Practical Non-interactive Publicly Verifiable Secret Sharing with Thousands of Parties

Craig Gentry[1], Shai Halevi[1], and Vadim Lyubashevsky[2]

[1] Algorand Foundation
[2] IBM Research

Abstract. Non-interactive publicly verifiable secret sharing (PVSS) schemes allow parties to re-share a secret in a decentralized setting in the presence of malicious parties. A recently proposed application of PVSS schemes is to enable permissionless proof-of-stake blockchains to "keep a secret" via a sequence of committees that share that secret. Such committees can use the secret to produce signatures on the blockchain's behalf, or to disclose hidden data conditioned on consensus that some event has occurred. Such a setting may involve thousands of parties, so the PVSS scheme that it uses must be very efficient, both in computation and communication. Yet, previous PVSS schemes have large proofs and/or require many exponentiations over large groups.

We present a non-interactive PVSS scheme in which the underlying encryption scheme is based on the learning with errors (LWE) problem. While lattice-based encryption schemes are very fast, they have issues with bandwidth (long ciphertexts and public keys). We deal with the bandwidth issue in two ways. First, we adapt the Peikert-Vaikuntanathan-Waters (PVW) encryption scheme to the multi-receiver setting so that the bulk of the parties' keys is a common random string, and so that we get good amortized communication: $\Omega(1)$ plaintext/ciphertext rate (rate $\approx 1/60$ for 100 parties, $\approx 1/8$ for 1000 parties, approaching $1/2$ as the number of parties grows). Second, we use bulletproofs over a DL-group of order about 256 bits to get compact proofs of correct encryption of shares. Switching from the lattice setting to the DL setting is relatively painless, as we equate the LWE modulus with the order of the group, and apply dimension reduction to vectors before the switch to minimize the number of exponentiations in the bulletproof. An implementation of our PVSS for 1000 parties showed that it's quite practical, and should remain so with up to a two order of magnitude increase in the group size.

Table of Contents

1 Introduction

A publicly-verifiable secret-sharing scheme (PVSS) lets a dealer share a secret among a committee of shareholders, in such a way that everyone (not just the shareholders) can verify that the secret was shared properly and be assured that it is recoverable. A *noninteractive* PVSS scheme lets the sender broadcast just a single message to the entire universe, from which the shareholders can get their shares and everyone else can check that sharing was properly done.[3] A *proactive* PVSS scheme further enables passing the secret from one committee of shareholders to the next, so that (a) the secret remains hidden from an adversary that only controls a minority in each committee, and (b) everyone can check that the secret is passed properly between consecutive committees.

Such protocols play crucial role in distributed cryptography, and were studied extensively in the literature [17, 28, 54, 23, 52, 12, 56, 21, 15, 51, 32, 33, 24, 37, 50, 31]. They were also recently proposed as enablers of secure computation on large-scale distributed networks such as public blockchains [8, 31]. Unfortunately, existing PVSS schemes in the literature fall short of what is needed for general-purpose secure computation in large-scale systems, where committees may scale to hundreds or even thousands of parties [8, 26]. See related work in Section 1.1.

In this work we propose a new system for (proactive, noninteractive) PVSS, that remains feasible even with huge committees. In asymptotic terms, with security parameter λ and k-party committees, the PVSS protocol that we propose has the dealer and each committee member perform only $O(\lambda+k)$ exponentiations and broadcast $O(\lambda+k)$ scalars in $\mathbb{Z}_p$ and $O(\log(\lambda+k))$ group elements. (In addition, each party needs to perform $O(\lambda^2 + \lambda k)$ scalar multiplications in $\mathbb{Z}_p$, which comes to dominate the running time.)

In terms of actual numbers, we wrote a preliminary, single-threaded, implementation of our system and tested it on committees of up to 1000 members.[4] With a 1000-member committee, the dealer runs in about 40 seconds (single-threaded) and broadcasts a single message of size less than 300KB, while each committee member requires about 20 seconds to obtain its share and verify the proofs. As we explain in the sequel, this system can be extended to a proactive PVSS protocol for very large-scale systems, where the wall-clock time to refresh a secret is measured in just a few minutes.

We also point out that while our goal of using LWE encryption was motivated by practical consideration, a side effect is that the *secrecy* of the PVSS scheme is preserved even against quantum attackers. This protects the PVSS scheme from potential "harvest-and-decrypt" attacks using future quantum computers. This feature may be especially important for blockchain applications, where all the data is "harvested" by design.

1.1 The PVSS Problem and Related Work

Verifiable secret sharing (VSS) was introduced by Chor et al. [17], with the objective of making secret sharing robust against malicious parties – i.e., a malicious dealer distributing incorrect shares, or malicious shareholders submitting incorrect shares in the reconstruction protocol.

Stadler [54] introduced publicly verifiable secret sharing (PVSS), in which the correctness of shares is verifiable by everyone (not just shareholders). As Stadler notes, the idea appears implicitly in earlier works. Chor et al.'s VSS protocol [17] happened to be publicly verifiable. GMW [28] also

---

[3] Clearly such schemes must rely on some form of PKI.

[4] The implementation should also support committees that are one or two orders of magnitude larger, with only a mild increase in runtime.

includes a PVSS protocol (section 3.3), in which shareholders generate public keys independently, and the encrypter sends encryptions of shares of the secret to the shareholders, together with NIZK proofs that the ciphertexts are well-formed and indeed encrypt shares. These early schemes can be made non-interactive, by using NIZKs with the PVSS protocol in [28], or by applying the Fiat-Shamir heuristic to the Σ-protocols in [54].

Later PVSS works focused primarily on improving the efficiency of non-interactive ZK proofs for the ciphertexts, and minimizing the assumptions underlying those proofs [23, 52, 12, 56, 21, 15, 51, 32, 33, 24, 37, 50, 31]. Below, we will focus on PVSS schemes that follow the GMW approach to PVSS, where shareholders receive shares encrypted under their own independently generated public keys. In [48], this approach to PVSS is called "threshold encryption with transparent setup". We can categorize these PVSS schemes according to what underlying encryption scheme they use to encrypt shares. For the most part, these schemes all use 1) Paillier encryption, 2) ElGamal encryption of scalars "in the exponent", 3) pairing-based encryption of elements of the source group of the bilinear map, or 4) lattice-based encryption.

Paillier encryption [45] might at first appear ill-suited to PVSS in the "threshold encryption with transparent setup" setting, as shareholders have different Paillier public keys, and therefore have incompatible plaintext spaces that make it awkward to prove relationships among shares. However, this problem can be overcome by using a common interval that is inside the plaintext spaces of all of the Paillier keys, and using a proof system that proves (among other things) that the encrypted message is indeed within this interval. Camenisch and Shoup [15] build an encryption scheme with verifiable encryption and decryption, based on Paillier's decision composite residuosity assumption, that uses such an "interval" approach; the Σ-protocols for verifiable encryption and decryption each require only $O(1)$ exponentiations.[5] Recently, Lindell et al. [37] used essentially a version of Camenisch-Shoup to construct a PVSS scheme with $O(k)$ exponentiations per committee member (during re-sharing), for committees of size k (see Section 6.2).[6] Later schemes using variants of Paillier to encrypt PVSS shares include [51, 33, 24]. All of these PVSS schemes have the usual disadvantage of schemes related to Paillier, namely that exponentiations are expensive, as the exponentiations are over a group whose size should in principle be about $\exp(O(\lambda^3))$ for security parameter λ to maintain sufficient security against the number field sieve, and which in practice is much larger than, say, an elliptic curve group with comparable security (against classical computers). Also, the size of the proofs is linear in the size of the ciphertexts.

PVSS schemes that encrypt shares "in the exponent" include [52, 37, 31]. For example, Groth's PVSS scheme [31, 30], affiliated with the Dfinity blockchain, shares the secret for BLS signing [10] by dividing it "into small chunks, which can be encrypted in the exponent and later extracted using the Baby-step Giant-step method". The scheme employs a weak range proof to demonstrate that the chucks in the exponent are small enough to be recovered. The scheme has numerous optimizations, such as using the same randomness for ciphertexts in the multi-receiver setting. The paper [31] mentions an implementation, but does not provide details.

Bilinear-map-based PVSS schemes can verifiably encrypt source group elements, as opposed to scalars [20, 55]. An advantage of these schemes is that proofs of smallness – such as those needed in Camenisch-Shoup and Groth's PVSS scheme – are unnecessary, as the bilinear map makes verifiable encryption very natural [9, 22]. A disadvantage is that these schemes are limited to settings where

---

[5] In earlier work, Fouque and Stern [21] informally present a somewhat similar scheme.

[6] Lindell et al. also constructed a scheme that avoids Paillier, but with much higher bandwidth.

one is content to have the secret be a source group element – e.g., as when the secret is being used as a signing or decryption key in a pairing-based cryptosystem.

Lattice-based encryption schemes can encrypt large scalars, and have encryption and decryption procedures that are much faster than group-based schemes.[7] The main disadvantage of lattice-based schemes is high bandwidth, as lattice-based ciphertexts and public keys are in the order of kilobytes. The high bandwidth issue, however, can often be amortized away, since many plaintexts can be packed into a single ciphertext, as in the Peikert-Vaikuntanathan-Waters encryption scheme [47]. In principle, ciphertext expansion in lattice-based schemes can be arbitrarily small [13]. Also, very small ciphertext expansion (e.g., close to 2) can be compatible with very high performance that can be orders of magnitude better than Paillier-based schemes [25]. (See also [44, 43], cf. [53].)

Proving that lattice-based ciphertexts are well-formed requires proofs of smallness (for vectors that should be small, such as the secret key, encryption randomness). Some lattice-based schemes [36, 18] have used the approach of decomposing the coefficients of the vectors into their binary representations, and then proving that each purported bit in the representation is indeed in $\{0, 1\}$. Alternatively, one can use an approach somewhat similar to Camenisch-Shoup: a Σ-protocol that proves that a vector is inside a certain ball by revealing a statistically masked version of that vector. In the lattice setting, Lyubashevsky [38] showed how to use rejection sampling to reduce the required size gap between the masking vector and masked vector. Some other works on proofs of smallness are: [5, 19].

In this paper, we are motivated in part by the blockchain setting, where PVSS can help enable a blockchain to "keep a secret" [8] that it can use to sign or decrypt conditioned upon events, but where bandwidth is at a premium. Currently, blockchains almost exclusively use proof systems based on QAPs [46, 29] or bulletproofs [14], because these have the most concise proofs.[8]

1.2 An Overview of Our PVSS Construction

We assume we have a PKI, in which each party (and potential shareholder) has independently generated its own key pair for public-key encryption. Based on this PKI, our goal is to design a practical non-interactive PVSS scheme that allows a dealer to share a secret by verifiably (in zero-knowledge) encrypting shares of the secret to a "committee" of shareholders under their keys. The scheme should also allow each committee member to act as a dealer and verifiably "re-share" its share to the next committee of shareholders. We use Shamir secret sharing, though essentially any linear secret sharing will do.

Our PVSS scheme arises out of two design choices – namely, 1) to use lattice-based encryption, and 2) to use bulletproofs. Below, we explain these choices and their consequences.

Lattice-based encryption Lattice-based encryption is a good fit for PVSS, not only because it is exceptionally fast, but also because its disadvantages turn out *not* to be big problems in the PVSS setting. One apparent disadvantage is that lattice-based encryption has long public keys and ciphertexts. However, in the multi-receiver setting of PVSS, this disadvantage can be amortized away by using the Peikert-Vaikuntanathan-Waters (PVW) encryption scheme [47]. Another apparent disadvantage is that, for lattice-based PVSS, to prove that ciphertexts are well-formed, we

---

[7] Of course, this statement refers to basic, possibly additively homomorphic lattice-based encryption schemes, not fully homomorphic encryption.

[8] Despite being compact, bulletproofs have linear verification complexity. The Dory scheme [35] is similar to bulletproofs, but with logarithmic verification complexity.

require zero-knowledge proofs of smallness – e.g., that the "noise" in the ciphertexts is small. However, as we have seen in Section 1.1, PVSS and verifiable encryption schemes based on Paillier and ElGamal "in the exponent" also employ weak range proofs, and therefore they have no advantage over lattices here.

We briefly review the PVW lattice-based encryption scheme, as used in our PVSS scheme. The scheme uses a public random matrix A that is common to all parties. Each party i generates a secret vector s_i, and sets $\mathbf{b}_i = \mathbf{s}_i \cdot A + \mathbf{e}_i$ to be its public key.[9] The parties' public key vectors (let us suppose that there are k of them) are collected into a matrix B. The collective public key of the PVSS system is $\begin{bmatrix} A \\ B \end{bmatrix}$. The encryption of a message vector $\mathbf{m} = (m_1, \ldots, m_k) \in \mathbb{Z}_q^k$ is

$$\begin{bmatrix} A \\ B \end{bmatrix} \mathbf{r} + \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix}, \tag{1}$$

where $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$ are vectors with small coefficients and all operations take place in $\mathbb{Z}_q$. A committee member will use this scheme to encrypt k re-shares of its share to the next k-member committee.[10]

Note how well the PVW scheme is suited to the multi-receiver setting. In the basic setting of (single-user) Regev encryption [49], each user has its own matrix A as part of its public key, while here A is amortized across all parties. Moreover, note that an encryption to an extra user costs just an extra element in $\mathbb{Z}_q$. When the number of users becomes large, the ciphertext expansion factor becomes a small constant.

As far as we know, however, the PVW lattice-based encryption scheme has not been used in the multi-receiver setting before. (Instead, one uses PVW to encrypt a vector to a single user who effectively holds k secret keys.) Indeed, proving the security of PVW in the multi-receiver setting is subtle, since when decrypting each user implicitly obtains the inner product of $\mathbf{s}_i$ and $\mathbf{r}$, which leaks something about $\mathbf{r}$. One therefore needs to show that, for practical parameters, the secrets are still hidden despite the leakage. We cover this issue in Section 2.3.

Bulletproofs Our second design choice is to use bulletproofs. We are aiming for a PVSS scheme that can be used on a blockchain, as blockchains provide an especially compelling platform for PVSS. Linear-size proofs are not suitable for blockchains, as such proofs (which might appear in many blocks) need to be downloaded and verified by everyone that is confirming the blockchain state. For this reason, proof systems in use on actual blockchains are almost exclusively based on QAPs [46, 29] or bulletproofs [14]. Bulletproofs have some advantages over proof systems based on QAPs, such as being based on more natural assumptions, not requiring bilinear maps, and having only linear (versus quasi-linear) prover time complexity. Bulletproofs also work over small groups (a feature not shared by PVSS schemes based on Paillier encryption).

Recently, Bootle et al. [11] described a variant of bulletproofs based on lattice problems. In this variant, the proofs are not as compact, but proof generation and verification presumably would be faster. As future work, it may be interesting to investigate how using this variant affects the performance of our scheme.

---

[9] In the real scheme, each user creates several such vectors, but we defer this discussion to the body of the paper.

[10] For convenience, we have described the system as having only k members total, but consecutive k-member committees could be non-overlapping subsets of a larger set of parties.

Using lattice-based encryption and bulletproofs together Now our goal is to construct a proof system, ultimately based on bulletproofs, that allows a shareholder to prove that incoming and outgoing PVW ciphertexts correctly encrypt re-shares associated to its share.

As a first step to make our encryption scheme and bulletproofs compatible, we set our LWE modulus q to be the order of the bulletproof group. The plaintext space of our encryption scheme – i.e., the space the shares live in – is also $\mathbb{Z}/(q\mathbb{Z})$.[11] Now we "simply" need to create a commitment of the messages and prove that the ciphertext encrypts them. After this, all the proofs can be done using bulletproofs. The main contribution of this work is a collection of techniques, optimized for efficiency, to prove that a lattice encryption is valid and that the message corresponds to some DL committed value.

In more detail, we create a Pedersen commitment to all the coefficients of $\mathbf{r}, \mathbf{e}_i$, and $\mathbf{m}$. We now would like to prove that the committed values satisfy the linear relationship in (1). Also, very importantly, we need a proof that $\mathbf{r}$ and $\mathbf{e}_i$ have small coefficients. Proving exact relationships is the bread and butter of bulletproofs. We handle proofs of smallness in a multi-stage process that carefully calibrates the transition from "lattice world" to "bulletproof world". Namely, in some cases, we first reduce the dimension of the vectors involved, and instead prove that this dimension-reduced vector has small coefficients. This dimension reduction in turn reduces the number of exponentiations we eventually need to perform in the bulletproof world. Before moving to bulletproof world, we also invoke a lattice-based (without bulletproofs) proof of smallness with a large gap. While this proof is "slacky", it is sufficient to prove certain expressions do not "wrap" modulo q, so that we can now consider these expressions over $\mathbb{Z}$. Now that we have reduced the dimension and are assured that mod-q statements can be lifted to statements over $\mathbb{Z}$, we can use bulletproofs to prove the exact l_2 norm of the vectors. We provide additional techniques to hide the exact l_2 norm if only a bound on the norm is desired. The bulletproofs for the linear relationships and for smallness are aggregated to the extent possible. Details are provided below.

Dimension reduction and slacky lattice-based proofs of smallness Our dimension reduction technique is based on the Johnson-Lindenstrauss lemma [34]. The idea is that for all vectors $\mathbf{v}$, we have $\|\mathbf{v}\| \approx \frac{1}{\sqrt{n}} \cdot \|\mathbf{v}R\|$, where R is an n-column matrix whose entries are chosen from a normal distribution of variance 1. When R is chosen in this way, the distribution of $\|\mathbf{v}R\|^2$ follows the chi-squared distribution and its confidence intervals are known. When the coefficients of R are instead chosen from a discrete distribution over $\{0, \pm 1\}$ where the probability of 0 is $1/2$, one can heuristically verify that these confidence intervals are bounded by the continuous ones.[12] If we would like to be in a $1 - 2^{-128}$ interval, then R can have around 256 columns and then the ratio between the smallest value of $\|\mathbf{v}R\|$ and the largest is under 4. This means that we can project an arbitrary-dimensional vector into just 256 dimensions and prove the ℓ_2 norm of the resulting vector, and be within a small factor of the correct result. And, of course, the projection operation is linear. The concrete bounds for the dimension-reduction technique are described in Section 3.2.

Everything in the above discussion was based on the fact that we were working over the integers, rather than over $\mathbb{Z}_q$. When working modulo q, it is possible that $\mathbf{v}$ has a large norm, but $\mathbf{v}R \bmod q$

---

[11] Unlike the more standard LWE encryption in which the message also needs to be small, we use a version of the scheme implicit in [27] where the messages can be arbitrarily large in $\mathbb{Z}_q$, but the length of $\vec{m}$ has to increase to encode all of the message. We describe this in Section 2.2.

[12] There are concrete bounds for tails of some of these distributions (e.g. [1]), but they are asymptotic and are looser than necessary for our concrete parameters.

has a small one. This event can clearly only occur if the coefficients of $\mathbf{v}$ are large enough that multiplication with R causes a wraparound modulo q. It is therefore important to show that this does not happen, and we do this in the manner as in the lattice-based proofs from [41]. We now explain how the technique applies to our context. The main idea is to show that all the elements of $\mathbf{v}$ are not too big. This seems a bit circular, as our goal is already to prove that $\|\mathbf{v}\|$ is small. But our requirement now is not to get a very tight bound on the norm, but simply to show that all the elements of $\mathbf{v}$ are small enough to not cause a wrap around. For this, one employs a simple fact that is sometimes useful in lattice cryptography [7, Lemma 2.3], which states that if a vector $\mathbf{v}$ has a large coefficient, then for any $y \in \mathbb{Z}_q$, $\langle \mathbf{v}, \mathbf{r} \rangle + y \bmod q$ has a large coefficient with probability at least $1/2$, where the coefficients of $\mathbf{r}$ are randomly chosen from $\{0, \pm 1\}$ as above. One would therefore prove that the coefficients of $\mathbf{v}$ are small by committing to some masking vector $\mathbf{y}$, receiving a 128-column matrix R as a challenge, and then outputting $\mathbf{v}R + \mathbf{y}$. The purpose of $\mathbf{y}$ is to hide $\mathbf{v}$, and so some rejection sampling [39] is necessary to keep the distribution of $\mathbf{v}R + \mathbf{y}$ independent of $\mathbf{v}$. Note that the gap between the actual ℓ_∞ norm of $\mathbf{v}$ and that of what we can prove is increased by a factor of at least the dimension of $\mathbf{v}$. This is because the ℓ_∞ norm is not well-preserved under transformations and also due to the masking which is needed because we will actually be outputting the value $\mathbf{v}R + \mathbf{y}$. This is much larger than the factor of approximately 4 in the ℓ_2-dimension reduction above, and this is why we only employ this technique for proving that no wrap-around occurs.

In the context of our encryption scheme, instead of proving that the *long* vectors $\mathbf{e}_i$ (with dimension dependent on the number of users) have small norm, we can instead prove that the *short* 256-dimensional vector

$$\left(\begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} - \begin{bmatrix} A \\ B \end{bmatrix} \mathbf{r} - \begin{bmatrix} \mathbf{0} \\ \mathbf{m} \end{bmatrix} \right) \cdot R \tag{2}$$

has small norm. Also, we prove that $\mathbf{r} \cdot R$ has small norm instead of $\mathbf{r}$. Other purportedly short vectors are handled in the same way. For example, each of the k new committee members needs to prove that the public key $\mathbf{b}_i = \mathbf{s}_i A + \mathbf{e}_i$ is properly created. The combination of these techniques is described throughout Section 3.

Bulletproofs and precise proofs of smallness Suppose now that we want to prove a tighter upper bound β on the squared ℓ_2 norm of a vector $\mathbf{v} = (v_1, \ldots, v_k)$. (Proving tighter bounds allows us to use tighter parameters in our lattice-based encryption scheme.) Assume β is an integer. First, we pick a vector $\mathbf{x}$ such that the squared l_2 norm of the concatenated vector $\mathbf{v}\|\mathbf{x}$ is exactly β. For the vector $\mathbf{x}$, 4 coefficients suffice, as the non-negative integer $\beta - \sum v_i^2$ can always be expressed as the sum of at most 4 squares. We then use the "slacky" techniques above to prove that there is no wraparound modulo q in the computation of the squared l_2 norm of $\mathbf{v}\|\mathbf{x}$. Then, we commit to $\mathbf{v}\|\mathbf{x}$, and use bulletproofs to prove the exact quadratic relation.

We can aggregate the relations that we prove using bulletproofs – e.g., these exact proofs of smallness are combined together with proofs of the linear equations in (1).

1.3 Organization

In Section 2 we describe our lattice-based encryption scheme, and discuss the extension of PVW to the multi-receiver setting. In Section 3 we present the size-proof protocols that we use in our scheme and their parameters. In Appendix A we describe the various sub-protocols that the parties

run locally, for key-generation, encryption, decryption, and secret re-sharing. In Appendix B, we describe how to aggregate all exact relations into a small number of bulletproofs. In Section 4, we provide details about our implementation. Finally, in Appendix C we explain how we put all these components together in a (proactive) publicly-verifiable secret-sharing protocol. For self-containment, we also review in the appendix some slight variations of the Bulletproof protocol that we use.

2 The Underlying Encryption Scheme

In this section we develop the encryption scheme that is used by our protocol, starting from a (variant of) PVW encryption [47] and specializing it to our needs.

Below we denote integers and scalars by lowercase letters, vectors by bold lowercase letters, and matrices by uppercase letters. Vectors are considered row vectors by default. (Parameters are denoted by either lowercase English or lowercase Greek letters). For integers x, q, we denote by $x \bmod q$ the unique integer $x' \in [-\frac{q}{2}, +\frac{q}{2})$ such that $x' = x \pmod q$. We denote vectors by bold-lowercase letters, and it will usually be evident from context whether they are row or column. The l_2 and l_∞ norms of a vector $\mathbf{v}$ are denoted $\|\mathbf{v}\|_2, \|\mathbf{v}\|_\infty$, respectively. For a matrix A, we let $\|A\|_2$, (resp. $\|A\|_\infty$) denote the largest l_2, (resp. l_∞) norm of any row in A.

