Succinct Vector, Polynomial, and Functional Commitments from Lattices

Hoeteck Wee  
NTT Research and ENS, Paris  
wee@di.ens.fr

David J. Wu  
∗UT Austin  
dwu4@cs.utexas.edu

Abstract

Vector commitment schemes allow a user to commit to a vector of values $x \in \{0,1\}^\ell$ and later, open up the commitment to a specific set of positions. Both the size of the commitment and the size of the opening should be succinct (i.e., polylogarithmic in the length $\ell$ of the vector). Vector commitments and their generalizations to polynomial commitments and functional commitments are key building blocks for many cryptographic protocols.

We introduce a new framework for constructing non-interactive lattice-based vector commitments and their generalizations. A simple instantiation of our framework yields a new vector commitment scheme from the standard short integer solution (SIS) assumption that supports private openings and large messages. We then show how to use our framework to obtain the first succinct functional commitment scheme that supports openings with respect to arbitrary bounded-depth Boolean circuits. In this scheme, a user commits to a vector $x \in \{0,1\}^\ell$, and later on, open the commitment to any function $f(x)$. Both the commitment and the opening are non-interactive and succinct: namely, they have size $\text{poly}(\lambda, d, \log \ell)$, where $\lambda$ is the security parameter and $d$ is the depth of the Boolean circuit computing $f$. Previous constructions of functional commitments could only support constant-degree polynomials, or require a trusted online authority, or rely on non-falsifiable assumptions. The security of our functional commitment scheme is based on a new falsifiable family of “basis-augmented” SIS assumptions (BASIS) we introduce in this work.

We also show how to use our vector commitment framework to obtain (1) a polynomial commitment scheme where the user can commit to a polynomial $f \in \mathbb{Z}_q[x]$ and subsequently open the commitment to an evaluation $f(x) \in \mathbb{Z}_q$; and (2) an aggregatable vector (resp., functional) commitment where a user can take a set of openings to multiple indices (resp., function evaluations) and aggregate them into a single short opening. Both of these extensions rely on the same BASIS assumption we use to obtain our succinct functional commitment scheme.

1 Introduction

Vector commitment schemes [Mer87, CFM08, LY10, CF13] allow a user to commit to a vector of values $x \in \{0,1\}^\ell$ and subsequently, open up the commitment to a specific set of positions. Both the commitment and the openings should be succinct (i.e., have size that scales polylogarithmically with the vector length $\ell$) and non-interactive.1 There has recently been tremendous interest and progress in the design and application of vector commitments, and even a “Vector Commitment Research Day” [Res22]. Starting from the classic vector commitment scheme of Merkle [Mer87] based on collision-resistant hash functions, we now have a broad range of algebraic constructions from pairing-based assumptions [LY10, KZG10, CF13, LRY16, LM19, TAB+20, GRWZ20] as well as assumptions over groups of unknown order (e.g., RSA groups or class groups) [CF13, LM19, CFG+20, AR20, TXN20]. We refer to [Nit21] for a survey of recent schemes. As a primitive, vector commitment schemes have found numerous applications to verifiable outsourced databases [BGV11, CF13], cryptographic accumulators [CF13], pseudonymous credentials [KZG10], and to blockchain protocols [RMC17, CPSZ18, BBF19]. Moreover, the generalization of vector commitments to polynomial commitments [KZG10] has emerged as a key building block in many recent (random-oracle) constructions of succinct

1Part of this work was done while visiting NTT Research.

2We discuss interactive commitments (as well as constructions in the random oracle model) in Section 1.3.
non-interactive arguments of knowledge (SNARKs) [MBKM19, CHM*20, GWC19, BDFG21, BFS20, COS20] having various appealing properties (e.g., universal or transparent setup, recursive composability, and more).

In this work, we focus on two themes in the study of vector commitments where progress has been more limited: (1) post-quantum constructions based on lattices [PSTY13, LLNW16, PPS21, ACL+22, FSZ22]; and (2) functional commitments, a generalization of vector commitments that supports openings to various functions on the committed values [LRY16, LP20, PPS21, BNO21, ACL+22]. There are good technical reasons for the limited progress on these two fronts. First, many of the techniques developed for vector commitments crucially exploit algebraic structure in pairing and RSA/class groups that do not naturally extend to the lattice setting. Second, pairing and RSA/class groups only support limited homomorphic capabilities.

