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

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

[1] Algorand Foundation, USA
[2] IBM Research, Swisserland

Abstract. Non-interactive publicly verifiable secret sharing (PVSS) schemes enables (re-)sharing of secrets 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. These 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. That application needs very large committees with thousands of parties, so the PVSS scheme in use must be efficient enough to support such large committees, in terms of both 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 often have long ciphertexts and public keys. We use the following two techniques to conserve bandwidth: 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. The resulting scheme yields $\Omega(1)$ amortized plaintext/ciphertext rate, where concretely the rate is $\approx 1/60$ for 100 parties, $\approx 1/8$ for 1000 parties, and 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/decryption of shares.

Alternating between the lattice and DL settings is relatively painless, as we equate the LWE modulus with the order of the group. We also show how to reduce the the number of exponentiations in the bulletproofs by applying Johnson-Lindenstrauss-like compression to reduce the dimension of the vectors whose properties must be verified.

An implementation of our PVSS with 1000 parties showed that it is feasible even at that size, and should remain so even with one or two order of magnitude increase in the committee size.

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 [16, 28, 54, 22, 52, 11, 56, 20, 14, 51, 32, 33, 23, 37, 50, 31]. They were also recently proposed as enablers of secure computation on large-scale distributed networks such as public blockchains [7, 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 [7, 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. [16], 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.

[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.

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 [16] happened to be publicly verifiable. GMW [28] also 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 [22, 52, 11, 56, 20, 14, 51, 32, 33, 23, 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 [14] 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, 23]. 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.

[5] In earlier work, Fouque and Stern [20] informally present a somewhat similar scheme.

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

PVSS schemes that encrypt shares "in the exponent" include [52, 37, 31]. In those schemes, recovering the secret itself requires solving DL, which is only possible when the secret is small. For example, Groth's PVSS scheme [31, 30], affiliated with the Dfinity blockchain, shares the secret for BLS signing [9] by dividing it "into small chunks, which can be encrypted in the exponent and later extracted using the Baby-step Giant-step method". That 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 [19, 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 [8, 21]. A disadvantage is that these schemes are limited to settings where 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 [12]. 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 [24]. (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, 17] 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, 18].

In this paper, we are motivated in part by the blockchain setting, where PVSS can help enable a blockchain to "keep a secret" [7] that it can use to sign or decrypt conditioned upon events, but where bandwidth is at a premium.

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

Currently, blockchains almost exclusively use proof systems based on QAPs [46, 29] or bulletproofs [13], 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 adapting the Peikert-Vaikuntanathan-Waters (PVW) encryption scheme [47] to the multi-receiver setting. Another apparent disadvantage is that, for lattice-based PVSS, proving that ciphertexts are well-formed requires 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 (say 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

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

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

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

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, ours is the first use of the PVW lattice-based encryption scheme in the multi-receiver setting. Proving the security of PVW in this setting is subtle: 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 [13]. 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. [10] 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.

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

[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.

[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

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 $\|\mathbf{v}R\| \approx \sqrt{n}\|\mathbf{v}\|$, 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 $\mathbf{v}R \bmod q$ has a small one. This event can clearly

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.

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 [6, 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 Section 4, we provide details about our implementation. In the long version [25] we describe in more detail the various sub-protocols that the parties run locally, for key-generation, encryption, decryption, and secret re-sharing, explain how to aggregate aggregate the bulletproof instances from all these components into just two bulletproof instances, and finally put all these components together in a (proactive) publicly-verifiable secret-sharing protocol.

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 The long version [25]. 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

[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 $p \ll q$.

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.

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 $B_i := S_i A + E_i \bmod q$.

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 $B_i\mathbf{r} + \mathbf{e}_i$ 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 provide a proof of knowledge of S_i, E_i such that $B_i := S_iA + E_i \bmod q$, and moreover where the l_2 norm of the rows in S_i, E_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 $\mathbf{e}_1$.