2.1 Learning with Errors (LWE)

The LWE problem was introduced by Regev [49]. In the decision variant, the adversary is given pairs (A, B) where A is chosen uniformly from $\mathbb{Z}_q^{k \times m}$, and it needs to distinguish the cases where:

- B is chosen uniformly at random in $\mathbb{Z}_q^{n \times m}$, or
- B is set as $B := SA + E \bmod q$, where the entries of S, E are chosen from some public distributions χ_s, χ_e over $\mathbb{Z}_q$ that output integers of magnitude much smaller than q with overwhelming probability.

This problem is believed to be hard for many different settings of the parameters $k, m, n, q, \chi_s, \chi_e$. For some of them it is even proven to be as hard as solving some "famous" lattice problems in the worst case. In this work we assume that this problem is (exponentially) hard when the χ's are uniform distributions on integers is some symmetric interval $[\pm \sigma]$ with $\sigma \ll q/2$. The specific parameters that we use were chosen according to the LWE hardness estimator of Albrecht et al. [3]. See more details in Appendix A.4. Also in our protocol we always use $k = m$, so we drop the distinction between these parameters in the sequel.

2.2 Variants of Regev Encryption

In [49], Regev described a public-key encryption scheme whose security is based on the hardness of decision-LWE. Later, Peikert, Vaikuntanathan and Waters (PVW) described in [47] a variant with improved plaintext-to-ciphertext expansion ratio. Our protocol is based on a variant of the PVW construction. Underlying it is the following "approximate encryption" scheme, where decryption only recovers a noisy version of the plaintext:

Key-generation. The key-owner chooses a random $A \leftarrow \mathbb{Z}_q^{k \times k}$, $S \leftarrow \chi_s^{n \times k}$ and $E \leftarrow \chi_e^{n \times k}$ and computes $B := SA + E \bmod q$. The secret key is S and the public key is (A, B), which is pseudorandom under the decision LWE assumption.

Encryption. To encrypt an n-vector $\mathbf{x} \in \mathbb{Z}_q^n$, the encryptor chooses $\mathbf{r} \leftarrow \chi_s^k$, $\mathbf{e}_1 \leftarrow \chi_e^k$, $\mathbf{e}_2 \leftarrow \chi_e^n$, and sets $\mathbf{c}_1 := A\mathbf{r} + \mathbf{e}_1 \bmod q$ and $\mathbf{c}_2 := B\mathbf{r} + \mathbf{e}_2 + \mathbf{x} \bmod q$. The ciphertext is $(\mathbf{c}_1, \mathbf{c}_2)$, which is again pseudorandom under the decision LWE assumption.

Decryption. To decrypt (approximately), the key-owner outputs $\mathbf{x}' := \mathbf{c}_2 - S\mathbf{c}_1 \bmod q$. Substituting all the terms one can check that

$$\mathbf{x}' = \big((SA + E)\mathbf{r} + \mathbf{e}_2 + \mathbf{x}\big) - S(A\mathbf{r} + \mathbf{e}_1) \;=\; \mathbf{x} + \overbrace{E\mathbf{r} + \mathbf{e}_2 - S\mathbf{e}_1}^{\mathbf{e}'},$$

where for appropriate choices of χ_s, χ_e we will have $\|\mathbf{e}'\|_\infty \ll q$.

Plaintext Encoding To be able to fully recover the plaintext, Regev encryption uses some form of error-correction that allows the decryptor to compute $\mathbf{x}$ from the noisy $\mathbf{x}'$. Most variants of Regev encryption use encoding based on scaling, but for us it is more convenient to use a different form of encoding[13] (which was implicit in the homomorphic encryption scheme of Gentry, Sahai and Waters [27]). We encode a plaintext vector $\mathbf{x}^* \in \mathbb{Z}_q^n$ by a higher-dimension $\mathbf{x} \in \mathbb{Z}_q^{\ell n}$ that includes not just $\mathbf{x}^*$ but also a large multiple of it. Let $\Delta := \lfloor \sqrt{q} \rfloor$ and $\mathbf{g} := (\Delta, 1) \in \mathbb{Z}_q^2$. The dimension-$n$ vector $(x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ is encoded in the vector $(x_1 \mathbf{g} | \ldots | x_n \mathbf{g}) \in \mathbb{Z}_q^{2n}$.

More generally, we could use a parameter $\ell \geq 2$ and set $\Delta := \lfloor \sqrt[\ell]{q} \rfloor$ and the "gadget vector" $\mathbf{g} := (\Delta^{\ell-1}, \ldots, \Delta, 1) \in \mathbb{Z}_q^\ell$. We then encode a vector $(x_1, \ldots, x_n)$ in the higher-dimension $(x_1 \mathbf{g} | \ldots | x_n \mathbf{g}) \in \mathbb{Z}_q^{n\ell}$. The larger we set the parameter ℓ, the more redundant the encoded vector becomes, which lets us tolerate larger noise and still recover the original vector. (On the other hand, we need to increase the number of rows in the secret key from n to ℓn.) Specifically, for each entry x_i in the original plaintext vector, the approximate-decryption above yields a noisy ℓ-vector $\mathbf{x}' = x\mathbf{g} + \mathbf{e} \bmod q$, and x_i can then recovered using the decoding procedure from Fig. 1.

Decode($(x_1', \ldots, x_\ell') \in \mathbb{Z}_q^\ell$):	# $x_i' = x\Delta^{\ell-i} + e_i \bmod q$
1. For $i = 1, \ldots, \ell - 1$ let $y_i := x_{i+1}' - \Delta x_i' \bmod q$	# $y_i = e_i - \Delta e_{i+1}$ (**w/o mod-q reduction**)
2. Set $z := \sum_{i=1}^{\ell-1} \Delta^{\ell-i-1} \cdot y_i$	# telescopic cancellation, $z = e_1 - \Delta^{\ell-1} e_\ell$
3. Set $e := z \bmod \Delta^{\ell-1}$	# $e = e_1$
4. Output $(x_1' - e)/\Delta^{\ell-1}$	# $= x$

Fig. 1. The plaintext decoding procedure

As long as all the e_i's are bounded in magnitude below $q/2(\Delta+1) \approx \Delta^{\ell-1}/2$, then the equality $y_i = e_i - \Delta e_{i+1}$ in Row 2 holds not only modulo q but also over the integers. In that case we also have $z = e_1 - \Delta^{\ell-1} e_\ell$ over the integers, and since $|e_1| < \Delta^{\ell-1}/2$ then also $e = e_1$ in Row 3 holds over the integers, so we recover the correct output x.

For our implementation we stuck to the setting $\ell = 2$, which is somewhat simpler to implement. In general, however, setting a slightly larger value (such as $\ell = 4$) may lead to somewhat better parameters, since it can tolerate larger noise and therefore smaller lattice dimension for the same security level. We leave exploring this direction to future work.

---

[13] The reason that this encoding method is better for us, is that it allows us to work only with $\mathbb{Z}_q$ elements. In other variants of Regev encryption one usually must work with both $\mathbb{Z}_q$ and $\mathbb{Z}_p$ for some $p \ll q$.

2.3 The Multiparty Setting

A very useful property of the scheme above is that the i'th plaintext value x_i can be recovered using only rows $\{1 + (i-1)\ell, \ldots, i\ell\}$ of the secret key matrix S (indexing start at 1). To wit, denote by S_i the sub-matrix of S consisting only of these rows, and let $\mathbf{c}_{2,i}$ be the sub-vector of $\mathbf{c}_2$ consisting of entries $\{1 + (i-1)\ell, \ldots, i\ell\}$, then x_i can be recovered by setting $\mathbf{x}' := \mathbf{c}_{2,i} - S_i\mathbf{c}_1 \in \mathbb{Z}_q^\ell$, then using the decoding procedure from Fig. 1.

It is therefore possible to use the encryption scheme above in a multiparty setting, where all parties share the same random matrix A (a common-random-string which is chosen by a trusted party during setup), and each party i chooses its own secret key $S \leftarrow \chi_s^{\ell \times k}$ and noise $E_i \leftarrow \chi_e^{\ell \times k}$, and computes its own public key $B_i := S_iA + E_i \mod q$.

The global public key is then set to include the matrix A, followed by all the B_i's in order (which are viewed as sub-matrices of the public-key matrix B from above). Encryption works just as above, with the plaintext vector $\mathbf{x} \in \mathbb{Z}_q^n$ viewed as having one plaintext element $x_i \in \mathbb{Z}_q$ destined to each party i. For decryption, each party i uses its secret key S_i to get the noise vector $\mathbf{x}_i' = x_i\mathbf{g} + \mathbf{e}_i$, then apply the decoding procedure from Fig. 1 to recover x_i from $\mathbf{x}_i'$.

LWE with Leakage The multiparty setting above brings up a new problem: what happens when some of the parties are dishonest and deviate from the prescribed distribution for choosing their public keys? The issue is that encryption uses the same vector $\mathbf{r}$ for encrypting all the plaintext elements to all the parties. When party i is dishonest and B_i is chosen adversarially, seeing $B_i\mathbf{r} + \mathbf{e}$ may leak information about $\mathbf{r}$ to the adversary, potentially making it possible for it to distinguish some other $B_j\mathbf{r} + \mathbf{e}_j$ from random and maybe learn something about the plaintext encrypted for party j.

Luckily, some characteristics of our application make it possible to counter this threat. In particular, each party i in our protocol is required to prove that its public key is "well formed". Namely it must provides a proof of knowledge of S_i, E_i such that $B_i := S_iA + E_i \mod q$, and moreover where the l_2 norm of the rows in S_i, B_i is bounded by some known bounds β_s, β_e, respectively. In this setting, we can reduce security to plain LWE (without any leakage), as long as the encryptor chooses $\mathbf{e}_2$ from a somewhat wider distribution than $\mathbf{e}_1$.

Specifically, fix the LWE parameters $k, n, q, \chi_s, \chi_{e1}$. Also let $\rho_s, \rho_s \in \mathbb{R}$ be factors related to χ_s, χ_{e1}, such that for any fixed $\mathbf{v} \in \mathbb{Z}_q^k$, choosing $\mathbf{s} \leftarrow \chi_s^k$ and $\mathbf{e} \leftarrow \chi_e^k$ we get

$$|\langle \mathbf{v}, \mathbf{s} \rangle| \leq \rho_s \cdot \|\mathbf{v}\|_2 \text{ and } |\langle \mathbf{v}, \mathbf{e} \rangle| \leq \rho_e \cdot \|\mathbf{v}\|_2,$$

except perhaps with a probability negligible in κ. Let $\beta_s, \beta_e \in \mathbb{R}$ be the bounds that the parties in our protocol must prove, and let χ_{e2} be another noise distribution over $\mathbb{Z}$, which is wide enough so that χ_{e2} is statistically close[14] to $\chi_{e2} + \delta$ for any fixed integer offset $\delta \leq \lceil \rho_s\beta_e + \rho_e\beta_s \rceil$. Then consider the following game between an adversary and a challenger:

- The challenger chooses and sends to the adversary a random matrix $A \in \mathbb{Z}_q^{k \times k}$.
- The adversary chooses arbitrary $S \in \mathbb{Z}_q^{n \times k}$ and $E \in \mathbb{Z}_q^{n \times k}$ subject to the constraint that the l_2 norm of each row in S, E is bounded by β_s, β_e, respectively.
  The adversary computes $BL = SA + E \mod q$ and sends S, E, B to the challenger.[15]

---

[14] Up to a distance negligible in κ.

[15] The adversary sends not only B but also S, E to the challenger, since in our protocol it will have to prove knowledge of these matrices so they can be extracted from it.

- The challenger chooses $\mathbf{r} \leftarrow \chi_s^k$, $\mathbf{e}_1 \leftarrow \chi_{e1}^k$, $\mathbf{e}_2 \leftarrow \chi_{e2}^k$, and a uniformly random vector $\mathbf{u} \in \mathbb{Z}_q^k$. It also tosses a coin $\sigma \in \{0,1\}$.
  If $\sigma = 1$ then the challenger sets $\mathbf{c}_1 := A\mathbf{r} + \mathbf{e}_1 \bmod q$ and $\mathbf{c}_2 := B\mathbf{r} + \mathbf{e}_2 \bmod q$.
  If $\sigma = 0$ then the challenger sets $\mathbf{c}_1 := \mathbf{u}$ and $\mathbf{c}_2 := S\mathbf{c}_1 + \mathbf{e}_2 \bmod q$.
- The challenger sends $(\mathbf{c}_1, \mathbf{c}_2)$ to the adversary, and the adversary outputs a guess σ' for σ.

Lemma 2.1. *Let the parameters $k, n, q, \chi_s, \chi_{e1}$, and $\rho_s, \rho_s, \chi_{e2}$ be as above. Then under the hardness of decision-LWE with parameters k, n, χ_s, χ_{e1}, the adversary in the game above has only a negligible advantage in guessing the value of σ.*

Proof. Substituting all the variables above, we have

$$(A\mathbf{r} + \mathbf{e}_1, B\mathbf{r} + \mathbf{e}_2) = (A\mathbf{r} + \mathbf{e}_1, (SA + E)\mathbf{r} + \mathbf{e}_2) \tag{3}$$
$$= (A\mathbf{r} + \mathbf{e}_1, S(A\mathbf{r} + \mathbf{e}_1) - S\mathbf{e}_1 + E\mathbf{r} + \mathbf{e}_2)$$
$$\overset{(s)}{\approx} (A\mathbf{r} + \mathbf{e}_1, S(A\mathbf{r} + \mathbf{e}_1) + \mathbf{e}_2) \overset{(c)}{\approx} (\mathbf{u}, S\mathbf{u} + \mathbf{e}_2). \tag{4}$$

The last relation follows directly from the hardness of decision LWE with these parameters. To see why the penultimate relation holds, note that $\|E\mathbf{r} - S\mathbf{e}\|_\infty \leq \rho_s \beta_e + \rho_e \beta_s$ except with a negligible probability, and therefore $\mathbf{e}_2$ is statistically close to $E\mathbf{r} - S\mathbf{e} + \mathbf{e}_2$.

Semantic Security in the Multiparty Setting Lemma 2.1 implies that we can get semantic security for the honest parties in our protocol, even if the dishonest parties deviate from the prescribed distribution for choosing their public keys. (As long as they successfully prove knowledge of S, E as above.)

To that end, we modify the encryption procedure from Section 2.2 so that it uses the wider noise χ_{e2} rather than χ_e when choosing the noise vector $\mathbf{e}_2$. We then view the CRS matrix *together with all the honest public keys* as the matrix A from the lemma, and the dishonest public keys are viewed as the matrix B from the lemma. We note that with this view, the matrix A is pseudorandom from the adversary's perspective. Lemma 2.1 tells us that $A\vec{r} + \vec{e}_1$ is indistinguishable from random even given $B\mathbf{r} + \mathbf{e}_2$, and the encryption scheme uses the part of $A\vec{r} + \vec{e}_1$ corresponding to the honest parties' public keys to mask the plaintext values for these parties, hence we get semantic security.

How Wide Must χ_{e2} Be? Lemma 2.1 requires that χ_{e2} is very wide, enough to "flood" the term $\delta := E\mathbf{r} - S\mathbf{e}$, i.e., larger by at least the (statistical) security parameter. In our application, however, making χ_{e2} very wide is costly: For security of 128 bits, adding one bit to the width of χ_{e2} increases by about 40 the dimension of the LWE secret that we need to use. (So making it (say) 50-bit wider will increase the dimension by almost 2000.)

However, in our setting it seems likely that setting χ_{e2} only slightly larger than (the expected size of) δ is safe, since the encryption randomness and noise are only used once, and the adversary gets at most $\mathsf{t} < 1000$ samples from the "leakage". We therefore took a pragmatic approach, making χ_{e2} only large enough so the distributions χ_{e2}^{t} and $\chi_{e2}^{\mathsf{t}} + \delta$ are "not too far". Specifically, we set it large enough to ensure that the Rényi divergence between them is a small constant. While this is not enough to prove that decision-LWE remains hard, it is enough to show that the search problem remains hard. As we are not aware of any attack on decision-LWE that does not go via full recovery of the LWE secret, we take it as a strong indication of security even in our setting.

In more detail, in Appendix A.4 we will establish a high-probability bound on the l_∞ norm of δ (call it μ). We use the heuristic of modeling χ_{e2} as a zero-mean Normal random variable with variance σ^2 (where σ is the parameter that we need to set). Using analysis similar to [4, 2], we bound the Rényi divergence of order α between χ_{e2}^{t} and $\chi_{e2}^{\mathsf{t}} + \delta$ by $\rho := \exp\left(\alpha\pi\mathsf{t} \cdot (\mu/\sigma)^2\right)$, and use the probability-preservation property of Rényi divergence to conclude that for any event that depends on a vector $\mathbf{v}$, we have

$$\Pr_{\mathbf{v} \leftarrow \chi_{e2}^t} [E(\mathbf{v})] \geq \Pr_{\mathbf{v} \leftarrow \chi_{e2}^t + \delta}[E(\mathbf{v})]^{\alpha/(\alpha-1)}/\rho.$$

In particular the above holds for the event in which the adversary finds the LWE-secret $\mathbf{r}$. Setting $\sigma = b\sqrt{2\pi t}$ and using the (say) $\alpha = 2$, yields $\rho = \exp(1) = e$ and hence $\Pr_{\mathbf{v} \leftarrow \chi_{e2}^t}[E(\mathbf{v})] \geq \Pr_{\mathbf{v} \leftarrow \chi_{e2}^t + \delta}[E(\mathbf{v})]^2/e$. By the hardness of search-LWE, the probability on the left-hand side is negligible, and hence so is the probability on the right-hand side.

2.4 An Optimization: Using Module-LWE over Small Rings

As is common in lattice-based cryptosystems, we gain efficiency by using operations over higher-degree algebraic ring rather than directly over the integers. In our multiparty setting parties use ℓ-row public key (to enable or input encoding), so instead we use operations over a ring of dimension ℓ, namely $R_\ell = \mathbb{Z}[X]/(X^{\ell+1})$. (We also denote $R_{\ell,q} = R/qR = \mathbb{Z}_q[X]/(X^{\ell+1})$.) (Recall that our implementation uses $\ell = 2$, and more generally we may use slightly larger value such as $\ell = 4$.) This means that the parties' secret-key and noise vectors can now be specified using half as many scalars, so in our protocols the parties will need to commit and prove relations for half as many variables. The scheme thus needs to choose low-norm elements in R_ℓ, which is done by choosing their representation in the power basis using the same distributions $\chi_s, \chi_{e1}, \chi_{e2}$ over $\mathbb{Z}_q$. Below we use the same notations $\chi_s, \chi_{e1}, \chi_{e2}$ for both the $\mathbb{Z}$ distribution and the induced distributions over R_ℓ.

2.5 The Encryption Scheme in Our Protocol

Using all the components above, we describe here explicitly the encryption scheme as we use it in our protocol:

Parameters. Denote by n the number of parties and $t < n/2$ bound the number of dishonest parties. For LWE we have a modulus q, The dimension k of the LWE secrets and noise vectors, and the secret- and noise-distributions $\chi_s, \chi_{e1}, \chi_{e2}$.
We also have the redundancy parameter ℓ, and we denote $\mathsf{n} = n\ell$, $\mathsf{t} = t\ell$, and $\mathsf{k} = k\ell$. Let $\Delta = \lfloor \sqrt[\ell]{q} \rfloor$ and let the "gadget vector" be $\mathbf{g} = (\Delta^{\ell-1}, \ldots, \Delta, 1) \in \mathbb{Z}_q$, representing the element $g \in R_{\ell,q}$.

Common reference string. A random matrix $A \leftarrow R_q^{k \times k}$.

Key-generation. Each party i chooses the secret key and noise vectors in R_q^k, $\mathbf{s}_i \leftarrow \chi_s^k$ and $\mathbf{e}_i \leftarrow \chi_{e1}^k$, sets $\mathbf{b}_i := \mathbf{s}_i A + \mathbf{e}_i \in R_{\ell,q}$ as its public key, and broadcasts it to everyone.

Encryption. The global public key consists of the matrix A and all the $\mathbf{b}_i$'s. Let $B \in R_{\ell,q}^{n \times k}$ be a matrix whose rows are all the $\mathbf{b}_i$'s in order.
Given n plaintext scalars $x_1, \ldots, x_n \in \mathbb{Z}_q$, we encode them in a vector $\mathbf{x} = (x_1, \ldots, x_n)g \in R_{\ell,q}^n$. Namely we encode each x_i as the element $x_i g \in R_{\ell,q}$. The encryptor chooses three vectors $\mathbf{r} \leftarrow \chi_s^k$, $\mathbf{e}_1 \leftarrow \chi_{e1}^k$, and $\mathbf{e}_1 \leftarrow \chi_{e1}^k$, and computes the ciphertext vectors

$$\mathbf{c}_1 := A\mathbf{r}^T + \mathbf{e}_1^T \bmod q, \text{ and } \mathbf{c}_2 := B\mathbf{r}^T + \mathbf{e}_2^T + \mathbf{x}^T \bmod q.$$

Decryption. On ciphertext $(\mathbf{c}_1, \mathbf{c}_2)$ and secret key $\mathbf{s}_i$, party i uses the approximate decryption procedure to compute $\mathbf{y} := \mathbf{c}_2 - \langle \mathbf{s}_i, \mathbf{c}_1 \rangle \bmod q$.

This yields $\mathbf{y} = xg + e$ for some scalar $x \in \mathbb{Z}_q$ and small noise element $e \in R_{q,\ell}$, which can also be written as a vector equation $\mathbf{y} = x\mathbf{g} + \mathbf{e} \bmod q$. The decryptor then uses the decoding procedure from Fig. 1 w to recover the scalar x.

The discussion above implies that this scheme is correct as long as the decryption noise is smaller than $\Delta^{\ell-1}/2$, and and it offers semantic security for the honest parties under module-LWE (with leakage if χ_{e2} does not completely drown the other noise terms.)

3   Proofs of Smallness

Our scheme relies on parties committing to various vectors and broadcasting publicly-verifiable proofs about them. Some of the statements that are proven are simple linear constraints (e.g., when a party proves that it re-shared its secret properly). But most of the proofs that we use are proofs-of-smallness, when the prover needs to convince everyone that the norms of its vectors are bounded by some public bounds.

The main reason for proving smallness is that lattice-based cryptosystems only provide correctness guarantees when certain quantities are small enough. Another reason to use proofs-of-smallness is because the underlying proof systems that we use are only capable of proving constraints modulo some integer parameter P (e.g., discrete-logarithm-based commitments and proofs). To prove the same constraints over the integers, we augment these underlying proofs by also proving smallness of the relevant values, to establish that no wraparound modulo P occurs.

A publicly verifiable proof of smallness protocol lets a prover commit to a vector and convince everyone that the committed vector is smaller than some public bound. Such proofs are parametrized by the norm in question (l_2 or l_∞) and a gap parameter $\gamma \geq 1$. Completeness of such proofs for a bound b only holds when the vector of the honest prover has norm bounded by b/γ, while soundness ensures that even cheating provers cannot pass verification if their vector has norm larger than b. Such protocols can be modeled as special cases of the commit-and-prove functionality (e.g., [16]), except that the constraint enforced on honest parties is more strict than that for dishonest parties. This is captured in the functionality $\mathcal{SML}_l[\gamma]$ from Fig. 2.

Parameters: norm $l \in \{l_2, l_\infty\}$ and gap $\gamma \geq 1$

The functionality maintains a list $\overrightarrow{w}$ of (vector,commitment) pairs, initially empty.

