Vector and Functional Commitments from Lattices

Chris Peikert[1,2], Zachary Pepin[1], and Chad Sharp[1]

[1] University of Michigan
[2] Algorand, Inc.

Abstract. Vector commitment (VC) schemes allow one to commit concisely to an ordered sequence of values, so that the values at desired positions can later be proved concisely. In addition, a VC can be statelessly updatable, meaning that commitments and proofs can be updated to reflect changes to individual entries, using knowledge of just those changes (and not the entire vector). VCs have found important applications in verifiable outsourced databases, cryptographic accumulators, and cryptocurrencies. However, to date there have been relatively few post-quantum constructions, i.e., ones that are plausibly secure against quantum attacks.

More generally, functional commitment (FC) schemes allow one to concisely and verifiably reveal various functions of committed data, such as linear functions (i.e., inner products, including evaluations of a committed polynomial). Under falsifiable assumptions, all known functional commitments schemes have been limited to "linearizable" functions, and there are no known post-quantum FC schemes beyond ordinary VCs.

In this work we give post-quantum constructions of vector and functional commitments based on the standard Short Integer Solution lattice problem (appropriately parameterized):

- First, we present new statelessly updatable VCs with significantly shorter proofs than (and efficiency otherwise similar to) the only prior post-quantum, statelessly updatable construction (Papamanthou *et al.*, EUROCRYPT 13). Our constructions use private-key setup, in which an authority generates public parameters and then goes offline.
- Second, we construct functional commitments for *arbitrary (bounded) Boolean circuits* and branching programs. Under falsifiable assumptions, this is the first post-quantum FC scheme beyond ordinary VCs, and the first FC scheme of any kind that goes beyond linearizable functions. Our construction works in a new model involving an authority that generates the public parameters and remains online to provide public, reusable "opening keys" for desired functions of committed messages.

1 Introduction

Commitment schemes are an essential cryptographic primitive. They provide the cryptographic equivalent of a "locked box," allowing one to publicly lock

some desired value away and to later reveal it. By analogy, two central desiderata of such schemes are as follows. First, it should be *binding*: once a (possibly adversarially generated) commitment is published, there should be no way to open it to two different values. Second, it may also be *hiding*: no one should be able to see inside the box. That is, the commitment reveals essentially nothing about the underlying value.

First constructed and formalized in work by Libert and Yung [LY10] and by Catalano and Fiore [CF13], *vector commitment* (VC) schemes generalize commitments to ordered sequences of values. More specifically, one can commit to a d-dimensional vector $\mathbf{m}$ and later open the commitment at any desired indices, i.e., prove that the ith entry of $\mathbf{m}$ is m_i. Here the notion of binding is replaced with *position binding*: it should be infeasible to open a commitment at a position i as two different message entries $m_i \neq m'_i$. For hiding, we may require that the commitment and openings reveal nothing about the *unopened* message entries. (However, many applications of VCs turn out not to need hiding.) In order to rule out trivial implementations, commitments and proofs are required to be *concise*, meaning that they should be smaller than the entire message vector, i.e., sublinear in d (and the smaller the better).

Additionally, VCs often need to be *updatable*, meaning it is possible to update commitments and proofs to reflect changes in the underlying vector entries, faster than the trivial solution of just computing new commitments and proofs from scratch. Updatability can even be *stateless* (also known as *distributed*) [CPSZ18], meaning that updates require only the position at which the message vector changed, and the old and new entries (or even just their difference).

Libert, Ramanna, and Yung [LRY16] generalized the concept of VCs to the notion of *functional commitments* (FCs), with a focus on *linear* functions. For some particular class of functions $\mathcal{F}$, FCs allow one to commit to a vector $\mathbf{m}$ and then, for any desired functions $f \in \mathcal{F}$, open the function-value pairs $(f, y = f(\mathbf{m}))$. (Traditional VCs can be seen as FCs for the class $\mathcal{F} = \{f_i(\mathbf{m}) = m_i\}_{i \in [d]}$ of "coordinate projection" functions.) A special case of linear FCs, defined earlier by Kate, Zaverucha, and Goldberg [KZG10] considers committing to a *polynomial* and then opening its evaluations at desired points.

Applications. Vector commitments have found numerous cryptographic applications. Catalano and Fiore [CF13] demonstrated their usefulness for publicly verifiable databases with efficient updates and, more broadly, verifiable outsourcing of storage [BGV11], updatable zero-knowledge sets and databases [MRK03, Lis05], cryptographic accumulators [BdM93], and pseudonymous credentials [KZG10]. More recently, Chepurnoy *et al.* [CPSZ18] used statelessly updatable VCs to construct an architecture for cryptocurrencies with stateless transaction validation.

Beyond VCs, functional commitments for polynomials [KZG10] and, more generally, linear functions [LRY16] have additionally found applications in verifiable secret sharing [CGMA85], content extraction signatures [SBZ01], proof-carrying data systems, and zero-knowledge SNARKs [BFS20, BDFG20].

Constructions. Merkle trees [Mer87] provide the first instantiation of VCs, with $O(1)$ commitment size and $O(\log d)$ proof size (both as functions solely of the dimension d), but they are not statelessly updatable. Libert and Yung [LY10] gave a construction that can serve as a statelessly updatable VC (and more), based on a "q-type" pairing assumption where q is the vector dimension, with $O(1)$-sized commitments and openings. Catalano and Fiore [CF13] gave two constructions of statelessly updatable VCs, based respectively on the Computational Diffie-Hellman (CDH) assumption over pairing-friendly groups and the RSA assumption, where each commitment and opening is a single group element. Chepurnoy *et al.* [CPSZ18] gave a construction based on the q-Strong Bilinear Diffie-Hellman assumption, which has smaller public parameters than the aforementioned CDH-based scheme (linear in d rather than quadratic) but slightly slower proof updates, and improved time complexity for proof updates versus the aforementioned RSA-based scheme ($O(\log d \log\log d)$ versus $O(d)$).

The work of [KZG10] gave a polynomial commitment scheme where commitments and proofs are each a single group element, and which is secure under the q-Strong Diffie-Hellman assumption. More generally, the work of [LRY16] gave an FC scheme for *linear* functions $f_{\mathbf{w}}(\mathbf{m}) = \langle \mathbf{w}, \mathbf{m} \rangle$, based on a subgroup decision assumption on pairing-friendly composite-order groups, in which commitments and proofs are each a single group element. Recently, the work of [LP20] gave a functional commitment scheme for what the authors called the class of "semi-sparse" polynomials. This class is an example of a *linearizable* function class, i.e., every function f in the class can be (efficiently) decomposed as $f(\mathbf{m}) = L_f(P(\mathbf{m}))$ for some polynomial-time "preprocessing" function P (which depends only on the class, and not the specific function f) and a linear function L_f (which may depend on f). For any linearizable class, an FC can be generically implemented via a linear FC (though perhaps not as efficiently as with a specialized construction), simply by having the committer commit to $P(\mathbf{m})$.

For *non-linearizable* function classes, we are not aware of *any* FC construction, apart from a straightforward generic construction mentioned in [LRY16]: it combines any succinct commitment scheme with any succinct noninteractive argument of knowledge for NP (such as PCP-based ones [Kil92, Mic94] or more specialized ones like Bulletproofs [BBB+18]), for proving functions of the committed data. However, the latter component cannot be based on a *falsifiable* assumption via a black-box security reduction [GW11]; indeed, existing constructions like the ones cited above tend to rely on strong heuristics like the random oracle model. In summary, we do not have any FC for a non-linearizable function class based on a falsifiable assumption.

There has also been quite limited work on *post-quantum* vector commitment schemes, i.e., ones which are plausibly secure against quantum attacks. Merkle trees instantiated with a post-quantum hash function may be used, though they suffer from relatively inefficient and necessarily stateful updates. Papamanthou *et al.* [PSTY13] gave a Merkle-tree-like construction based on the Short Integer Solution (SIS) lattice problem, which straightforwardly yields a statelessly updatable VC scheme. At present, we are unaware of any post-quantum FC schemes

beyond the aforementioned VC schemes and generic FC construction (requiring non-falsifiable assumptions); in particular, even constructing a post-quantum *linear* FC from a falsifiable assumption is an open problem.

1.1 Our Contributions

We present two main sets of results. First, in Section 3 we give new constructions of vector commitments based on the (post-quantum) SIS lattice problem. The first of these is a "base" VC construction that is statelessly updatable; it is most appropriate for only moderately large d, due to the public parameters' quadratic dependence on d. Then, for larger dimensions d^h, we give a specialized tree transformation of our SIS-based VC that preserves stateless updates (unlike generic Merkle trees). This transformation uses a main idea from [PSTY13], but our construction's proofs are significantly more concise, by a d factor, because the transformation is based on a VC rather than a hash function. For a detailed comparison with the prior work on VCs see Figures 1 and 2 and the associated discussion in Section 1.2, and for an overview of the constructions see Section 1.3.

Our second main contribution, given in Section 4, is a functional commitment scheme for *arbitrary (bounded) Boolean circuits* and branching programs, also based on SIS (appropriately parameterized). Under falsifiable assumptions, this is the first functional commitment scheme that goes beyond linearizable functions, and is also the first post-quantum construction of functional commitments beyond vector commitments, e.g., for linear functions. Indeed, we specialize our general construction to linear functions over large finite fields, resulting in a relatively simple and potentially practical scheme.

Our functional commitment construction works in a model, which we introduce in this work, involving a trusted authority that sets up the public parameters and *remains online* to provide "opening keys" ok_f for any desired functions f of committed messages. We stress that, unlike superficially similar models (e.g., for identity- or attribute-based encryption), these opening keys are not tied to any particular party and do not need to be transmitted via a secret channel; they can be announced publicly and used by all committers any number of times. See Section 1.3 for an overview of the construction.

As an additional contribution, in the full version we give a formal definition and analysis of a generic tree transformation on VCs, which converts a VC for d-dimensional vectors into one for d^h-dimensional vectors for any desired positive integer h. The transformation is like a Merkle tree of height h and arity d, using the underlying VC rather than a hash function to commit to each node's children. The key advantage is that to open an entry, one only needs to open each step of the entry's root-to-leaf path, but *no "sibling" information is required.* This immediately saves about a factor of d in the proof size, thus allowing for the use of a larger arity d and hence smaller height h, which reduces proof size even further. This main idea has previously been used in other contexts like signatures [DN94] and zero-knowledge databases [CRFM08], and even VCs [Kus18] (where it is called a "Verkle tree"), though without a formal analysis or treatment of updates.

1.2 Comparisons to Related Work

For the purposes of comparison, we divide existing VC schemes into "primitive" and "tree-based" schemes. Primitive schemes operate directly on the input vector and are typically based on some concrete cryptographic assumption (RSA, CDH, SIS, etc.). A tree-based scheme transforms another (usually primitive) VC scheme, and treats its input vector as a tuple of subvectors, somehow recursively committing to each subvector and then committing to the tuple of results using the underlying scheme. Tree-based VCs are most suitable for vectors of large dimension, and generally sacrifice proof and/or commitment size for smaller public parameters.

Primitive schemes. Here and in Figure 1 we briefly compare our base SIS vector commitment scheme to other primitive VCs from the literature [CF13, CPSZ18]. The primary advantage of our scheme is that it is plausibly secure against quantum attacks, whereas the others are broken by them. However, this comes at the cost of commitment and proof sizes that are *logarithmic*, rather than *constant*, in the vector dimension d. Our scheme, along with the others in question, has stateless updates but requires private-coin setup, in contrast to the only other known post-quantum VC schemes (discussed below).

Scheme	$\lvert vp\rvert$	$\lvert cp\rvert$	$\lvert c\rvert$	$\lvert\pi\rvert$	Setup	Stateless	PQ
[CF13] (RSA)	d	d	1	1	Private	✓	✗
[CF13] (CDH)	d	d^2	1	1	Private	✓	✗
[CPSZ18]	d	d	1	1	Private	✓	✗
Construction 1	d	d^2	$\log d$	$\log d$	Private	✓	✓

Fig. 1. A comparison of "primitive" VC schemes. Object sizes are expressed asymptotically as functions of the vector dimension d, with logarithmic factors elided from the sizes of verifier parameters vp and committer parameters cp (but not commitments c or proofs π). PQ indicates that the scheme is plausibly secure against quantum attacks.

Tree-based schemes. Here and in Figure 2 we compare tree-based schemes that commit to vectors of (typically large) dimension D, using a tree of some chosen arity d and height $h = \log_d D$. It is important to bear in mind that asymptotically different values of d and h are optimal for different schemes, so the object sizes as functions of these parameters cannot be compared directly across schemes.

Selecting optimal parameters for Merkle trees is straightforward: simply minimize the asymptotic proof size $dh = d \log_d D$, yielding optimal values $d = O(1)$ and $h = O(\log d)$. For the remaining schemes, choosing larger d (and thereby smaller $h = \log_d D$) reduces the commitment and proof sizes, at the cost of larger public parameters. Therefore, one should maximize d subject to some reasonable constraint on the sizes of the public parameters. For large dimensions D, a

plausible choice is $d = D^\epsilon$ for some small constant $\epsilon > 0$, yielding $h = 1/\epsilon = O(1)$; we use this setting for the following comparisons.

Like the scheme described in [PSTY13], our specialized tree transformation (Construction 2) has stateless updates. However, ours has smaller proofs at the cost of larger public parameters, both by a factor of d. This seems like an advantageous tradeoff in most applications, since many proofs (about many commitments) are given for a single set of public parameters.

In comparison with Merkle trees, the primary benefit of our specialized tree transform is that it has stateless updates. However, its proof sizes are slightly larger than those of Merkle trees ($O(\log^2 D)$ rather than $O(\log D)$), and its commitment sizes are logarithmic rather than constant.

The generic tree transformation for VCs (in the full version) instantiated with our SIS-based scheme (Construction 1) has constant-factor improvements, by h or h^2, in commitment size and proof size compared to our specialized scheme and that of [PSTY13], with an additional factor-of-d improvement in proof size over [PSTY13]. Unlike those other works, however, it requires stateful updates.

Though not post-quantum, instantiating the generic tree transformation with the CDH-based scheme of [CF13] (which has constant-sized commitments and proofs) demonstrates more clearly the structural advantage it has over Merkle trees. Since sibling information is not necessary, the proof size depends only on $h = O(1)$. However, this comes at the cost of private setup and larger public parameters.

Scheme	$\lvert vp\rvert$	$\lvert cp\rvert$	$\lvert c\rvert$	$\lvert\pi\rvert$	Setup	Stateless	PQ
Merkle tree	1	1	1	hd	Public	✗	✓
[PSTY13]	$h^2 d$	$h^2 d$	$h \log d$	$h^3 d \log^2 d$	Public	✓	✓
Construction 2	$h^2 d$	$h^2 d^2$	$h \log d$	$h^3 \log^2 d$	Private	✓	✓
Generic tree with 1 (SIS)	d	d^2	$\log d$	$h \log d$	Private	✗	✓
Generic tree with [CF13] (CDH)	d	d^2	1	h	Private	✗	✗

Fig. 2. A comparison of "tree-based" VC schemes. Object sizes are expressed as functions of the tree arity d and height h (handling a vector dimension $D = d^h$), with logarithmic factors elided from the sizes of the verifier parameters vp and committer parameters cp (but not the commitments c or proofs π). PQ indicates that the scheme is plausibly secure against quantum attacks (for Merkle trees, when using a PQ hash function).