Fix the LWE parameters $k, n, q, \chi_s, \chi_{e1}$, and let $\rho_s, \rho_s \in \mathbb{R}$ be factors that bound the size of vector from χ_s, χ_{e1}, respectively, along any fixed direction. Specifically, we require 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

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

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 $\delta \le \lceil \rho_s\beta_e + \rho_e\beta_s \rceil$. Then consider the following game between an adversary and a challenger:

- The challenger chooses, sends to the adversary a random matrix $A \in \mathbb{Z}_q^{k\times k}$.
- The adversary chooses $S \in \mathbb{Z}_q^{n\times k}$ 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 sets $B = SA + E \bmod q$ and sends S, E, B to the challenger.[15]
- 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.

[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.

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

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

The last relation follows directly from the hardness of decision LWE with these parameters. To see why the penultimate relation holds, note that $\|E\mathbf{r}-S\mathbf{e}\|_\infty \le \rho_s\beta_e+\rho_e\beta_s$ except with a negligible probability, and therefore $\mathbf{e}_2$ is statistically close to $E\mathbf{r}-S\mathbf{e}_1+\mathbf{e}_2$.

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 $\mathbf{Br}+\mathbf{e}_2$, 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 $\boldsymbol{\chi_{e2}}$ 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 $\mathtt{t} < 1000$ samples from the "leakage". We therefore took a pragmatic approach, making χ_{e2} only large enough so the distributions $\chi_{e2}^{\mathtt{t}}$ and $\chi_{e2}^{\mathtt{t}}+\delta$ 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 the long version [25] we 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 $E(\mathbf{v})$ 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 (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 $\mathbf{g} = (\Delta^{\ell-1}, \ldots, \Delta, 1) \in \mathbb{Z}_q^\ell$, 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 $\mathbf{e}_2 \leftarrow \chi_{e2}^k$, 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., [15]), 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 $(\mathsf{commit}, sid, \mathbf{w} \in \mathbb{Z}^d, c \in \{0,1\}^*)$ 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 $(\mathsf{receipt}, sid, c, d)$ to everyone.

Proof. Given a message $(\mathsf{prove}, sid, c, b \in \mathbb{R})$ 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 $(\mathsf{proof}, sid, c, b)$ 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 Bulletproof protocol [13]. (In this case the parameter P is the order of the hard-discrete-logarithm group.) The Bulletproof variants that we use are described in the long version [25], where we also show how to use some homomorphic properties in order to aggregate them.

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

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

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 $\mathbf{u} = (1, u, u^2, \dots, u^{d-1})$. 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.

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 the long version [25]. 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:*

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

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

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

Corollary 3.2. *[heuristic] Let $\mathcal{D}$ be a distribution on $\{0,\pm1\}$ 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 $\mathbf{w} \in \mathbb{Z}^d$:*

$$\begin{aligned}\Pr_{\mathbf{r}\leftarrow\mathcal{D}^d}\;[|\langle\mathbf{w},\mathbf{r}\rangle| > 9.75\cdot\|\mathbf{w}\|] &\lessapprox 2^{-141},\\ \Pr_{R\leftarrow\mathcal{D}^{d\times 256}}[\|\mathbf{w}R\|^2 < 30\cdot\|\mathbf{w}\|^2] &\lessapprox 2^{-128},\\ \Pr_{R\leftarrow\mathcal{D}^{d\times 256}}[\|\mathbf{w}R\|^2 > 337\cdot\|\mathbf{w}\|^2] &\lessapprox 2^{-128},\end{aligned}$$

where $\lessapprox$ denotes a heuristic bound.

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:

Corollary 3.3. *Fix $d, P \in \mathbb{Z}$ and a bound $b \le P/45d$, and let $\mathbf{w} \in [\pm P/2]^d$ with $\|\mathbf{w}\| \ge b$. Let $\mathcal{D}[0] = 1/2$ and $\mathcal{D}[\pm1] = 1/4$, 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 $\|\mathbf{w}\|_\infty \ge P/4d$. 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 $\{0,\pm1\}$ 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 $b \le P/45d$ then $P/8d > b\sqrt{30}$ and therefore

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

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

– 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*

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

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 $\mathbf{y} \leftarrow [\ \pm\lceil \frac{b}{2}(1+\frac{1}{\lambda})\rceil\ ]^\lambda$ 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.

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

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

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

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

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

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

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

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 minuscule.)

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 $\|\mathbf{w}\| \le b$ 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

[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.