Commitment.   Upon receiving (commit, $sid, \mathbf{w} \in \mathbb{Z}^d, c \in \{0,1\}^*$) from the prover, if $\overrightarrow{w}$ does not contain any pairs with the commitment value c then add the pair $(\mathbf{w}, c)$ to $\overrightarrow{w}$, and send the message (receipt, sid, c, d) to everyone.
Proof.   Given a message (prove, $sid, c, b \in \mathbb{R}$) from the prover, if $\overrightarrow{w}$ contains a pair $(\mathbf{w}, c)$ such that
   – either the prover is honest and $\|\mathbf{v}\|_l \leq b/\gamma$,
   – or the prover is dishonest and $\|\mathbf{v}\|_l \leq b$,
   then send the message (proof, sid, c, b) to everyone. (Otherwise, ignore the message.)

Fig. 2. The proof-of-smallness functionality $\mathcal{SML}_l[\gamma]$

3.1 Underlying Commit-and-Prove Systems

Our scheme makes extensive use of underlying commit-and-prove systems, that let parties commit to integer values and prove relations among these committed values. Specifically, these systems lets a prover convince everyone of the veracity of two types of constraints:

Linear constraints. The prover commits to the secret vector $\mathbf{x}$, then given the public vector $\mathbf{a}$ it reveal the scalar b and proves that $\sum_i a_i x_i = b \pmod{P}$.

Quadratic constraints. The prover commits to $(\mathbf{x}|\mathbf{y})$, then given the public offset vectors[16] $\mathbf{u}, \mathbf{v}$ it reveals the scalar b and proves that $\sum(x_i + u_i)(y_i + v_i) = b \pmod{P}$.

In our implementation we use Pedersen commitments to vectors, and small variations of the Bullet-proof protocol [14]. (In this case the parameter P is the order of the hard-discrete-logarithm group.) For the sake of self-containment, we describe the Bulletproof variants that we use in Appendix E. We use some homomorphic properties of these commitments in order to aggregate them, see more details in Appendix B. An alternative implementation could instead use lattice-based schemes, such as the BDLOP scheme from [6], at the cost of giving up on compactness.

When using these commit-and-prove protocols in our scheme, we model them as ideal commit-and-prove functionalities. That is, we argue security of our higher-level protocols in a model where the prover presents the committed vectors to a trusted party, who verifies publicly that they satisfy the relevant constraint modulo P. Formally, we have a functionality similar to the one in Fig. 2 that verifies these constraints.

We note that for the systems that we use, proving linear constraints is cheaper than proving quadratic constraints, roughly because the prover only needs to commit to $\mathbf{x}$ rather than to both $\mathbf{x}$ and $\mathbf{y}$. We therefore strive to only prove quadratic constraints on *low-dimension vectors*, which leads to noticeable savings. The main novel tool that we use for that purpose, and which we believe will find other applications, is in showing how to use the Johnson-Lindenstrauss lemma to reduce the dimension of the vectors on which we need to perform quadratic proofs. That is, we replace a quadratic proof on a high-dimension vector with a linear proof on that vector, combined with a quadratic proof on a low-dimension one (i.e. 256-dimensional). See more details later in this section.

l_2 Norm Proofs Modulo P. In our scheme we often use commit-and-prove protocols for quadratic constraints to prove the l_2-norm of a vector modulo P, which is not entirely straightforward. Naively, we could try to let the prover commit to $(\mathbf{x}|\mathbf{x})$ and then directly use the underlying quadratic proofs to prove that $\sum_i x_i^2 = b^2 \pmod{P}$. This naive protocol doesn't quite work, however, since a cheating prover may commit to two different vectors $(\mathbf{x}|\mathbf{x}')$ rather than to the same vector twice. One solution could be to add linear proofs to establish that $x_i = x'_i$ for all i, but that could become expensive (as it may require commitments to each x_i separately).

Instead, after the prover commits to $(\mathbf{x}|\mathbf{x}') \in \mathbb{Z}_P^{2d}$ and publishes the bound b, the verifier chooses at random an offset vector $\mathbf{u} \in \mathbb{Z}_P^d$, and the prover uses the underlying quadratic proof protocol to prove that $\sum_i (x_i + u_i)(x_i - u_i) = b^2 - \|\mathbf{u}\|^2 \pmod{P}$. It is easy to see that if a cheating prover commits to some $(\mathbf{x}|\mathbf{x}')$ with $\mathbf{x} \neq \mathbf{x}'$, then this last constraint would only hold with probability $1/P$. In our implementation we let the verifier choose only a single random scalar $u \in \mathbb{Z}_P$, then use the offset vector $\mathbf{u} = (1, u, u^2, \ldots, u^{d-1})$. Again it is easy to see that in this case, if $\mathbf{x} \neq \mathbf{x}'$ then the constraint only holds with probability at most d/P.

---

[16] See Section 3.1 for the reason for the offset vectors.

3.2 Tails of Distributions and the Johnson-Lindenstrauss Lemma

As we mentioned above, an important component in our scheme is projecting high-dimension vectors down to lower dimension using the Johnson-Lindenstrauss Lemma. Namely, instead of directly proving smallness of a high-dimension vector $\mathbf{w}$, we choose a random rectangular matrix R, prove smallness of the lower-dimension $\mathbf{v} = \mathbf{w}R$, and use Johnson-Lindenstrauss to argue that this implies also tight approximation for the norm of the original $\mathbf{w}$. (Specifically, the distribution $\mathcal{D}$ that we use for the entries of R has $\mathcal{D}(0) = 1/2$ and $\mathcal{D}(\pm 1) = 1/4$.)

To obtain very tight bounds, we use a heuristic that roughly states that the tail of the distribution on $\|\mathbf{w}R\|$ can be bounded as if the entries of R were chosen from the zero-mean continuous Normal distribution of the same variance. A strong justification for this heuristic comes from the analysis of Achlioptas [1], who proved that for an arbitrary vector $\mathbf{w}$ and $R \leftarrow \mathcal{D}^{n \times k}$, *all the moments* of the induced distribution over $\|\mathbf{w}R\|^2$ are bounded by the corresponding moments of the distribution $\|\mathbf{w}R'\|^2$ where the entries of R' are chosen from the corresponding zero-mean continuous Normal distribution. This intuitively implies that the tails of the continuous distribution are fatter, and so bounding them will imply bounds on the discrete distribution. This intuition generally holds except that the discretization may cause some minor discrepancies that vanish exponentially with the dimension k. See more discussion in Appendix D. This heuristic lets us use the following bounds when setting concrete parameters:

Fact 3.1 *Let $\mathcal{N}$ be the continuous normal distribution centered at 0 with variance 1, and $\chi^2[k]$ be the χ^2 distribution with k degrees of freedom.[17] Then for every vector $\mathbf{w} \in \mathbb{Z}^d$ it holds that:*

1. $\Pr_{\mathbf{r} \leftarrow \mathcal{N}^d}\left[\left|\left\langle \mathbf{w}, \frac{1}{\sqrt{2}}\mathbf{r} \right\rangle\right| > 9.75 \cdot \|\mathbf{w}\|\right] = \Pr_{y \leftarrow \mathcal{N}}\left[|y| > 9.75 \cdot \sqrt{2}\right] < 2^{-141}.$

2. $\Pr_{R \leftarrow \mathcal{N}^{d \times 256}}\left[\left\|\frac{1}{\sqrt{2}}\mathbf{w}R\right\|^2 < 30 \cdot \|\mathbf{w}\|^2\right] = \Pr_{y \leftarrow \chi^2[256]}[y < 60] < 2^{-128}.$

3. $\Pr_{R \leftarrow \mathcal{N}^{d \times 256}}\left[\left\|\frac{1}{\sqrt{2}}\mathbf{w}R\right\|^2 > 337 \cdot \|\mathbf{w}\|^2\right] = \Pr_{y \leftarrow \chi^2[256]}[y > 674] < 2^{-128}.$

Corollary 3.2. *[heuristic] Let $\mathcal{D}$ be a distribution on $\{0, \pm 1\}$ such that $\mathcal{D}(1) = \mathcal{D}(-1) = \frac{1}{4}$ and $\mathcal{D}(0) = \frac{1}{2}$. Under the heuristic substitution of $\mathcal{D}$ with $\frac{1}{\sqrt{2}}\mathcal{N}$, for every vector $\mathbf{w} \in \mathbb{Z}^d$:*

1. $\Pr_{\mathbf{r} \leftarrow \mathcal{D}^d}[|\langle \mathbf{w}, \mathbf{r} \rangle| > 9.75 \cdot \|\mathbf{w}\|] \lesssim 2^{-141}.$
2. $\Pr_{R \leftarrow \mathcal{D}^{d \times 256}}[\|\mathbf{w}R\|^2 < 30 \cdot \|\mathbf{w}\|^2] \lesssim 2^{-128}.$
3. $\Pr_{R \leftarrow \mathcal{D}^{d \times 256}}[\|\mathbf{w}R\|^2 > 337 \cdot \|\mathbf{w}\|^2] \lesssim 2^{-128}.$

3.3 A Modular Johnson–Lindenstrauss Variant

In some cases we need a high probability bounds on the size of $\mathbf{w}R \bmod P$ rather than the size of $\mathbf{w}R$ itself. When the bound that we seek is sufficiently smaller than P, we get this as an easy corollary:

---

[17] The χ^2 distribution with k degrees of freedom is the distribution of $\sum_{i=1}^{k} x_i^2$ where $x_i \leftarrow \mathcal{N}$.

Corollary 3.3. *Fix $d, P \in \mathbb{Z}$ and a bound $b \leq P/45d$, and let $\mathbf{w} \in [\pm P/2]^d$ with $\|\mathbf{w}\| \geq b$. Let $\mathcal{D}[0] = 1/2$ and $\mathcal{D}[\pm 1] = 1/4$, then $\Pr_{R \leftarrow \mathcal{D}^{d \times 256}}[\|\mathbf{w}R \bmod P\| < b\sqrt{30}] < 2^{-128}$.*

Proof. We have two cases:

– The first case is when $\|\mathbf{w}\|_\infty \geq P/4d$. Let i be an index of an entry in $\mathbf{w}$ with magnitude at least $P/4d$, and consider any column of R (denoted $\mathbf{r}$): After choosing all but the i'th entry in $\mathbf{r}$, at most one of the three values $\{0, \pm 1\}$ yields $|\langle \mathbf{w}, \mathbf{r}\rangle \bmod P| < P/8d$. Hence the probability that all the columns of R yield entries smaller than $P/8d$ is at most $(1/2)^{256}$. Since $b \leq P/45d$ then $P/8d > b\sqrt{30}$ and therefore

$$\Pr_{R \leftarrow \mathcal{D}^{d \times 256}}[\|\mathbf{w}R \bmod P\| < b\sqrt{30}] \leq \Pr_R[\|\mathbf{w}R \bmod Pq\| < P/8d] \leq 2^{-256}.$$

– The second case is when $\|\mathbf{w}\|_\infty < P/4d$. Here with probability one we have $\mathbf{w}R \in [\pm P/2]^{256}$, so mod-$P$ reduction has no effect and the assertion follows directly from Corollary 3.2.

3.4 Approximate Proofs of Smallness

A tool from previous work that will be used as a subroutine in most of our new proofs is a zero-knowledge proof that proves that a committed vector has small coefficients. We use the approximate proofs of l_∞-smallness of Lyubashevsky et al. [42] (which also utilize rejection sampling, as is common in lattice-based proofs). This proof system has a fairly large gap between the l_∞ norm of the vector used by honest provers and what the prover can prove. But this gap will not show up in the rest of our scheme, because these proofs are only used to show that there is no wraparound modulo P (after which we use an exact proof for l_2 norm modulo P). The main feature of this proof is that the dimension of the transmitted vector is just 128, irrespective of how long the vector whose smallness we would like to prove.

To bound the size of a vector $\mathbf{w}$, the prover commits to $\mathbf{w}$ and to a masking vector $\mathbf{y}$ (chosen at random to be somewhat larger than $\mathbf{w}$), and sends the commitments to the verifier. The verifier chooses a small random matrix R, and the prover opens $\mathbf{z} = \mathbf{w}R + \mathbf{y}$ (and convinces the verifier that it is indeed the right $\mathbf{z}$ wrt $\mathbf{w}$ and $\mathbf{y}$), and the verifier checks that $\mathbf{z}$ is small. Soundness relies on the following lemma.

Lemma 3.4 ([42], Lemma 2.5). *Fix $q, d \in \mathbb{Z}$ and any two vectors $\mathbf{y} \in [\pm q/2]^{128}$ and $\mathbf{w} \in [\pm q/2]^d$. Let $\mathcal{D}[0] = 1/2$ and $\mathcal{D}[\pm 1] = 1/4$, then choosing $R \leftarrow \mathcal{D}^{d \times 128}$ we have*

$$\Pr_R\left[\|\mathbf{w}R + \mathbf{y} \bmod q\|_\infty < \tfrac{1}{2}\|\mathbf{w}\|_\infty\right] < 2^{-128}. \square$$

Describing the proof system in more detail, we use a hard-DL group of order P for the underlying commit-and-prove protocols, as follows. The prover holds a vector $\mathbf{w}$, and the verifier holds a discrete-log-based commitment to $\mathbf{w}$ (e.g., Pedersen). The goal of the protocol is to prove that $\mathbf{w}$ has l_∞ norm bounded by some known b, where for the honest prover we assume that $\|\mathbf{w}\|_\infty \leq b/\gamma$ (with γ our gap parameter).

0. We use security parameter $\lambda = 128$ and the size gap is $\gamma = 2 \cdot 9.75\lambda\sqrt{d} < 2500\sqrt{d}$.
1. The prover has a vector $\mathbf{w} \in \mathbb{Z}^d$ of bounded size $\|\mathbf{w}\|_\infty \leq b/\gamma$, and the verifier knows a commitment to $\mathbf{w}$.
2. The prover chooses a uniform masking vector $\mathbf{y} \leftarrow [\pm\lceil \tfrac{b}{2}(1 + \tfrac{1}{\lambda})\rceil]^\lambda$ and sends to the verifier a commitment to $\mathbf{y}$.

3. Let $\mathcal{D}(0) = 1/2$ and $\mathcal{D}(\pm 1) = 1/4$, the verifier chooses $R \leftarrow \mathcal{D}^{d \times \lambda}$ and sends it to the prover.
4. The prover computes $\mathbf{u} := \mathbf{w}R$ and $\mathbf{z} := \mathbf{u} + \mathbf{y}$. It restarts the protocol from Step 2 if either $\|\mathbf{u}\|_\infty > b/2\lambda$ or $\|\mathbf{z}\|_\infty > b/2$.
   If the two tests above passed, then the prover sends $\mathbf{z}$ to the verifier along with a ZKPOK that indeed $\mathbf{z} = \mathbf{w}R + \mathbf{y} \pmod{P}$.
5. The verifier accepts if the ZKPOK succeeds, and in addition $\|\mathbf{z}\|_\infty \leq b/2$.

Lemma 3.5. *The protocol above is an approximate proof-of-smallness for the l_∞ norm, with size gap $\gamma < 2500\sqrt{d}$.*

Proof. The honest prover has $\|\mathbf{w}\|_\infty \leq b/\gamma$, so by the first part of Claim 3.2 and the union bound, we have that $\|\mathbf{u}\|_\infty \leq 9.75\sqrt{d}\|\mathbf{w}\|_\infty \leq 9.75\sqrt{d}b/\gamma < b/2\lambda$ except with probability $2^7 \cdot 2^{-141} = 2^{-134}$. A restart due to this check therefore only happens with negligible probability.

Conditioned on $\|\mathbf{u}\|_\infty \leq b/2\lambda$, the rejection sampling check for $\|\mathbf{u} + \mathbf{y}\|_\infty \leq b/2$ leaks nothing about $\mathbf{u}$ (or $\mathbf{w}$), by [39]. Furthermore, using the analysis from [40, Section 5.2], the probability of the prover restarting due to this check is about $1 - \frac{1}{e} \approx 0.63$. Hence the expected number of repetitions is constant.

It is left to show soundness, so consider a cheating prover with $\|\mathbf{w}\|_\infty > b$. By Lemma 3.4 such prover has probability at most 2^{-128} of getting $\|\mathbf{w}R + \mathbf{y} \bmod P\|_\infty \leq b/2$, regardless of $\mathbf{y}$. This completes the proof.

3.5 Exact Proofs of Smallness

Using the protocol from Section 3.4, combined with a sum-of-squares proof, we can get an efficient exact proofs of smallness, provided that the bound b that we need to prove is sufficiently smaller than $\sqrt{P}$. Roughly, to prove that a value x has magnitude smaller than some public bound b, it is sufficient to show that $b^2 - x^2$ is non-negative,[18] which can be done by representing it as a sum of squares: After committing to x, the prover finds and commits to four other integers $\alpha, \beta, \gamma, \delta$ such that $b^2 - x^2 = \alpha^2 + \beta^2 + \gamma^2 + \delta^2$. The prover uses the underlying commit-and-prove systems to show that this equality holds modulo P, and also uses the approximate proof from above to show that the numbers are small enough so that they do not trigger a wraparound modulo P. Taken together, this means that this constraint holds over the integers, hence proving that indeed $|x| < b$.

In our implementation we actually use a slightly more general version, where the prover may wish to amortize over m instances of this problem. The upside of amortizing is that he will only need one l_∞ proof (as opposed to one per vector). The downside is that the size-bounds that we can prove this way are slightly more restricted, since the gap in the approximate proofs grows with (the square root of) the total dimension of all the vectors combined.

The protocol is described below. In this description we assume that commitments to different vectors can be combined to a single commitment for the concatenated vector (as needed for the underlying proofs systems). This clearly holds for the Pedersen commitments that we use in our implementation.

1. The prover has m vectors $\mathbf{w}_1, \ldots \mathbf{w}_m \in \mathbb{Z}^d$, and the verifier has commitments to all these vectors. For each vector $\mathbf{w}_i$, the prover wants to prove that $\|\mathbf{w}_i\| \leq b_i$ (where the b_i's are public). Denote $b = \max_i b_i$, and assume that $b < \sqrt{P}/(3536(d+4)\sqrt{m})$.

---

[18] More generally, to show that $x \in [a, b]$ it is sufficient to show that $(x - a)(b - x)$ is non-negative.

2. For each $\mathbf{w}_i$, the prover finds four non-negative integers $\alpha_i, \beta_i, \gamma_i, \delta_i$ such that $\alpha_i^2 + \beta_i^2 + \gamma_i^2 + \delta_i^2 = b_i^2 - \|\mathbf{w}_i\|^2$.
   Let $\mathbf{u}_i := (\alpha_i, \beta_i, \gamma_i, \delta_i)$ and $\mathbf{v}_i := (\mathbf{w}_i | \mathbf{u}_i) \in \mathbb{Z}^{d+4}$. The prover sends to the verifier commitments to all the $\mathbf{u}_i$'s, and they both combine them with the commitments to $\mathbf{w}_i$'s to get commitment for the $\mathbf{v}_i$'s.
3. The prover provides a ZKPOK that for all i, $\|v_i\|^2 = b_i^2 \pmod{P}$ (cf. Section 3.1).
4. The prover provides an l_∞ ZKPOK showing that
   $\|(\mathbf{v}_1 | \cdots | \mathbf{v}_m)\|_\infty < \sqrt{P/2(d+4)}$.

Lemma 3.6. *If $b = \max_i b_i < \sqrt{P}/(3536(d+4)\sqrt{m})$, then the protocol above is correct, and a zero-knowledge proof of knowledge that $\|\mathbf{w}_i\| \le b_i$ for all i.*

Proof. ZK follows from the ZK of the two underlying proofs.

For soundness, note that proving statement (3) implies that for all the $\mathbf{v}_i$'s we have $\|\mathbf{v}_i\|_\infty < \sqrt{P/2(d+4)}$, and therefore $\|\mathbf{v}_i\|^2 = \sum_{j=1}^{d+4} \mathbf{v}_{i,j}^2 < P/2$. This implies that statement (2) holds over the integers and not just modulo P, hence $b_i^2 - \|\mathbf{w}_i\|^2$ is positive.

The only thing left to show is that the bound $b = \max_i b_i$ is small enough to allow the use of the l_∞ approximate proof from Section 3.4 To prove that all the coefficients in the concatenated vector $(\mathbf{v}_1 | \cdots | \mathbf{v}_m)$ of dimension $m(d+4)$ are of size at most $\sqrt{P/2(d+4)}$ using that proof, the honest prover must have all the coefficients smaller than $\sqrt{P/2(d+4)}/\gamma$, where $\gamma = 2500\sqrt{m(d+4)}$. Hence we need

$$b \le \frac{\sqrt{P/2(d+4)}}{2500\sqrt{m(d+4)}} = \sqrt{P}/\left(\sqrt{2} \cdot 2500 \cdot (d+4)\sqrt{m}\right) \approx \sqrt{P}/\left(3536(d+4)\sqrt{m}\right),$$

which is exactly the bound in the statement of the lemma.

As a side remark, if we can tolerate a one-bit leakage on each $\|\mathbf{w}_i\|^2$, then the prover can instead find *three integers* $\alpha_i, \beta_i, \gamma_i$ such that $\alpha_i^2 + \beta_i^2 + \gamma_i^2 = b_i^2 - \|\mathbf{w}_i\|^2 \pm 1$ (such three integers always exist since every integer which is congruent to 1 or 2 modulo 4 is a sum of three squares). The prover then does the same proof as above, but sending $\delta_i = \pm 1$ to the verifier in the clear. (We do not use this option in our protocol.)

Exact Proofs of Smallness with Larger Bounds In our scheme we sometimes need to prove exact bounds on vectors with entries that are larger than the bound above. To do that, we let the prover break each coefficients into (say) two digits of size $\le \lceil \sqrt{b} \rceil$, commit to these digits and prove exact smallness for them separately, and then prove that combining these digits indeed yields the original coefficient.

Namely, the honest prover has a dimension-d vector $\mathbf{w}$ with $\|\mathbf{w}\| \le b$, and the verifier has a commitment to $\mathbf{w}$. The prover uses radix $\phi \in \mathbb{Z}$, chosen as small as possible subject to $\phi^2 - \phi \ge b\sqrt{d}/2$. It breaks $\mathbf{w}$ into two "digit vectors", $\mathbf{w}^{lo} := \mathbf{w} \bmod \phi$ (with entries in $[\pm\phi/2]$) and $\mathbf{w}^{hi} := (\mathbf{w} - \mathbf{w}^{lo})/\phi$. It commits to these vectors, produces a linear-constraint proof showing that $\mathbf{w} = \rho \cdot \mathbf{w}^{hi} + \mathbf{w}^{lo} \pmod{P}$, and uses the exact proof protocol from above to prove that

$$\|\mathbf{w}^{lo}\| \le \sqrt{d} \cdot \phi/2, \text{ and } \|\mathbf{w}^{hi}\| \le b/\phi + \sqrt{d}/2. \tag{5}$$

To see why the last inequality must hold, observe that

$$\|\mathbf{w}^{hi}\| = \|\mathbf{w} - \mathbf{w}^{lo}\|/\phi \le (\|\mathbf{w}\| + \|\mathbf{w}^{lo}\|)/\phi \le (b + \phi\sqrt{d}/2)/\phi = b/\phi + \sqrt{d}/2.$$

17

Let $b^* := \sqrt{P}/(3536(d+4)\sqrt{m})$ be the bound that we need in order to be able to use the exact proofs from above. The condition $\phi^2 - \phi \geq b\sqrt{d}/2$ ensures that $b/\phi + \sqrt{d}/2 \leq \sqrt{d} \cdot \phi/2$, so we can use the above proofs as long as we are able to set the radix ϕ small enough such that $\sqrt{d} \cdot \phi/2 \leq b^*$. It is not hard to verify that when $\sqrt{b} < b^* \cdot (4/d)^{3/4} - (4/d)^{1/4}$, the two conditions $\phi^2 - \phi \geq b\sqrt{d}/2$ and $\sqrt{d} \cdot \phi/2 \leq b^*$ can always be satisfied.[19]

Combining the two bounds from Eq. (5) and the linear-relation proof, we can therefore conclude that the size of the original $\mathbf{w}$ is bounded by

$$\|\mathbf{w}\| \leq \phi\|\mathbf{w}^{hi}\| + \|\mathbf{w}^{lo}\| \leq \phi(b/\phi + \sqrt{d}/2) + \phi\sqrt{d}/2 = b + \phi\sqrt{d}.$$

Therefore, this technique induces a multiplicative size gap of $\gamma = 1 + \frac{\phi\sqrt{d}}{b}$ between what the honest prover holds and what we can conclude about the vector of a cheating prover. (In our setting this gap will be miniscule.)