Other related work. The work of [BGJS16] considered a variant of functional commitment (potentially with a bounded number of function queries), which does not require succinctness but also does not need any trusted setup, and showed that it is implied by verifiable functional encryption (and thereby from assumptions on pairing-friendly groups).

The work of [HW15] considers a primitive called "somewhat statistically binding" (SSB) hash, which is a strengthening of vector commitments: the

commitment and openings of individual entries are concise, and additionally, the commitment is *statistically* binding at one (hidden) location. However, the construction relies on Merkle trees plus (comparatively heavy) fully homomorphic encryption, so it has no better efficiency than ordinary Merkle trees, nor is it statelessly updatable.

1.3 Technical Overview

In this section we give a brief overview of the key ideas underlying our constructions; we assume some basic familiarity with the abstract functionality of "trapdoors" for lattices [GPV08] and "third-generation" fully homomorphic encryption/commitments [GSW13, GVW15].

SIS-Based Vector Commitments Our base VC scheme is conceptually inspired by the ones proposed in [CF13]. To convey the key ideas, we describe a technically simpler, unoptimized version of our scheme for vectors whose entries belong to $\mathcal{M} = \{0,1\}^\ell$ for some desired ℓ. To generate the public parameters, we first choose a uniformly random matrix $\mathbf{U} = [\mathbf{U}_0 \mid \cdots \mid \mathbf{U}_{d-1}] \in \mathbb{Z}_q^{n \times \ell d}$, where each $\mathbf{U}_i \in \mathbb{Z}_q^{n \times \ell}$. Then we generate d (statistically close to) uniformly random matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ along with respective trapdoors $\mathbf{T}_i$. We use each trapdoor $\mathbf{T}_i$ to sample a "short" (discrete Gaussian-distributed) $\mathbf{R}_i \in \mathbb{Z}^{m \times \ell(d-1)}$ such that $\mathbf{A}_i \mathbf{R}_i = \mathbf{U}_{-i}$, where $\mathbf{U}_{-i} \in \mathbb{Z}_q^{n \times \ell(d-1)}$ is $\mathbf{U}$ with its ith block $\mathbf{U}_i$ removed. The public parameters are $\mathbf{U}$ and all the $\mathbf{A}_i$ and $\mathbf{R}_i$ matrices.

The commitment to a vector $\mathbf{m} \in \{0,1\}^{\ell d}$ is simply $\mathbf{c} = \mathbf{U}\mathbf{m} \in \mathbb{Z}_q^n$.[3] To open the commitment at position i as $\mathbf{m}_i \in \{0,1\}^\ell$, output the proof $\mathbf{p}_i = \mathbf{R}_i \mathbf{m}_{-i} \in \mathbb{Z}^m$ (which is short), where $\mathbf{m}_{-i} \in \{0,1\}^{\ell(d-1)}$ is $\mathbf{m}$ with its ith ℓ-bit block removed. The verifier simply checks that $\mathbf{p}_i$ is sufficiently short and that $\mathbf{c} = \mathbf{A}_i \mathbf{p}_i + \mathbf{U}_i \mathbf{m}_i$, which holds since

$$\mathbf{A}_i \mathbf{p}_i + \mathbf{U}_i \mathbf{m}_i = \mathbf{A}_i \mathbf{R}_i \mathbf{m}_{-i} + \mathbf{U}_i \mathbf{m}_i = \mathbf{U}_{-i} \mathbf{m}_{-i} + \mathbf{U}_i \mathbf{m}_i = \mathbf{U}\mathbf{m} = \mathbf{c}.$$

Breaking position binding of this scheme means producing a tuple $(\mathbf{c}^*, i, \mathbf{m}_i, \mathbf{m}'_i, \mathbf{p}, \mathbf{p}')$ such that for commitment $\mathbf{c}^*$ and position i, proofs $\mathbf{p}, \mathbf{p}'$ respectively verify for distinct $\mathbf{m}_i \neq \mathbf{m}'_i$, i.e., $\mathbf{p}, \mathbf{p}'$ are sufficiently short and

$$\mathbf{c}^* = \mathbf{A}_i \mathbf{p} + \mathbf{U}_i \mathbf{m}_i = \mathbf{A}_i \mathbf{p}' + \mathbf{U}_i \mathbf{m}'_i.$$

From this, we have that $\mathbf{x} := \begin{bmatrix} \mathbf{p}_i - \mathbf{p}'_i \\ \mathbf{m}_i - \mathbf{m}'_i \end{bmatrix} \neq \mathbf{0}$ is an SIS solution to the (statistically close to) uniformly random matrix $[\mathbf{A}_i \mid \mathbf{U}_i]$. Our security reduction shows how to embed an external SIS instance as this matrix, and generate all the rest of the public parameters to have the proper joint distribution (even though the reduction does not have a trapdoor for $\mathbf{A}_i$).

In our actual construction, we use the "trapdoor puncturing" technique of [PW08, ABB10, MP12] to reduce the size of the public parameters, allowing

[3] In this overview we aim only for position binding, and dispense with hiding.

us to generate each $\mathbf{A}_i$ from a single uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. The (stateless) updatability of this scheme follows from the linearity of commitments and openings.

Specialized tree transformation. In the full version, we construct a generic tree transformation for VC schemes for vectors of high dimension d^h, which uses any VC for dimension d as a black box. (Its practical efficiency was analyzed in [Kus18], but without a formal security analysis or treatment of updates.) The main idea is very similar to that of Merkle trees [Mer87], but with the public hash function replaced by a vector commitment scheme. In particular, it can be instantiated with our SIS-based VC scheme, where commitments in $\mathbb{Z}_q^n$ are treated as messages by representing them as bit vectors. More formally, we can use the nonlinear "bit decomposition" transformation $\mathbf{G}^{-1} \colon \mathbb{Z}_q^n \to \mathbb{Z}^w$, where $w \approx n \log q$ and we take $\ell = w$ in the above construction, to bring commitments back into the message space (and this can be inverted by multiplying by the "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$). However, this nonlinearity makes the commitment function of the transformed scheme non-linear, and thus breaks stateless updatability.

To preserve linearity and thereby stateless updatability, we give a specialized tree transformation of our SIS-based VC scheme, which takes inspiration from the Merkle-tree-like construction of [PSTY13]. For simplicity, we briefly outline this transformation for small parameters $d = h = 2$ (though a significantly larger d yields better efficiency).

The public parameters are identical to those of the base scheme (for $d = 2$), and using them we additionally define a matrix

$$\mathbf{U}^{(2)} := \mathbf{U}(\mathbf{I}_2 \otimes \mathbf{G}^{-1}(\mathbf{U})) = [\mathbf{U}_0\mathbf{G}^{-1}(\mathbf{U}) \mid \mathbf{U}_1\mathbf{G}^{-1}(\mathbf{U})] \in \mathbb{Z}_q^{n \times \ell d^2}.$$

To commit to a vector $\overline{\mathbf{m}} = (\overline{\mathbf{m}}_{00}, \overline{\mathbf{m}}_{01}, \overline{\mathbf{m}}_{10}, \overline{\mathbf{m}}_{11}) \in \{0,1\}^{\ell d^h} = \{0,1\}^{4\ell}$, we compute $\overline{\mathbf{c}} = \mathbf{G}^{-1}(\mathbf{U}^{(2)})\overline{\mathbf{m}}$; observe that this is a linear function of $\overline{\mathbf{m}}$, which ultimately allows for stateless updates. Also notice that

$$\mathbf{G}\overline{\mathbf{c}} = \mathbf{U}^{(2)}\overline{\mathbf{m}} = \mathbf{U}(\mathbf{I}_2 \otimes \mathbf{G}^{-1}(\mathbf{U}))\overline{\mathbf{m}}$$

is a commitment (under the base scheme) to $\overline{\mathbf{m}}' := (\mathbf{I}_2 \otimes \mathbf{G}^{-1}(\mathbf{U}))\overline{\mathbf{m}} \in \mathbb{Z}^{\ell d}$, which is relatively short. Additionally, $\overline{\mathbf{m}}'$ itself can be seen essentially as a pair of commitments, as

$$\begin{aligned} \overline{\mathbf{m}}' &= (\mathbf{I}_2 \otimes \mathbf{G}^{-1}(\mathbf{U}))\overline{\mathbf{m}} \\ &= \begin{bmatrix} \mathbf{G}^{-1}(\mathbf{U}) & \\ & \mathbf{G}^{-1}(\mathbf{U}) \end{bmatrix} \begin{bmatrix} \overline{\mathbf{m}}_{00} \\ \overline{\mathbf{m}}_{01} \\ \overline{\mathbf{m}}_{10} \\ \overline{\mathbf{m}}_{11} \end{bmatrix} =: \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \end{bmatrix}, \end{aligned}$$

because $\mathbf{G}\mathbf{c}_i \in \mathbb{Z}_q^n$ is a commitment (under the base scheme) to $(\overline{\mathbf{m}}_{i0}, \overline{\mathbf{m}}_{i1})$. So, to prove a particular entry of the message $\overline{\mathbf{m}}$, say $\overline{\mathbf{m}}_{01}$, we simply provide $\mathbf{c}_0$ and a proof that it is the 0th entry of $\overline{\mathbf{m}}'$ (viewing $\mathbf{G}\overline{\mathbf{c}}$ as a commitment to $\overline{\mathbf{m}}'$),

along with a proof that $\overline{\mathbf{m}}_{01}$ is the 1st entry committed to by $\mathbf{G}\mathbf{c}_0$. Breaking position binding of this scheme requires breaking position binding of the base scheme somewhere along this path.

We stress that the above proof structure is more concise than for Merkle trees and [PSTY13], because the proofs need not contain any "sibling" information. Essentially, our construction opens a vector commitment at each level, just as in the generic VC transformation, but in a manner that preserves linearity and hence stateless updates. This also allows for the use of a larger tree arity d, which reduces the tree height and thereby the number of elements in proofs. However, preserving statelessness in this way does come at a cost: in the general version of the transformation, the norm of $\overline{\mathbf{m}}'$ can grow linearly with its dimension ℓd^h, so the SIS parameters of the underlying VC scheme must be increased to accommodate these larger messages. This introduces a poly-logarithmic dependence on d^h in the sizes of commitments and proofs. The SIS-based Merkle-like construction of [PSTY13] (when used as a VC) has exactly the same dependence, so our commitment sizes match, and our proof sizes are strictly better by a d factor. (See Figure 2.)

Functional Commitments for Arbitrary Functions In Section 4 we give an SIS-based construction of functional commitments for arbitrary Boolean functions of bounded size S, via an online authority that provides opening keys for desired functions. The construction relies heavily upon the fully homomorphic commitment scheme that was implicit in the homomorphic encryption scheme of Gentry, Sahai, and Waters [GSW13], and was made explicit by Gorbunov, Vaikuntanathan, and Wichs [GVW15]. We remark that while the latter work gives a commitment scheme in which one can commit to data and then later open any bounded function of it, this falls short of a true functional commitment scheme because the commitment is not *succinct*—its size is a substantial factor larger than the data itself. In our construction, the sizes of both the commitments and proofs depend only on the complexity of the supported functions, and not directly on the message size. More specifically, they grow poly-logarithmically in the size and polynomially in the depth of the functions; see Section 4.3 for details.

In our construction, the commitment function is essentially identical to the first stage (homomorphic evaluation) of the correlation-intractable hash functions of [CCH+19, PS19]. However, in those works' main application (noninteractive zero knowledge proofs), the resulting output is never "opened." In functional commitments, the result needs to be opened in several different ways, once for each function that is proved about the committed message.

We also mention that using just tagged-trapdoor techniques, which predate the above-referenced fully homomorphic schemes but provide only linear homomorphism, we specialize our functional commitment scheme to arbitrary linear functions over large finite fields. The resulting scheme is asymptotically quite efficient, especially when adapted to use rings, and potentially practical. In this

overview we just focus on the general scheme for arbitrary functions, and refer to Section 4.3 for the linear specialization.

Public parameters and commitment. The public parameters in our scheme are a uniformly random matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times Sw}$ (where again, $w \approx n \log q$ is the width of the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$) and a (statistically close to) uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, which is generated with a trapdoor $\mathbf{T}$ that serves as the authority's secret key. Using standard tagged-trapdoor techniques, for each size-S function f we implicitly define a public matrix $\mathbf{A}_f \in \mathbb{Z}_q^{n \times m}$ that is efficiently computable from $\mathbf{A}$ and f; the authority's trapdoor $\mathbf{T}$ allows it to sample short preimages with respect to any of these $\mathbf{A}_f$.

It is helpful to interpret the public parameter $\mathbf{C}$ as a homomorphic commitment to some as-yet-unspecified function(s) f, with respect to some corresponding public key. To commit to a message m, one simply computes $\mathbf{C}_m = \mathsf{Eval}(U_m, \mathbf{C})$, i.e., the homomorphic evaluation of the universal function $U_m(f) := U(f, m) = f(m)$ on the committed function f. Therefore, we can think of $\mathbf{C}_m$ as a homomorphic commitment to $f(m)$—but we again stress that f is still unspecified.[4] We stress that it is vital that the homomorphic evaluation can indeed be done using just $\mathbf{C}$ and m, and not the unspecified public key relative to which $\mathbf{C}$ is viewed as a commitment.

Opening. In order to prove for a commitment $\mathbf{C}_m$ that $f(m) = y$ for some desired f, one first needs an appropriate opening key from the authority. Such a key is simply some randomness that opens $\mathbf{C}$ as a homomorphic commitment to the desired function f, *with respect to public key* $\mathbf{A}_f$. That is, the authority uses its trapdoor $\mathbf{T}$ to sample a "short" integer matrix such that $\mathbf{C} = \mathsf{FHCom}_{\mathbf{A}_f}(f; \mathbf{R}_f)$. We point out that each opening key essentially equivocates $\mathbf{C}$ as committing to a different value—but this does not violate security in any way, because these openings are with respect to *different* $\mathbf{A}_f$. (Indeed, it is important that the authority publishes at most one opening key for each function, which can be ensured by standard techniques.)