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 $\mathbf{b} = \mathbf{s}A + \mathbf{e} \pmod q$, 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 $\mathbf{e}_i, \mathbf{x}_i \in \mathbb{Z}_P^{d_i}$, where $\|\mathbf{s}_i\| \le b_i/\gamma$ and $\|\mathbf{e}_i\| \le b'_i/\gamma$. 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 awkward.

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 $b_i \le \beta/\sqrt{30}$. 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 $8b^*/\sqrt{b_*} \le \beta$.

The radix for breaking integers into digits is set to $\phi \in \mathbb{Z}$, taken as large as possible subject to $\sqrt{30}b_*/\phi + 8 \ge 8\phi$, specifically we use $\phi := \lfloor \sqrt{b_* \cdot 30/64} \rfloor$. Denoting $\gamma_1 := \sqrt{337/30} \le 3.36$ 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.

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

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 $\|\mathbf{u}_i\| \le \hat{b}_i$, $\|\mathbf{v}_i^{lo}\| \le \hat{b}^{lo}$, and $\|\mathbf{v}_i^{hi}\| \le \hat{b}_i^{hi}$.
   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 $b_i \le \beta/\sqrt{30}$, and for vectors in L we have $8b^*/\sqrt{b_*} \le \beta$.*
– *For all i, $b_i \le P/45k_i$ and $b'_i \le P/45d_i$.*

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 $\|\mathbf{s}_i\| \le b_i$ and $\|\mathbf{e}_i\| \le b'_i$. The size gap for both the $\mathbf{s}_i$'s and $\mathbf{e}_i$'s is $\gamma := \sqrt{337/30} \cdot (1 + \frac{16\phi}{b^*}) \le 3.36(1 + \frac{20}{\sqrt{b^*}})$.

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 $\hat{b}_i, \hat{b}_i^{hi}, \hat{b}^{lo}$ that are used in the exact-smallness proofs do not exceed $\sqrt{P}/\big(\sqrt{2} \cdot 2500 \cdot 260\sqrt{m_s + 2m_l}\big) = \beta$. 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 $\phi = \lfloor \sqrt{b_* \cdot 30/64} \rfloor$ to get $\hat{b}_i^{hi} \ge \hat{b}^{lo}$, and since $b'_i \le b^*$ we get:

$$\begin{aligned}\hat{b}^{lo} \le \hat{b}_i^{hi} &\le (\sqrt{30}\, b'_i/\phi + 8)/\gamma_2 \le \frac{(\sqrt{30}\, b^* + 8\phi)/\phi}{(\sqrt{30}\, b_* + 16\phi)/(\sqrt{30}\, b_*)} \\ &= \frac{\sqrt{30}\, b^* + 8\phi}{\sqrt{30}\, b_* + 16\phi} \cdot \frac{\sqrt{30}\, b_*}{\lfloor \sqrt{b_* \cdot 30/64} \rfloor} \\ &\le (b^*/b_*) \cdot 8\sqrt{b_*} \;=\; 8b^*/\sqrt{b_*} \;\le\; \sqrt{P}/(\sqrt{2} \cdot 2500 \cdot 260 \cdot \sqrt{m_s + 2m_l}).\end{aligned}$$

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 $\mathbf{b}_i = \mathbf{s}_i A_i + \mathbf{e}_i + \mathbf{x}_i$

(mod 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 $(\phi\mathbf{v}^{hi} + \mathbf{v}^{lo}) = \mathbf{e}_i R'_i$ (mod 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 $\phi_i = \lfloor\sqrt{b'_i \cdot 30/64}\rfloor$ 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

In the long version [25] we describe how to put the techniques from the previous sections together into a PVSS scheme. We implemented the components 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	Keygen	Encrypt	Decrypt	Prove		Verify		RAM
parties	time(sec)	time(sec)	time(ms)	# exp	time(sec)	# exp	time(sec)	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 insignificant. Implementing this optimization could yield an almost 2× speedup for the prover and about 1.5× speedup for

	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.