We remark that when using this technique, the prover needs to commit to more vectors and prove quadratic constraints on them, incurring a somewhat higher computational cost. Also, in the amortized setting, we can deal with a mix of some "small" and "large" vectors by breaking into digits only the large vectors and keeping the small vectors intact.

Approximate Proofs of Smallness for l_2 Norm The protocol in the previous section for proving that $\|\mathbf{w}\| \leq b$ require proving quadratic constraints on the $\mathbf{v}_i$'s to show that $\|\mathbf{v}_i\|^2 = b_i^2$, which may be costly. We note, however, that a simple application of Corollary 3.2 allows us to reduce the number of coefficients that are involved in the quadratic proof to $256 + 4 = 260$, regardless of the dimension of $\mathbf{w}$. The price that we pay is a small gap between what we can prove and what the honest prover actually uses (and the restriction on the bound that the protocol supports becomes somewhat smaller).

The idea is to first project the d-dimensional vector down to a 256-dimensional one by setting $\mathbf{u} = \mathbf{w}R$, for a random matrix R, and then apply the proof from the previous section to the projected vector $\mathbf{u}$. Using Corollary 3.2, an exact bound on $\|\mathbf{u}\|$ yields a very narrow range for the bound on $\|\mathbf{w}\|$. In our protocol, however, we use a more general form of this approximate proof, which is tailored to proving LWE relations, as we described next.

3.6 Proofs of Smallness for LWE

In the encryption scheme from Section 2.5, the prover sometimes has an LWE instance $\mathbf{b} = \mathbf{s}A + \mathbf{e}$ (mod q), and it needs to prove that $\mathbf{s}, \mathbf{e}$ are small. While the prover can commit to $\mathbf{s}, \mathbf{e}$ and use the proofs above, in this case we can save about half the cost by skipping the commitment to $\mathbf{e}$, since $\mathbf{e}$ is implicitly committed by seeing the commitment to $\mathbf{s}$ and knowing A and $\mathbf{b}$.

Below we describe this more efficient protocol, for the case $q = P$ (with P the parameter of the underlying commit-and-prove systems). In fact we need a slightly more general variant that includes a committed "offset vector", and as in previous sections we also let the prover amortize over m such proofs. We also use the technique from Section 3.5 to handle vectors with larger norm by splitting the projected vectors into high and low digits.

In more detail, both prover and verifier know public matrices $A_i \in \mathbb{Z}_P^{k_i \times d_i}$, $i = 1, \ldots, m$ and bounds b_i, b_i', and let γ be the size gap (to be defined below). The prover has vectors $\mathbf{s}_i \in \mathbb{Z}_P^k$ and

---

[19] Jumping ahead, in our setting we have $b^* > 2^{104}$ and $d = 256$, so we can handle bounds up to $b \approx 2^{190}$. The bounds that we actually need to prove will all be much much smaller.

$\mathbf{e}_i, \mathbf{x}_i \in \mathbb{Z}_P^{d_i}$, where $\|\mathbf{s}_i\| \leq b_i/\gamma$ and $\|\mathbf{e}_i\| \leq b_i'/\gamma$. The $2m$ vectors $\mathbf{s}_i, \mathbf{e}_i$ are partitioned into a set L of m_l "large" vectors and a set S of m_s "small" ones (so $m_l + m_s = 2m$). The designation of which vector belongs to what set is also public.

To simplify notations somewhat, below we assume that the LWE secrets are all "small" and the noise vectors are all "large", which would be the case in our application. The protocol can be easily extended to handle an arbitrary mix of "large" and "small", but the notations get rather swkward.

Let $\beta := \sqrt{P}/(\sqrt{2} \cdot 2500 \cdot 260 \cdot \sqrt{m_s + 2m_l}) \approx \sqrt{P}/(2^{19.9}\sqrt{m_s + 2m_l})$. For correctness of the protocol below, we require that the the bounds on the "small" vectors in S all satisfy $b_i \leq \beta/\sqrt{30}$. For the "large" vectors in L, let $b_* = \min_i(b_i')$ (i.e., the smallest "large" bound) and $b^* = \max_i(b_i')$, and we require that $8b^*/\sqrt{b_*} \leq \beta$.

The radix for breaking integers into digits is set to $\phi \in \mathbb{Z}$, taken as large as possible subject to $\sqrt{30}b_*/\phi + 8 \geq 8\phi$, specifically we use $\phi := \lfloor\sqrt{b_* \cdot 30/64}\rfloor$. Denoting $\gamma_1 := \sqrt{337/30} \leq 3.36$ and $\gamma_2 := 1 + \frac{16\phi}{\sqrt{30}b_*} < 1 + \frac{2}{\sqrt{b_*}}$, the size-gap that the protocol below achieves is $\gamma_1 \cdot \gamma_2$. [20] The protocol proceeds as follows:

0. For all i, let $\hat{b}_i := \sqrt{30}\, b_i/\gamma_2$ and $\hat{b}_i^{hi} := (\sqrt{30}\, b_i'/\phi + \sqrt{256}/2)/\gamma_2 = (\sqrt{30}\, b_i'/\phi + 8)/\gamma_2$, and also let $\hat{b}^{lo} := \sqrt{256}\phi/(2\gamma_2) = 8\phi/\gamma_2$.
1. The prover sets $\mathbf{b}_i := \mathbf{s}_i A_i + \mathbf{e}_i + \mathbf{x}_i \bmod P$ for all i, and sends to the verifier the $\mathbf{b}_i$'s and also commitments to the $\mathbf{s}_i$'s and $\mathbf{x}_i$'s.
2. Let $\mathcal{D}[0] = 1/2$ and $\mathcal{D}[\pm 1] = 1/4$. The verifier chooses $R_i \leftarrow \mathcal{D}^{k_i \times 256}$ and $R_i' \leftarrow \mathcal{D}^{d_i \times 256}$, and sends to the prover.
3. The prover computes $\mathbf{u}_i := \mathbf{s}_i R_i$, $\mathbf{v}_i := \mathbf{e}_i R_i'$. If $\|u_i\| > \sqrt{30}b_i/\gamma_2$ or $\|v_i\| > \sqrt{30}b_i'/\gamma_2$ then the prover aborts.

   Otherwise it splits the $\mathbf{v}_i$'s into digits, $\mathbf{v}_i^{lo} = \mathbf{v}_i \bmod \phi$ (with entries in $[\pm\phi/2]$), and $\mathbf{v}_i^{hi} = (\mathbf{v}_i - \mathbf{v}_i^{lo})/\phi$.

   The prover commits to all the $\mathbf{u}_i$'s, $\mathbf{v}_i^{lo}$'s, and $\mathbf{v}_i^{hi}$'s and sends to the verifier.
4. the parties then engage in the following ZKPOK protocols:
   A. Exact smallness proofs (cf. Section 3.5): For all i the prover proves that $\|\mathbf{u}_i\| \leq \hat{b}_i$, $\|\mathbf{v}_i^{lo}\| \leq \hat{b}^{lo}$, and $\|\mathbf{v}_i^{hi}\| \leq \hat{b}_i^{hi}$.
   B. Linear-constraint proofs for the projected LWE secrets, $\mathbf{s}_i R_i = \mathbf{u}_i \pmod{P}$ for all i.
   C. Linear-constraint proof for the LWE relation: For each all i it proves that

$$\mathbf{b}_i R_i' = \mathbf{s}_i A_i R_i' + \phi\mathbf{v}_i^{hi} + \mathbf{v}_i^{lo} + \mathbf{x}_i R_i' \pmod{P}.$$

5. The verifier accepts if all the proofs passed.

Lemma 3.7. *Assume that the dimensions and bounds satisfy the following conditions:*

- *For vectors in S we have $b_i \leq \beta/\sqrt{30}$, and for vectors in L we have $8b^*/\sqrt{b_*} \leq \beta$.*
- *For all i, $b_i \leq P/45k_i$ and $b_i' \leq P/45d_i$.*

Then the protocol is correct ZKPOK, proving that $\mathbf{b}_i = \mathbf{s}_i A_i + \mathbf{e}_i + \mathbf{x}_i \bmod P$ holds for some $\|\mathbf{s}_i\| \leq b_i$ and $\|\mathbf{e}_i\| \leq b_i'$. The size gap for both the $\mathbf{s}_i$'s and $\mathbf{e}_i$'s is $\gamma := \sqrt{337/30} \cdot (1 + \frac{16\phi}{b^*}) \leq 3.36(1 + \frac{20}{\sqrt{b^*}})$.

---

[20] In our setting we have $b_* > 2^{90}$, so the term $\frac{2}{\sqrt{b_*}}$ is insignificant.

Proof. ZK follows from the ZK of all the components. For completeness, first note that since the honest prover has $\mathbf{s}_i \le b_i/\gamma$ and $\mathbf{s}_i \le b'_i/\gamma$ then by Corollary 3.2 the prover only aborts in Step 3 with negligible probability.

We also need to show that the bounds used in Step 4A satisfy the constraints from Lemma 3.6. As we have $m_s + 2m_l$ projected vectors $\mathbf{u}_i, \mathbf{v}_i \in \mathbb{Z}_P^{256}$, we need to ensure that the bounds $\hat{b}_i, \hat{b}_i^{hi}, \hat{b}_i^{lo}$ that are used in the exact-smallness proofs do not exceed $\sqrt{P}/\left(\sqrt{2} \cdot 2500 \cdot 260\sqrt{m_s + 2m_l}\right) = \beta$. For vectors in S we have $b_i \le \beta/\sqrt{30}$ and therefore $\hat{b}_i \le \sqrt{30}b_i \le \beta$. For vectors in L, recall that we set $\phi = \lfloor\sqrt{b_* \cdot 30/64}\rceil$ to get $\hat{b}_i^{hi} \ge \hat{b}^{lo}$, and since $b'_i \le b^*$ we get:

$$\hat{b}^{lo} \le \hat{b}_i^{hi} \le (\sqrt{30}\,b'_i/\phi + 8)/\gamma_2 \le \frac{(\sqrt{30}\,b^* + 8\phi)/\phi}{(\sqrt{30}\,b_* + 16\phi)/(\sqrt{30}\,b_*)}$$

$$= \frac{\sqrt{30}\,b^* + 8\phi}{\sqrt{30}\,b_* + 16\phi} \cdot \frac{\sqrt{30}\,b_*}{\lfloor\sqrt{b_* \cdot 30/64}\rceil}$$

$$\le (b^*/b_*) \cdot 8\sqrt{b_*} \;=\; 8b^*/\sqrt{b_*} \;\le\; \sqrt{P}/(\sqrt{2} \cdot 2500 \cdot 260 \cdot \sqrt{m_s + 2m_l}).$$

It remains to prove soundness. Due to the proofs in Step 4 we can extract concrete $\mathbf{s}_i, \mathbf{x}_i, \mathbf{u}_i, \mathbf{v}_i^{hi}, \mathbf{v}_i^{lo}$ even from cheating provers. For each i, we can therefore define $\mathbf{e}_i := \mathbf{b}_i - \mathbf{s}_i A_i - \mathbf{x}_i \bmod P \in [\pm P/2]^{d_i}$ (so the constraint $\mathbf{b}_i = \mathbf{s}_i A_i + \mathbf{e}_i + \mathbf{x}_i \pmod{P}$ holds by definition). All we need to show, then, is that $\|\mathbf{s}_i\| \le b_i$ and $\|\mathbf{e}_i\| \le b'_i$.

Due to constraint 4C, it holds by definition of $\mathbf{e}_i$ that $(\phi\mathbf{v}^{hi} + \mathbf{v}^{lo}) = \mathbf{e}_i R'_i \pmod{P}$. Letting $\mathbf{v}_i := \phi\mathbf{v}^{hi} + \mathbf{v}^{lo}$, the bounds that we proved on the size of $\|\mathbf{v}^{hi}\|$ and $\|\mathbf{v}^{lo}\|$, together with the setting $\gamma_2 = 1 + 16\phi/(\sqrt{30}b_*) \ge 1 + 16\phi/(\sqrt{30}b'_i)$, imply that

$$\|\mathbf{v}_i\| \le \phi\hat{b}_i^{hi} + \hat{b}^{lo} = (\sqrt{30}\,b'_i + 8\phi)/\gamma_2 + 8\phi/\gamma_2 \le \frac{\sqrt{30}\,b'_i + 16\phi}{1 + 16\phi/(\sqrt{30}b'_i)} = \sqrt{30}b'_i.$$

Since $b_i \le P/45k_i$ and $b'_i \le P/45d_i$ then we can use Corollary 3.3. By this corollary, it must be the case that $\|\mathbf{s}_i\| \le b_i$ and $\|\mathbf{e}_i\| \le b'_i$ for all i, or else we would only have negligible probability of getting $\|\mathbf{u}_i\| \le b_i\sqrt{30}$ or $\|\mathbf{v}_i\| \le b'_i\sqrt{30}$. This completes the proof.

Using different ϕ_i for different LWE equations. The protocol above uses the same radix ϕ for all the "large" vectors, adding an extra factor of b^*/b_* in the conditions of Lemma 3.7. In our application this factor does not make a difference, but it can be avoided by using a different radix $\phi_i = \lfloor\sqrt{b'_i \cdot 30/64}\rceil$ for splitting the i'th "large" vector $\mathbf{v}_i$. This would have the effect of only requiring $8\sqrt{b^*} \le \beta$ (rather than $8b^*/\sqrt{b_*} \le \beta$).

Sharing LWE secrets across instances. When using the proof above in our protocol, we often need to prove multiple LWE instances for the same LWE secret. For example the same secret key is used in both the proof of key generation and the proof of decryption.

In this case, the prover will only send a single commitment to that LWE secret $\mathbf{s}$, the verifier will only send a single challenge matrix R, and the parties will only run a single exact-smallness proof for $\mathbf{u} = \mathbf{s}R$ in Step 4A and a single instance of the linear proof for it in Step 4B. On the other hand, they will run a separate instance of the proof in Step 4C for each LWE relation. The bounds will remain exactly as in Lemma 3.7 (although in this case we may have $m_s + 2m_l < 2m$).

Proofs for Module-LWE As mentioned in Section 2.4, our implementation actually uses Module-LWE over a low dimension extension field $\mathbb{F}_{P^\ell}$ rather than over the integers (specifically we use $\ell = 2$).

The proofs-of-smallness protocols above can easily be extended to this case, treating $\mathbf{b} = \mathbf{s}A + \mathbf{e}$ (mod q) as an equation over the $\mathbb{F}_{P^\ell}$, which can be written as $B = SA' + E$ (mod P) in matrix notation over the integers.

Given A' and B, every entry in E can be expressed as an affine expression in the entries of S, and moreover, the entries in S are all known linear combinations of the (representation over $\mathbb{Z}_p$ of) $\mathbf{s}$. We can therefore arrange the entries of E in a vector $\tilde{\mathbf{e}}$, and get a new equation over the integers $\tilde{\mathbf{b}} = \mathbf{s}\tilde{A} + \tilde{\mathbf{e}}$ (mod P), which we can prove using the protocol above.

4   Implementation and Performance

We implemented the scheme and proofs above in C++, with operations in the Curve 25519 using `libsodium` and operations in $\mathbb{F}_{P^2}$ using `NTL`. The implementation is available under MIT license from `https://github.com/shaih/cpp-lwevss`. Our implementation is still quite naive, operating single-threaded, and making direct call to the exponentiation routines of `libsodium` without any optimizations for multi-exponentiations.

We run this program on an old server that we had access to, featuring Intel Xeon CPU, E5-2698 v3 running at 2.30GHz (which is a Haswell processor) with 32 cores and 250GB RAM. The software configurations included `libsodium` 1.0.18, `NTL` version 11.3.0, and `GMP` version 6.2.0, all compiled with `gcc` 7.3.1 and running on CentOS Linux 7, kernel version 3.10.0.

The performance results with number of parties from 128 to 1024 are summarized in Tables 1 and 2. In Table 1 we specify for each setting the time spent in each of the high-level subroutine: key-generation, encryption, decryption, proving, and verifying. We also specify there the number of scalar-point multiplications (denoted #exp) performed in each subroutine, and the total RAM consumption.

# of parties	Keygen time(sec)	Encrypt time(sec)	Decrypt time(ms)	Prove # exp	Prove time(sec)	Verify # exp	Verify time(sec)	RAM usage
128	5.1	4.2	1.4	80392	22.9	23145	15.3	$2.26GB$
256	5.2	4.4	1.4	82608	23.7	23451	15.9	$2.73GB$
512	5.2	5.0	1.4	84030	25.3	24063	17.4	$3.74GB$
1024	5.3	5.8	1.4	87524	28.2	24939	20.0	$5.28GB$

Table 1. Performance results with 128-1024 parties, by high-level subroutine.

In Table 2 we specify for each setting the running-time spent in some of the lower-level subroutines: In particular the time spent by vector-matrix multiplication by the CRS matrix over Z_q, and the time spend performing scalar-point multiplications on the curve.

As can be seen in Table 2, only about 25-30% of the prover time and about 15% of the verifier time was spent performing scalar-point multiplications on the curve. The reason is that the number of these curve operations is linear in the dimensions k, while the number of scalar multiplications modulo q is quadratic (since we compute a few vector-matrix multiplications.) We also note that switching to a structured CRS matrix (by moving to operations over dimension-k extension field/ring and relying on ring-LWE) would have reduced the multiply-by-CRS time, making it

	Prover			Verifier		
# of parties	multiply by CRS	point-scalar multiply	total time	multiply by CRS	point-scalar multiply	total time
128	15.2	9.6	32.2	6.1	2.8	15.3
256	15.3	9.9	33.3	6.1	2.8	15.9
512	15.8	10.1	35.5	6.4	2.9	17.4
1024	16.1	10.5	39.4	6.5	3.0	20.0

Table 2. Running time (seconds) with 128-1024 parties, by low-level subroutine.

insignificant. Implementing this optimization could yield an almost $2\times$ speedup for the prover and about $1.5\times$ speedup for the verifier. It is clear from these tables that this PVSS scheme is quite feasible, even for committees with many hundreds of parties and with our rather naive, single-thread implementation.

References

1. D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. Special Issue on PODS 2001.
2. S. Agrawal, D. Stehlé, and A. Yadav. Towards practical and round-optimal lattice-based threshold and blind signatures. *IACR Cryptol. ePrint Arch.*, 2021:381, 2021.
3. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9:169–203, October 2015. `https://bitbucket.org/malb/lwe-estimator/src/master/`.
4. S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *Journal of Cryptology*, 31(2):610–640, Apr. 2018.
5. C. Baum, I. Damgård, K. G. Larsen, and M. Nielsen. How to prove knowledge of small secrets. In *Annual International Cryptology Conference*, pages 478–498. Springer, 2016.
6. 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. Springer, Heidelberg, Sept. 2018.
7. C. Baum and V. Lyubashevsky. Simple amortized proofs of shortness for linear relations over polynomial rings. *IACR Cryptol. ePrint Arch.*, page 759, 2017.
8. F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin. Can a public blockchain keep a secret? In *TCC*, 2020. https://eprint.iacr.org/2020/464.
9. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *International conference on the theory and applications of cryptographic techniques*, pages 416–432. Springer, 2003.
10. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security*, pages 514–532. Springer, 2001.
11. J. Bootle, A. Chiesa, and K. Sotiraki. Sumcheck arguments and their applications. *IACR Cryptol. ePrint Arch.*, 2021:333, 2021.
12. F. Boudot and J. Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *International Conference on Information and Communications Security*, pages 87–102. Springer, 1999.
13. Z. Brakerski, N. Döttling, S. Garg, and G. Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *Theory of Cryptography Conference*, pages 407–437. Springer, 2019.
14. 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, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.
15. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Annual International Cryptology Conference*, pages 126–144. Springer, 2003.
16. R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

17. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 383–395. IEEE, 1985.

18. N. Costa, R. Martínez, and P. Morillo. Proof of a shuffle for lattice-based cryptography. In *Nordic Conference on Secure IT Systems*, pages 280–296. Springer, 2017.

19. R. Del Pino and V. Lyubashevsky. Amortization with fewer equations for proving knowledge of small secrets. In *Annual International Cryptology Conference*, pages 365–394. Springer, 2017.

20. R. D'Souza, D. Jao, I. Mironov, and O. Pandey. Publicly verifiable secret sharing for cloud-based key management. In *International Conference on Cryptology in India*, pages 290–309. Springer, 2011.

21. P.-A. Fouque and J. Stern. One round threshold discrete-log key generation without private channels. In *International Workshop on Public Key Cryptography*, pages 300–316. Springer, 2001.

22. G. Fuchsbauer. Commuting signatures and verifiable encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–245. Springer, 2011.

23. E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 32–46. Springer, 1998.

24. R. Gennaro and S. Goldfeder. Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.

25. C. Gentry and S. Halevi. Compressible fhe with applications to pir. In *Theory of Cryptography Conference*, pages 438–464. Springer, 2019.

26. C. Gentry, S. Halevi, B. Magri, J. B. Nielsen, and S. Yakoubov. Random-index PIR with applications to large-scale secure MPC. https://eprint.iacr.org/2020/1248, 2020.

27. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

28. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

29. J. Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

30. J. Groth. Applied crypto: Introducing noninteractive distributed key generation. `https://medium.com/dfinity/applied-crypto-one-public-key-for-the-internet-computer-ni-dkg-4af800db869d`, accessed June 2021, 2021.

31. J. Groth. Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339, 2021. `https://eprint.iacr.org/2021/339`.

32. S. Heidarvand and J. L. Villar. Public verifiability from pairings in secret sharing schemes. In *International Workshop on Selected Areas in Cryptography*, pages 294–308. Springer, 2008.

33. M. P. Jhanwar, A. Venkateswarlu, and R. Safavi-Naini. Paillier-based publicly verifiable (non-interactive) secret sharing. *Designs, codes and Cryptography*, 73(2):529–546, 2014.

34. W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26, 1984.

35. J. Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. *IACR Cryptol. ePrint Arch.*, 2020:1274, 2020.

36. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 101–131. Springer, 2016.

37. Y. Lindell and A. Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *ACM CCS 18*, pages 1837–1854. ACM Press, 2018.

38. V. Lyubashevsky. Lattice-based identification schemes secure under active attacks. In *International workshop on public key cryptography*, pages 162–179. Springer, 2008.

39. V. Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, Heidelberg, Dec. 2009.

40. V. Lyubashevsky. Basic lattice cryptography: Encryption and Fiat-Shamir signatures. `https://www.tinyurl.com/latticesurvey`, accessed Apr-2021, 2020.

41. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *CCS*, pages 1051–1070. ACM, 2020.

42. V. Lyubashevsky, N. K. Nguyen, and G. Seiler. Shorter lattice-based zero-knowledge proofs via one-time commitments. *IACR Cryptol. ePrint Arch.*, 2020:1448, 2020.

43. C. A. Melchor, J. Barrier, L. Fousse, and M.-O. Killijian. Xpir: Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, 2016:155–174, 2016.

44. F. Olumofin and I. Goldberg. Revisiting the computational practicality of private information retrieval. In *International Conference on Financial Cryptography and Data Security*, pages 158–172. Springer, 2011.

45. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

46. B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.

47. C. Peikert, V. Vaikuntanathan, and B. Waters. A framework for efficient and composable oblivious transfer. In D. A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

48. M. Rambaud and A. Urban. Almost-asynchronous mpc under honest majority, revisited. *IACR Cryptol. ePrint Arch.*, 2021:503, 2021.

49. O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

50. L. Reyzin, A. Smith, and S. Yakoubov. Turning hate into love: Compact homomorphic ad hoc threshold encryption for scalable mpc. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 361–378. Springer, 2021.

51. A. Ruiz and J. L. Villar. Publicly verifiable secret sharing from paillier's cryptosystem. In *WEWoRC 2005–Western European Workshop on Research in Cryptology*. Gesellschaft für Informatik eV, 2005.

52. B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Annual International Cryptology Conference*, pages 148–164. Springer, 1999.

53. R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, pages 2006–06. Internet Society, 2007.

54. M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer, 1996.

55. T.-Y. Wu and Y.-M. Tseng. A pairing-based publicly verifiable secret sharing scheme. *Journal of Systems Science and Complexity*, 24(1):186–194, 2011.

56. A. Young and M. Yung. A pvss as hard as discrete log and shareholder separability. In *International Workshop on Public Key Cryptography*, pages 287–299. Springer, 2001.

A   Components and Parameters

$q = P \approx 2^{252}$	The lattice modulus and order of the hard-DL group for commitments & proofs.
$\ell = 2$	Redundancy parameter, encryption scheme operates over $\mathbb{F} = \mathbb{F}_{q^\ell}$.
$\Delta = 2^{126} \approx \lfloor \sqrt[\ell]{q} \rfloor$ $\mathbf{g} = (\Delta^{\ell-1}, \ldots, 1)$	Also $g \in \mathbb{F}$ is the element represented by $\mathbf{g}$. To encrypt a plaintext scalar $x \in \mathbb{Z}_q$, it is expanded to the field element $x' = x \cdot g$, represented by the vector $x \cdot \mathbf{g} \in \mathbb{Z}_q^\ell$.
k $\mathsf{k} = k\ell \approx 6000$	Dimension of LWE secrets and noise vectors, over $\mathbb{F}_{q^\ell}$. Dimension of integer vector representation of LWE secrets and noise.
$n, \ t < n/2$	Number of parties and secret-sharing threshold.
$\mathsf{n} = n\ell, \mathsf{t} = t\ell$	Dimension of integer vector representation of vectors in $\mathbb{F}_{q^\ell}^n, \mathbb{F}_{q^\ell}^t$.
$\mathcal{D}$	distribution for verifier challenges: $\mathcal{D}[0] = 1/2$ and $\mathcal{D}[\pm 1] = 1/4$.
$\sigma_s = 3$	l_∞ norm of LWE secrets. Entries in the integer representation of LWE secrets are drawn from χ_s: $\chi_s[0] = 1/\sigma_s$ and $\chi_s[x] = 1/2\sigma_s$ for all $x \in [\pm \sigma_s], x \neq 0$.
$\sigma_{e1} = 2^{94} - 1$ $\sigma_{e2} = 2^{114} - 1$	l_∞ norm of LWE noise. Integers in the representation of the noise vectors are drawn from χ_{e1}, χ_{e2}: $\chi_{ei}[0] = 1/\sigma_{ei}$ and $\chi_s[x] = 1/2\sigma_{ei}$ for $x \in [\pm \sigma_{ei}], x \neq 0$.

Table 3. Notations for encryption and commit-and-prove schemes

In this section we describe the main components of our scheme, combining the encryption scheme from Section 2 with the proofs from Section 3, and setting concrete parameters for them. Notation that were introduced in previous sections and used here are summarized in Table 3, together with their setting as described in Appendix A.4. Throughout this section we use $\tilde{\square}$ to denote quantities held by dishonest parties. For example, if an honest party needs to prove something about a vector $\mathbf{v}$, then we denote by $\tilde{\mathbf{v}}$ the quantity that a dishonest party holds (which can be extracted from the relevant ZKPOK).

A.1 Key Generation

Given the CRS matrix $A \in \mathbb{F}^{k \times k}$, each key holder i chooses $\mathbf{s}_i \leftarrow \chi_s^k$ and $\mathbf{e}_i \leftarrow \chi_{e1}^k$. Each entry in the integer representation of $\mathbf{s}_i$ has variance $E[\chi_s^2] = 3.5$, hence $E[\|\mathbf{s}_i\|^2] = 3.5\mathsf{k}$. The key-holder resets its choice until it gets $\|\mathbf{s}_i\| \leq 2\sqrt{\mathsf{k}}$ (with less than two expected trials). Similarly each entry in the integer representation of $\mathbf{e}_i$ has variance $\leq \sigma_{e1}^2/3$, hence $E[\|\mathbf{e}_i\|^2] \leq \mathsf{k}\sigma_{e1}^2/3$. The key-holder resets its choice until it gets $\|\mathbf{e}_i\| \leq \sigma_{e1}\sqrt{\mathsf{k}/2}$ (with less than two expected trials).

The key holder sets $\mathbf{b}_i := \mathbf{s}_i A + \mathbf{e}_i$ (over $\mathbb{F}$), then broadcasts $\mathbf{b}_i$ along with a commitment to $\mathbf{s}_i$, then uses the LWE-smallness protocol from Section 3.6, with $\mathbf{s}_i \in S$ and $\mathbf{e}_i \in L$ (i.e., the projected vector for $\mathbf{e}_i$ is broken into two digits, and the one for $\mathbf{s}_i$ is not). Since that protocol has a size gap of $\gamma = \sqrt{337/30}$, we can conclude that even for dishonest parties we have

$$\|\tilde{\mathbf{s}}_i\| \leq \mathsf{b_s} := \sqrt{337/30} \cdot 2\sqrt{\mathsf{k}} \approx 6.7\sqrt{\mathsf{k}}, \quad \text{and}$$
$$\|\tilde{\mathbf{e}}_i\| \leq \mathsf{b_{ek}} := \sqrt{337/30} \cdot \sigma_{e1}\sqrt{\mathsf{k}/2} \approx 2.37\sigma_{e1}\sqrt{\mathsf{k}}.$$

A.2 Encryption

Given the matrices $A \in \mathbb{F}^{k \times k}, B \in \mathbb{F}^{n \times k}$ and the plaintext vector $\mathbf{x} \in \mathbb{Z}_q^n \subset \mathbb{F}$, the encryptor chooses $\mathbf{r} \leftarrow \chi_s^k$, $\mathbf{e1} \leftarrow \chi_{e1}^k$, and $\mathbf{e2} \leftarrow \chi_{e2}^n$. Similarly to above, the encryptor resets its choices until $\|\mathbf{r}\| \leq 2\sqrt{\mathsf{k}}$, $\|\mathbf{e1}\| \leq \sigma_{e1}\sqrt{\mathsf{k}/2}$, and $\|\mathbf{e2}\| \leq \sigma_{e2}\sqrt{\mathsf{n}/2}$ (with less than two expected trials each).

The encryptor sets $\mathbf{c}_1 := A\mathbf{r}^T + \mathbf{e1}^T$ and $\mathbf{c}_2 := B\mathbf{r}^T + \mathbf{e2}^T + g \cdot \mathbf{x}^T$ (both over $\mathbb{F}$), then broadcasts $\mathbf{c}_1, \mathbf{c}_2$ along with a commitment to $\mathbf{r}$ and $\mathbf{x}$ (which implies a commitment also to $\mathbf{x}' = g\mathbf{x} \in \mathbb{F}$). It then uses the LWE-smallness protocol from Section 3.6 with $\mathbf{r} \in S$ and $\mathbf{e1}, \mathbf{e2} \in L$, ensuring that even for dishonest encryptors we have

$$\|\tilde{\mathbf{r}}\| \leq \mathsf{b_r} := \sqrt{337/30} \cdot 2\sqrt{\mathsf{k}} \approx 6.7\sqrt{\mathsf{k}},$$
$$\|\tilde{\mathbf{e1}}\| \leq \mathsf{b_{e1}} := \sqrt{337/30} \cdot \sigma_{e1}\sqrt{\mathsf{k}/2} \approx 2.37\sigma_{e1}\sqrt{\mathsf{k}}, \quad \text{and}$$
$$\|\tilde{\mathbf{e2}}\| \leq \mathsf{b_{e2}} := \sqrt{337/30} \cdot \sigma_{e2}\sqrt{\mathsf{n}/2} \approx 2.37\sigma_{e2}\sqrt{\mathsf{n}}.$$

A.3 Proof of Decryption

On the receiving end of our protocol, each party i considers the first t ciphertexts with valid proofs of encryption (ordered by the sender's IDs). It collects the $\mathbf{c}_1$ vectors from all these ciphertexts (corresponding to $A\mathbf{r}^T + \mathbf{e1}^T$) as well as "its entries" from the $\mathbf{c}_2$ vectors (corresponding to $\langle \mathbf{b}_i, \mathbf{r}\rangle + e2_i + x_i g$). Hence the receiver gets a k-vector plus one more element of $\mathbb{F}$ from each of these t ciphertexts.

The receiver arranges these ciphertexts in a matrix $C_1 \in \mathbb{F}^{k \times t}$ and a vector $\mathbf{c}_2 \in \mathbb{F}^t$, with the columns corresponding to the different senders. It uses its secret key to decrypt the noisy plaintext

vector $\mathbf{x}^* := \mathbf{c}_2 - \mathbf{s}_i C_1$, then apply the decoding procedure to obtain the plaintext vector $\mathbf{x} \in \mathbb{Z}_q^t$, and recovers the error vector $\mathbf{e} := \mathbf{x}^* - g\mathbf{x}$.

Let $\mathbf{x}' := g\mathbf{x} \in \mathbb{F}^t$ be the encoded plaintext vector, the receiver broadcasts a commitment to $\mathbf{x}$ (that implies a commitment to $\mathbf{x}'$) and uses the LWE-smallness protocol from Section 3.6 with $\mathbf{s}_i \in S, \mathbf{e} \in L$. Note that a commitment to $\mathbf{s}_i$ and a bound on its norm were already included in the proof of key-generation above. The bound that we use for the norm of $\mathbf{e}$ is specified below.

By the proof of correct encryption, each column of C_1 and the corresponding entry of $\mathbf{c}_2$ have the form $A\tilde{\mathbf{r}}^T + \tilde{\mathbf{e1}}^T$ and $\langle \mathbf{b}_i, \mathbf{r} \rangle + e2_i + x_i g$, respectively, with $\|\tilde{\mathbf{r}}\| \le \mathsf{b_r}$ and $\|e2_i\| \le \|\mathbf{e2}\| \le \mathsf{b_{e2}}$. This holds even if that sender was dishonest.

For an honest receiver we also have $\mathbf{b}_i = \mathbf{s}_i A + \mathbf{e}$, so each entry of the decryption noise vector is of the form $\tilde{e2}_i - \langle \mathbf{e}, \tilde{\mathbf{r}} \rangle - \langle \mathbf{s}_i, \tilde{\mathbf{e1}} \rangle$. Let us denote $\eta := \langle \mathbf{e}, \tilde{\mathbf{r}} \rangle - \langle \mathbf{s}_i, \tilde{\mathbf{e1}} \rangle$. Since the honest receiver chooses $\mathbf{s} \leftarrow [\pm 3]$ and $\mathbf{e} \leftarrow [\pm\sigma_{e1}]$, then η is a zero mean random variable, with a bounded variance

$$
\begin{aligned}
E[|\eta|^2] &\le \|\tilde{\mathbf{r}}\|^2 \cdot \sigma_{e1}^2/3 + \|\tilde{\mathbf{e1}}\|^2 \cdot 3.5 \quad \le \quad \mathsf{b_r}^2 \cdot \sigma_{e1}^2/3 + \mathsf{b_{e1}}^2 \cdot 3.5 \\
&= (337 \cdot 4\mathsf{k} \cdot 3.5)/30 + (337 \cdot \sigma_{e1}^2 \cdot \mathsf{k})/60 \quad = \quad (\sigma_{e1}^2 + 28) \cdot \mathsf{k} \cdot 337/60.
\end{aligned}
$$

Using the same heuristic as in Section 3.2, we can bound the tails of the distribution of η as if $\mathbf{s}$ and $\mathbf{e}$ were Gaussian random vectors, obtaining $\Pr[|\eta| > 9.75 \cdot \sqrt{2 \cdot (\sigma_{e1}^2 + 28) \cdot \mathsf{k} \cdot 337/60}] < 2^{-141}$. In our setting we use $\sigma_{e1} > 2^{90}$ and hence $\sqrt{\sigma_{e1}^2 + 28} \approx \sigma_{e1}$, so we can simplify the bound above as

$$
\Pr\left[|\eta| > 33\sigma_{e1}\sqrt{\mathsf{k}}\right] \quad < \quad \Pr\left[|\eta| > 9.75\sqrt{337/30} \cdot \sigma_{e1} \cdot \sqrt{\mathsf{k}}\right] \quad < \quad 2^{-141}.
$$

Thus we can bound the l_∞ norm of the decryption noise with overwhelming probability by: $\|\mathbf{e}_d\|_\infty \le \mathsf{b_{e2}} + 33\sigma_{e1}\sqrt{\mathsf{k}} \approx 2.37\sigma_{e2}\sqrt{\mathsf{n}} + 33\sigma_{e1}\sqrt{\mathsf{k}}$, which implies that

$$
\begin{aligned}
\|\mathbf{e}_d\| &\le \mathsf{b_{ed}} := \sqrt{t}\left(2.37\sigma_{e2}\sqrt{\mathsf{n}} + 33\sigma_{e1}\sqrt{\mathsf{k}}\right) \quad < \quad \sqrt{\mathsf{n}/2}\left(2.37\sigma_{e2}\sqrt{\mathsf{n}} + 33\sigma_{e1}\sqrt{\mathsf{k}}\right) \\
&\approx 1.7\sigma_{e2}\mathsf{n} + 24\sigma_{e1}\sqrt{\mathsf{kn}}.
\end{aligned}
$$

Validity of Decrypted Values Consider a pair of sender and receiver, where the sender produces a ciphertext that passes the proof of encryption, encrypting some scalar $x \in \mathbb{Z}_q$ to that receiver. We argue that if the receiver produced passing proofs for key generation and decryption of some x' from that sender, then necessarily $x = x' \pmod q$. Note that this guarantee holds even if both sender and receiver are dishonest, so long as their proofs pass verification.

- By validity of the receiver key, we know that $\mathbf{b} = \tilde{\mathbf{s}}A + \tilde{\mathbf{e}}_{kg}$ with $\|\tilde{\mathbf{s}}\| < \mathsf{b_s}$ and $\|\tilde{\mathbf{e}}_{kg}\| \le \mathsf{b_{ek}}$.
- By the proof of correct encryption we also know that $\mathbf{c}_1 = A\tilde{\mathbf{r}} + \tilde{\mathbf{e1}}$ and $c_2 = \langle \mathbf{b}, \tilde{\mathbf{t}} \rangle + \tilde{e2} + xg$, where $\|\tilde{\mathbf{r}}\| \le \mathsf{b_r}$, $\|\tilde{\mathbf{e1}}\| \le \mathsf{b_{e1}}$, and $\|\tilde{e2}\| \le \mathsf{b_{e2}}$.
- Combining these two we get $c_2 - \langle \tilde{\mathbf{s}}, \mathbf{c}_1 \rangle = \tilde{e2} + \langle \tilde{\mathbf{e}}_{kg}, \tilde{\mathbf{r}} \rangle - \langle \tilde{\mathbf{s}}, \tilde{\mathbf{e1}} \rangle + xg$, with $\|\tilde{e2} + \langle \tilde{\mathbf{e}}_{kg}, \tilde{\mathbf{r}} \rangle - \langle \tilde{\mathbf{s}}, \tilde{\mathbf{e1}} \rangle\| \le \mathsf{b_{e2}} + \mathsf{b_{ek}}\mathsf{b_r} + \mathsf{b_s}\mathsf{b_{e1}}$.
- At the same time, the proof of correct decryption tells us that $c_2 - \langle \tilde{\mathbf{s}}, \mathbf{c}_1 \rangle = \tilde{\mathbf{e}}_d + x'g$ for some x' (that may or may not be the same as x), with $\|\tilde{\mathbf{e}}_d\| < \mathsf{b_{ed}}$.

The last two bullets imply that

$$
d := (x - x')g = \tilde{e}_{e2} + \langle \tilde{\mathbf{e}}_{kg}, \tilde{\mathbf{r}} \rangle - \langle \tilde{\mathbf{s}}, \tilde{\mathbf{e}}_{e1} \rangle - \tilde{\mathbf{e}}_d. \tag{6}
$$

Denoting the representation of d by $\mathbf{d} \in \mathbb{Z}^\ell$, we have

$$
\begin{aligned}
\|\mathbf{d}\|_\infty &\leq b_{e2} + b_{ek}b_r + b_s b_{e1} + b_{ed} \\
&\leq 2.37\sigma_{e2}\sqrt{n} + 2(2.37\sigma_{e1}\sqrt{k} \cdot 6.7\sqrt{k}) + (1.7\sigma_{e2}n + 24\sigma_{e1}\sqrt{kn}) \\
&\approx \sigma_{e1}\left(24\sqrt{kn} + 31.8k\right) + \sigma_{e2}\left(2.37\sqrt{n} + 1.7n\right)
\end{aligned}
\tag{7}
$$

At the same time, Eq. (6) tells us that $\mathbf{d}$ belongs to the integer lattice spanned by $\mathbf{g}$ modulo q, namely

$$
\mathbf{d} \in L_g := \{\mathbf{z} \in \mathbb{Z}^\ell : \exists x \in \mathbb{Z} \ s.t. \ \mathbf{z} = x\mathbf{g} \pmod{q}\}.
$$

But it is easy to see that every non-zero vector in L_g must have at least one entry of size $\Delta^{\ell-1}$: Indeed for any index $i > 0$, if the i'th entry is of size $b < \Delta^{\ell-1}$ then the $i-1$'st entry has size $b \cdot \Delta$.

If q is large enough so that the bound from Appendix A.3 is smaller than $\Delta^{\ell-1}$, then $\mathbf{d}$ cannot be a non-zero vector in L_g, it must therefore be the zero vector which means that $x = x' \pmod{q}$. In other words, a sufficient condition for correct decryption is

$$
\Delta^{\ell-1} \approx q^{(\ell-1)/\ell} > \sigma_{e1}\left(24\sqrt{kn} + 31.8k\right) + \sigma_{e2}\left(2.37\sqrt{n} + 1.7n\right).
\tag{8}
$$

(As we explain in Appendix A.4, the term $1.7n\sigma_{e2}$ turns out to be by far the largest term above.)

Proof of Re-Sharing In our proactive secret sharing protocol, each party needs to prove that the plaintext values that it received in the previous round are consistent with the ones that it is sending in the current one. Specifically, denote the t values that it received in the previous round by $x_1, \ldots, x_t$ and the n values that it is sending in the current round by $y_1, \ldots, y_n$. Let $\lambda_1, \ldots, \lambda_t$ be the Lagrange coefficients that this party used to reconstruct its share in the previous round, then it needs to prove that the vector

$$
\mathbf{v} = \left(\textstyle\sum_{i=1}^t \lambda_i x_i, \ y_1, y_2, \ldots, y_n\right) \in \mathbb{Z}_q^{n+1}
$$

consists of the evaluations of some polynomial F of degree (at most) $t-1$ at the points $0, 1, \ldots, n$ in $\mathbb{Z}_q$. The set of all vectors with this property is a linear subspace of $\mathbb{Z}_q^{n+1}$, whose rank is t. Let $H \in \mathbb{Z}_q^{(n+1) \times (n+1-t)}$ be the parity-check matrix for this linear subspace, namely $\mathbf{v}$ has the right format is and only if $\mathbf{v}H = 0 \pmod{q}$.

The values x_i, y_j were committed by this party as part of its proofs of decryption and encryption, respectively. So it just remains to prove that the linear relation holds.

A.4 Setting the Parameters

Below we collect all the constraints from the analysis above and explain how we set our concrete parameters.

– We begin by looking at the setting of σ_{e2}. Let $\mathbf{e}^*$ be the vector that should be "flooded" in the proof of Lemma 2.1, namely each entry of $\mathbf{e}^*$ has the form $\delta := \langle \tilde{\mathbf{s}}, \mathbf{e1}\rangle + \langle \tilde{\mathbf{e}}_{ek}, \mathbf{r}\rangle$.
Recall from Section 2.3 that we want to set σ_{e2} just large enough to ensure that the Rényi divergence (say, of order $\alpha = 2$) between $\mathbf{e2}$ and $\mathbf{e2} + \mathbf{e}^*$ is a small constant. The random variable δ has zero-mean, and its variance is bounded by

$$
E[|\delta|^2] = \|\tilde{\mathbf{s}}\|^2 \cdot \sigma_{e1}^2/3 + \|\tilde{\mathbf{e}}_{kg}\|^2 \cdot 3.5 < (6.7\sqrt{k})^2 \cdot \sigma_{e1}^2/3 + (2.37\sigma_{e1}\sqrt{k})^2 \cdot 3.5 < 35\sigma_{e1}^2 k.
$$

We use the same heuristic from Section 3.2, bounding the size of $\mathbf{e}^*$ as if the honestly-chosen $\mathbf{r}, \mathbf{e1}$ are zero-mean Normal random vectors. Setting $b_\delta := 9.75\sqrt{2} \cdot \sigma_{e1}\sqrt{35\mathsf{k}} < 82\sigma_{e1}\sqrt{\mathsf{k}}$, we get the high -probability bound $\Pr\left[\|\mathbf{e}^*\|_\infty > b_\delta\right] < \mathsf{t} \cdot 2^{-141} < 2^{-128}$. This also implies that $\|\mathbf{e}^*\| > b_\delta\sqrt{\mathsf{t}}$ (except with negligible probability).

Next, we also model $\mathbf{e2}$ as a zero-mean Normal vector whose entries have variance $\sigma_{e2}^2/3$. Then we can use the analysis from [2] (see Claim 3.4 there) to bound the Rényi divergence of order α between $\mathbf{e2}$ and $\mathbf{e2} + \mathbf{e}^*$ by

$$\rho \leq \exp\left(2\pi\|\mathbf{e}^*\|^2/(\sigma_{e2}^2/3)\right) < \exp(124080\,\mathsf{kt} \cdot \sigma_{e1}^2/\sigma_{e2}^2).$$

Therefore, setting $\sigma_{e2} \geq \sigma_{e1}\sqrt{124080\,\mathsf{kt}}$ we can bound the Rényi divergence of order $\alpha = 2$ between these distributions below $\exp(1) = e$, as needed.

- The correctness condition from Eq. (8) can now be written as

$$q^{\frac{\ell-1}{\ell}} > \sigma_{e1}\left(24\sqrt{\mathsf{kn}} + 31.8\mathsf{k}\right) + \sigma_{e1}\sqrt{124080\,\mathsf{kt}}\left(2.37\sqrt{\mathsf{n}} + 1.7\mathsf{n}\right)$$

$$\approx \sigma_{e1}\sqrt{\mathsf{k}}\left(31.8\sqrt{\mathsf{k}} + 24\sqrt{\mathsf{n}} + 835\sqrt{\mathsf{nt}} + 600\mathsf{n}\sqrt{\mathsf{t}}\right). \tag{9}$$

We remark that as n grows, the term $600\sigma_{e1}\mathsf{n}\sqrt{\mathsf{kt}}$ becomes by far the most significant term in the expression above, so this condition is implied by $q^{\frac{\ell-1}{\ell}} \approx 2^{126} \geq 2^{9.3}\sigma_{e1}\mathsf{n}\sqrt{\mathsf{kt}}$, or $\sigma_{e1}\mathsf{n}\sqrt{\mathsf{kt}} \leq 2^{126-9.3} = 2^{116.7}$.