With an opening key $\mathbf{R}_f$ in hand, one proves that $f(m) = y$ by "tracing" the evolution of the randomness through the homomorphic computation of $\mathbf{C}_m$ from $\mathbf{C}$. More specifically, a key feature of the homomorphic commitment scheme is that, given the function U_m and the "short" randomness $\mathbf{R}_f$ underlying the commitment $\mathbf{C} = \mathsf{FHCom}_{\mathbf{A}_f}(f; \mathbf{R}_f)$, one can efficiently compute "relatively short" randomness $\mathbf{R}_{f,m}$ underlying $\mathbf{C}_m$, i.e., for which

$$\mathbf{C}_m := \mathsf{Eval}(U_m, \mathbf{C}) = \mathsf{FHCom}_{\mathbf{A}_f}(y = f(m); \mathbf{R}_{f,m}).$$

To prove that $f(m) = y$, one just computes and reveals this $\mathbf{R}_{f,m}$. The verifier checks that it is sufficiently short and that $\mathbf{C}_m = \mathsf{FHCom}_{\mathbf{A}_f}(y; \mathbf{R}_{f,m})$.

For security, we prove that under the SIS assumption (appropriately parameterized), it is infeasible for an adversary to break *function binding*, i.e., to output

[4] As an optimization, for Boolean functions f the matrix $\mathbf{C}_m$ can be further compressed to just a single vector $\mathbf{c}_m \in \mathbb{Z}_q^n$; we omit the details in this overview.

a tuple $(\mathbf{C}^*, f^*, y, y', \mathbf{R}, \mathbf{R}')$ such that $y \neq y'$ and yet the verifier accepts on both $(\mathbf{C}^*, f^*, y, \mathbf{R})$ and $(\mathbf{C}^*, f^*, y', \mathbf{R}')$. We prove this for a *selective function attack*, where the adversary must announce the targeted function f^* before seeing the public parameters, but may produce the rest of its output $(\mathbf{C}^*, y, y', \mathbf{R}, \mathbf{R}')$ after getting arbitrary opening keys for functions f of its choice, *even including* $f = f^*$. It is well known that security against such selective attacks can be boosted to security against fully adaptive attacks using complexity leveraging, i.e., the reduction is loose by a factor of the size of the function class. In addition, there are lattice-trapdoor techniques for obtaining adaptive security in related settings (e.g., [CHKP10, Yam16]), which seem compatible with our construction techniques.

1.4 Open Problems and Future Work

Our work raises several interesting questions for further research. First, recall that all of our constructions require private-coin setup (i.e., a trusted authority that uses private randomness to generate the public parameters). An important question is whether there are lattice-based or other post-quantum vector commitments with public setup and having similar or better properties, including for updates. (Recall that post-quantum Merkle trees can have public setup, but require stateful updates.)

Recall that our functional commitment scheme requires an *online* authority to generate opening keys for the desired functions, using some secret trapdoor.[5] A very interesting question is whether functional commitments for some large non-linear function class can be obtained using an *offline* authority, which only generates and publishes some setup parameters, based on a falsifiable assumption.[6] Even more ambitiously, can such a scheme be constructed with just *public*-coin setup?

Finally, the literature contains several variants of vector commitments. For example, *subvector* commitments [LM19, CFG+20] allow one to open a committed vector at any subset of positions, via a proof that is smaller than proofs for all the positions individually. Subvector commitments can even be *aggregatable* [TAB+20], meaning that openings for two different subsets can be aggregated, producing an opening for their union. So far, subvector commitments have been constructed only from assumptions on pairing-friendly groups. It is a very interesting question whether any kind of subvector commitments (with or without aggregation) can be constructed from lattices or other post-quantum assumptions.

Acknowledgments. We thank the anonymous TCC reviewers for many helpful comments and suggestions. This material is based upon work supported by

[5] However, if the class has only polynomially many functions, then the authority can publish all the opening keys and then go offline (disappear).

[6] Recall that the naive construction proposed in [LRY16], which combines a succinct commitment scheme with a succinct noninteractive argument for NP, cannot have a black-box security proof from a falsifiable assumption [GW11].

DARPA under Agreement No. HR00112020025. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

2 Preliminaries

For any non-negative integer i, denote $[i] = \{0, \ldots, i-1\}$ (where $[0] = \emptyset$). For a real vector $\mathbf{v}$, let $\|\mathbf{v}\| := \left(\sum_i v_i^2\right)^{1/2}$ denote its Euclidean norm and $\|\mathbf{v}\|_1 = \sum_i |v_i|$ denote its ℓ_1 norm. For a real matrix $\mathbf{V}$, let $\|\mathbf{V}\| := \max_j \|\mathbf{v}_j\|$ denote the maximum Euclidean norm of its column vectors $\mathbf{v}_j$, and let $s_1(\mathbf{V}) := \max_{\mathbf{u} \neq \mathbf{0}} \|\mathbf{V}\mathbf{u}\| / \|\mathbf{u}\|$ denote its maximum singular value (also known as its spectral norm). Observe that for any matrix $\mathbf{V}$ and vector $\mathbf{u}$, we have $\|\mathbf{V}\mathbf{u}\| \leq \|\mathbf{V}\| \cdot \|\mathbf{u}\|_1$ by the triangle inequality.

2.1 Vector Commitments

Definition 1 (Vector commitment). *A* vector commitment scheme *with message space $\mathcal{M}$, commitment space $\mathcal{C}$, and proof space $\mathcal{P}$ (which may be functions of the setup parameters) is a set of algorithms with the following interfaces:*

- $\mathsf{Setup}(1^\lambda, 1^d)$ *outputs (public)* committer parameters *cp and* verifier parameters *vp*.
- $\mathsf{Commit}(cp, \mathbf{m} \in \mathcal{M}^d)$ *outputs a* commitment $c \in \mathcal{C}$ *and some committer state st*.
- $\mathsf{Open}(cp, st, i \in [d])$ *outputs a* proof p_i *for the ith entry of the committed message associated to st*.
- $\mathsf{Verify}(vp, c \in \mathcal{C}, i \in [d], m \in \mathcal{M}, p \in \mathcal{P})$ *either accepts or rejects.*

These algorithms should satisfy the following correctness condition: for any $d = \text{poly}(\lambda)$, $\mathbf{m} \in \mathcal{M}^d$, and $i \in [d]$, and for $(cp, vp) \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, $(c, st) \leftarrow \mathsf{Commit}(cp, \mathbf{m})$, and $p_i \leftarrow \mathsf{Open}(cp, st, i)$, $\mathsf{Verify}(vp, c, i, m_i, p_i)$ accepts with probability $1 - \text{negl}(\lambda)$ (over all the randomness of the experiment).

Additionally, the scheme is updatable *if it has a set of algorithms with the following interfaces:*

- $\mathsf{PrepareUpdates}(cp, st, j \in [d], m'_j \in \mathcal{M})$ *outputs a* commitment update δ_c, *a* proof update δ_p, *and a* state update δ_s *for changing the jth entry of the committed message vector to m'_j*.
- $\mathsf{UpdateC}(vp, c \in \mathcal{C}, \delta_c)$ deterministically *outputs an updated commitment c'*.[7]
- $\mathsf{UpdateP}(vp, i \in [d], p_i \in \mathcal{P}, \delta_p)$ deterministically *outputs an updated proof p'_i*.
- $\mathsf{UpdateS}(cp, st, \delta_s)$ deterministically *outputs an updated committer state st'*.

[7] The determinism is without loss of generality, because $\mathsf{PrepareUpdates}$ can include any needed random coins in δ_c; the same also applies for $\mathsf{UpdateP}, \delta_p$ and $\mathsf{UpdateS}, \delta_s$.

Additionally, the scheme is statelessly updatable *if* $\mathsf{PrepareUpdates}$ *can be implemented via:*

- $\mathsf{PrepareUpdates}^{no\text{-}st}(cp, j \in [d], m_j \in \mathcal{M}, m'_j \in \mathcal{M})$*, which has the same outputs as* $\mathsf{PrepareUpdates}$*, and differs only in its inputs: it does not get the committer state* st*, and instead receives only the old and new* j*th entries* m_j, m'_j *of the message vector* $\mathbf{m}$*. Then* $\mathsf{PrepareUpdates}$ *can be written generically in terms of* $\mathsf{PrepareUpdates}^{no\text{-}st}$ *(assuming that* $\mathbf{m}$ *is part of* st*, which is without loss of generality).*

Moreover, the scheme is differentially updatable *if* $\mathsf{PrepareUpdates}^{no\text{-}st}$ *(and hence* $\mathsf{PrepareUpdates}$*) can be implemented via:*

- $\mathsf{PrepareUpdates}^{diff}(cp, j \in [d], \delta)$*, which has the same outputs as* $\mathsf{PrepareUpdates}^{no\text{-}st}$*, and differs only in its inputs: rather than receiving* m_j *and* m'_j *separately, it receives only the "difference"* $\delta = m'_j - m_j$*, where* $-$ *denotes some abstract operation on* $\mathcal{M}$ *(whose output may be more compact than its two inputs). Then* $\mathsf{PrepareUpdates}^{no\text{-}st}$ *can be written generically in terms of* $\mathsf{PrepareUpdates}^{diff}$*.*

These algorithms should satisfy the following correctness condition: for any $d = \mathrm{poly}(\lambda)$*, any* $(cp, vp) \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*, any* $\mathbf{m}, \mathbf{m}' \in \mathcal{M}^d$ *that differ in at most the* j*th coordinate, and any* $i \in [d]$*, the outputs of the following two experiments are statistically indistinguishable; if they are identically distributed, we say that the updatability is* perfect*:*

1. *Let* $(c, st) \leftarrow \mathsf{Commit}(cp, \mathbf{m})$*,* $p_i \leftarrow \mathsf{Open}(cp, st, i)$*,* $(\delta_c, \delta_p, \delta_s) \leftarrow \mathsf{PrepareUpdates}(cp, st, j, m'_j)$*,*[8] $c' \leftarrow \mathsf{UpdateC}(vp, c, \delta_c)$*,* $p'_i \leftarrow \mathsf{UpdateP}(vp, i, p_i, \delta_p)$*,* $st' \leftarrow \mathsf{UpdateS}(cp, st, \delta_s)$*. Output* (st', c', p'_i)*.*
2. *Let* $(c', st') \leftarrow \mathsf{Commit}(cp, \mathbf{m}')$*,* $p'_i \leftarrow \mathsf{Open}(cp, st', i)$*. Output* (st', c', p'_i)*.*

In words, the results of updating a commitment and proof to a new entry of the message vector should be essentially the same as generating a "fresh" commitment and proof on the updated message vector. (The state information is included in the results for compositionality, so that the same goes for polynomially many updates.)

Remark 1. We note that our vector commitment interface differs slightly from the one introduced in [CF13]. First, we split our public parameters into separate committer and verifier parameters, to highlight the different values needed by each role. Second, we break out $\mathsf{PrepareUpdates}$ from the algorithms that do the actual updating, to delineate what work can be performed by the committer rather than the verifier. However, any VC implementing our interface can trivially be converted to the interface from [CF13], simply by merging the public parameters, and merging the code in $\mathsf{PrepareUpdates}$ that generates δ_c and δ_p into $\mathsf{UpdateC}$ and $\mathsf{UpdateP}$, respectively.

[8] For statelessly updatable schemes, this step can be replaced by $\mathsf{PrepareUpdates}^{\text{no-st}}(cp, j, m_j, m'_j)$, and for differentially updatable ones, it can be replaced by $\mathsf{PrepareUpdates}^{\text{diff}}(cp, j, \delta = m'_j - m_j)$.

Position binding. We now recall the main security property of vector commitments, known as position binding. Essentially, it should be infeasible to output a (possibly malformed) commitment along with two valid openings for different message entries at a particular position.

Definition 2. *A vector commitment scheme VCS is* position binding *if, for every $d = d(\lambda) = \mathrm{poly}(\lambda)$ and every probabilistic polynomial-time adversary $\mathcal{A}$,*

$$\mathbf{Adv}^{pba}_{VCS}(\mathcal{A}) := \Pr[m \neq m' \text{ and } \mathsf{Verify}(vp, c^*, i, m, p) = \mathsf{Verify}(vp, c^*, i, m', p') = accept] = \mathrm{negl}(\lambda),$$

where the probability is over the choice of $(cp, vp) \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$, $(c^*, i, m, p, m', p') \leftarrow \mathcal{A}(1^\lambda, 1^d, cp, vp)$.

Position binding alone is a sufficient security property for many applications of vector commitments, and can be obtained entirely with deterministic algorithms, excepting Setup; indeed, our own constructions achieve this. Of course, a deterministic Commit algorithm cannot hide the message vector, at least not in the sense of indistinguishability.

2.2 Short Integer Solution and (Tagged) Trapdoors

We recall the Short Integer Solution (SIS), and its hardness based on worst-case lattice problems.

Definition 3. *The (homogeneous) $SIS_{n,q,m,\beta}$ problem is: given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero integral vector $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Az} = \mathbf{0} \pmod q$ and $\|\mathbf{z}\| \leq \beta$. The* normal form *of the problem is to find a non-zero integral vector $\mathbf{z} = (\mathbf{x} \in \mathbb{Z}^m, \mathbf{e} \in \mathbb{Z}^n)$ such that $\mathbf{Ax} = \mathbf{e} \pmod q$ and $\|\mathbf{z}\| \leq \beta$.*

When $q \geq \beta \cdot \tilde{O}(\sqrt{n})$ and m is polynomial in n and $\log q$, solving $\mathrm{SIS}_{n,q,m,\beta}$ (in either its homogeneous or normal form) is at least as hard as approximating certain worst-case lattice problems on n-dimensional lattices to within a $\beta \cdot \tilde{O}(\sqrt{n})$ factor [MR04, GPV08].

Gadget and trapdoors. Our constructions use standard techniques for SIS-based trapdoors and preimage sampling as developed in [GPV08, MP12]. These rely on a publicly known "gadget" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ for some w. The prototypical example is $\mathbf{G} = \mathbf{I}_n \otimes (1, 2, \ldots, 2^{\lceil \log_2 q \rceil - 1})$ where $w = n\lceil \log_2 q \rceil$, but any other suitable $\mathbf{G}$ supporting an efficient preimage-sampling algorithm can work just as well in our applications (possibly after adjusting parameters); see [MP12] for the precise requirements.

The basic gadget-based inversion operation is a deterministic function denoted $\mathbf{G}^{-1} \colon \mathbb{Z}_q^n \to \mathbb{Z}^w$, which for some small $g = g_{\mathbf{G}}$ satisfies $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{u}) = \mathbf{u}$ and $\|\mathbf{G}^{-1}(\mathbf{u})\| \leq g$ for all $\mathbf{u} \in \mathbb{Z}_q^n$.[9] (For example, $g_{\mathbf{G}} = 1$ for the prototypical

[9] We stress that $\mathbf{G}^{-1}$ is a *function*, not a matrix, and it does not necessarily satisfy $\mathbf{G}^{-1}(\mathbf{G} \cdot \mathbf{z}) = \mathbf{z}$ for $\mathbf{z} \in \mathbb{Z}^w$.

gadget $\mathbf{G}$ defined above.) We extend the definition of $\mathbf{G}^{-1}$ to matrices simply by applying it column-wise.