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. Achlioptas, D.: Database-friendly random projections: Johnson-lindenstrauss with binary coins. Journal of Computer and System Sciences **66**(4), 671–687 (2003), `https://doi.org/10.1016/S0022-0000(03)00025-4`, special Issue on PODS 2001
2. Agrawal, S., Stehlé, D., Yadav, A.: Towards practical and round-optimal lattice-based threshold and blind signatures. IACR Cryptol. ePrint Arch. **2021**, 381 (2021), `https://eprint.iacr.org/2021/381`
3. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Journal of Mathematical Cryptology **9**, 169–203 (October 2015). https://doi.org/10.1515/jmc-2015-0016, `https://bitbucket.org/malb/lwe-estimator/src/master/`
4. Bai, S., Lepoint, T., Roux-Langlois, A., Sakzad, A., Stehlé, D., Steinfeld, R.: 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). https://doi.org/10.1007/s00145-017-9265-9
5. Baum, C., Damgård, I., Larsen, K.G., Nielsen, M.: How to prove knowledge of small secrets. In: Annual International Cryptology Conference. pp. 478–498. Springer (2016)
6. Baum, C., Lyubashevsky, V.: Simple amortized proofs of shortness for linear relations over polynomial rings. IACR Cryptol. ePrint Arch. p. 759 (2017)
7. Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a public blockchain keep a secret? In: TCC (2020), https://eprint.iacr.org/2020/464
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: International conference on the theory and applications of cryptographic techniques. pp. 416–432. Springer (2003)
9. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: International conference on the theory and application of cryptology and information security. pp. 514–532. Springer (2001)
10. Bootle, J., Chiesa, A., Sotiraki, K.: Sumcheck arguments and their applications. IACR Cryptol. ePrint Arch. **2021**, 333 (2021)
11. Boudot, F., Traoré, J.: Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In: International Conference on Information and Communications Security. pp. 87–102. Springer (1999)

12. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: Theory of Cryptography Conference. pp. 407–437. Springer (2019)
13. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: 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. pp. 315–334. IEEE Computer Society (2018), https://doi.org/10.1109/SP.2018.00020
14. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Annual International Cryptology Conference. pp. 126–144. Springer (2003)
15. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press (May 2002). https://doi.org/10.1145/509907.509980
16. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: 26th Annual Symposium on Foundations of Computer Science (sfcs 1985). pp. 383–395. IEEE (1985)
17. Costa, N., Martínez, R., Morillo, P.: Proof of a shuffle for lattice-based cryptography. In: Nordic Conference on Secure IT Systems. pp. 280–296. Springer (2017)
18. Del Pino, R., Lyubashevsky, V.: Amortization with fewer equations for proving knowledge of small secrets. In: Annual International Cryptology Conference. pp. 365–394. Springer (2017)
19. D'Souza, R., Jao, D., Mironov, I., Pandey, O.: Publicly verifiable secret sharing for cloud-based key management. In: International Conference on Cryptology in India. pp. 290–309. Springer (2011)
20. Fouque, P.A., Stern, J.: One round threshold discrete-log key generation without private channels. In: International Workshop on Public Key Cryptography. pp. 300–316. Springer (2001)
21. Fuchsbauer, G.: Commuting signatures and verifiable encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 224–245. Springer (2011)
22. Fujisaki, E., Okamoto, T.: 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. pp. 32–46. Springer (1998)
23. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ecdsa with fast trustless setup. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1179–1194 (2018)
24. Gentry, C., Halevi, S.: Compressible fhe with applications to pir. In: Theory of Cryptography Conference. pp. 438–464. Springer (2019)
25. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. https://eprint.iacr.org/2021/1397 (2021)
26. Gentry, C., Halevi, S., Magri, B., Nielsen, J.B., Yakoubov, S.: Random-index PIR with applications to large-scale secure MPC. https://eprint.iacr.org/2020/1248 (2020)
27. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer (2013), https://doi.org/10.1007/978-3-642-40041-4_5