- Another size constraint stems from our use of Lemma 3.7, where we must ensure that the bounds are small enough so we can apply that lemma. In our protocol we have a total of 10 projected vectors in all the sub-protocols specifically the "small" vectors are the LWE secrets $S = \{\mathbf{s}_i, \mathbf{r}\}$,and the "large" vectors are the LWE noise vectors $L = \{\mathbf{e}_i, \mathbf{e1}, \mathbf{e2}, \mathbf{e}\}$, we have $m_s = 2$ and $m_l = 4$ hence $m_s + 2m_l = 10$.

To use that lemma we need to ensure that for the small vectors we have $\mathsf{b_s}, \mathsf{b_r} < \sqrt{P}/(2^{19.9}\sqrt{10} \cdot \sqrt{30}) \approx 2^{102}$. For the large vectors we have $b^* = \max(\mathsf{b_{ek}}, \mathsf{b_{e1}}, \mathsf{b_{e2}}, \mathsf{b_{ed}}) = \mathsf{b_{ed}}$ and $b_* = \min(\mathsf{b_{ek}}, \mathsf{b_{e1}}, \mathsf{b_{e2}}, \mathsf{b_{ed}}) = \mathsf{b_{ek}}$, and we need $8b^*/\sqrt{b_*} \leq \sqrt{P}/(2^{19.9}\sqrt{10}) \approx 2^{104}$.

The first condition easily holds, as we see below we will have $\mathsf{k} < 2^{13}$ and therefore $\mathsf{b_s} = \mathsf{b_r} \approx 6.7\sqrt{k} \approx 2^{10.3} \ll 2^{102}$. For the second condition, with out settings we have $b_* = \mathsf{b_{ek}} = 2.37\sigma_{e1}\sqrt{\mathsf{k}}$ and

$$b^* = \mathsf{b_{ed}} \approx 1.7\sigma_{e2}\mathsf{n} + 24\sigma_{e1}\sqrt{\mathsf{kn}} \approx \sigma_{e1}\sqrt{\mathsf{k}}(600\mathsf{n} + 24\sqrt{\mathsf{n}}) < 2^{10}\sigma_{e1}\mathsf{n}\sqrt{\mathsf{k}}.$$

Hence we get the constraint

$$\frac{8 \cdot 2^{10}\sigma_{e1}\mathsf{n}\sqrt{\mathsf{k}}}{\sqrt{2.37\sigma_{e1}\sqrt{\mathsf{k}}}} \leq 2^{104} \quad \Leftrightarrow \quad \sigma_{e1}\mathsf{n}^2\sqrt{\mathsf{k}} \lessapprox 2^{184}.$$

Clearly, this constraint is subsumed by the constraint from the previous bullet, for any reasonable value of n.

- Finally, for LWE security we must set the dimension of the LWE secrets large enough relative to the bit-size of the modulus q vs. the noise magnitude. Targeting security level 128, the LWE estimators [3] implies LWE secrets of dimension roughly $37.5\log_2(q/\sigma) + 75$. [21] With $q \approx 2^{252}$, we therefore need to set

$$\mathsf{k} \geq 37.5\log_2(q/\sigma_{e1}) + 75 \approx 37.5 \cdot (254 - \log_2(\sigma_{e1})). \tag{10}$$

---

[21] The formula $\mathsf{dim} \geq 37.5\log_2(q/\sigma) + 75$ is a useful shorthand, giving results that are very close to what the estimator gives for 128-bit security level.

Given the number of parties n (which implies the values $\mathsf{n} = n \cdot \ell$ and $\mathsf{t} \approx n/2$), we therefore need to find σ_{e1} and k that satisfy both $\sigma_{e1}\mathsf{n}\sqrt{\mathsf{kt}} \leq 2^{116.7}$ and $\mathsf{k} \geq 37.5(254 - \log_2(\sigma_{e1}))$. It is easy to verify that the for parameter regime that we care about (where n is at most a few thousands), these two constraints are satisfied with

$$\sigma_{e1} := 2^{108 - \lceil (3 \log n)/2 \rceil} - 1, \ \ \sigma_{e2} := 2^{123 - \lceil (3 \log n)/2 \rceil} - 1, \ \ \mathsf{k} \approx 5438 + 37.5 \lceil (3 \log n)/2 \rceil. \quad (11)$$

For example, for $n = 1024$ we have $\sigma_{e1} = 2^{93} - 1, \sigma_{e2} = 2^{113} - 1$, and $\mathsf{k} = 6038$.

B   Aggregating DL-based Commit-and-Prove Protocols

Our scheme uses different proofs for the different components, with different vectors (or parts of them) participating in different proofs. Recalling that the DL-proofs that we use cost (roughly) two exponentiations for each variable that participate in a norm-squared proof and one exponentiation for each variable in linear proofs, we would like to reduce as much as possible the intersection between the different proofs.

In this section we describe a few transformations that we apply to these proofs, aggregating them into just two instances, once for a linear constraint and the other for a quadratic constraint. This aggregation maintain the soundness of these proofs (so that all the original constraints still hold), while minimizing the number of exponentiations needed to verify them. Specifically, variables that only participated in linear proofs in the original system will only participate in the linear constraint in the resulting system, and all the other variables will only participate in the quadratic constraint. In addition, the transformation adds *just a single variable* to the system, which is the only variable in the intersection between the linear and quadratic constraints.

B.1   The Initial System

Our starting point is a system of linear and quadratic constraints, with some vectors only participate in linear constraints (i.e., the original high-dimension vectors), some participate in both linear and norm constraints (i.e., the projected 265-dimensional vectors), and some only participate in norm constraints (i.e., the dimension-4 vectors for the sum-of-squares).

Below we denote the linear-only vectors by $\mathbf{u}_i$, the ones participating in both linear and norm constraints by $\mathbf{v}_i$ and the ones participating only in norm constraints by $\mathbf{w}_i$, $i = 1, \ldots, n$. That is, the set of constraints that need to be proven are as follows:

$$\left\{ \langle \mathbf{a}_i, \mathbf{u}_i \rangle + \langle \mathbf{b}_i, \mathbf{v}_i \rangle = p_i \right\}_{i=1}^n \text{ and } \left\{ \langle \mathbf{v}_i, \mathbf{v}_i \rangle + \langle \mathbf{w}_i, \mathbf{w}_i \rangle = q_i \right\}_{i=1}^n.$$

The prover knows all these vectors explicitly, while the verifier only knows the public $\mathbf{a}_i$, $\mathbf{b}_i$, p_i and q_i, and has commitments to everything else.

In the actual system we do not have the same number of vectors of each type, and moreover some of the linear-only vectors $\mathbf{u}_i$ participate in multiple linear proofs. But here we only describe this simplified variant, since it makes indexing much simpler while still conveying all the technical ideas. On the other hand, it is important for the transformation below that the norm constraints are disjoint on their variables.

In the Bulletproof-like system that we use for these proofs, different variable are committed relative to different generators. Specifically, for the vectors $\mathbf{v}_i$ and $\mathbf{w}_i$ that participate in the norm

constraints, the prover is supposed to provide "double commitments", namely commitment to the vectors $(\mathbf{v}_i|\mathbf{v}_i)$ $(\mathbf{w}_i|\mathbf{w}_i)$ with the different halves committed wrt different generators. For vectors $\mathbf{u}_i$ that only participate in linear constraints, the prover only commits with respect to one set of generators.

In more detail, we have vectors of generators $\mathbf{G}_i$, $\mathbf{G}'_i$, $\mathbf{G}''_i$, $\mathbf{H}'_i$, and $\mathbf{H}''_i$, $i = 1, \ldots, n$, and another generator F. These are chosen such that it is hard for the prover to find non-trivial representation of any of these generators relative to the others (using hash-to-curve). The honest prover computes and sends to the verifier the following commitments:

$$
\begin{aligned}
C_i &:= r_i F + \langle \mathbf{u}_i, \mathbf{G}_i \rangle, & \text{\# linear-only vectors} \\
C'_i &:= s_i F + \langle \mathbf{v}_i, \mathbf{G}'_i \rangle + \langle \mathbf{v}_i, \mathbf{H}'_i \rangle, & \text{\# both linear and norm} \\
C''_i &:= t_i F + \langle \mathbf{w}_i, \mathbf{G}''_i \rangle + \langle \mathbf{w}_i, \mathbf{H}''_i \rangle, & \text{\# norm-only vectors}
\end{aligned}
$$

where r_i, s_i, t_i are randomizing scalars that the prover chooses uniformly at random in $\mathbb{Z}_P$.

B.2 Step 1: Aggregating the Linear Constraints

We begin by aggregating all the linear constraints into a single one, simply by taking a random linear combination of them. Specifically, the verifier chooses a random scalar $\rho \in \mathbb{Z}_P$ and asks that the prover replaces all the linear constraints from above by the single constraint

$$
\sum_{i=1}^{n} \langle \rho^{i-1} \mathbf{a}_i, \mathbf{u}_i \rangle + \langle \rho^{i-1} \mathbf{b}_i, \mathbf{v}_i \rangle = \sum_{i=1}^{n} \rho^{i-1} p_i.
$$

This constraint can be written as $\langle \mathbf{a}^*, \mathbf{u}^* \rangle + \langle \mathbf{b}^*, \mathbf{v}^* \rangle = p^*$, where $p^* := \sum_{i=1}^{n} \rho^{i-1} p_i$ and

$$
\begin{aligned}
\mathbf{a}^* &:= (\mathbf{a}_1 | \rho \mathbf{a}_2 | \ldots | \rho^{n-1} \mathbf{a}_n), \, \mathbf{b}^* := (\mathbf{b}_1 | \rho \mathbf{b}_2 | \ldots | \rho^{n-1} \mathbf{b}_n), \\
\mathbf{u}^* &:= (\mathbf{u}_1 | \, \mathbf{u}_2 | \ldots | \mathbf{u}_n), \qquad \mathbf{v}^* := (\mathbf{v}_1 | \, \mathbf{v}_2 \, | \ldots | \mathbf{v}_n).
\end{aligned}
$$

Clearly (), if the vectors committed by the prover fail to satisfy any of the linear constraints from above, they will also fail to satisfy the aggregated constraint (except with probability n/P).

This aggregation step can be used even when some vectors appear in multiple linear constraints. For example, if a vector $\mathbf{u}$ appears in the linear constraints i and j with coefficients $\mathbf{a}_i$ and $\mathbf{a}_j$, respectively, then it will appear in the aggregate constraint with coefficients $\rho^{i-1} \mathbf{a}_i + \rho^{j-1} \mathbf{a}_j$. After this aggregation step, each of the vectors $\mathbf{u}_i, \mathbf{v}_i$ appears only once in the the aggregate linear constraint, and each of the vectors $\mathbf{v}_i, \mathbf{w}_i$ appears only once in one of the norm constraints.

So far, this transformation did not modify any of the commitments.

B.3 Step 2: Aggregating the Norm Constraints

Next we aggregate the norm constraints into a single one by taking a random linear combination of them. Specifically, the verifier chooses a random scalar $\tau \in \mathbb{Z}_P$ and asks that the prover replaces all the norm constraints from above by the single norm constraint

$$
\sum_{i=1}^{n} \langle \tau^{i-1} \mathbf{v}_i, \tau^{i-1} \mathbf{v}_i \rangle + \langle \tau^{i-1} \mathbf{w}_i, \tau^{i-1} \mathbf{w}_i \rangle = \sum_{i=1}^{n} \tau^{2(i-1)} q_i.
$$

This constraint can be written as $\langle \hat{\mathbf{v}}, \hat{\mathbf{v}} \rangle + \langle \hat{\mathbf{w}}, \hat{\mathbf{w}} \rangle = \hat{q}$, where $\hat{\mathbf{v}} := (\mathbf{v}_1 | \tau \mathbf{v}_2 | \dots | \tau^{n-1} \mathbf{v}_n)$, $\hat{\mathbf{w}} := (\mathbf{w}_1 | \tau \mathbf{w}_2 | \dots | \tau^{n-1} \mathbf{w}_n)$, and $\hat{q} := \sum_{i=1}^{n} \tau^{2(i-1)} q_i$.

As before, if the vectors committed by the prover fail to satisfy any of the original norm constraints, they will also fail to satisfy the aggregated constraint (except with probability $2n/P$).

Differently from the first step, here the prover does change the witness that it uses to prove the new constraint, multiplying the vectors that appear in the i'th constraints by τ^{i-1}. (This is where we use the condition that the different norm constraints are disjoints on their variables.) The prover and verifier will therefore also change the commitments to these vectors, modifying each C_i' to $\hat{C}_i' := \tau^{i-1} C_i'$ and similarly $\hat{C}_i'' := \tau^{i-1} C_i''$.

In addition, the verifier will also adjust the linear constraint, so that it will use the same witness $\hat{\mathbf{v}}$ as the new norm constraint. (This is important for the steps below.) Specifically, the verifier replaces the linear constraint $\langle \mathbf{a}^*, \mathbf{u}^* \rangle + \langle \mathbf{b}^*, \mathbf{v}^* \rangle = p^*$ by the equivalent constraint $\langle \mathbf{a}^*, \mathbf{u}^* \rangle + \langle \hat{\mathbf{b}}, \hat{\mathbf{v}} \rangle = p^*$, where

$$\hat{\mathbf{b}} := (\mathbf{b}_1 | (\rho/\tau) \mathbf{b}_2 | \dots | (\rho/\tau)^{n-1} \mathbf{b}_n) \text{ and } \hat{\mathbf{v}} := (\mathbf{v}_1 | \tau \mathbf{v}_2 | \dots | \tau^{n-1} \mathbf{v}_n).$$

Change of notations. To avoid dragging extra notations to the following steps, we switch notations now and denote the resulting single linear constraint and single norm constraint by

$$\langle \mathbf{a}, \mathbf{u} \rangle + \langle \mathbf{b}, \mathbf{v} \rangle = p \text{ and } \langle \mathbf{v}, \mathbf{v} \rangle + \langle \mathbf{w}, \mathbf{w} \rangle = q.$$

The honest prover has the secret vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$, and both prover and verifier know the public quantities $\mathbf{a}, \mathbf{b}, p, q$ as well as commitments to the secret vectors, which can be computed from the commitments above via $C^* := \sum_i C_i$, $C^{**} := \sum_i \hat{C}_i' + \sum_i \hat{C}_i''$. These commitments are supposed to be

$$C^* = r^* F + \langle \mathbf{u}, \mathbf{G}_u \rangle, \ C^{**} = r^{**} F + (\langle \mathbf{v}, \mathbf{G}_v \rangle + \langle \mathbf{v}, \mathbf{H}_v \rangle) + (\langle \mathbf{w}, \mathbf{G}_w \rangle + \langle \mathbf{v}, \mathbf{H}_w \rangle),$$

where the generator vectors $\mathbf{G}_u, \mathbf{G}_v, \mathbf{G}_w, \mathbf{H}_v, \mathbf{H}_w$ are the concatenation of the generators from above, and the prover knows r^*, r^{**}.

B.4 Step 3: Enforcing the Norm Constraint

Next we apply the transformation from Section 3.1 to ensure that C_v, C_w are indeed "double commitments" with the same vectors committed relative to both the G's and the H's. Let d_v be the dimension of $\mathbf{v}$ and d_w be the dimension of $\mathbf{w}$. The verifier chooses a random scalar $x \in \mathbb{Z}_P$ and sets $\mathbf{x}_v := (1, x, \dots, x^{d_v - 1})$ and $\mathbf{x}_w := (x^{d_v}, \dots, x^{d_v + d_w - 1})$, then asks the prover to replace the norm constraint by the quadratic constraint

$$\langle \mathbf{v} + \mathbf{x}_v, \mathbf{v} - \mathbf{x}_v \rangle + \langle \mathbf{w} + \mathbf{x}_w, \mathbf{w} - \mathbf{x}_w \rangle = q - \|\mathbf{x}_v\|^2 - \|\mathbf{x}_w\|^2.$$

The committed witnesses remain unchanged (i.e., the vectors $\mathbf{v}, \mathbf{w}$), but we use the fact that our underlying commit-and-prove system (as noted in Section 3.1 above) supports public offset vectors.

B.5 Step 4: Removing v from the Linear Constraint

Finally, we make the quadratic and linear constraints almost disjoint on their variables by removing $\mathbf{v}$ from the linear constraint and replacing it by just a single variable. Specifically, the prover computes the scalar $z := \langle \mathbf{a}, \mathbf{u} \rangle \pmod{P}$ and commits to it via $Z := r_z F + z G_z$ (where G_z is yet

another random generator, computed by hashing to the curve). The prover sends the commitment Z to the verifier, who replies with yet another random scalar $y \in \mathbb{Z}_P$. The verifier then asks the prover to prove the two new constraints

Linear: $\langle(\mathbf{a}|-1), (\mathbf{u}|z)\rangle = 0,$ and
Quadratic: $\langle(\mathbf{v}|\mathbf{w}|z) + \mathbf{offset}_1, (\mathbf{v}|\mathbf{w}|0) + \mathbf{offset}_2\rangle = q - \|\mathbf{x}_v\|^2 - \|\mathbf{x}_w\|^2 + y(p + \langle\mathbf{x_v}, \mathbf{b}\rangle).$

where the offset vectors are $\mathbf{offset}_1 = (\mathbf{x}_v|\mathbf{x}_w|0)$, $\mathbf{offset}_2 = (y\mathbf{b} - \mathbf{x}_v| - \mathbf{x}_w|y)$. Completeness of the linear constraint holds by definition. For the quadratic constraint, we have

$$
\begin{aligned}
&\langle(\mathbf{v}|\mathbf{w}|z) + \mathbf{offset}_1, (\mathbf{v}|\mathbf{w}|0) + \mathbf{offset}_2\rangle \\
&= \langle\mathbf{v} + \mathbf{x}_v, \mathbf{v} + y\mathbf{b} - \mathbf{x}_v\rangle + \langle\mathbf{w} + \mathbf{x}_w, \mathbf{w} - \mathbf{x}_w\rangle + yz \\
&= \langle\mathbf{v} + \mathbf{x}_v, \mathbf{v} - \mathbf{x}_v\rangle + \langle\mathbf{w} + \mathbf{x}_w, \mathbf{w} - \mathbf{x}_w\rangle + y(z + \langle\mathbf{v} + \mathbf{x}_v, \mathbf{b}\rangle) \\
&= \langle\mathbf{v} + \mathbf{x}_v, \mathbf{v} - \mathbf{x}_v\rangle + \langle\mathbf{w} + \mathbf{x}_w, \mathbf{w} - \mathbf{x}_w\rangle + y(\langle\mathbf{u}, \mathbf{a}\rangle + \langle\mathbf{v}, \mathbf{b}\rangle + \langle\mathbf{x}_v, \mathbf{b}\rangle) \\
&= q - \|\mathbf{x}_v\|^2 - \|\mathbf{x}_w\|^2 + y(p + \langle\mathbf{x}_v, \mathbf{a}\rangle),
\end{aligned}
$$

as needed. For soundness, we can assume that indeed $z := \langle\mathbf{a}, \mathbf{u}\rangle \pmod{P}$ (or else the linear constraint will fail). Then, if either of the previous two constraints was not satisfied, then there is at most one value of y for which the new quadratic constraint will be satisfied, hence this transformation only adds $1/P$ to the soundness error. The commitments are modified by incorporating also the commitment to z (as described below).

B.6 The End Result

At this point we are left with only two constraints: a linear constraints over the vector $\mathbf{u}' = (\mathbf{u}|z)$, and a quadratic constraint over the vectors $\mathbf{v}'_1 = (\mathbf{v}|\mathbf{w}|z)$ and $\mathbf{v}'_2 = (\mathbf{v}|\mathbf{w}|0)$, with the public offset vectors $\mathbf{offset}_1$, $\mathbf{offset}_2$ from above.

Both prover and verifier have commitments to the vectors $\mathbf{u}', \mathbf{v}'_1, \mathbf{v}'_2$, computed as $C_{\mathbf{u}'} := C^* + Z$, $C_{\mathbf{v}'_1} := C^{**} + Z$, and $C_{\mathbf{v}'_2} := C^{**}$. At this point the prover and verifier have everything they need for the underlying Bulletproof-like protocols that we use (cf. Appendix E). They run two instances of these protocols, one for the linear constraint and the other for the quadratic constraint.

C The Proactive VSS Scheme

Putting all these components in a proactive VSS scheme is straightforward in principle, but some optimizations are still possible. The parties maintain a Shamir sharing of some global secret s, and need to refresh that sharing in every step. Since we aim at a protocol that will be usable in the secrets-on-blockchain architecture of Benhamouda et al.[8], our high-level protocol is essentially the one described there, with the components from the current work. Some important optimizations are possible even at this high level, however, and are in fact needed to make the overall scheme practical.

Recall that in the protocol from [8], refreshing the shares is done by each party preparing a second-level sharing of its share, and then recovering its next-step share of the global secret from all the shares-of-shares that it get from everyone else. Specifically

In more detail, each party does the following in every epoch:

- Generates a re-sharing of its share, and encrypts the different shares-of-share under the public keys of the respective recipients;
- Generate proofs of valid encryption, decryption, and re-sharing, as per Appendix A;
- Generates a new secret/public key pair with a proof of valid key-generation as in Appendix A;
- Broadcasts a message consisting of the new public key, the ciphertext, and all the proofs.

The parties all listen to messages on the broadcast channel, and each party verifies all the proofs in these messages. Each party then uses its secret key to decrypt the messages addressed to it by "good senders" (i.e. those senders whose proofs verify). Since all honest parties agree on the set of good senders, and since there are more than t of them, then the honest parties can recover a Shamir sharing of the underlying secret from the messages that they received.

C.1   Reducing the Verification Cost

As described above, each party must generate one set of proofs but verify n of them. While verification of the proofs from Appendix A is somewhat cheaper than proof generation, it is not all that much cheaper. For example, for 1024 parties each verification takes about 20 seconds, so each party would need to spend close to six CPU-hours verifying all the proofs.

Such verification cost is a gross overkill, however. We note that all we need is that (a) all the honest parties agree on the set of "good senders" (of cardinality more than t), and (b) all the senders in this set have valid proofs. When the committee size is significantly more than the security parameter, $n \gg \lambda$, we can easily get that effect without every party verifying all proofs. For example, we can assign to each sender a (pseudo)random verification-committee of size only $O(\lambda)$, and decide whether or not that sender is "good" by a majority vote of that committee.

This is already an improvement, but we can still do better: We can choose smaller verification committees, then designate a sender as "good" if it has *a large majority* among the votes of this committee, and "undetermined" otherwise. Each party can then go down the list of the remaining "undetermined" senders, verifying their proofs and adding them to the good set, until the set is large enough. A standard analysis of this solution shows that it significantly reduces the verification load. For example, with a committees of $n = 1024$ parties, of which less than $1/3$ are corrupted, we can get the error probability below 2^{-128} while having each party verify only about 200-300 proofs (vs. 1024 in the naive protocol). The verification cost is therefore reduced to 1-2 CPU-hours (that can be done in only a few minutes of wall-clock if we use multiple CPUs).

The verification protocol proceeds as follows: Let k be parameter denoting the size of the verification committees, and $\tau > 1/2$ be a threshold value, both to be determined later. We first choose for each sender a (pseudo)random verification committee of size k, derived publicly from the messages of that sender, using a random oracle or VRFs. (In the analysis below we treat this set as if it were truly random.) Each member of the verification-committee verifies the proofs of their sender, and broadcasts a vote as to whether or not the proofs are valid. Once all the votes for all the senders are sent, every party does the following:

1. Set the initial good set G to include all the senders for which at least a τ-fraction of the votes are YES;
2. If $|G| < t + 1$ then order the remaining senders in order of the fraction of their YES votes, breaking ties in an arbitrary (but consistent) manner. Then verify their proofs in that order, adding every sender whose proofs verify, until the set G reaches cardinality of $t + 1$.