The more advanced inversion operation is the (randomized) preimage-sampling algorithm, whose properties are described below in Theorem 1. For this purpose the gadget matrix $\mathbf{G}$ comes with a small factor $\omega = \omega_{\mathbf{G}}$ that appears in the bounds associated with the sampling algorithm. (E.g., for the prototypical $\mathbf{G}$ given above, we can take ω to be any $\omega(\sqrt{\log n})$ function.)

Preimage sampling works for "tagged" trapdoors, which rely on an efficiently computable *invertible-differences* encoding from elements of some particular set F to $\mathbb{Z}_q^{n\times n}$. In such an encoding, for any distinct $f, f' \in F$, the difference $\mathbf{H}_{f'} - \mathbf{H}_f$ between their respective encodings $\mathbf{H}_{f'}, \mathbf{H}_f$ is invertible. A standard construction (see, e.g., [PW08, ABB10, MP12]) for prime q uses any (efficiently computable) injective map from F into the finite field $\mathbb{F}_{q^n}$, which is viewed as an n-dimensional vector space over $\mathbb{F}_q$ with some arbitrary basis. Then relative to that basis, multiplication by any fixed (nonzero and hence invertible) field element $f \in \mathbb{F}_{q^n}$ basis corresponds, via an additive homomorphism, to multiplication by an (invertible) matrix $\mathbf{H}_f \in \mathbb{Z}_q^{n\times n}$; this correspondence therefore yields an invertible-differences encoding. (This can be extended to arbitrary non-prime q in a natural way; see, e.g., [MP12].)

The following theorem summarizes the trapdoor functionality that our constructions will use; our presentation abstracts away the precise distributions sampled by the various algorithms. Typically $\mathsf{SampleD}$ samples from a discrete Gaussian distribution (as in [GPV08]), but the theorem holds equally well (but with somewhat looser bounds) for other distributions over the integers, like uniform over a sufficiently wide interval; see [LW15].

Theorem 1 ([GPV08, MP12]). *There are probabilistic polynomial-time algorithms* $\mathsf{TrapGen}$, $\mathsf{SampleD}$, $\mathsf{SamplePre}$ *and a "gadget" matrix* $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$ *having the following properties, where* $m = \bar{m} + w$*:*

1. $\mathsf{TrapGen}(\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}, \mathbf{H}^* \in \mathbb{Z}_q^{n\times n})$ *outputs some* $(\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T} \in \mathbb{Z}^{\bar{m}\times m})$ *such that* $s_1(\mathbf{T}) \le s_T = O(\sqrt{m})$ *and*
$$\mathbf{A} = \bar{\mathbf{A}}\mathbf{T} + [\mathbf{0} \mid \mathbf{H}^*\mathbf{G}] \in \mathbb{Z}_q^{n\times m}.$$
2. *For any* $\bar{m} \ge 2n\log q$ *and any* $\mathbf{H}^* \in \mathbb{Z}_q^{n\times n}$*, and for uniformly random* $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n\times\bar{m}}$ *and* $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}^*)$*, the distribution of* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *is within* $\mathrm{negl}(n)$ *statistical distance of uniform.*
3. *For any* $\mathbf{H}^*, \mathbf{H} \in \mathbb{Z}_q^{n\times n}$ *such that* $\mathbf{H}^* - \mathbf{H}$ *is invertible, any* $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}^*)$ *for any* $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times\bar{m}}$*, any* $s \ge s_T \cdot \omega$*, any positive integer* k*, and letting* $\mathbf{A}' = \mathbf{A} - [\mathbf{0} \mid \mathbf{H}\mathbf{G}]$*:*
   (a) *For* $\mathbf{R} \leftarrow \mathsf{SampleD}(1^{m\times k}, s)$ *the distribution of* $\mathbf{U} = \mathbf{A}'\mathbf{R} \in \mathbb{Z}_q^{n\times k}$ *is within* $k \cdot \mathrm{negl}(n)$ *statistical distance of uniform.*
   (b) *For any* $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$*,* $\mathsf{SamplePre}(\mathbf{T}, \mathbf{A}', \mathbf{U}, s)$ *outputs some* $\mathbf{R} \in \mathbb{Z}^{m\times k}$ *such that* $\|\mathbf{R}\| \le s\sqrt{m}$*,* $s_1(\mathbf{R}) = O(s\sqrt{m+k})$*, and the distribution of* $\mathbf{R}$ *is within* $k \cdot \mathrm{negl}(n)$ *statistical distance of* $D_{\mathbf{A}',\mathbf{U},s}$*, the* conditional distribution *of* $\mathbf{R} \leftarrow \mathsf{SampleD}(1^{m\times k}, s)$ *conditioned on the event* $\mathbf{A}'\mathbf{R} = \mathbf{U}$*.*

In particular, the output distributions of the following two experiments are within $k \cdot \mathrm{negl}(n)$ statistical distance:

- *choose* $\mathbf{R} \leftarrow \mathsf{SampleD}(1^{m \times k}, s)$ *and output* $(\mathbf{R}, \mathbf{U} = \mathbf{A}'\mathbf{R} \in \mathbb{Z}_q^{n \times k})$.
- *choose uniformly random* $\mathbf{U} \leftarrow \mathbb{Z}_q^{n \times k}$ *and* $\mathbf{R} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}', \mathbf{U}, s)$, *and output* $(\mathbf{R}, \mathbf{U})$.

We stress that the distribution $D_{\mathbf{A}',\mathbf{U},s}$ from Item 3b does not involve $\mathbf{T}$, so $\mathsf{SamplePre}$'s output essentially reveals nothing about its $\mathbf{T}$ argument. More precisely, in probability experiments we can replace calls to $\mathsf{SamplePre}(\mathbf{T}, \mathbf{A}', \mathbf{U}, s)$, which use $\mathbf{T}$, with (not necessarily efficient) samplings from $D_{\mathbf{A}',\mathbf{U},s}$, which do not use $\mathbf{T}$.

3 Vector Commitments

In this section we construct a vector commitment scheme based on the SIS problem (for suitable parameters). By convention, in this section we let n be the security parameter.

3.1 Construction

We let messages be vectors over $\mathcal{M} = I^\ell$ for some desired ℓ and interval $I \subset \mathbb{Z}$ of contiguous integers of maximum magnitude $M_I = \max_{i \in I} |i|$, e.g., $I = \{0, 1\}$ and $M_I = 1$.

The construction uses a suitable gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ and an injective, efficiently computable invertible-differences encoding that maps any $i \in [d+1]$ to a matrix $\mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$, so that $\mathbf{H}_{i'} - \mathbf{H}_i$ is invertible for any distinct $i, i' \in [d+1]$; see Section 2.2 for instantiations. (In fact, because we will later need to take $q \gg d$, if q is prime then we can simply take $\mathbf{H}_i = i\mathbf{I}$ to be a scaled identity matrix.)

Construction 1 (SIS-based vector commitment). For suitable parameters $\bar{m}, s, \gamma$ (which are functions of the security parameter n), define the following differentially updatable vector commitment scheme.

- $\mathsf{Setup}(1^n, 1^d)$: choose uniformly random $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and let $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}_d)$. In all that follows, let $m = \bar{m} + w$ and for any $i \in [d]$ let
$$\mathbf{A}_i = \mathbf{A} - [\mathbf{0} \mid \mathbf{H}_i\mathbf{G}] \in \mathbb{Z}_q^{n \times m}. \tag{1}$$
Choose uniformly random $\mathbf{U} = [\mathbf{U}_0 \mid \cdots \mid \mathbf{U}_{d-1}] \leftarrow \mathbb{Z}_q^{n \times \ell d}$, where each $\mathbf{U}_j \in \mathbb{Z}_q^{n \times \ell}$.
For each $i \in [d]$ let $\mathbf{R}_{i,i} = \mathbf{0} \in \mathbb{Z}^{m \times \ell}$, and for each $j \in [d] \setminus \{i\}$, let $\mathbf{R}_{i,j} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_i, \mathbf{U}_j, s)$. Observe that $i \neq d$ and hence $\mathbf{H}_d - \mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$ is invertible, as needed by $\mathsf{SamplePre}$ (see Item 3 of Theorem 1). In particular,
$$\mathbf{R}_{i,j} \in \mathbb{Z}^{m \times \ell} \text{ is "short" and } \mathbf{A}_i\mathbf{R}_{i,j} = \mathbf{U}_j. \tag{2}$$
Output the committer parameters $cp = (\mathbf{U}, \mathbf{R} = (\mathbf{R}_{i,j})_{i,j \in [d]} \in \mathbb{Z}^{md \times \ell d})$ and the verifier parameters $vp = (\mathbf{A}, \mathbf{U})$.

- $\mathsf{Commit}(cp, \mathbf{m} \in \mathcal{M}^d = I^{\ell d})$: output commitment $\mathbf{c} = \mathbf{U}\mathbf{m} = \sum_{j \in [d]} \mathbf{U}_j \mathbf{m}_j \in \mathbb{Z}_q^n$ and state $st = \mathbf{m}$.
- $\mathsf{Open}(cp, st = \mathbf{m}, i \in [d])$: output $\mathbf{p}_i = \mathbf{R}_{i,\star}\mathbf{m} = \sum_{j \in [d]} \mathbf{R}_{i,j}\mathbf{m}_j \in \mathbb{Z}^m$.
- $\mathsf{Verify}(vp, \mathbf{c} \in \mathbb{Z}_q^n, i \in [d], \mathbf{m}_i \in \mathcal{M}, \mathbf{p}_i \in \mathbb{Z}^m)$: accept if $\|\mathbf{p}_i\| \leq \gamma$ and $\mathbf{c} = \mathbf{A}_i\mathbf{p}_i + \mathbf{U}_i\mathbf{m}_i$; otherwise, reject.

In addition, define the following update algorithms.

- $\mathsf{PrepareUpdates}^{\mathrm{diff}}(cp, j \in [d], \boldsymbol{\delta} \in \mathbb{Z}^\ell)$: output commitment update[10] $\delta_c = \tilde{\mathbf{c}} = \mathbf{U}_j\boldsymbol{\delta} \in \mathbb{Z}_q^n$, proof update[11] $\delta_p = (\mathbf{r}_i)_{i \in [d]}$ where $\mathbf{r}_i = \mathbf{R}_{i,j}\boldsymbol{\delta} \in \mathbb{Z}^m$, and state update $\delta_s = (j, \boldsymbol{\delta})$.
- $\mathsf{UpdateC}(vp, \mathbf{c} \in \mathbb{Z}_q^n, \delta_c = \tilde{\mathbf{c}} \in \mathbb{Z}_q^n)$: output $\mathbf{c}' = \mathbf{c} + \tilde{\mathbf{c}} \in \mathbb{Z}_q^n$.
- $\mathsf{UpdateP}(vp, i, \mathbf{p}_i \in \mathbb{Z}^m, \delta_p = (\mathbf{r}_i)_{i \in [d]})$: output $\mathbf{p}_i' = \mathbf{p}_i + \mathbf{r}_i \in \mathbb{Z}^m$.
- $\mathsf{UpdateS}(cp, st = \mathbf{m}, \delta_s = (j, \boldsymbol{\delta}))$: output $st' = \mathbf{m}'$, which is $\mathbf{m}$ with its jth $\mathcal{M}$-entry $\mathbf{m}_j$ replaced by $\mathbf{m}_j' = \mathbf{m}_j + \boldsymbol{\delta}$.

Parameters and sizes. An appropriate choice of the parameters $\bar{m}, s, \gamma$ is as follows:

- let $\bar{m} = \lceil 2n \log q \rceil$ so that Item 2 of Theorem 1 applies;
- let $s = s_T \cdot \omega$ where ω and $s_T = O(\sqrt{m})$ are as in Section 2.2, so that $s_1(\mathbf{T}) \leq s_T$ and $\mathsf{SamplePre}$ can sample with parameter s, by Items 1 and 3 (respectively) of Theorem 1;
- let $\gamma = O(sM_I\sqrt{(m + \ell d)\ell d}) = O(sM_I(m + \ell d))$ be sufficiently large so that the norm of a proof $\mathbf{p}_i = \mathbf{R}_{i,\star}\mathbf{m}$ is bounded by γ. (This is because $s_1(\mathbf{R}_{i,\star}) = O(s\sqrt{m + \ell d})$ by Item 3b of Theorem 1, and $\|\mathbf{m}\| \leq M_I\sqrt{\ell d}$ because $\mathbf{m} \in I^{\ell d}$.)[12]

Letting the modulus q be sufficiently large relative to β (Equation (3)), as needed for the hardness of the relevant SIS problem, we obtain the following asymptotic object sizes.

- Commitments are in $\mathbb{Z}_q^n$ and hence have size $O(n \log q)$ bits.
- Proofs are vectors in $\mathbb{Z}^m$ of Euclidean norm bounded by $\gamma < q$, and hence can have size $O(n \log q)$ bits.
- Committer parameters are dominated by the $d^2 - d$ short matrices $\mathbf{R}_{i,j} \in \mathbb{Z}^{m \times \ell}$, and hence have size $\tilde{O}(n\ell d^2)$ bits. This can be reduced by a factor of n using a ring-based construction and Ring-SIS.

[10] Alternatively, depending on how much space it requires to represent $\boldsymbol{\delta}$ compared to an element of $\mathbb{Z}_q^n$, it may be preferable to output $\delta_c = (j, \boldsymbol{\delta})$ as the commitment update, and have $\mathsf{UpdateC}$ compute $\mathbf{U}_j\boldsymbol{\delta}$.

[11] Alternatively, if $\mathsf{UpdateP}$ has access to all the $\mathbf{R}_{i,j}$ (via the committer parameters), then we can output a smaller proof update $\delta_p = \boldsymbol{\delta}$, and $\mathsf{UpdateP}$ can compute $\mathbf{r}_i = \mathbf{R}_{i,j}\boldsymbol{\delta}$ itself.

[12] A smaller γ can be used if we have a tighter bound on $\|\mathbf{m}\|$, e.g., if we know from the surrounding application that $\mathbf{m}$ is sparse.

– Verifier parameters are dominated by the d matrices $\mathbf{U}_j \in \mathbb{Z}_q^{n\times m}$, and hence have size $O(n^2 d \log q)$ bits. This also can be reduced by a factor of n using rings. Separately, the verifier parameters can be reduced to just the size of $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, which is $O(n^2 \log^2 q)$ bits (or $O(n \log^2 q)$ bits using rings), by including the appropriate authenticated $\mathbf{U}_i$ in each proof.

Combinable commitments and proofs. The scheme also has the following combinability properties.[13] Suppose we have several commitments $\mathbf{c}_j = \mathbf{U}\mathbf{m}_j$ to respective message vectors $\mathbf{m}_j \in \mathbb{Z}^{\ell d}$ for $j = 1, \ldots, t$. Gathering these into the columns of matrices $\mathbf{C} \in \mathbb{Z}_q^{n\times t}$ and $\mathbf{M} \in \mathbb{Z}^{\ell d \times t}$ (respectively), we have $\mathbf{C} = \mathbf{U}\mathbf{M}$. Then for any integer vector $\mathbf{e} \in \mathbb{Z}^t$ representing a linear combination of the message vectors, $\mathbf{c} := \mathbf{C}\mathbf{e} = \mathbf{U}(\mathbf{M}\mathbf{e}) \in \mathbb{Z}_q^n$ is the commitment to the combined message $\mathbf{m} := \mathbf{M}\mathbf{e} \in \mathbb{Z}^{\ell d}$.