1.1 Our Results

In this work, we introduce a general framework for constructing lattice-based vector commitments that simultaneously encapsulates recent lattice-based vector commitment schemes [PPS21, ACL+22] and enables us to achieve stronger functionality and security properties. As we describe below, our framework readily generalizes to also yield polynomial commitments, functional commitments, and aggregatable commitments from (falsifiable) lattice-based assumptions.

A new family of SIS assumptions. The security of our schemes relies on a new “basis-augmented” family of short integer solution (SIS) assumptions we introduce in this work. We refer to our basis-augmented SIS assumption as the BASIS assumption (Assumption 3.2). The basic version of our assumption (denoted BASIS\text{std}) suffices for constructing standard vector commitments and is implied by the standard SIS assumption. The structured version of the assumption (denoted BASIS\text{struct}) we need for our extensions has a similar flavor as the k-SIS-like assumptions introduced in [ACL+22] for constructing lattice-based succinct arguments (c.f., Section 6). While the BASIS\text{struct} assumption is not a standard lattice-based assumption, it is a falsifiable assumption [Nao03]. We view our assumption as a “q-type” lattice assumption and at a conceptual level, it shares a similar flavor as the q-type assumptions used in the pairing-based world for constructing vector commitments [CF13] and polynomial commitments [KZG10].

Vector commitments with private opening. An immediate consequence of our framework is a vector commitment scheme that supports private openings. In this setting, a user can commit to a vector \( \mathbf{x} \in \{0, 1\}^t \) with a short commitment \( \sigma \) and then open \( \sigma \) to an index-value pair \((i, x_i)\) with a short opening \( \pi_i \). We say the vector commitment scheme supports private openings if the commitment \( \sigma \) and any collection of openings \( \{(i, x_i, \pi_i)\}_{i \in S} \) reveal no additional information about \( x_j \) for any \( j \notin S \). Notably and in contrast to previous lattice-based vector commitment schemes [PPS21, ACL+22], our scheme also does not impose any restrictions on the magnitude of the entries of \( \mathbf{x} \) (the vectors can be arbitrary elements of \( \mathbb{Z}_q^\ell \) and the commitment as well as the opening are vectors over \( \mathbb{Z}_q \)). Previous lattice-based schemes [PPS21, ACL+22] require that the components of \( \mathbf{x} \) be small and this property was essential for both correctness and security.

Our vector commitment scheme has the same efficiency properties as the earlier scheme of Peikert et al. [PPS21] which did not support private openings and was limited to a small message space. Our scheme provides the same functionality (e.g., support for “stateless updates”) and like the scheme of [PPS21], security can be based on the standard SIS assumption. Thus, relative to [PPS21], our framework achieves private openings and supports a large message space with essentially no overhead.

We could alternatively obtain a lattice-based vector commitment by instantiating Merkle’s classic construction [Mer87] with a lattice-based collision-resistant hash function (e.g., Ajtai’s hash function from SIS [Ajt96, GGH96]). Our vector commitment scheme improves upon this generic approach in two main ways: (1) we support (bounded) stateless updates like [PPS21] (where a user can update a commitment to a vector \( \mathbf{x} \) into a commitment to a vector \( \mathbf{x}' \) given only knowledge of the difference \( \mathbf{x}' - \mathbf{x} \) and not the entirety of \( \mathbf{x} \) or \( \mathbf{x}' \)); and (2) we can support private openings directly. It is possible to extend Merkle hashing to support private openings via zero-knowledge proofs, but this would either need non-black-box use of cryptography or require interaction, random oracles, or correlation-intractable hash functions [CCH*19, PS19]. More broadly, as we illustrate below, our algebraic scheme serves as a stepping stone for realizing polynomial and functional commitment schemes (for which we crucially exploit algebraic structure).
**Functional commitments.** A functional commitment [GVW15, LRY16] is a generalization of a vector commitment with the property that given a commitment to an input \( x \in \{0, 1\}^\ell \), one can then construct an opening \( \pi_f \) to \( y = f(x) \), for some function \( f \). The basic binding property of the commitment scheme says that the adversary cannot come up with openings \( \pi_f \) and \( \pi'_f \) that open \( \sigma \) to different values \( y \neq y' \) with respect to the same function \( f \). The efficiency requirements are that the size of the commitment and the opening should be sublinear in both the size of the function \( f \) and the length of the input \( x \). Previously, Peikert et al. [PPS21] showed how to construct functional commitments for bounded-depth Boolean circuits in an online model where a central trusted authority issues opening keys for functions \( f \), with security based on the standard SIS assumption. Albrecht et al. [ACL*22] subsequently showed how to construct functional commitments for constant-degree polynomials from new variants of the SIS assumption in the standard setting without an online authority. Earlier pairing-based functional commitments could only support linear functions [LRY16] or sparse polynomials [LP20]. Functional commitments can also be obtained generically by combining a vanilla vector commitment (e.g., a Merkle tree [Mer87]) with a succinct non-interactive argument of knowledge (for NP). However, existing constructions of SNARKs (for NP) either rely on making non-falsifiable assumptions [GW11] or working in idealized models.

