligible probability.

We argue that conditioned on $\overline{\mathsf{E}_1} \wedge \overline{\mathsf{E}_2} \wedge \overline{\mathsf{E}_3}$, the view of $\mathcal{A}$ in the simulation is indistinguishable from its view in the real experiment. This is because the only differences between the two experiments are:

1. The proofs of shuffle of the honest parties are generated using the NIZK simulator.

2. The randomness values passed by $\mathcal{B}$ to the challenger in the unlinkability game are extracted from the proofs provided by $\mathcal{A}$ for the corrupted parties, via the NIZK extractor. If $\mathcal{A}$ used admissible randomness, but the extractor failed to extract admissible randomness, this is a deviation between the two experiments.

By the zero-knowledge and simulation-sound knowledge soundness of NIZK, it follows that conditioned on $\overline{\mathsf{E}_1} \wedge \overline{\mathsf{E}_2} \wedge \overline{\mathsf{E}_3}$, the two experiments are indistinguishable in the view of $\mathcal{A}$. Hence, the probability that it guesses the leader in the simulated experiment is negligibly close to the probability it guesses it in the real experiment.

Whenever $\overline{\mathsf{E}_1} \wedge \overline{\mathsf{E}_2} \wedge \overline{\mathsf{E}_3}$ occurs $\mathcal{A}$ guesses the leader, $\mathcal{B}$ correctly guesses the bit b. Hence, if we denote by j^* the index of the party chosen as leader, $\mathcal{B}$ guesses the bit b correctly with probability at least

$$\frac{1}{2} \cdot \frac{n-2}{n} + \frac{2}{n} \cdot \mathbb{E}_{j_1 \neq j_2 \xleftarrow{\$} [n] \setminus \mathcal{I}} \left[\Pr \left[\ell = j^* \mid j^* \in \{j_1, j_2\} \right] \right] - \nu(\lambda),$$

for some negligible function ν. Therefore, the unlinkability of R implies that there is negligible function ν' such that

$$\begin{aligned}
\Pr \left[\ell = j^* \mid j^* \in [n] \setminus \mathcal{I} \right] &= \mathbb{E}_{j_1 \neq j_2 \xleftarrow{\$} [n] \setminus \mathcal{I}} \left[\Pr \left[j^* \in \{j_1 \neq j_2\} \mid j^* \in [n] \setminus \mathcal{I} \right] \cdot \Pr \left[\ell = j^* \mid j^* \in \{j_1, j_2\} \right] \right] \\
&= \mathbb{E}_{j_1 \neq j_2 \xleftarrow{\$} [n] \setminus \mathcal{I}} \left[\frac{2}{n-c} \cdot \Pr \left[\ell = j^* \mid j^* \in \{j_1, j_2\} \right] \right] \\
&= \frac{2}{n-c} \cdot \mathbb{E}_{j_1 \neq j_2 \xleftarrow{\$} [n] \setminus \mathcal{I}} \left[\Pr \left[\ell = j^* \mid j^* \in \{j_1, j_2\} \right] \right] \\
&\leq \frac{2}{n-c} \cdot \left(\frac{1}{2} + \nu'(\lambda) \right) \\
&= \frac{1}{n-c} + \nu''(\lambda),
\end{aligned}$$

where ν'' is negligible. This concludes the proof. □