Proofs are similarly combinable, for any fixed position. Let $\mathbf{p}_i^j \in \mathbb{Z}^m$ denote the opening of $\mathbf{c}_j$ at position i, and collect these into the columns of a matrix $\mathbf{P}_i \in \mathbb{Z}^{m\times t}$. Then $\mathbf{P}_i = \mathbf{R}_{i,\star}\mathbf{M}$, and hence $\mathbf{p}_i := \mathbf{P}_i\mathbf{e} = \mathbf{R}_{i,\star}\mathbf{m}$ is the proof for position i of the combined message $\mathbf{m} = \mathbf{M}\mathbf{e}$. Note that the entries of $\mathbf{m}$ may be larger than those of the $\mathbf{m}_j$—their magnitudes depend on the norm of $\mathbf{e}$—so the norm bound γ used in Verify and related parameters need to be adjusted accordingly.

In the full version we discuss variants of this scheme, including a more efficient one based on Ring-SIS.

3.2 Correctness

Lemma 1. *For the parameters given above, Construction 1 is a correct, perfectly updatable vector commitment scheme (according to Definition 1).*

Proof. We first show that openings are accepted by the verifier. Let $(cp, vp) \leftarrow \mathsf{Setup}(1^n, 1^d)$ and let $\mathbf{m} \in \mathcal{M}^d$, $c = \mathsf{Commit}(cp, \mathbf{m}) = \mathbf{U}\mathbf{m}$ and $\mathbf{p}_i = \mathsf{Open}(cp, \mathbf{m}, i) = \mathbf{R}_{i,\star}\mathbf{m}$ for some $i \in [d]$. Then

$$\mathbf{A}_i\mathbf{p}_i = \mathbf{A}_i \sum_{j\in[d]} \mathbf{R}_{i,j}\mathbf{m}_j = \sum_{j\in[d]} (\mathbf{A}_i\mathbf{R}_{i,j})\mathbf{m}_j = \sum_{j\in[d]\setminus\{i\}} \mathbf{U}_j\mathbf{m}_j,$$

since by definition of Setup we have that $\mathbf{A}_i\mathbf{R}_{i,j} = \mathbf{U}_j$ for $i \neq j$ (Equation (2)) and $\mathbf{R}_{i,i} = \mathbf{0}$. Finally, we have

$$\sum_{j\in[d]\setminus\{i\}} \mathbf{U}_j\mathbf{m}_j = \mathbf{U}\mathbf{m} - \mathbf{U}_i\mathbf{m}_i = \mathbf{c} - \mathbf{U}_i\mathbf{m}_i,$$

as required. Moreover, by our choice of γ above we have that $\|\mathbf{p}_i\| = \|\mathbf{R}_{i,*}\mathbf{m}\| \leq \gamma$. Therefore, $\mathsf{Verify}(vp, \mathbf{c}, i, \mathbf{m}_i, \mathbf{p}_i)$ accepts.

Due to space limitations, we defer the proof of perfect differential updatability to the full version. □

[13] We use the term *combinability* rather than *aggregatability* because the latter is often used for the ability to combine proofs for several different locations, which we do not see how to do for our construction.

3.3 Security

Theorem 2. *Construction 1 is position binding if $SIS_{n,q,\bar{m}+\ell,\beta}$ is hard for*

$$\beta := 2\sqrt{\gamma^2 s_T^2 + M_I^2 \cdot \ell} = O(\gamma \cdot s_T) = O(M_I \cdot m(m+\ell d)) \cdot \omega. \tag{3}$$

More specifically, for any adversary $\mathcal{A}$ against the position binding of the scheme, there is an $SIS_{n,q,\bar{m}+\ell,\beta}$ adversary $\mathcal{B}$ for which

$$\mathbf{Adv}^{SIS}(\mathcal{B}) \geq \frac{1}{d}\mathbf{Adv}^{pba}(\mathcal{A}) - \mathrm{negl}(n),$$

and whose running time is essentially that of $\mathcal{A}$, plus a small polynomial in n.

Proof. Let $\mathcal{A}$ be any adversary that attacks the position binding (Definition 2) of Construction 1. That is, for some $d = \mathrm{poly}(n)$, and letting $(cp, vp) \leftarrow \mathsf{Setup}(1^n, 1^d)$, $\mathcal{A}(1^n, 1^d, cp, vp)$ attempts to output some $(\mathbf{c}, i, \mathbf{m}, \mathbf{m}', \mathbf{p}, \mathbf{p}')$ with distinct $\mathbf{m}, \mathbf{m}' \in I^{\ell d}$ and where $\mathsf{Verify}(vp, \mathbf{c}, i, \mathbf{m}, \mathbf{p})$, $\mathsf{Verify}(vp, \mathbf{c}, i, \mathbf{m}', \mathbf{p}')$ both accept.

We use $\mathcal{A}$ to construct an SIS adversary $\mathcal{B}$ which, on input $[\bar{\mathbf{A}} \mid \bar{\mathbf{U}}] \in \mathbb{Z}_q^{n \times (\bar{m}+\ell)}$, seeks to output a nonzero vector $\mathbf{x} \in \mathbb{Z}^{\bar{m}+\ell}$ such that $[\bar{\mathbf{A}} \mid \bar{\mathbf{U}}]\mathbf{x} = \mathbf{0}$ and $\|\mathbf{x}\| \leq \beta$. It operates as follows:

1. Choose uniformly random $i^* \in [d]$ as a guess of the position where $\mathcal{A}$ will attempt to break binding.
2. Let $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}_{i^*})$ and note that $s_1(\mathbf{T}) \leq s_T$ and $\mathbf{A}_{i^*} = \mathbf{A} - [\mathbf{0} \mid \mathbf{H}_{i^*}\mathbf{G}] = \bar{\mathbf{A}}\mathbf{T} \in \mathbb{Z}_q^{n \times m}$, by Item 1 of Theorem 1.
3. Define $\mathbf{U}_{i^*} = \bar{\mathbf{U}}$.
4. For each $j \neq i^*$ let $\mathbf{R}_{i^*,j} \leftarrow \mathsf{SampleD}(1^{m \times \ell}, s)$ and set $\mathbf{U}_j = \mathbf{A}_{i^*}\mathbf{R}_{i^*,j} \in \mathbb{Z}_q^{n \times \ell}$.
5. For each $i \neq i^*$ and $j \neq i$, let $\mathbf{R}_{i,j} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_i, \mathbf{U}_j, s)$.
6. For each $i \in [d]$ let $\mathbf{R}_{i,i} = \mathbf{0}$.
7. Let $cp = (\mathbf{U} = (\mathbf{U}_j)_{j \in [d]} \in \mathbb{Z}_q^{n \times \ell d}, \mathbf{R} = (\mathbf{R}_{i,j})_{i,j \in [d]} \in \mathbb{Z}^{md \times \ell d})$ and $vp = (\mathbf{A}, \mathbf{U})$.
8. Let $(\mathbf{c}, i, \mathbf{m}, \mathbf{m}', \mathbf{p}, \mathbf{p}') \leftarrow \mathcal{A}(1^n, 1^d, cp, vp)$.
9. If $i \neq i^*$, abort. Otherwise, output $\mathbf{x} = \begin{bmatrix} \mathbf{T}(\mathbf{p} - \mathbf{p}') \\ \mathbf{m} - \mathbf{m}' \end{bmatrix} \in \mathbb{Z}^{\bar{m}+\ell}$.

By inspection, it is clear that $\mathcal{B}$ runs in the same time as $\mathcal{A}$, plus a small polynomial.

First, we show that if $\mathcal{A}$ successfully breaks binding at position $i = i^*$, then $\mathcal{B}$ outputs an SIS solution for $[\bar{\mathbf{A}} \mid \bar{\mathbf{U}}]$. In this case, we have $\mathbf{m} \neq \mathbf{m}'$ and both $\mathsf{Verify}(vp, \mathbf{c}, i^*, \mathbf{m}, \mathbf{p})$ and $\mathsf{Verify}(vp, \mathbf{c}, i^*, \mathbf{m}', \mathbf{p}')$ accept, so $\mathbf{c} = \mathbf{A}_{i^*}\mathbf{p} + \mathbf{U}_{i^*}\mathbf{m} = \mathbf{A}_{i^*}\mathbf{p}' + \mathbf{U}_{i^*}\mathbf{m}'$ and $\|\mathbf{p}\|, \|\mathbf{p}'\| \leq \gamma$. Recalling from above that $\mathbf{A}_{i^*} = \bar{\mathbf{A}}\mathbf{T}$, we

therefore have

$$\begin{aligned}[\bar{\mathbf{A}} \mid \bar{\mathbf{U}}]\mathbf{x} &= [\bar{\mathbf{A}} \mid \bar{\mathbf{U}}] \begin{bmatrix} \mathbf{T}(\mathbf{p}-\mathbf{p}') \\ \mathbf{m}-\mathbf{m}' \end{bmatrix} \\ &= \bar{\mathbf{A}}\mathbf{T}(\mathbf{p}-\mathbf{p}') + \bar{\mathbf{U}}(\mathbf{m}-\mathbf{m}') \\ &= \mathbf{A}_{i^*}(\mathbf{p}-\mathbf{p}') + \mathbf{U}_{i^*}(\mathbf{m}-\mathbf{m}') \\ &= (\mathbf{A}_{i^*}\mathbf{p} + \mathbf{U}_{i^*}\mathbf{m}) - (\mathbf{A}_{i^*}\mathbf{p}' + \mathbf{U}_{i^*}\mathbf{m}') \\ &= \mathbf{c} - \mathbf{c} = \mathbf{0}.\end{aligned}$$

Moreover, $\mathbf{x} \neq \mathbf{0}$ because $\mathbf{m} \neq \mathbf{m}'$, and by the triangle inequality and the above bound $s_1(\mathbf{T}) \leq s_T$ we have

$$\|\mathbf{x}\| = \sqrt{\|\mathbf{T}(\mathbf{p}-\mathbf{p}')\|^2 + \|\mathbf{m}-\mathbf{m}'\|^2} \leq \sqrt{(2\gamma \cdot s_T)^2 + (2M_I \cdot \ell)^2} = \beta.$$

Therefore, $\mathbf{x}$ is an SIS solution for $[\bar{\mathbf{A}} \mid \bar{\mathbf{U}}]$.

It remains to analyze the probability that $\mathcal{A}$ breaks position binding at position $i = i^*$ in the above experiment run by $\mathcal{B}$. To do this we consider the following hybrid experiments.

- H_0 corresponds exactly to the "real" attack experiment with Setup, with some convenient presentational changes to aid comparison with the other experiments.
  1. Sample $i^* \leftarrow [d]$ uniformly at random.
  2. Sample $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ uniformly at random and $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}_d)$.
  3. Sample $\mathbf{U}_{i^*} \in \mathbb{Z}_q^{n \times \ell}$ uniformly at random.
  4. For each $j \neq i^*$, choose uniform $\mathbf{U}_j \leftarrow \mathbb{Z}_q^{n \times \ell}$ and let $\mathbf{R}_{i^*,j} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_{i^*}, \mathbf{U}_j, s)$.
  5. For each $i \neq i^*$ and $j \neq i$, let $\mathbf{R}_{i,j} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_i, \mathbf{U}_j, s)$.
  6. Set $\mathbf{R}_{i,i} = \mathbf{0}$ for all $i \in [d]$.
  7. Set $cp = (\mathbf{U}, \mathbf{R} = (\mathbf{R}_{i,j})_{i,j \in [d]})$ and $vp = (\mathbf{A}, \mathbf{U} = (\mathbf{U}_j)_{j \in [d]})$.
  8. Let $(\mathbf{c}, i, \mathbf{m}, \mathbf{m}', \mathbf{p}, \mathbf{p}') \leftarrow \mathcal{A}(1^n, 1^d, cp, vp)$.

  Note that H_0 is identical to the position-binding attack experiment against the scheme, as the choice of i^* only affects the order in which the $\mathbf{R}_{i,j}$ and $\mathbf{U}_j$ are selected. This does not affect the distribution of (cp, vp), and indeed, i^* is independent of the input to $\mathcal{A}$. Hence, the probability of the event GOOD that $\mathcal{A}$ breaks position binding at index $i = i^*$ is $d^{-1} \cdot \mathbf{Adv}^{\mathrm{pba}}(\mathcal{A})$.
- H_1 is identical to H_0 except that we replace Step 4 with the following:
  4. For each $j \neq i^*$, sample $\mathbf{R}_{i^*,j} \leftarrow \mathsf{SampleD}(1^{m \times \ell}, s)$ and let $\mathbf{U}_j = \mathbf{A}_{i^*}\mathbf{R}_{i^*,j} \in \mathbb{Z}_q^{n \times \ell}$.

  By Item 3 of Theorem 1, each independent $(\mathbf{R}_{i^*,j}, \mathbf{U}_j)$ pair generated in H_0 is within $\ell \cdot \mathrm{negl}(n)$ statistical distance of the corresponding pair generated in H_1. Because $\ell = \mathrm{poly}(n)$, the probability of the event GOOD in H_1 is within $\mathrm{negl}(n)$ of its probability in H_0.
- H_2 is identical to H_1 except we replace Step 5 with the following:
  5. For $i \neq i^*$ and $j \neq i$, (inefficiently) sample $\mathbf{R}_{i,j} \leftarrow D_{\mathbf{A}_i, \mathbf{U}_j, s}$.

Note that $\mathbf{T}$ is unused in H_2, because the experiment instead samples inefficiently (e.g., using brute force); this is acceptable because we will only make *statistical* comparisons between H_2 and its adjacent experiments. For each i, we have $i \neq d$ and hence $\mathbf{H}_d - \mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$ is invertible. So by Item 3b of Theorem 1, each of the independent $\mathbf{R}_{i,j}$ (for $i \neq i^*$ and $j \neq i$) sampled from $\mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_i, \mathbf{U}_j, s)$ in H_1 is within $\ell \cdot \mathrm{negl}(n)$ statistical distance of the corresponding $\mathbf{R}_{i,j}$ sampled from $D_{\mathbf{A}_i, \mathbf{U}_j, s}$ in H_2. Hence, the probability of the event GOOD in H_2 is within $\mathrm{negl}(n)$ of its probability in H_1.

- H_3 is identical to H_2 except that we replace Step 2 with the following:
  2. Sample $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times \bar{m}}$ uniformly at random and let $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}_{i^*})$.