Our vector commitment framework directly yields a succinct functional commitment scheme for all bounded-depth Boolean circuits in the standard offline model without an authority and from falsifiable assumptions. The size of the commitment and the openings are \( \poly(\lambda, d, \log \ell) \), where \( \lambda \) is a security parameter, \( d \) is the depth of the Boolean circuit computing \( f : \{0, 1\}^\ell \rightarrow \{0, 1\} \), and \( \ell \) is the length of the input. Security relies on the new non-standard, but falsifiable, BASIS\textsuperscript{struct} assumption we introduce in this work (with a sub-exponential noise bound). Notably, this is the first succinct functional commitment scheme for general circuits from a falsifiable assumption, and answers an open question posed by Peikert et al. [PPS21]. Our construction can be viewed as a succinct analog of the homomorphic commitments and signatures introduced by [GSW13, GVW15].

**Polynomial commitments.** In a polynomial commitment [KZG10], a user can commit to a polynomial \( f \in \mathbb{Z}_q[x] \) over \( \mathbb{Z}_q \) and later open to an evaluation \( f(x) \) at any point \( x \in \mathbb{Z}_q \). A polynomial commitment can be viewed as a succinct commitment to the vector of evaluations of \( f \) on all inputs \( x \in \mathbb{Z}_q \). While a polynomial commitment can be built from a succinct functional commitment for Boolean circuits, this incurs a \( \poly(\log q) \) overhead to encode the polynomial evaluation over \( \mathbb{Z}_q \) as a Boolean circuit and also relies on the BASIS\textsuperscript{struct} assumption with a sub-exponential noise bound. In this work, we show that a simple adaptation of our succinct functional commitments to the setting of linear functions directly gives a polynomial commitment over \( \mathbb{Z}_q \). Notably, this construction can be based on our BASIS\textsuperscript{struct} assumption with only a polynomial noise bound. This is the first polynomial commitment scheme from lattices based on falsifiable assumptions.

An important feature of our framework that enables the direct construction of polynomial commitments is that it natively supports values over \( \mathbb{Z}_q \). Previous lattice-based vector commitments [PPS21, ACL*22] required that the committed value \( x \) and the opened value \( f(x) \) be “small,” and moreover, that the modulus \( q \) scale with the norm of the output (i.e., \( f(x) \)) when computed over the integers. This is not suitable when constructing polynomial commitments directly, as the size of \( f(x) \) computed over the integers scales with the degree of \( f \). Correspondingly, if the modulus \( q \) scales linearly with the degree of \( f \), then the resulting scheme is no longer succinct. The ability to directly work over the entirety of \( \mathbb{Z}_q \) is an appealing property of our new framework.

**Aggregatable commitments.** A simple modification to our basic vector commitment scheme yields a scheme that supports aggregation. We say a vector commitment scheme is aggregatable [BBF19, CFG*20] if given a commitment \( \sigma \) along with a set of openings \( \pi_1, \ldots, \pi_t \) to indices \( i_1, \ldots, i_t \in [\ell] \) and values \( x_{i_1}, \ldots, x_{i_t} \), there is an efficient aggregation algorithm that outputs a short aggregate opening \( \tilde{\pi} \) that validates the full set of values \( \{(i_j, x_{i_j})\}_{j \in [t]} \). The requirement is that the size of \( \tilde{\pi} \) scale sublinearly, or better yet, polylogarithmically with \( t \). Aggregatable commitments immediately imply subvector commitments [LM19] (i.e., a vector commitment scheme that supports batch openings to a set of indices \( S \subseteq [\ell] \)). Our framework yields an aggregatable commitment scheme for short messages from the same falsifiable BASIS\textsuperscript{struct} assumption used to construct succinct functional commitments. This is the first aggregatable

---

2The homomorphic commitments from [GSW13, GVW15] are non-succinct; in particular, the size of the commitment scales linearly with the input length \( \ell \).
commitment scheme from lattice assumptions without relying on general-purpose succinct arguments [ACL+22] or batch arguments [CJ21, DGKV22], and answers another open question posed by Peikert et al. [PPS21].