At a high level, the reason that this procedure works is that the expected fraction of YES for good senders is at least $2/3$ (over the choice of the verification committee), while the expected fraction for bad senders is at most $1/3$. We set τ to be the smallest fraction such that for bad senders we have $\Pr[\text{more than } \tau \text{ fraction of YES}] < 2^{-128}$, using $\tau > 1/2$ to get smaller committee-size k. This may mean that some good senders will fail to clear the threshold, but we will get them back in the second step above. It is left to analyze the expected number of proofs that parties need to verify in the second step, and show that it is only a very small number.

D   More on the Normal-Distribution Heuristic

We would like to compute the following two quantities:

$$\arg \sup_{\alpha \in \mathbb{R}} \left[\forall \mathbf{w} \in \mathbb{Z}^d \Pr_{R \leftarrow \mathcal{D}^{d \times 256}} \left[\|\mathbf{w}R\|^2 < \alpha \cdot \|\mathbf{w}\|^2 \right] < 2^{-128} \right] \tag{12}$$

$$\arg \inf_{\beta \in \mathbb{R}} \left[\forall \mathbf{w} \in \mathbb{Z}^d \Pr_{R \in \mathcal{D}^{d \times 256}} \left[\|\mathbf{w}R\|^2 > \beta \cdot \|\mathbf{w}\|^2 \right] < 2^{-128} \right] \tag{13}$$

We will obtain the values of α and β using two heuristics. The first heuristic assumes that the fattest tails are achieved by the vector $\mathbf{w}$ all of whose coefficients are the same. Hence bounding α and β involves just considering, for simplicity, the element $\mathbf{w} = 1^d$. The second heuristic substitutes the distribution $\mathcal{D}$ by the normal distribution with the same mean and standard deviation – i.e. $\frac{1}{\sqrt{2}}\mathcal{N}$.

The intuition for both heuristics stems from two lemmas proved in [1]. It is shown in [1, Lemma 7] that all the respective moments of the distribution of $\|\mathbf{w}R\|^2$ are largest among $\mathbf{w}$ of norm $\sqrt{d}$ when $\mathbf{w} = 1^d$. And Lemma 8 of that paper states that when the distribution of the coefficients of R is switched to a normal distribution with the same mean and variance as $\mathcal{D}$, then the moments of the distribution of $\|\mathbf{w}R\|^2$ are larger.

The distribution of $\|1^d \cdot R\|^2$, where the coefficients of $R \in \mathbb{R}^{d \times 256}$ are normally distributed, is the (scaled) χ^2 distribution with 256 degrees of freedom. More specifically, it's the distribution $\frac{1}{2}d \cdot \chi^2[256]$ because the variance of each coefficient of R is $\frac{1}{2}$, and so the variance of each coefficient of $\mathbf{w}R$ (which is normally distributed) is $d/2$. The two lemmas from [1] therefore intuitively imply that the tails of the scaled χ^2 distribution are fatter than those of the distribution we're trying to bound in (12) and (13). And so we can hope that by bounding those, we bound the quantities that we're interested in. Specifically,

$$\forall \mathbf{w} \in \mathbb{Z}^d \Pr_{R \leftarrow \mathcal{D}^{d \times k}} \left[\|\mathbf{w}R\|^2 < \alpha \cdot \|\mathbf{w}\|^2 \right] \lessapprox \Pr[\chi^2[k] < 2\alpha] = cdf_{\chi^2[k]}(2\alpha) \tag{14}$$

$$\forall \mathbf{w} \in \mathbb{Z}^d \Pr_{R \leftarrow \mathcal{D}^{d \times k}} \left[\|\mathbf{w}R\|^2 > \beta \cdot \|\mathbf{w}\|^2 \right] \lessapprox \Pr[\chi^2[k] > 2\beta] = 1 - cdf_{\chi^2[k]}(2\beta) \tag{15}$$

Thus computing heuristic bounds for (12) and (13) involves simply evaluating the inverse cdf of the $\chi^2[k]$ distribution, which can be done via, for example, a python command.[22] We will now look at how well the heuristic holds. Let us first examine the validity of (14) and (15) for $\mathbf{w} = 1^d$.

---

[22] See https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2.html

degree of freedom k	16	32	64	128	256
$cdf^{-1}_{\chi^2[k]}\left(2^{-k/2}\right)$	4.929	8.787	16.374	31.398	61.278
$cdf^{-1}_{\chi^2[k]}\left(1-2^{-k/2}\right)$	35.055	76.747	161.318	331.612	673.328

Table 4. The largest α (middle row) such that $cdf_{\chi^2[k]}(\alpha)$ is $< 2^{-128}$. And the smallest β (bottom row) such that $1 - cdf_{\chi^2[k]}(\beta)$ is $< 2^{-128}$.

d \ k	16	32	64	128	256
16 (left tail)	-8.08	-16.15	-32.55	-64.75	-129.62
16 (right tail)	-8.25	-16.69	-33.72	-67.76	-135.96
64 (left tail)	-8.04	-16.03	-32.13	-64.23	-128.44
64 (right tail)	-8.06	-16.17	-32.41	-64.91	-129.93
256 (left tail)	-8.01	-16.02	-32.03	-64.06	-128.11
256 (right tail)	-8.03	-16.04	-32.10	-64.23	-128.48
512 (left tail)	-8.01	-16.01	-32.02	-64.03	
512 (right tail)	-8.01	-16.02	-32.05	-64.11	

Table 5. The "left tail" is the base-2 logarithm of the cdf of the distribution $\|1^d \cdot R\|^2$, where $R \leftarrow \mathcal{D}^{d \times k}$, evaluated at $\alpha \cdot \|1^d\|^2 = \alpha d = \frac{d}{2} \cdot cdf^{-1}_{\chi^2[k]}(2^{-k/2})$. The right tail is the base-2 logarithm of $1-$cdf of the distribution $\|1^d \cdot R\|^2$,where $R \leftarrow \mathcal{D}^{d \times k}$, evaluated at $\beta \cdot \|1^d\|^2 = \beta d = \frac{d}{2} \cdot cdf^{-1}_{\chi^2[k]}(1 - 2^{-k/2})$. The $cdf^{-1}_{\chi^2[k]}$ values, which correspond to 2α (resp. 2β), are taken from Table 4.

In Table 4, we compute the inverse cdf of the $\chi^2[k]$ distribution at $2^{-k/2}$ and $1 - 2^{-k/2}$ for various values of k. [23]

If (14) and (15) were true (rather than heuristic) inequalities, then the cdf of the distribution of $\|1^d \cdot R\|^2$ evaluated at $\alpha \cdot \|\mathbf{w}\|^2 = \alpha d$ for

$$\alpha = \frac{1}{2} \cdot cdf^{-1}_{\chi^2[k]}\left(2^{-k/2}\right) \tag{16}$$

(and resp. $\beta \cdot \|\mathbf{w}\|^2 = \beta d$ for $\beta = \frac{1}{2} \cdot cdf^{-1}_{\chi^2[k]}\left(1 - 2^{-k/2}\right)$) would always be less than $2^{-k/2}$ (resp. greater than $1 - 2^{-k/2}$). Based on some experiments, for $\mathbf{w} = 1^d$, the $\lesssim$ in (14) and (15) indeed appear to be true inequalities (at least when we would like the area under the tails to be $2^{-k/2}$). In Table 5, we compute the true values of the areas under the left and right tails of the cdf of $\|1^d \cdot R\|^2$ when the boundaries are the aforementioned appropriately-scaled values derived from Table 4. For example, from Table 4, we know that $cdf^{-1}_{\chi^2[128]}\left(2^{-64}\right) = 31.398 = 2\alpha$. From Table 5, we then see that $\Pr_{R \leftarrow \mathcal{D}^{512 \times 128}}[\|1^d \cdot R\|^2 < \alpha \cdot d] < 2^{-64.03} < 2^{-64}$. We also see that the values in Table 5 approach $2^{-k/2}$ as d increases, which is consistent with the central limit theorem.

We now move on to the heuristic where we assume that the fattest tails in the distribution of $\|\mathbf{w}R\|^2$ occur when $\mathbf{w}$ is balanced. For simplicity, we will assume that the squared norm of $\mathbf{w}$ is d,

---

[23] The $k = 256$ column gives us the value of 2α and 2β that we need for (heuristically) obtaining the values for α and β in (12) and (13). Even though we are only interested in $k = 256$, we also compute for other values of k just to get a sense of how well the heuristic holds in general.
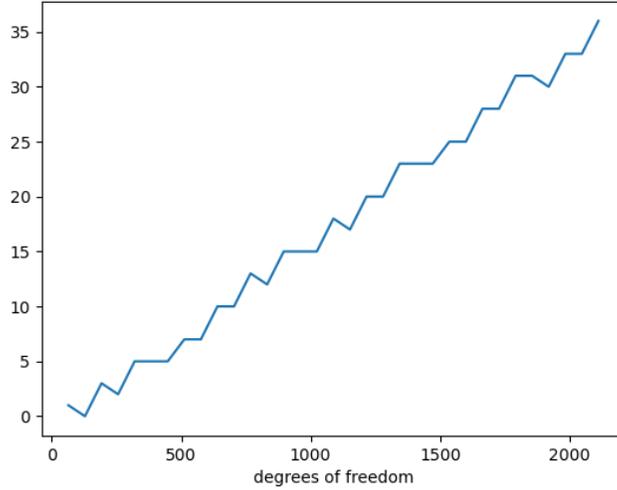
Fig. 3. The X-axis is the degrees of freedom k. If the value of (19) for $\alpha = \frac{1}{2} \cdot cdf^{-1}_{\chi^2[k]}\left(2^{-k/2}\right)$ is $2^{-k/2+\delta}$, then the Y coordinate is set to δ. For example, for $k = 256$, the value of (19) is $\approx 2^{-126}$, and thus the Y coordinate is 2. The lack of monotonicity is due to the discrete nature of the sum in (19); in particular the fact that the upper limit of the sum is defined by $\lfloor \alpha \rfloor$. A rough approximation of δ is $k/64$.

and so the fattest tails supposedly occur for $\mathbf{w} = 1^d$. In other words, we assume the heuristic:

$$\forall \mathbf{w} \in \mathbb{Z}^d \text{ s.t. } \|\mathbf{w}\|^2 = d, \Pr_{R \leftarrow \mathcal{D}^{d \times k}}\left[\|\mathbf{w}R\|^2 < \alpha\right] \lessapprox \Pr_{R \leftarrow \mathcal{D}^{d \times k}}\left[\|1^d \cdot R\|^2 < \alpha\right] \qquad (17)$$

$$\forall \mathbf{w} \in \mathbb{Z}^d \text{ s.t. } \|\mathbf{w}\|^2 = d, \Pr_{R \leftarrow \mathcal{D}^{d \times k}}\left[\|\mathbf{w}R\|^2 > \beta\right] \gtrapprox \Pr_{R \leftarrow \mathcal{D}^{d \times k}}\left[\|1^d \cdot R\|^2 > \beta\right] \qquad (18)$$

The lower bound in (18) may very well hold, but unfortunately the upper bound in (17) is false at the extreme tails. If $\mathbf{w}$ is 0 everywhere except for its first coefficient, then the distribution $\|\mathbf{w}R\|^2$ eventually flattens out at the left tail. In particular, for all positive α, $\Pr_{R \leftarrow \mathcal{D}^{d \times k}}[\|\mathbf{w} \cdot R\|^2 < \alpha] \geq 2^{-k}$. The reason is that $\mathbf{w}R = 0$ whenever the first row of R is 0, which happens with probability exactly 2^{-k}. More generally, if $\|\mathbf{w}\|^2 = d$ and all the coefficients of $\mathbf{w}$ except the first are 0 (so the first coefficient is $\sqrt{d}$), then

$$\Pr_{R \leftarrow \mathcal{D}^{d \times k}}[\|\mathbf{w} \cdot R\|^2 \leq \alpha \cdot \|\mathbf{w}\|^2] = \Pr[\text{first row of } R \text{ has at most } \lfloor \alpha \cdot \|\mathbf{w}\|^2/d \rfloor \pm 1\text{'s}]$$

$$= \Pr[\text{first row of } R \text{ has at most } \lfloor \alpha \rfloor \pm 1\text{'s}]$$

$$= \sum_{i=0}^{\lfloor \alpha \rfloor} \binom{k}{i} \cdot 2^{-k}. \qquad (19)$$

In our specific case, we would like the area under the left tail between 0 and $\alpha \cdot \|\mathbf{w}\|^2 = \alpha d$, for α as in (16), of the distribution $\|\mathbf{w}R\|^2$ to be less than $2^{-k/2}$. For $k = 256$, the value in (19) for such α is $\approx 2^{-126}$. This is slightly larger than 2^{-128} and so (14) is not an inequality, but an approximation. In Figure 3 we plot how the value of (19) differs from $2^{-k/2}$ as k grows for such α. While the difference in the exponent between the heuristic value and the actual value increases

36

(in an absolute sense), for the k and the tails that we're interested in, the heuristic appears to be fairly good.

One might also wonder whether there is yet a different shape of $\mathbf{w}$, apart from having its weight being concentrated completely on one coordinate, that may completely break our heuristic assumption. We do not believe this to be the case, as any choice of $\mathbf{w}$ that is not concentrated all on one element would result in the distribution of $\|\mathbf{w}R\|$ being closer to that of $\frac{1}{2}\chi^2$, via by the central limit theorem. It would certainly be worthwhile to have a formal proof of some version of our heuristic assumption, and we leave this to future work. But as it stands, we are fairly confident of the statements in Corollary 3.2 which is used throughout the paper.

E   Bulletproof Variations

The base recursion in Bulletproof can be generalized slightly: Recall that both prover and verifier know the generator vectors $\mathbf{G}_1, \mathbf{G}_2, \mathbf{H}_1, \mathbf{H}_2 \in \mathbb{G}^n$ and another generator F, and in addition a "commitment element" C, and the prover claims to know scalar vectors $\mathbf{a}_1, \mathbf{a}_2, \mathbf{b}_1, \mathbf{b}_2 \in \mathbb{Z}^n$ such that

$$C = \langle (\mathbf{a}_1|\mathbf{a}_2), (\mathbf{G}_1|\mathbf{G}_2) \rangle + \langle (\mathbf{b}_1|\mathbf{b}_2), (\mathbf{H}_1|\mathbf{H}_2) \rangle + \langle (\mathbf{a}_1|\mathbf{a}_2), (\mathbf{b}_1|\mathbf{b}_2) \rangle \cdot F. \tag{20}$$

The heart of the Bulletproof system is a method for reducing this system to a system of half the dimension, by having the verifier send a random challenge x and the prover replying with just two group elements L, R. A slight generalization of this method will use two challenges x, y rather than one, then the prover will send the exact same L, R and the commitment-update rule will use xy and $(xy)^{-1}$ instead of x^2 and x^{-2}, respectively. Specifically, the prover sets as normal

$$L := \langle \mathbf{a}_1, \mathbf{G}_2 \rangle + \langle \mathbf{b}_2, \mathbf{H}_1 \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle \cdot F$$
$$R := \langle \mathbf{a}_2, \mathbf{G}_1 \rangle + \langle \mathbf{b}_1, \mathbf{H}_2 \rangle + \langle \mathbf{a}_2, \mathbf{b}_1 \rangle \cdot F$$

and sends them to the verifier. The verifier sends back x, y, and then the prover sets

$$\mathbf{a}' := x \cdot \mathbf{a}_1 + y^{-1}\mathbf{a}_2$$
$$\mathbf{b}' := x^{-1} \cdot \mathbf{b}_1 + y\mathbf{b}_2$$
$$\text{and } C' := C + xy \cdot L + (xy)^{-1} \cdot R$$

and sends C' to the verifier. The prover and verifier agree on the new generator-vectors $\mathbf{G}' = x^{-1}\mathbf{G}_1 + y\mathbf{G}_2$ and $\mathbf{H}' = x\mathbf{H}_1 + y^{-1}\mathbf{H}_2$, and they now have a new instance where the prover claims that for the vectors $\mathbf{a}', \mathbf{b}'$ above it holds that

$$C' = \langle \mathbf{a}', \mathbf{G}' \rangle + \langle \mathbf{b}', \mathbf{H}' \rangle + \langle \mathbf{a}', \mathbf{b}' \rangle \cdot F. \tag{21}$$

The Bulletproof system as described in [14] corresponds to $x = y$, but one can also use other choices. In particular choosing x at random and setting $y = 1$ would be (very) slightly more efficient.

E.1 Correctness and Soundness

Seeing that Eq. (21) holds involves some mind-numbing parenthesis-opening, as follows:

$$
\begin{aligned}
C' &= C + xy \cdot L + (xy)^{-1} \cdot R \\
&= \langle(\mathbf{a}_1|\mathbf{a}_2), (\mathbf{G}_1|\mathbf{G}_2)\rangle + \langle(\mathbf{b}_1|\mathbf{b}_2), (\mathbf{H}_1|\mathbf{H}_2)\rangle + \langle(\mathbf{a}_1|\mathbf{a}_2), (\mathbf{b}_1|\mathbf{b}_2)\rangle \cdot F \\
&\quad + xy \cdot \big(\langle\mathbf{a}_1, \mathbf{G}_2\rangle + \langle\mathbf{b}_2, \mathbf{H}_1\rangle + \langle\mathbf{a}_1, \mathbf{b}_2\rangle \cdot F\big) \\
&\quad + (xy)^{-1}\big(\langle\mathbf{a}_2, \mathbf{G}_1\rangle + \langle\mathbf{b}_1, \mathbf{H}_2\rangle + \langle\mathbf{a}_2, \mathbf{b}_1\rangle \cdot F\big) \\
&= \langle\mathbf{a}_1 + (xy)^{-1}\mathbf{a}_2, \mathbf{G}_1\rangle + \langle\mathbf{a}_2 + xy\mathbf{a}_1, \mathbf{G}_2\rangle \;+\; \langle\mathbf{b}_1 + xy\mathbf{b}_2, \mathbf{H}_1\rangle \\
&\quad + \langle\mathbf{b}_2 + (xy)^{-1}\mathbf{b}_1, \mathbf{H}_2\rangle + \big(\langle\mathbf{a}_1, \mathbf{b}_1\rangle + \langle\mathbf{a}_2, \mathbf{b}_2\rangle \\
&\quad + xy\langle\mathbf{a}_1, \mathbf{b}_2\rangle + (xy)^{-1}\langle\mathbf{a}_2, \mathbf{b}_1\rangle\big) \cdot F \\
&= \langle(x\mathbf{a}_1 + y^{-1}\mathbf{a}_2), x^{-1}\mathbf{G}_1\rangle + \langle(y^{-1}\mathbf{a}_2 + x\mathbf{a}_1), y\mathbf{G}_2\rangle \\
&\quad + \langle(x^{-1}\mathbf{b}_1 + y\mathbf{b}_2), x\mathbf{H}_1\rangle + \langle(y\mathbf{b}_2 + x^{-1}\mathbf{b}_1), y^{-1}\mathbf{H}_2\rangle \\
&\quad + \big(\langle x\mathbf{a}_1, x^{-1}\mathbf{b}_1\rangle + \langle y^{-1}\mathbf{a}_2, y\mathbf{b}_2\rangle + \langle x\mathbf{a}_1, y\mathbf{b}_2\rangle + \langle x^{-1}\mathbf{a}_2, y^{-1}\mathbf{b}_1\rangle\big) \cdot F \\
&= \langle\mathbf{a}', \mathbf{G}'\rangle + \langle\mathbf{b}', \mathbf{H}'\rangle + \langle\mathbf{a}', \mathbf{b}'\rangle \cdot F
\end{aligned}
$$

The soundness proof is exactly the same as in [14], showing that rewinding the prover four times with the same L, R but with different x_i and y_i, each time getting different $\mathbf{G}'_i, \mathbf{H}'_i$ and $\mathbf{a}'_i, \mathbf{b}'_i$, $i = 1, 2, 3, 4$, we can extract the original $\mathbf{a}, \mathbf{b}$ (or else find a nontrivial discrete logarithm among the generators). Fix L, R, and we assume that for $i = 1, 2, 3, 4$ we get

$$
\begin{aligned}
C'_i &= C + x_i y_i L + (x_i y_i)^{-1} R = \langle\mathbf{a}'_i, \mathbf{G}'_i\rangle + \langle\mathbf{b}'_i, \mathbf{H}'_i\rangle + \langle\mathbf{a}'_i, \mathbf{b}'_i\rangle \cdot F \\
&= \langle x_i^{-1}\mathbf{a}'_i, \mathbf{G}_1\rangle + \langle y_i\mathbf{a}'_i, \mathbf{G}_2\rangle + \langle x_i\mathbf{b}'_i, \mathbf{H}_1\rangle + \langle y_i^{-1}\mathbf{b}'_i, \mathbf{H}_2\rangle + \langle\mathbf{a}'_i, \mathbf{b}'_i\rangle \cdot F
\end{aligned}
\tag{22}
$$

Assuming linear independence, we can use the values x_i, y_i for $i = 1, 2, 3$ to get u_i, v_i, w_i satisfying

$$
\sum_{i=1}^{3} u_i(x_i y_i) = 1, \quad \text{and} \quad \sum_{i=1}^{3} u_i = \sum_{i=1}^{3} u_i(x_i y_i)^{-1} = 0,
$$

$$
\sum_{i=1}^{3} v_i(x_i y_i)^{-1} = 1, \quad \text{and} \quad \sum_{i=1}^{3} v_i = \sum_{i=1}^{3} v_i(x_i y_i) = 0,
$$

$$
\sum_{i=1}^{3} w_i = 1, \quad \text{and} \quad \sum_{i=1}^{3} w_i(x_i y_i) = \sum_{i=1}^{3} u_i(x_i y_i)^{-1} = 0.
$$

This yields three linear combinations of Eq. (22) for $i = 1, 2, 3$, satisfying:

$$L = \sum_{i=1}^{3} u_i C_i' = \langle \underbrace{\sum_i u_i x_i^{-1} \mathbf{a}_i'}_{\mathbf{a}_{L,1}}, \mathbf{G}_1 \rangle + \langle \underbrace{\sum_i u_i y_i \mathbf{a}_i'}_{\mathbf{a}_{L,2}}, \mathbf{G}_2 \rangle$$

$$+ \langle \underbrace{\sum_i u_i x_i \mathbf{b}_i'}_{\mathbf{b}_{L,1}}, \mathbf{H}_1 \rangle + \langle \underbrace{\sum_i u_i y_i^{-1} \mathbf{b}_i'}_{\mathbf{b}_{L,2}}, \mathbf{H}_2 \rangle + \underbrace{\sum_i u_i \langle \mathbf{a}_i', \mathbf{b}_i' \rangle}_{\gamma_L} \cdot F \qquad (23)$$

$$R = \sum_{i=1}^{3} v_i C_i' = \langle \underbrace{\sum_i v_i x_i^{-1} \mathbf{a}_i'}_{\mathbf{a}_{R,1}}, \mathbf{G}_1 \rangle + \langle \underbrace{\sum_i v_i y_i \mathbf{a}_i'}_{\mathbf{a}_{R,2}}, \mathbf{G}_2 \rangle$$

$$+ \langle \underbrace{\sum_i v_i x_i \mathbf{b}_i'}_{\mathbf{b}_{R,1}}, \mathbf{H}_1 \rangle + \langle \underbrace{\sum_i v_i y_i^{-1} \mathbf{b}_i'}_{\mathbf{b}_{R,2}}, \mathbf{H}_2 \rangle + \underbrace{\sum_i v_i \langle \mathbf{a}_i', \mathbf{b}_i' \rangle}_{\gamma_R} \cdot F \qquad (24)$$

$$C = \sum_{i=1}^{3} w_i C_i' = \langle \underbrace{\sum_i w_i x_i^{-1} \mathbf{a}_i'}_{\mathbf{a}_{C,1}}, \mathbf{G}_1 \rangle + \langle \underbrace{\sum_i w_i y_i \mathbf{a}_i'}_{\mathbf{a}_{C,2}}, \mathbf{G}_2 \rangle$$

$$+ \langle \underbrace{\sum_i w_i x_i \mathbf{b}_i'}_{\mathbf{b}_{C,1}}, \mathbf{H}_1 \rangle + \langle \underbrace{\sum_i w_i y_i^{-1} \mathbf{b}_i'}_{\mathbf{b}_{C,2}}, \mathbf{H}_2 \rangle + \underbrace{\sum_i w_i \langle \mathbf{a}_i', \mathbf{b}_i' \rangle}_{\gamma_C} \cdot F \qquad (25)$$