  By Item 2 of Theorem 1, for uniformly random $\bar{\mathbf{A}}$ we have that the $\mathbf{A}$ obtained from $\mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H})$ is within $\mathrm{negl}(n)$ statistical distance from uniform, for *any* tag $\mathbf{H}$. Hence, the $\mathbf{A}$ generated in H_2 is within $\mathrm{negl}(n)$ statistical distance from the $\mathbf{A}$ generated in H_3. Note that this is *not* necessarily true for the $\mathbf{T}$s generated in each experiment, however $\mathbf{T}$ is entirely unused (following its creation) in both experiments. Hence, the probability of the event GOOD in H_3 is within $\mathrm{negl}(n)$ of its probability in H_2.
- H_4 is identical to H_3 except that we replace Step 5 with the following:
  5. For each $i \neq i^*$ and $j \neq i$, sample $\mathbf{R}_{i,j} \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_i, \mathbf{U}_j, s)$.

  For all $i \neq i^*$ we have that $\mathbf{H}_{i^*} - \mathbf{H}_i \in \mathbb{Z}_q^{n \times n}$ is invertible. So by Item 3b of Theorem 1, each (independently chosen) $\mathbf{R}_i$ that is generated in this way is within $\mathrm{negl}(n)$ statistical distance of the corresponding one in H_3. By the same reasoning given above for H_2 versus H_1, the probability of the event GOOD in H_4 is within $\mathrm{negl}(n)$ of its probability in H_3.

  Finally, notice that H_4 is identical to the experiment that $\mathcal{B}$ simulates to $\mathcal{A}$, because the $\bar{\mathbf{A}}$ and $\mathbf{U}_{i^*} = \bar{\mathbf{U}}$ from the SIS instance are uniformly random and independent.

Combining all of the above completes the proof of the theorem. □

3.4 Specialized Tree Transformation

In the vector commitment scheme from Construction 1, the sizes of the committer and verifier parameters are respectively quadratic and linear in the message dimension d, which makes the construction unsuitable as-is for large dimensions. In the full version, we give a formal treatment of a generic d-ary tree construction that transforms a VC scheme for dimension d into one for dimension d^h for any desired positive integer h, with no increase in the sizes of the parameters or commitments. Only the proof and proof-update sizes increase, growing linearly in h but independently of d. However, this transformation fails to preserve the combinability of commitments and proofs, as well as the stateless updatability property of the base scheme, which is important for distributed VCs [CPSZ18].

Here we give a specialized tree-like transformation of our SIS-based VC scheme. In contrast with the generic transform, ours preserves combinability and (differential) stateless updates, essentially because commitments are linear in the committed messages, but at the price of somewhat larger objects and a stronger

SIS assumption. The transformation is based on the main idea from [PSTY13], which was used in the context of an SIS-based Merkle-tree-like construction (which can be used as a VC). In that context, a proof must include all of the "sibling" information for each step in a root-to-leaf path, so the proof size ends up being proportional to $hd\log^2(d^h)$. (The $\log^2(d^h)$ factor comes from the length of the proofs along the path as well as the sizes of the integers within them.) Here we show that the main idea from [PSTY13] also applies to our SIS-based VC scheme, but with the advantage that *proofs need not contain any sibling information*, so the proof size grows only as $h\log^2(d^h) = h^3\log^2 d$.

In summary, our construction is quantitatively a strict improvement over the VC obtained from [PSTY13], for any choice of arity d and tree height h (but at the price of private setup). Its efficiency profile also recommends using a moderately large d and correspondingly smaller h, which can ultimately yield the same asymptotic proof size as for the *generic* tree transformation for VCs, while preserving combinability and (differential) stateless updates (but at the price of private setup and a stronger SIS assumption).

Construction 2 (SIS-based tree vector commitment). Let $\mathbf{G} \in \mathbb{Z}_q^{n\times w}$ be a suitable gadget matrix with associated magnitude bound $g_{\mathbf{G}}$ for the $\mathbf{G}^{-1}$ operation; see Section 2.2 for details. (This gadget need not be the same as the one used in Construction 1.)

Let message vectors be over $\overline{\mathcal{M}} = \overline{I}^w$ for some desired range $\overline{I} = \{-M_{\overline{I}}, \ldots, M_{\overline{I}}\} \subset \mathbb{Z}$. Let h be any positive integer, and adopt the algorithms defined in Construction 1 for dimension-d vectors over $\mathcal{M} = I^w$, where $I = \{-M_I, \ldots, M_I\}$ for $M_I = M_{\overline{I}} \cdot g_{\mathbf{G}} \cdot w \cdot d^h$.[14] Define the following algorithms:

- $\overline{\mathsf{Setup}}_h(1^n, 1^{d^h})$: output $(cp = (\mathbf{U} = [\mathbf{U}_0 \mid \cdots \mid \mathbf{U}_{d-1}], \_), vp) \leftarrow \mathsf{Setup}(1^n, 1^d)$. In all that follows, let $\mathbf{S}^{(1)} = \mathbf{I}_{wd} \in \mathbb{Z}^{wd\times wd}$ and $\mathbf{U}^{(1)} = \mathbf{U} \in \mathbb{Z}_q^{n\times wd}$, and for $1 < k \le h$ let

$$\mathbf{S}^{(k)} = \mathbf{I}_d \otimes \mathbf{G}^{-1}(\mathbf{U}^{(k-1)}) \in \mathbb{Z}^{wd\times wd^k}$$
$$\mathbf{U}^{(k)} = \mathbf{U}\mathbf{S}^{(k)} \in \mathbb{Z}_q^{n\times wd^k}.$$

  Note that $\mathbf{U}^{(k)} = [\mathbf{U}_0^{(k)} \mid \cdots \mid \mathbf{U}_{d^k-1}^{(k)}]$ can be viewed as a block matrix where each $\mathbf{U}_{\bar{\jmath}}^{(k)} \in \mathbb{Z}_q^{n\times w}$ and $\bar{\jmath} \in [d^k]$. Moreover, each such block can be computed independently given just $\mathbf{U}$ and $\bar{\jmath}$, without needing to compute the entire matrix.[15]
- For $1 \le k \le h$, $\overline{\mathsf{Commit}}_k(cp, \overline{\mathbf{m}} \in \overline{\mathcal{M}}^{d^k} = \overline{I}^{wd^k})$ does:

[14] The factor $M_{\overline{I}} \cdot w \cdot d^h$ in M_I may be replaced by an upper bound on $\|\mathbf{m}\|_1$ for only those message vectors $\mathbf{m}$ that may be used in an application, e.g., sparse vectors.

[15] Specifically, $\mathbf{U}_{\bar{\jmath}}^{(k)} = \mathbf{U}_j\mathbf{G}^{-1}(\mathbf{U}_{\bar{\jmath}'}^{(k-1)})$ where $\bar{\jmath} = jd^{k-1} + \bar{\jmath}'$ for $\bar{\jmath} \in [d^k]$, $j \in [d]$, and $\bar{\jmath}' \in [d^{k-1}]$. Unrolling this fully, $\mathbf{U}_{\bar{\jmath}}^{(k)} = \mathbf{U}_{j_{k-1}}\mathbf{G}^{-1}(\mathbf{U}_{j_{i-2}}\mathbf{G}^{-1}(\cdots\mathbf{G}^{-1}(\mathbf{U}_{j_1}\mathbf{G}^{-1}(\mathbf{U}_{j_0}))\cdots))$ where $j_{k-1}\cdots j_1 j_0$ is the base-d representation of $\bar{\jmath}$.

- let $\overline{\mathbf{c}} = \mathbf{G}^{-1}(\mathbf{U}^{(k)})\overline{\mathbf{m}} \in I^w = \mathcal{M}$,[16]
- output $(\overline{\mathbf{c}}, \overline{st} = \overline{\mathbf{m}})$.

– $\overline{\mathsf{Open}}_1 = \mathsf{Open}$. For $1 < k \leq h$, $\overline{\mathsf{Open}}_k(cp, \overline{st} = \overline{\mathbf{m}} \in \overline{\mathcal{M}}^{d^k}, \overline{\imath} \in [d^k])$ does:
- write $\overline{\imath} = id^{k-1} + \overline{\imath}'$ where $i \in [d]$ and $\overline{\imath}' \in [d^{k-1}]$,
- parse $\overline{\mathbf{m}} = (\overline{\mathbf{m}}_0, \ldots, \overline{\mathbf{m}}_{d-1})$ where each $\overline{\mathbf{m}}_i \in \overline{\mathcal{M}}^{d^{k-1}}$,
- let $\mathbf{p}_i \leftarrow \mathsf{Open}(cp, \mathbf{S}^{(k)}\overline{\mathbf{m}} \in I^{wd} = \mathcal{M}^d, i)$,
- let $(\overline{\mathbf{c}}_i, \_) \leftarrow \overline{\mathsf{Commit}}_{k-1}(cp, \overline{\mathbf{m}}_i)$,[17]
- let $\overline{p}'_{\overline{\imath}'} \leftarrow \overline{\mathsf{Open}}_{k-1}(cp, \overline{\mathbf{m}}_i, \overline{\imath}')$,
- output $\overline{p}_{\overline{\imath}} = (\mathbf{p}_i, \overline{\mathbf{c}}_i, \overline{p}'_{\overline{\imath}'})$.

– $\overline{\mathsf{Verify}}_1(vp, \overline{\mathbf{c}} \in I^w = \mathcal{M}, \overline{\imath} \in [d], \overline{\mathbf{m}}_{\overline{\imath}} \in \overline{\mathcal{M}}, \overline{p}_{\overline{\imath}}) = \mathsf{Verify}(vp, \mathbf{G}\overline{\mathbf{c}}, \overline{\imath}, \overline{\mathbf{m}}_{\overline{\imath}}, \overline{p}_{\overline{\imath}})$.
For $1 < k \leq h$, $\overline{\mathsf{Verify}}_k(vp, \overline{\mathbf{c}} \in I^w = \mathcal{M}, \overline{\imath} \in [d^k], \overline{\mathbf{m}}_{\overline{\imath}} \in \overline{\mathcal{M}}, \overline{p}_{\overline{\imath}})$ does:
- define $i, \overline{\imath}'$ in terms of $\overline{\imath}$ as in $\overline{\mathsf{Open}}_k$ and parse $\overline{p}_{\overline{\imath}} = (\mathbf{p}_i, \overline{\mathbf{c}}_i, \overline{p}'_{\overline{\imath}'})$,
- if $\mathsf{Verify}(vp, \mathbf{G}\overline{\mathbf{c}}, i, \overline{\mathbf{c}}_i, \mathbf{p}_i)$ rejects, then reject,
- if $\overline{\mathsf{Verify}}_{k-1}(vp, \overline{\mathbf{c}}_i, \overline{\imath}', \overline{\mathbf{m}}_{\overline{\imath}}, \overline{p}'_{\overline{\imath}'})$ rejects, then reject; else, accept.

Update algorithms follow rather straightforwardly from the linearity of the commitment, and are given in the full version.

Object sizes. Instantiating the underlying scheme with parameters discussed above and choosing sufficiently large q relative to β (where β is as in Equation (3)), so that $\log q = O(h \log d + \log n) = O(h \log d)$ under the (very mild) assumption that the vector dimension $d^h = n^{\Omega(1)}$ is at least polynomial in n, we obtain the following asymptotic object sizes.

– A commitment is a vector in $\mathcal{M} = I^w$, with each entry bounded by $M_I = M_{\overline{I}} \cdot g_{\mathbf{G}} \cdot w \cdot d^h$. Since $w = O(n \log q) = O(nh \log d)$, a commitment requires $O(wh \log d) = O(nh^2 \log^2 d)$ bits to represent.
– A proof consists of $h - 1$ vectors in $\mathcal{M}$ and h proofs from the underlying scheme; the latter are vectors in $\mathbb{Z}^m$ of Euclidean norm bounded by γ. Since $m = O(n \log q) = O(nh \log d)$ and $\log \gamma = O(h \log d)$, a full proof requires $O(nh^3 \log^2 d)$ bits to represent.
– The committer parameters cp are dominated by the $d^2 - d$ short matrices $\mathbf{R}_{i,j} \in \mathbb{Z}^{m \times w}$ from the underlying scheme, and hence have size $\tilde{O}(n^2h^2d^2)$. This can be reduced by a factor of n using a ring-based construction
– The verifier parameters vp are dominated by the d matrices $\mathbf{U}_j \in \mathbb{Z}_q^{n \times w}$, and hence have size $O(n^2h^2d \log^2 d)$ bits. This also can be reduced by a factor of n using a ring-based construction.

In the full version, we prove that for any positive integer h, Construction 2 is correct, perfectly updatable, and position binding.

[16] Note that $\mathbf{G}\overline{\mathbf{c}} = \mathbf{U}\mathbf{S}^{(k)}\overline{\mathbf{m}} \in \mathbb{Z}_q^n$ is the commitment output of $\mathsf{Commit}(cp, \mathbf{S}^{(k)}\overline{\mathbf{m}})$, but we cannot necessarily compute $\overline{\mathbf{c}}$ from that output.

[17] Alternatively, the computation of all the d^{h-1} intermediate commitments $\overline{\mathbf{c}}_i$ could be done at commitment time, and stored in st.

4 Functional Commitments (with Authority)

In this section we define *functional commitments with authority*, which enable concise commitments and proofs of *arbitrary functions* (from a particular family) of committed messages. We introduce a new model in which a trusted authority both performs the system setup, and remains online to give out *opening keys* ok_f that enable committers to open desired functions f of committed messages. We stress that, unlike with identity/attribute-based encryption, where each key extracted by the authority must be transmitted confidentially to its intended recipient and kept secret, with functional commitments all opening keys can be made public and used by any party. For example, any party can query the authority for any supported function f, and the authority can post the opening key ok_f on a public bulletin board for all to see and use.

Of course, if the supported function family has only polynomially many functions, then the authority can immediately post all the associated opening keys and then go offline forever. However, many families of interest have super-polynomially (or even exponentially) many functions, so in these cases the authority needs to remain online to answer new queries. It is a very interesting question whether our construction can be modified to remove the need for an online authority.

4.1 Definitions

Here we formally define functional commitments with authority, and the security notions we consider for them.

Definition 4. *A* functional commitment scheme with authority *for a function class $\mathcal{F}$, and having message space $\mathcal{M}$, commitment space $\mathcal{C}$, and proof space $\mathcal{P}$ (all of which may depend on the security parameter), is a tuple of algorithms with the following interfaces:*

- $\mathsf{Setup}(1^\lambda)$ *outputs* committer parameters *cp,* verifier parameters *vp, and an* extraction key *ek.*
- $\mathsf{Extract}(ek, f \in \mathcal{F})$ *outputs an* opening key ok_f *for the function f.*
- $\mathsf{Commit}(cp, m \in \mathcal{M})$ *outputs a* commitment $c \in \mathcal{C}$ *and some* auxiliary data *aux.*
- $\mathsf{Open}(cp, aux, ok_f)$ *outputs a* proof $p_{f,m} \in \mathcal{P}$ *for the value of $f(m)$, where m is the committed message associated to aux.*
- $\mathsf{Verify}(vp, c \in \mathcal{C}, f \in \mathcal{F}, y, p_{f,m})$ *either accepts or rejects.*

The scheme should satisfy the following correctness property: for any $m \in \mathcal{M}$ and $f \in \mathcal{F}$, and for $(cp, vp, ek) \leftarrow \mathsf{Setup}(1^\lambda)$, $ok_f \leftarrow \mathsf{Extract}(ek, f)$, $(c, aux) \leftarrow \mathsf{Commit}(cp, m)$, and $p_{f,m} \leftarrow \mathsf{Open}(cp, aux, ok_f)$, $\mathsf{Verify}(vp, c, f, y = f(m), p_{f,m})$ should accept with $1 - \mathrm{negl}(\lambda)$ probability (over all the randomness of the experiment).