28. Goldreich, O., Micali, S., Wigderson, A.: 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. Groth, J.: On the size of pairing-based non-interactive arguments. In: Annual international conference on the theory and applications of cryptographic techniques. pp. 305–326. Springer (2016)
30. Groth, J.: Applied crypto: Introducing noninteractive distributed key generation (2021), `https://medium.com/dfinity/applied-crypto-one-public-key-for-the-internet-computer-ni-dkg-4af800db869d`
31. Groth, J.: Non-interactive distributed key generation and key resharing. Cryptology ePrint Archive, Report 2021/339 (2021), `https://eprint.iacr.org/2021/339`
32. Heidarvand, S., Villar, J.L.: Public verifiability from pairings in secret sharing schemes. In: International Workshop on Selected Areas in Cryptography. pp. 294–308. Springer (2008)
33. Jhanwar, M.P., Venkateswarlu, A., Safavi-Naini, R.: Paillier-based publicly verifiable (non-interactive) secret sharing. Designs, codes and Cryptography **73**(2), 529–546 (2014)
34. Johnson, W.B., Lindenstrauss, J.: Extensions of lipschitz mappings into a hilbert space 26. Contemporary mathematics **26** (1984)
35. Lee, J.: Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. IACR Cryptol. ePrint Arch. **2020**, 1274 (2020)
36. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: 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. pp. 101–131. Springer (2016)
37. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: ACM CCS 18. pp. 1837–1854. ACM Press (2018). https://doi.org/10.1145/3243734.3243788
38. Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: International workshop on public key cryptography. pp. 162–179. Springer (2008)
39. Lyubashevsky, V.: Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (Dec 2009), `https://doi.org/10.1007/978-3-642-10366-7_35`
40. Lyubashevsky, V.: Basic lattice cryptography: Encryption and Fiat-Shamir signatures. `https://www.tinyurl.com/latticesurvey`, accessed Apr-2021 (2020)
41. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Practical lattice-based zero-knowledge proofs for integer relations. In: CCS. pp. 1051–1070. ACM (2020)
42. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Shorter lattice-based zero-knowledge proofs via one-time commitments. In: Garay, J.A. (ed.) Public-Key Cryptography - PKC 2021, Part I. Lecture Notes in Computer Science, vol. 12710, pp. 215–241. Springer (2021), `https://doi.org/10.1007/978-3-030-75245-3_9`
43. Melchor, C.A., Barrier, J., Fousse, L., Killijian, M.O.: Xpir: Private information retrieval for everyone. Proceedings on Privacy Enhancing Technologies **2016**, 155–174 (2016)
44. Olumofin, F., Goldberg, I.: Revisiting the computational practicality of private information retrieval. In: International Conference on Financial Cryptography and Data Security. pp. 158–172. Springer (2011)

45. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)
46. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE (2013)
47. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D.A. (ed.) Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings. Lecture Notes in Computer Science, vol. 5157, pp. 554–571. Springer (2008), https://doi.org/10.1007/978-3-540-85174-5_31
48. Rambaud, M., Urban, A.: Almost-asynchronous mpc under honest majority, revisited. IACR Cryptol. ePrint Arch. **2021**, 503 (2021)
49. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6), 34:1–34:40 (2009), http://doi.acm.org/10.1145/1568318.1568324
50. Reyzin, L., Smith, A., Yakoubov, S.: Turning hate into love: Compact homomorphic ad hoc threshold encryption for scalable mpc. In: International Symposium on Cyber Security Cryptography and Machine Learning. pp. 361–378. Springer (2021)
51. Ruiz, A., Villar, J.L.: Publicly verifiable secret sharing from paillier's cryptosystem. In: WEWoRC 2005–Western European Workshop on Research in Cryptology. Gesellschaft für Informatik eV (2005)
52. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Annual International Cryptology Conference. pp. 148–164. Springer (1999)
53. Sion, R., Carbunar, B.: On the computational practicality of private information retrieval. In: Proceedings of the Network and Distributed Systems Security Symposium. pp. 2006–06. Internet Society (2007)
54. Stadler, M.: 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. Lecture Notes in Computer Science, vol. 1070, pp. 190–199. Springer (1996), https://doi.org/10.1007/3-540-68339-9_17
55. Wu, T.Y., Tseng, Y.M.: A pairing-based publicly verifiable secret sharing scheme. Journal of Systems Science and Complexity **24**(1), 186–194 (2011)
56. Young, A., Yung, M.: A pvss as hard as discrete log and shareholder separability. In: International Workshop on Public Key Cryptography. pp. 287–299. Springer (2001)