Substituting the expressions for L, R, C from Eqs. (23-25) back into Eq. (22) we get for all $i = 1, 2, 3, 4$

$$\begin{aligned}
& \langle \mathbf{a}_{C,1}, \mathbf{G}_1 \rangle + \langle \mathbf{a}_{C,2}, \mathbf{G}_2 \rangle + \langle \mathbf{b}_{C,1}, \mathbf{H}_1 \rangle + \langle \mathbf{b}_{C,2}, \mathbf{H}_2 \rangle + \gamma_C \cdot F && // \ C \\
& + \ x_i y_i \cdot \big(\langle \mathbf{a}_{L,1}, \mathbf{G}_1 \rangle + \langle \mathbf{a}_{L,2}, \mathbf{G}_2 \rangle + \langle \mathbf{b}_{L,1}, \mathbf{H}_1 \rangle + \langle \mathbf{b}_{L,2}, \mathbf{H}_2 \rangle + \gamma_L \cdot F \big) && // \ + x_i y_i L \\
& + \ (x_i y_i)^{-1} \big(\langle \mathbf{a}_{R,1}, \mathbf{G}_1 \rangle + \langle \mathbf{a}_{R,2}, \mathbf{G}_2 \rangle + \langle \mathbf{b}_{R,1}, \mathbf{H}_1 \rangle + \langle \mathbf{b}_{R,2}, \mathbf{H}_2 \rangle + \gamma_R \cdot F \big) && // \ + (x_i y_i)^{-1} R \\
& = \langle x_i^{-1} \mathbf{a}_i', \mathbf{G}_1 \rangle + \langle y_i \mathbf{a}_i', \mathbf{G}_2 \rangle + \langle x_i \mathbf{b}_i', \mathbf{H}_1 \rangle + \langle y_i^{-1} \mathbf{b}_i', \mathbf{H}_2 \rangle + \langle \mathbf{a}_i', \mathbf{b}_i' \rangle \cdot F
\end{aligned}$$

Note that the extractor knows all the scalars $\mathbf{a}_\star, \mathbf{a}_\star', \mathbf{b}_\star, \mathbf{b}_\star'$ and $\gamma_\star$ in the equations above. So either is can find a nontrivial discrete logarithm between these generators, or else for all i the following equalities hold:

$$\begin{aligned}
x_i^{-1} \mathbf{a}_i' &= \mathbf{a}_{C,1} + x_i y_i \cdot \mathbf{a}_{L,1} + (x_i y_i)^{-1} \mathbf{a}_{R,1} \\
y_i \cdot \mathbf{a}_i' &= \mathbf{a}_{C,2} + x_i y_i \cdot \mathbf{a}_{L,2} + (x_i y_i)^{-1} \mathbf{a}_{R,2} \\
x_i \cdot \mathbf{b}_i' &= \mathbf{b}_{C,1} + x_i y_i \cdot \mathbf{b}_{L,1} + (x_i y_i)^{-1} \mathbf{b}_{R,1} \\
y_i^{-1} \mathbf{b}_i' &= \mathbf{b}_{C,2} + x_i y_i \cdot \mathbf{b}_{L,2} + (x_i y_i)^{-1} \mathbf{b}_{R,2} \\
\langle \mathbf{a}_i', \mathbf{b}_i' \rangle &= \gamma_C + x_i y_i \cdot \gamma_L + (x_i y_i)^{-1} \gamma_R \qquad (26)
\end{aligned}$$

The first four equalities above imply that we gave for all $i = 1, 2, 3, 4$:

$$x_i^2 y_i \cdot \mathbf{a}_{L,1} \ + \ x_i(\mathbf{a}_{C,1} - \mathbf{a}_{L,2}) \ + \ y_i^{-1}(\mathbf{a}_{R,1} - \mathbf{a}_{C,2}) \ - \ x_i^{-1} y_i^{-2} \mathbf{a}_{R,2} \ = \ 0 \qquad (27)$$

$$x_i^{-2} y_i^{-1} \mathbf{b}_{R,1} + \ x_i^{-1}(\mathbf{b}_{C,1} - \mathbf{b}_{R,2}) \ + \ y_i(\mathbf{b}_{L,1} - \mathbf{b}_{C,2}) \ - \ x_i y_i^2 \cdot \mathbf{b}_{L,2} \ = \ 0 \qquad (28)$$

We can view the last two equations as a homogeneous linear systems with:

– four equations ($i = 1, 2, 3, 4$),

- in four variables (the expression in the **a**'s and **b**'s),
- with the coefficients being the expressions the the x's and y's.

Note again that the terms $\mathbf{a}_\star, \mathbf{b}_\star$ in these equalities do not depend on the choice of x_i, y_i, or else we could extract from the prover nontrivial discrete logarithm relative to the bases $\mathbf{G}, \mathbf{H}, F$. Hence they must satisfy these equalities for any choice of $x_i, y_i,$[24] and since whp the coefficients are linearly independent it must be that the expressions in the **a**'s and **b**'s are all zero. Namely we must have

$$\mathbf{a}_{L,1} = \mathbf{a}_{C,1} - \mathbf{a}_{L,2} = \mathbf{a}_{R,1} - \mathbf{a}_{C,2} = \mathbf{a}_{R,2} = \mathbf{b}_{R,1} = \mathbf{b}_{C,1} - \mathbf{b}_{R,2} = \mathbf{b}_{L,1} - \mathbf{b}_{C,2} = \mathbf{b}_{L,2} = 0. \quad (29)$$

Plugging these equalities back into the top equalities from Eq. (26) we get

$$\mathbf{a}'_i = x_i \cdot \mathbf{a}_{C,1} + y_i^{-1}\mathbf{a}_{C,2} \ \text{ and } \ \mathbf{b}'_i = x_i^{-1}\mathbf{b}_{C,1} + y_i \cdot \mathbf{b}_{C,2}$$

and plugging these into the bottom equality from Eq. (26) we have for all i

$$\gamma_C + x_i y_i \cdot \gamma_L + (x_i y_i)^{-1}\gamma_R = \langle \mathbf{a}'_i, \mathbf{b}'_i \rangle$$
$$= \langle \mathbf{a}_{C,1}, \mathbf{b}_{C,1} \rangle + \langle \mathbf{a}_{C,2}, \mathbf{b}_{C,2} \rangle + x_i y_i \cdot \langle \mathbf{a}_{C,1}, \mathbf{b}_{C,2} \rangle + (x_i y_i)^{-1}\langle \mathbf{a}_{C,2}, \mathbf{b}_{C,1} \rangle$$

Observing yet again that the terms $\mathbf{a}_\star, \mathbf{b}_\star$ do not depend on the choice of x_i, y_i, it must therefore be the case that $\gamma_C = \langle \mathbf{a}_{C,1}, \mathbf{b}_{C,1} \rangle + \langle \mathbf{a}_{C,2}, \mathbf{b}_{C,2} \rangle$. The extractor therefore sets $\mathbf{a} = (\mathbf{a}_{C,1}|\mathbf{a}_{C,2})$ and $\mathbf{b} = (\mathbf{b}_{C,1}|\mathbf{b}_{C,2})$, which together with γ_C are a valid witness for the original statement.

E.2 Zero Knowledge

The core system above is not zero-knowledge, in particular because L, R contain information about $\mathbf{a}, \mathbf{b}$. The Bulletproof paper [14] shows how to use it as-is to get zero-knowledge range proofs (as well as proofs for general arithmetic circuits), by changing the statement being proved. This may be a generic transformation from non-ZK to ZK inner-product argument system, but in [14] it is conflated with the range proof (or Hadamard product). This transformation requires that the prover computes at least $2n + 1$ more exponentiations (and a handful more for the verifier).

Here we describe another approach, modifying the core protocol itself to make it directly zero-knowledge. This modification induces (almost) no additional exponentiations for the prover or verifier. The construction consists of two steps: in the first step the verifier sends a new random generator F', then ask the prover to prove a weaker inner-product statement, namely one that contains an arbitrary exponent for F'. Since F' is fresh and random, then barring finding nontrivial discrete-logarithm, the only such representation must include $(F')^0$. Also as it turns out, it is easier to recuse on this weak inner-product proof while keeping it zero-knowledge.

The public input to the protocol are the generators $\mathbf{G}, \mathbf{H}, F$ and the element C, and the prover claims to know $\mathbf{a}, \mathbf{b}$ such that $C = \langle (\mathbf{a}_1|\mathbf{a}_2), (\mathbf{G}_1|\mathbf{G}_2) \rangle + \langle (\mathbf{b}_1|\mathbf{b}_2), (\mathbf{H}_1|\mathbf{H}_2) \rangle + (\langle (\mathbf{a}_1|\mathbf{a}_2), (\mathbf{b}_1|\mathbf{b}_2) \rangle) \cdot F$.

The protocol begins with the verifier choosing a new random generator F' and sending it to the prover, asking the prover to prove the knowledge of $\mathbf{a}, \mathbf{b}$ *and also another scalar* δ such that

$$C = \langle (\mathbf{a}_1|\mathbf{a}_2), (\mathbf{G}_1|\mathbf{G}_2) \rangle + \langle (\mathbf{b}_1|\mathbf{b}_2), (\mathbf{H}_1|\mathbf{H}_2) \rangle + (\langle (\mathbf{a}_1|\mathbf{a}_2), (\mathbf{b}_1|\mathbf{b}_2) \rangle) \cdot F \underbrace{+ \delta \cdot F'}_{\text{new}}.$$

---

[24] More precisely, only for those x_i's and y_i's for which the prover does not abort. Observe that we assume that it answers with noticeable probability, and the probability of the linear system being degenerate is negligible.

The prover then chooses at random δ_L and δ_R and sets

$$L := \langle \mathbf{a}_1, \mathbf{G}_2 \rangle + \langle \mathbf{b}_2, \mathbf{H}_1 \rangle + \langle \mathbf{a}_1, \mathbf{b}_2 \rangle \cdot F + \delta_L \cdot F'$$
$$R := \langle \mathbf{a}_2, \mathbf{G}_1 \rangle + \langle \mathbf{b}_1, \mathbf{H}_2 \rangle + \langle \mathbf{a}_2, \mathbf{b}_1 \rangle \cdot F + \delta_R \cdot F'$$

and sends them to the verifier. The rest of this step is as usual, the verifier replies with random x, y, the prover sets

$$\mathbf{a}' := x \cdot \mathbf{a}_1 + y^{-1} \mathbf{a}_2$$
$$\mathbf{b}' := x^{-1} \cdot \mathbf{b}_1 + y \mathbf{b}_2$$
$$\text{and } C' := C + xy \cdot L + (xy)^{-1} \cdot R$$

and sends C' to the verifier, and they both agree on the new generator-vectors $\mathbf{G}' = x^{-1} \mathbf{G}_1 + y \mathbf{G}_2$ and $\mathbf{H}' = x \mathbf{H}_1 + y^{-1} \mathbf{H}_2$. They now have a new instance of half the dimension, where the prover claims that the vectors $\mathbf{a}'$, $\mathbf{b}'$ and $\delta' = \delta + xy\delta_L + (xy)^{-1}\delta_R$ satisfy

$$C' = \langle \mathbf{a}', \mathbf{G}' \rangle + \langle \mathbf{b}', \mathbf{H}' \rangle + \langle \mathbf{a}', \mathbf{b}' \rangle \cdot F + \delta' \cdot F'.$$

The parties continue by recursion, each time cutting in half the dimension. At the base of the recursion ($n = 1$), the parties use an arbitrary ZKPOK protocol to prove the last step. For example they can use a Schnorr-based protocol, or a ZK transformation similar to the Bulletproof paper (as described in Appendix E.2 below).

To see that the modified protocol remains sound, note that given three δ_i's for three different choices of x_i, y_i, it is possible to recover the original δ and the values δ_L', δ_R'. This allows the extractor to recover the transcript of the underlying non-ZK Bulletproof protocol, and then it can use the extraction procedure from above. Moreover, the final extracted δ must be zero, since any other value would imply finding a nontrivial discrete logarithm for the base F'.

To argue zero-knowledge, note that the L, R values sent in the protocol are completely random and independent elements of G, so the simulator can just choose them at random. Then the simulator only needs to supply the transcript of the base step, for which it can use the ZK-simulator of that step.

A different "generic" ZK transformation The weaker inner-product protocol is also easier to make zero knowledge directly. Fix an order-P group and $2n + 2$ generators $\mathbf{G}, \mathbf{H}, F, F'$. The public input is a group element C, and the prover claims that it knows $\mathbf{a}, \mathbf{b}, \delta$ such that $C = \langle \mathbf{a}, \mathbf{G} \rangle + \langle \mathbf{b}, \mathbf{H} \rangle + \langle \mathbf{a}, \mathbf{b} \rangle \cdot F + \delta \cdot F'$. To get a zero-knowledge proof of knowledge, the prover does the following:

- Chooses at random two n-vectors $\mathbf{r}, \mathbf{s} \leftarrow \mathbb{Z}_p^n$ and two more scalars $u, v \leftarrow \mathbb{Z}_p$.
- Send to the verifier $R := \langle \mathbf{r}, \mathbf{G} \rangle + \langle \mathbf{s}, \mathbf{H} \rangle + (\langle \mathbf{a}, \mathbf{s} \rangle + \langle \mathbf{b}, \mathbf{r} \rangle) \cdot F + u \cdot F'$ and $S := \langle \mathbf{r}, \mathbf{s} \rangle \cdot F + v \cdot F'$.

The verifier replies with a random challenge $x \leftarrow \mathbb{Z}_p$, and the two parties compute the new public input

$$C' := C + xR + x^2 S.$$

The prover then proves knowledge of $\mathbf{a}', \mathbf{b}', \delta'$ such that $C' = \langle \mathbf{a}', \mathbf{G} \rangle + \langle \mathbf{b}', \mathbf{H} \rangle + \langle \mathbf{a}', \mathbf{b}' \rangle + \cdot F + \delta' \cdot F'$ (either by just sending them, or by engaging in some other POK protocol). The honest prover computes these quantities as

$$\mathbf{a}' := \mathbf{a} + x\mathbf{r}, \ \mathbf{b}' := \mathbf{b} + x\mathbf{s}, \text{ and } \delta' = \delta + ux + vx^2.$$

41

The prover overhead in this transformation is one multi-exponentiation of dimension $(2n + 2)$ for R and two dimension-2 multi-exponentiations, one for S and one for C'. The verifier's overhead is just the dimension-2 multi-exponentiation for C'.

Soundness is proved as usual: it is possible to recover the original $\mathbf{a}, \mathbf{b}, \delta$ given the prover's answers $\mathbf{a}', \mathbf{b}', \delta'$ for three different x'es. To prove honest-verifier zero-knowledge we need to show a simulator that given a random $x \in \mathbb{Z}_p$ generates an accepted transcript which is consistent with that x.

E.3 Lightweight Bulletproof for Linear Relations

The power of Bulletproof comes from its ability to prove quadratic relations (i.e. inner-product) on committed values, but in our setting we often just need to prove linear relations. It turns out that in that case we can save a factor of two in the number of exponentiations that are needed, making the proof as computationally efficient as a Schnorr proof (but a lot shorter). Proving a linear relation can be casted as an inner-product proof where $\mathbf{a}$ is secret but $\mathbf{b}$ is public. In that case, we do not need to commit to $\mathbf{b}$ and therefore we can use a Bulletproof version where we replace the generators $\mathbf{H}$ by the identity element in the group, namely setting $\mathbf{H} = \mathbf{0}$.

This lets us prove a dimension-n linear relation using multi-exponentiation of dimension only (slightly more than) n, which is the same as in Schnorr proofs. One can verify that the soundness arguments above carry over also to this case, when we drop all the terms corresponding to $\mathbf{H}$ from all the formulas. Below is a complete description of the scheme, including the ZK part and the implementation using a single multi-exponentiation.

Parameters. A dimension $n = 2^\ell$, and an order-p group $\mathbb{G}$ with $n+2$ generators $B, F, G_0, \ldots, G_{n-1}$. Below we denote $\mathbf{G} = (G_0, \ldots, G_{n-1})$, $\mathbf{G}_0 = (G_0, \ldots, G_{\frac{n}{2}-1})$, and $\mathbf{G}_1 = (G_{\frac{n}{2}}, \ldots, G_{n-1})$.

Common input. An n-vector of scalars $\mathbf{b} = (b_0, \ldots, b_{n-1}) \in \mathbb{Z}_p^n$ and a group element $C \in \mathbb{G}$. Below we denote $\mathbf{b}_0 = (b_0, \ldots, b_{\frac{n}{2}-1})$ and $\mathbf{b}_1 = (b_{\frac{n}{2}}, \ldots, b_{n-1})$.

Prover input. The prover claims to know an n-vector of scalars $\mathbf{a} = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}_p^n$ and another scalar $r \in \mathbb{Z}_p$ such that $C = r \cdot B + \langle \mathbf{a}, \mathbf{b} \rangle \cdot F + \langle \mathbf{a}, \mathbf{G} \rangle$. Below we denote $\mathbf{a}_0 = (a_0, \ldots, a_{\frac{n}{2}-1})$ and $\mathbf{a}_1 = (a_{\frac{n}{2}}, \ldots, a_{n-1})$.

Base case, $n = 1$: We just use Schnorr proof in this case.

- The prover chooses at random $s, t \leftarrow \mathbb{Z}_p$ and sends to the verifier $S := tB + sb_0F + sG_0$.
- The verifier sends a random challenge $x \leftarrow \mathbb{Z}_p$.
- The prover replies with $a' = s + xa_0$ and $r' = t + xr$.
- The verifier accepts if $S + xC = r'B + a'b_0F + a'G_0$, and rejects otherwise.

For completeness, we have

$$
\begin{aligned}
S + xC &= tB + sb_0F + sG_0 + x(rB + a_0b_0F + a_0G_0) \\
&= (t + xr)B + (s + xa_0)b_0F + (s + xa_0)G_0.
\end{aligned}
$$

Soundness and honest-verifier zero-knowledge follows since this is a simple Schnorr proof.

Recursive step, $n = 2n'$: Here we use the lightweight Bulletproof recursion: Recall that both parties know C and $\mathbf{b} = (\mathbf{b}_0|\mathbf{b}_1) \in \mathbb{Z}_p^n$, and the prover also knows $\mathbf{a} = (\mathbf{a}_0|\mathbf{a}_1) \in \mathbb{Z}_p^n$ and $r \in \mathbb{Z}_p$.

- The prover chooses at random $s, t \leftarrow \mathbb{Z}_p$ and sends to the verifier

$$L := sB + \langle \mathbf{a}_0, \mathbf{b}_1 \rangle F + \langle \mathbf{a}_0, \mathbf{G}_1 \rangle \text{ and } R := tB + \langle \mathbf{a}_1, \mathbf{b}_0 \rangle F + \langle \mathbf{a}_1, \mathbf{G}_0 \rangle.$$

- The verifier sends a random challenge $x \leftarrow \mathbb{Z}_p$.
- The two parties continue recursively with an instance of dimension n', where they both know

$$C' := C + xL + x^{-1}R \text{ and } \mathbf{G}' := \mathbf{G}_0 + x\mathbf{G}_1, \ \mathbf{b}' := \mathbf{b}_0 + x\mathbf{b}_1.$$

The prover claims to know $\mathbf{a}' \in \mathbb{Z}_p^{n'}$ and $r' \in \mathbb{Z}_p$ such that $C' = r'B + \langle \mathbf{a}', \mathbf{b}' \rangle F + \langle \mathbf{a}', \mathbf{G}' \rangle$, which the honest prover computes as $\mathbf{a}' := \mathbf{a}_0 + x^{-1}\mathbf{a}_1$ and $r' := r + xs + x^{-1}t$.

Putting it together.

Below we number the recursive rounds in the protocol from $\ell - 1$ (first) to 0 (last), and use the following notation: For an ℓ-vector $\mathbf{x} = (x_{\ell-1}, \ldots, x_0) \in \mathbb{Z}_p^\ell$ and an index $0 \le i < n = 2^\ell$, let $\mathbf{x}\langle i \rangle \in \mathbb{Z}_p$ be the subset-product of the x_j's, consisting of all the indexes j corresponding to 1-bits in the binary representation of i (with 0 being the LSB and $\ell - 1$ the MSB). For example, $\mathbf{x}\langle 0 \rangle = 1$, $\mathbf{x}\langle 1 \rangle = x_0$, $\mathbf{x}\langle 2 \rangle = x_1$, and $\mathbf{x}\langle 3 \rangle = x_0 x_1$. Unrolling the recursion, the overall protocol is as follows:

Input: $\text{Both}(B, F, G_0, \ldots, G_{2^\ell - 1}, C, b_0, \ldots, b_{2^\ell - 1})$, $\text{Prover}(a_0, \ldots, a_{2^\ell - 1}, r)$:

\# Run through the ℓ recursive steps
1. For $j = \ell - 1$ down to 0:
2.   Set $n := 2^j$ \# Instance of size $2n$ at this level
3.   Prover chooses $s_j, t_j \leftarrow \mathbb{Z}_p$
4.   Prover computes $L_j := s_j \cdot B + (\sum_{i=0}^{n-1} a_i b_{i+n}) \cdot F + \sum_{i=0}^{n-1} a_i \cdot G_{i+n}$
              and $R_j := t_j \cdot B + (\sum_{i=0}^{n-1} a_{i+n} b_i) \cdot F + \sum_{i=0}^{n-1} a_{i+n} \cdot G_i$
5.   Prover sends L_j, R_j to verifier, who chooses $x_j \leftarrow \mathbb{Z}_p$ and sends back to prover
6.   For $i = 0$ to $n - 1$:   \# Update the first half of the a_i, b_i, G_i'ss
7.   Both prover and verifier set $b_i := b_i + x_j \cdot b_{i+n}$
8.   Prover sets $G_i := G_i + x_j \cdot G_{i+n}$ and $a_i := a_i + x_j^{-1} a_{i+n}$
9.   Prover sets $r := r + x_j s_j + x_j^{-1} t_j$

\# Do the final base step
10. Prover chooses $s^*, t^* \leftarrow \mathbb{Z}_p$ and computes $S := t^*B + s^*b_0 F + s^*G_0$
11. Prover sends S to verifier, who chooses $x^* \leftarrow \mathbb{Z}_p$ and sends to prover
12. Prover computes $a^* := s^* + x^*a_0$ and $r^* := t^* + x^*r$, and sends both to verifier

Finally, the verifier accepts if and only if

$$S = r^* \cdot B + a^*b_0 \cdot F - x^* \cdot \left(C + \sum_{j=0}^{\ell-1} x_j^{-1} \cdot L_j + \sum_{j=0}^{\ell-1} x_j \cdot R_j\right) + \sum_{i=0}^{2^\ell - 1} a^* \mathbf{x}\langle i \rangle \cdot G_i.$$

The prover in this protocol computes a total of $3 \cdot 2^\ell + 2\ell - 1$ exponentiations (most of them part of large multi-exponentiations), and the verifier computes a single multi-exponentiation of dimension $2^\ell + 2\ell + 3$.