Definition 5. *For a functional commitment scheme with authority FCS, the* selective-function attack *game with an adversary is defined as follows:*

1. *The adversary is given the security parameter* 1^λ *and outputs a function* $f^* \in \mathcal{F}$ *to the challenger.*
2. *The challenger lets* $(cp, vp, ek) \leftarrow \mathsf{Setup}(1^\lambda)$ *and gives* cp, vp *to the adversary.*
3. *The adversary is given adaptive oracle (query) access to* $\mathsf{Extract}(ek, \cdot)$.[18]
4. *Finally, the adversary outputs a commitment* c^* *and two value-proof pairs* (y, p) *and* (y', p')*. It* wins *the game if* $y \neq y'$ *and both* $\mathsf{Verify}(vp, c^*, f^*, y, p)$ *and* $\mathsf{Verify}(vp, c^*, f^*, y', p')$ *accept.*

The advantage *of an adversary* $\mathcal{A}$ *in the above game, denoted* $\mathbf{Adv}^{sfa}_{FCS}(\mathcal{A})$*, is the probability that it wins the game (as a function of the security parameter).*

We say that FCS has the selective function binding *property if* $\mathbf{Adv}^{sfa}_{FCS}(\mathcal{A}) = \mathrm{negl}(\lambda)$ *for every probabilistic polynomial-time adversary* $\mathcal{A}$*.*

Remark 2. One can strengthen Definition 5 by changing the attack game so that the adversary does not specify the target function f^* until Item 4 (rather than in Item 1, before seeing the public parameters and the queried opening keys); we call the resulting security notion *adaptive*, or *full*, functional binding. Generically, any scheme with selective security also has adaptive security, up to a loose reduction whose advantage is smaller by a factor of the size of the function family. (This follows by the standard technique of complexity leveraging—i.e., initially "guessing" the function f^* that the adversary will eventually choose, and succeeding when this guess turns out to be correct.)

4.2 Homomorphic Commitments

A main tool used in our functional commitment scheme is a homomorphic commitment implicit in the FHE scheme of Gentry, Sahai, and Waters (GSW) [GSW13], and made explicit in the works of Gorbunov, Vaikuntanathan, and Wichs (GVW) [GVW15] and Peikert and Shiehian [PS19], which we recall in this section. For better parameters and efficiency when working with certain function families (e.g., linear functions), we generalize the scheme somewhat using standard "tagged trapdoor" and homomorphic techniques developed in works such as [PW08, AFV11, MP12, Xag13]. The following theorem abstracts what we need from these works and others like [BV14, AP14]; see the cited works and the full version for further details on the implementations of the claimed algorithms.

Theorem 3 (Homomorphic commitment). *Let* $\mathcal{U}$ *denote one of the following families* $\mathcal{U}_{linear}$*,* $\mathcal{U}_{circuit}$*,* $\mathcal{U}_{BP}$ *of "size-*T*" functions from* X^S *to* X^L*, for a certain domain* X*, input size* S*, and output length* L*:*

- *for* $X = \mathbb{Z}_q^{n \times n}$*,* $T = S$*, and* $L = 1$*, the family* $\mathcal{U}_{linear}$ *of functions* $U_{\vec{\mathbf{M}}}$ *with* $\vec{\mathbf{M}} = (\mathbf{M}_1, \ldots, \mathbf{M}_S) \in X^S$*, defined as* $U_{\vec{\mathbf{M}}}(\vec{\mathbf{F}}) := \sum_{i=1}^{S} \mathbf{F}_i \mathbf{M}_i$ *for* $\vec{\mathbf{F}} = (\mathbf{F}_1, \ldots, \mathbf{F}_S) \in X^S$*;*

[18] Note that we allow the adversary to query the oracle on any $f \in \mathcal{F}$, even $f = f^*$. This is because having an opening key for f^* does not inherently allow for breaking function binding for f^*—as opposed to, say, identity-based encryption, where a decryption key for the target identity trivially allows decryption of the challenge ciphertext.

– *for* $X = \{0,1\}$*, the set* $\mathcal{U}_{circuit}$ *of size-*T*, depth-*D *Boolean circuits* $U\colon \{0,1\}^S \to \{0,1\}^L$*;*
– *for* $X = \{0,1\}$*, the set* $\mathcal{U}_{BP}$ *of size-*T *(for a given width) branching programs* $U\colon \{0,1\}^S \to \{0,1\}^L$*.*

There exist deterministic polynomial-time algorithms Encode*,* Eval *having the following properties. Each input in square brackets is optional, and when provided, the additional output (also in square brackets) is also produced. The algorithm's main output is the same whether or not the optional input is provided.*

1. $\mathsf{Eval}(U \in \mathcal{U}, \mathbf{C} \in \mathbb{Z}_q^{n \times Sw}[, \mathbf{R}_x \in \mathbb{Z}^{m \times Sw}])$ *outputs a commitment matrix* $\mathbf{C}_U \in \mathbb{Z}_q^{n \times Lw}$ *[and an integral matrix* $\mathbf{R}_{U,x} \in \mathbb{Z}^{m \times Lw}$*].*
   If $\mathbf{C} = \mathbf{A}\mathbf{R}_x + \mathsf{Encode}(x)$ *for some* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and* $x \in X^S$*, then* $\mathbf{C}_U = \mathbf{A}\mathbf{R}_{U,x} + \mathsf{Encode}(U(x))$*, and:*
   (a) for $\mathcal{U} = \mathcal{U}_{linear}$*,* $\|\mathbf{R}_{U,x}\| \le \|\mathbf{R}_x\| \cdot Sw$*;*
   (b) for $\mathcal{U} = \mathcal{U}_{circuit}$*,* $\|\mathbf{R}_{U,x}\| \le \|\mathbf{R}_x\| \cdot (w+1)^D$*;*
   (c) for $\mathcal{U} = \mathcal{U}_{BP}$*,* $\|\mathbf{R}_{U,x}\| \le \|\mathbf{R}_x\| \cdot w^{O(1)} \cdot T$*.*
2. *There is an efficient deterministic polynomial-time algorithm that, given any invertible* $\mathbf{Y} \in \mathbb{Z}_q^{n \times n}$ *and any* $\mathbf{u} \in \mathbb{Z}_q^n$*, outputs a binary* $\mathbf{e} = \mathbf{G}^{-1}(\mathbf{Y}^{-1}\mathbf{u}) \in \{0,1\}^w$ *such that* $\mathsf{Encode}(\mathbf{Y}) \cdot \mathbf{e} = \mathbf{Y}(\mathbf{G}\mathbf{e}) = \mathbf{u}$*.*
3. *There is an efficient deterministic polynomial-time algorithm that, given any* $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$*, outputs a binary* $\mathbf{e} \in \{0,1\}^{Lw}$ *such that* $\mathsf{Encode}(\mathbf{y}) \cdot \mathbf{e} = \mathbf{M}\mathbf{y} \in \mathbb{Z}_q^n$ *for every* $\mathbf{y} \in \{0,1\}^L$*.*

4.3 Functional Commitment Construction

Here we give a functional commitment (with authority) for various families $\mathcal{F}$ of functions from a given message space $\mathcal{M}$ to X^L, for some domain X and output length L (typically, $X = \mathbb{Z}_q^{n \times n}$ or $X = \{0,1\}$).

Let $\mathcal{F}' = \mathcal{F} \cup \{d\}$ where $d \notin \mathcal{F}$ is some distinguished "dummy" function, and define the family of functions

$$\mathcal{U} := \{U_m\colon \mathcal{F}' \to X^L\}_{m \in \mathcal{M}}$$
$$U_m(f) := f(m);$$

we emphasize that this switches the roles of the message m and function f, letting the message define a function U_m that takes f as input data.

The construction requires homomorphic evaluation of any $U_m \in \mathcal{U}$ (for known $m \in \mathcal{M}$) on a commitment to a function f under the GSW/GVW scheme; let S denote the size of f under a suitable representation for this purpose. Naturally, the choice of function family $\mathcal{F}$ influences the scheme's efficiency and parameters (norm bound γ, modulus q, etc.); we describe some example instantiations of interest following the construction.

The construction also uses an injective, efficiently computable invertible-differences encoding that maps any $f \in \mathcal{F}'$ to a matrix $\mathbf{H}_f \in \mathbb{Z}_q^{n \times n}$, so that $\mathbf{H}_f - \mathbf{H}_{f'}$ is invertible for any distinct $f, f' \in \mathcal{F}'$. (See Section 2.2 for instantiations.)

For simplicity we assume that the family $\mathcal{F}'$ is small enough to support such an encoding. Alternatively, we can map each f to a *tuple* $(\mathbf{H}_{f,1}, \ldots, \mathbf{H}_{f,t})$ for sufficiently large t so that $\mathbf{H}_{f,i} - \mathbf{H}_{f',i}$ is invertible for some i (and modify the construction below in the natural way), or we can first apply a collision-resistant (or even just universal one-way) hash function to f and use an invertible-differences encoding on the hash values.[19]

Construction 3 (SIS-based functional commitment). Let function families $\mathcal{F}, \mathcal{F}' = \mathcal{F} \cup \{d\}$ (which may depend on the security parameter n) and $\mathcal{U} = \{U_m(\cdot)\}$ be as described above. For suitable parameters $\bar{m}, s, \gamma$, we define the following functional commitment scheme (with authority) for the family $\mathcal{F}$.

- $\mathsf{Setup}(1^n)$: choose uniform $\bar{\mathbf{A}} \leftarrow \mathbb{Z}_q^{n \times \bar{m}}$ and let $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{T} \in \mathbb{Z}^{\bar{m} \times m}) \leftarrow \mathsf{TrapGen}(\bar{\mathbf{A}}, \mathbf{H}_d)$, where $d \in \mathcal{F}'$ is the special "dummy" function. In all that follows, for any $f \in \mathcal{F}$ let

$$\mathbf{A}_f = \mathbf{A} - [\mathbf{0} \mid \mathbf{H}_f \mathbf{G}] \in \mathbb{Z}_q^{n \times m}. \tag{4}$$

  Choose uniformly random $\mathbf{C} \leftarrow \mathbb{Z}_q^{n \times Sw}$. Output the committer parameters $cp = \mathbf{C}$, the verifier parameters $vp = \mathbf{A}$, and the extraction key $ek = (\mathbf{C}, \mathbf{A}, \mathbf{T})$.
- $\mathsf{Extract}(ek = (\mathbf{C}, \mathbf{A}, \mathbf{T}), f \in \mathcal{F})$: output $\mathbf{R}_f \leftarrow \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_f, \mathbf{C} - \mathsf{Encode}(f), s)$ as the opening key. Observe that $f \neq d$ and hence $\mathbf{H}_d - \mathbf{H}_f \in \mathbb{Z}_q^{n \times n}$ is invertible, as needed by $\mathsf{SamplePre}$ (see Item 3 of Theorem 1). In particular, $\mathbf{R}_f \in \mathbb{Z}^{m \times Sw}$ is "short" and

$$\mathbf{C} = \mathbf{A}_f \mathbf{R}_f + \mathsf{Encode}(f), \tag{5}$$

  i.e., $\mathbf{R}_f$ is randomness that opens $\mathbf{C}$ as a commitment to f with respect to $\mathbf{A}_f$.
  Note: for security, it is *essential* that repeated calls to $\mathsf{Extract}$ on the same input (ek, f) produce the *same* output $\mathbf{R}_f$. This can be ensured by the standard techniques of memoization (e.g., using a public bulletin board of all prior queries and answers), or by applying a pseudorandom function to (ek, f) to generate randomness for the call to $\mathsf{SamplePre}$, so that $\mathbf{R}_f$ is a deterministic function of the input.
- $\mathsf{Commit}(cp = \mathbf{C}, m \in \mathcal{M})$: output $\mathbf{C}_m = \mathsf{Eval}(U_m, \mathbf{C}) \in \mathbb{Z}_q^{n \times Lw}$ and $aux = m$. [For Boolean functions the commitment can be compressed significantly; see Section 4.3 below.]
- $\mathsf{Open}(cp = \mathbf{C}, aux = m \in \mathcal{M}, ok_f = \mathbf{R}_f)$: compute $(\mathbf{C}_m, \mathbf{R}_{m,f}) = \mathsf{Eval}(U_m, \mathbf{C}, \mathbf{R}_f)$ and output $\mathbf{R}_{m,f} \in \mathbb{Z}^{m \times Lw}$.
  [For Boolean functions the proof can be compressed significantly; see Section 4.3 below.]
- $\mathsf{Verify}(vp = \mathbf{A}, \mathbf{C}^*, f \in \mathcal{F}, \mathbf{y} \in X^L, \mathbf{R}^*)$: accept if $\|\mathbf{R}^*\| \leq \gamma$ and $\mathbf{C}^* = \mathbf{A}_f \mathbf{R}^* + \mathsf{Encode}(\mathbf{y})$, i.e., if $\mathbf{R}^*$ is sufficiently short randomness that opens $\mathbf{C}^*$ as a commitment to $\mathbf{y}$ with respect to $\mathbf{A}_f$. Otherwise, reject.

[19] No homomorphic properties (only invertible differences) are needed for this encoding of the functions, so hashing is acceptable.

Combinability. Similarly to our SIS-based vector commitments from Section 3, the functional commitment scheme has the following combinability property: given commitments $\mathbf{C}_m, \mathbf{C}_{m'}$ and respective proofs $\mathbf{R}_{m,f}, \mathbf{R}_{m',f}$ that $f(m) = \mathbf{y}, f(m') = \mathbf{y}'$ for some function f, we can take the same arbitrary small linear combination of each pair to get a combined commitment and a proof (under a suitably relaxed norm bound) that $f(m) + f(m') = \mathbf{y} + \mathbf{y}'$. For linear functions f this is equivalent to $f(m + m') = \mathbf{y} + \mathbf{y}'$, but we caution that in general it may not correspond to the application of f on any legal message.

Parameters. Here we give a convenient choice of parameters that works for all of our instantiations:

- let $\bar{m} = \lceil 2n \log q \rceil$ so that the output $\mathbf{A}$ of $\mathsf{TrapGen}$ is statistically close to uniform;
- let $s = s_T \cdot \omega$ where ω and $s_T = O(\sqrt{m})$ are as in Section 2.2, so that $s_1(\mathbf{T}) \leq s_T$ and $\mathsf{SamplePre}$ can sample with parameter s, by Items 1 and 3 (respectively) of Theorem 1;
- let γ be defined based on the particular function family $\mathcal{F}$, as in the following instantiations.

Instantiation: Linear Functions Over Finite Fields Let $\mathbb{F} = \mathbb{F}_{p^n}$ for some prime p that divides q, let $\mathcal{M} = \mathbb{F}^S$ for some $S = S(\lambda)$, and let $\mathcal{F}'$ be the family of all $\mathbb{F}$-linear functions from $\mathbb{F}^S$ to $\mathbb{F}$. We can represent any such function as a vector $\vec{f} = (f_1, \ldots, f_S) \in \mathbb{F}^S$ of field elements, and can define $U_{\vec{m}}(\vec{f}) := \sum_{i=1}^{S} f_i m_i \in \mathbb{F}$ for each $\vec{m} \in \mathbb{F}^S$. By simulating $\mathbb{F}$ using the matrix ring $\mathcal{R} = \mathbb{Z}_q^{n \times n}$ (simply by reducing modulo p), Theorem 3 with family $\mathcal{U}_{\text{linear}}$ yields a suitable homomorphic commitment. For this instantiation, following Item 1a of Theorem 3 we set

$$\gamma = \gamma_{\text{linear}} := s\sqrt{m} \cdot Sw.$$

Let the "dummy" function $d \in \mathcal{F}'$ be the trivial function that always outputs zero, leaving the family $\mathcal{F} = \mathcal{F}' \setminus \{d\}$ of all *nontrivial* linear functions as the one supported by the scheme (note that the trivial function is not needed, since its output is fixed).

Remark 3. The restriction to linear functions over *finite fields*, rather than more general matrix rings $\mathbb{Z}_p^{n \times n}$ where p divides q, is mostly for convenience of presentation in the SIS-based security proof (see the full version). Using more sophisticated techniques and exploiting the fact that every column of the trapdoor $\mathbf{T}$ is well hidden information theoretically (conditioned on the adversary's view), it is plausible that the proof could be adapted to work for the full matrix ring $\mathbb{Z}_p^{n \times n}$ (though we do not do so here, to keep the proof simpler).

Instantiation: Boolean Functions of Bounded Complexity Let $\mathcal{F}'$ be the family of all functions from some set $\mathcal{M}$ to $\{0,1\}^L$ that are computable by Boolean circuits of some depth $D' = D'(\lambda)$ and size $S = S(\lambda) = \text{poly}(\lambda)$, under

a suitable representation as binary strings. There is a (uniformly generated) universal Boolean circuit U of size $T = T(S) = \text{poly}(S)$ and depth $D = O(D')$ for which $U(f, m) = f(m)$ for all $f \in \mathcal{F}', m \in \mathcal{M}$. Defining the size-$T$, depth-$D$ circuits $U_m(\cdot) = U(\cdot, m)$, Theorem 3 with the family $\mathcal{U}_{\text{circuit}} = \{U_m(\cdot)\}$ yields a suitable homomorphic commitment. For this instantiation, following Item 1b of Theorem 3 we set

$$\gamma = \gamma_{\text{circuit}} := s\sqrt{m} \cdot (w+1)^D.$$

We proceed analogously for the family $\mathcal{F}'$ of functions from $\mathcal{M}$ to $\{0,1\}^L$ computable by size-S branching programs of some fixed width, using a (uniformly generated) universal branching program $U(f, m)$ of some size $T = T(S) = \text{poly}(S)$, and invoking Theorem 3 with the family $\mathcal{U}_{\text{BP}} = \{U_m(\cdot) = U(\cdot, m)\}$ yields a suitable homomorphic commitment. For this instantiation, following Item 1c of Theorem 3 we set

$$\gamma = \gamma_{\text{BP}} := s\sqrt{m} \cdot w^{O(1)} \cdot T.$$

In both instantiations we let the "dummy" function $d \in \mathcal{F}'$ be the trivial function that always outputs zero, leaving the family $\mathcal{F} = \mathcal{F}' \setminus \{d\}$ of all *nontrivial* size-S circuits (or branching programs) as the one supported by the scheme. (There is no need to suport the trivial function, since its output is fixed.)

Compressing commitments and proofs. Finally, for functions with outputs in $\{0,1\}^L$ for some $L \leq n$, we can reduce the sizes of the commitments and proofs by a factor of Lw.[20] Define $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ to be the identity matrix $\mathbf{I} \in \mathbb{Z}_q^{L \times L}$ padded with $n - L$ all-zero rows, and let $\mathbf{e} \in \{0,1\}^{Lw}$ be as in Item 3 of Theorem 3. Then any commitment $\mathbf{C}_m \in \mathbb{Z}_q^{n \times Lw}$ can be replaced by the single column vector $\mathbf{c}_m = \mathbf{C}_m \cdot \mathbf{e} \in \mathbb{Z}_q^n$, and any proof $\mathbf{R}_{m,f} \in \mathbb{Z}^{m \times Lw}$ can be replaced by $\mathbf{r}_{m,f} = \mathbf{R}_{m,f} \cdot \mathbf{e} \in \mathbb{Z}^m$. We then redefine $\mathsf{Verify}(\mathbf{A}, \mathbf{c}^*, f, \mathbf{y} \in \{0,1\}^L, \mathbf{r}^*)$ to accept if $\|\mathbf{r}^*\| \leq \gamma' := \gamma \|\mathbf{e}\|_1 \leq \gamma L w$ and $\mathbf{c}^* = \mathbf{A}_f \mathbf{r}^* + \binom{\mathbf{y}}{\mathbf{0}}$. This works because $\|\mathbf{r}_{m,f}\| \leq \|\mathbf{R}_{m,f}\| \cdot \|\mathbf{e}\|_1$, and $\mathsf{Encode}(\mathbf{y}) \cdot \mathbf{e} = \mathbf{M}\mathbf{y} = \binom{\mathbf{y}}{\mathbf{0}}$. We note that the above-described combinability property is also preserved.

Lemma 2. *For the instantiations and parameters given above, Construction 3 is a correct functional commitment scheme with authority.*

Proof. Let $m \in \mathcal{M}$ and $f \in \mathcal{F}$ be arbitrary, and let:

- $(cp = \mathbf{C}, vp = \mathbf{A}, ek = (\mathbf{C}, \mathbf{A}, \mathbf{T})) \leftarrow \mathsf{Setup}(1^n)$,
- $ok_f = \mathbf{R}_f \leftarrow \mathsf{Extract}(ek, f) = \mathsf{SamplePre}(\mathbf{T}, \mathbf{A}_f, \mathbf{C} - \mathsf{Encode}(f), s)$
  (note that $f \neq d$ and s is large enough, so $\mathsf{SamplePre}$ works on these arguments), and
- $(\mathbf{C}_m, \mathbf{R}_{m,f}) = \mathsf{Open}(\mathbf{C}, m, \mathbf{R}_f) = \mathsf{Eval}(U_m, \mathbf{C}, \mathbf{R}_f)$
  (note that $\mathbf{C}_m = \mathsf{Commit}(\mathbf{C}, m) = \mathsf{Eval}(U_m, \mathbf{C})$ by definition of Eval).

[20] For $L > n$, we can simply treat the function as the concatenation of multiples functions, and extract keys and generate proofs for each of them individually.

We show that $\mathsf{Verify}(\mathbf{A}, \mathbf{C}_m, f, y = f(m), \mathbf{R}_{m,f})$ accepts. By definition of $\mathsf{Extract}$ and Item 3 of Theorem 1, we have $\|\mathbf{R}_f\| \leq s\sqrt{m}$ and

$$\mathbf{C} = \mathbf{A}_f \mathbf{R}_f + \mathsf{Encode}(f),$$

so by the correctness of $U_m(\cdot) = U(\cdot, m)$ and of $\mathsf{Eval}(U_m, \mathbf{C}, \mathbf{R}_f)$ (Theorem 3) we have

$$\mathbf{C}_m = \mathbf{A}_f \mathbf{R}_{m,f} + \mathsf{Encode}(U_m(f)) = \mathbf{A}_f \mathbf{R}_{m,f} + \mathsf{Encode}(y),$$

as required. In addition, $\|\mathbf{R}_{m,f}\| \leq \gamma$ because $\|\mathbf{R}_{m,f}\|/\|\mathbf{R}_f\|$ is bounded as given in Item 1 of Theorem 3, so Verify accepts.

Finally, for the compressed variant with commitment $\mathbf{c}_m := \mathbf{C}_m \mathbf{e}$ and proof $\mathbf{r}_{m,f} := \mathbf{R}_{m,f} \cdot \mathbf{e}$, we have

$$\mathbf{c}_m = \mathbf{A}_f \mathbf{r}_{m,f} + \mathsf{Encode}(y) \cdot \mathbf{e} = \mathbf{A}_f \mathbf{r}_{m,f} + \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix}$$

and has norm $\|\mathbf{r}\| \leq \|\mathbf{R}_{m,f}\| \cdot \|\mathbf{e}\|_1 \leq \gamma \|\mathbf{e}\|_1 = \gamma'$, so Verify accepts. □

Theorem 4. *For the instantiations and parameters given above, Construction 3 satisfies selective function binding (Definition 5) if normal-form $SIS_{n,q,\bar{m},\beta}$ is hard for sufficiently large $\beta = O(\gamma w \sqrt{m})$ [or for the compressed variant, $\beta = O(\gamma' \sqrt{m})$ where $\gamma' = \gamma \|\mathbf{e}\|_1$ for the special short vector $\mathbf{e}$ used for compression].*

More specifically, for any adversary $\mathcal{A}$ against the selective function binding of the scheme that makes at most $Q = Q(n)$ queries to its $\mathsf{Extract}$ oracle, there is a normal-form $SIS_{n,q,\bar{m},\beta}$ adversary $\mathcal{B}$ for which

$$\mathbf{Adv}^{SIS}(\mathcal{B}) \geq \mathbf{Adv}^{sfa}(\mathcal{A}) - (Q+1) \cdot \mathrm{negl}(n),$$

and whose running time is that of $\mathcal{A}$ plus a small polynomial in n.

Due to space constraints, the proof of Theorem 4 is deferred to the full version. (It has many similarities with the proof of Theorem 2.)

References

ABB10. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572. 2010.

AFV11. S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*. 2011.

AP14. J. Alperin-Sheriff and C. Peikert. Faster bootstrapping with polynomial error. In *CRYPTO*, pages 297–314. 2014.

BBB+18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy*, pages 315–334. 2018.

BDFG20. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Recursive zk-SNARKs from any additive polynomial commitment scheme. Cryptology ePrint Archive, Report 2020/1536, 2020. `https://eprint.iacr.org/2020/1536`.

BdM93. J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures (extended abstract). In *EUROCRYPT*, volume 765, pages 274–285. 1993.

BFS20. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT*, pages 677–706. 2020.

BGJS16. S. Badrinarayanan, V. Goyal, A. Jain, and A. Sahai. Verifiable functional encryption. In *ASIACRYPT*, pages 557–587. 2016.

BGV11. S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *CRYPTO*, pages 111–131. 2011.

BV14. Z. Brakerski and V. Vaikuntanathan. Lattice-based FHE as secure as PKE. In *ITCS*, pages 1–12. 2014.

CCH+19. R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, R. D. Rothblum, and D. Wichs. Fiat-Shamir: from practice to theory. In *STOC*, pages 1082–1090. 2019.

CF13. D. Catalano and D. Fiore. Vector commitments and their applications. In *PKC*, pages 55–72. 2013.

CFG+20. M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In *ASIACRYPT*, pages 3–35. 2020.

CGMA85. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *FOCS*, pages 383–395. 1985.

CHKP10. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012. Preliminary version in Eurocrypt 2010.

CPSZ18. A. Chepurnoy, C. Papamanthou, S. Srinivasan, and Y. Zhang. Edrax: A cryptocurrency with stateless transaction validation. Cryptology ePrint Archive, Report 2018/968, 2018. `https://eprint.iacr.org/2018/968`.

CRFM08. D. Catalano, M. D. Raimondo, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. *IEEE Transactions on Information Theory*, 57(4):2488–2502, 2011. Preliminary version in EUROCRYPT 2008.

DN94. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. In *CRYPTO*, volume 839, pages 234–246. 1994.

GPV08. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. 2008.

GSW13. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92. 2013.

GVW15. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, pages 469–477. 2015.

GW11. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108. 2011.

HW15. P. Hubácek and D. Wichs. On the communication complexity of secure function evaluation with long output. In *ITCS*, pages 163–172. 2015.

Kil92. J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, pages 723–732. 1992.

Kus18. J. Kuszmaul. Verkle trees, 2018. Unpublished manuscript, available at `https://math.mit.edu/research/highschool/primes/materials/2018/Kuszmaul.pdf`.

KZG10. A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, pages 177–194. 2010.

Lis05. M. D. Liskov. Updatable zero-knowledge databases. In *ASIACRYPT*, pages 174–198. 2005.

LM06. V. Lyubashevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. In *ICALP (2)*, pages 144–155. 2006.

LM19. R. W. F. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In *CRYPTO*, pages 530–560. 2019.

LP20. H. Lipmaa and K. Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. In *ASIACRYPT*, pages 686–716. 2020.

LPR13. V. Lyubashevsky, C. Peikert, and O. Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54. 2013.

LRY16. B. Libert, S. C. Ramanna, and M. Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In *ICALP*, pages 30:1–30:14. 2016.

LW15. V. Lyubashevsky and D. Wichs. Simple lattice trapdoor sampling from a broad class of distributions. In *PKC*, pages 716–730. 2015.

LY10. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC*, pages 499–517. 2010.

Mer87. R. C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378. 1987.

Mic94. S. Micali. CS proofs. In *FOCS*, pages 436–453. 1994.

Mic02. D. Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16(4):365–411, 2007. Preliminary version in FOCS 2002.

MP12. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. 2012.

MR04. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004.

MRK03. S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. In *FOCS*, pages 80–91. 2003.

PR06. C. Peikert and A. Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC*, pages 145–166. 2006.

PS19. C. Peikert and S. Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, pages 89–114. 2019.

PSTY13. C. Papamanthou, E. Shi, R. Tamassia, and K. Yi. Streaming authenticated data structures. In *EUROCRYPT*, pages 353–370. 2013.

PW08. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. *SIAM J. Comput.*, 40(6):1803–1844, 2011. Preliminary version in STOC 2008.

SBZ01. R. Steinfeld, L. Bull, and Y. Zheng. Content extraction signatures. In *ICISC*, pages 285–304. 2001.

TAB+20. A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In *SCN*, Lecture Notes in Computer Science, pages 45–64. 2020.

Xag13. K. Xagawa. Improved (hierarchical) inner-product encryption from lattices. In *PKC*, pages 235–252. 2013.

Yam16. S. Yamada. Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In *EUROCRYPT*, pages 32–62. 2016.