A limitation of our aggregatable commitment is that it only satisfies same-set binding, which guarantees that for every subset of indices $S \subseteq [\ell]$, the adversary can only open to a single set of values. However, there is still the possibility that an adversary could open the commitment to different sets $S$ and $T$ that are inconsistent (i.e., $x_i = 0$ with respect to $S$ while $x_i = 1$ with respect to $T$).\(^3\) Constructing aggregatable commitments that satisfy the stronger notion of different-set binding directly from falsifiable lattice-based assumptions is an interesting open problem.

The same techniques we use to construct aggregatable vector commitments also applies to our succinct functional commitment scheme, and we obtain an aggregatable functional commitment scheme from the same underlying hardness assumption. In this setting, a user can take openings $\pi_1, \ldots, \pi_\ell$ for function-value pairs $(f_1, y_1), \ldots, (f_\ell, y_\ell)$ and aggregate the openings into a single short opening $\pi$ that validates all $\ell$ function-value pairs and where the size of the aggregated opening scales polylogarithmically with $\ell$.

**Summary.** Similar to previous lattice-based vector commitments [PPS21, ACL+22], we rely on a structured reference string in all of our constructions. We refer to the structured reference string as a common reference string (CRS). To summarize, our new lattice-based vector commitment framework yields the following constructions:

- A vector commitment scheme with private openings based on the standard SIS assumption with polynomial noise bound (Corollary 3.17). For vectors of dimension $\ell$, the size of the commitment is $O(\lambda(\log \lambda + \log \ell))$ and the size of an opening is $O(\lambda(\log^2 \lambda + \log^2 \ell))$.\(^4\) The size of the CRS is $\ell^2 \cdot \text{poly} (\lambda, \log \ell)$.

- A succinct functional commitment scheme supporting all bounded-depth Boolean circuits from the BASIS\textsubscript{struct} assumption with a sub-exponential noise bound (Corollary 4.8). A variant of this construction supports private openings under a weaker notion of target binding (Corollary 4.33). For both constructions, to support functions on $\ell$-bit inputs and computable by Boolean circuits of depth $d$, the sizes of the commitment and openings are $\text{poly}(\lambda, d, \log \ell)$. The size of the CRS is $\ell^2 \cdot \text{poly}(\lambda, d, \log \ell)$.

- A polynomial commitment (for polynomials of a priori bounded degree) under the BASIS\textsubscript{struct} assumption with a polynomial noise bound (Corollary 4.21). To support polynomials of degree up to $d$ over $\mathbb{Z}_q$ (where $q = \text{poly}(\lambda)$), the sizes of the commitment and openings are $\text{poly}(\lambda, d, \log \ell)$. The size of the CRS is $d^2 \cdot \text{poly}(\lambda, \log d)$.

- An aggregatable vector commitment scheme (over a small message space) based on the BASIS\textsubscript{struct} assumption with polynomial noise bound (Corollary 5.11). The sizes of the commitment, openings, and CRS match those above for our vanilla vector commitment.

- An aggregatable functional commitment scheme for all bounded-depth Boolean circuits from the BASIS\textsubscript{struct} assumption used to obtain succinct functional commitments (Corollary 5.23). To support aggregating $T$ openings for functions on $\ell$-bit inputs and computable by Boolean circuits of depth $d$, the sizes of the commitment and opening are $\text{poly}(\lambda, d, \log \ell, \log T)$. The size of the CRS is $(\ell^2 + T) \cdot \text{poly}(\lambda, d, \log \ell, \log T)$. In the random oracle model, we can reduce the CRS size to $\ell^2 \cdot \text{poly}(\lambda, d, \log \ell)$ and support an arbitrary polynomial number of aggregations.

### 1.2 Technical Overview

In this section we provide a general overview of our new framework for constructing vector commitments from lattices as well as the family of basis-augmented SIS assumptions (BASIS) we use to prove hardness. In the following description, for a matrix $A \in \mathbb{Z}_q^{n \times m}$ and a target vector $t \in \mathbb{Z}_q^n$, we write $A^{-1}(t)$ to denote a random variable $x \in \mathbb{Z}_q^m$ whose entries are distributed according to a discrete Gaussian conditioned on $Ax = t$. Sampling $x \leftarrow A^{-1}(t)$ can be done efficiently given a trapdoor for $A$ (see Section 2.1). Here, we will use the Micciancio-Peikert gadget trapdoors [MP12]; namely, a matrix $R$ is a gadget trapdoor for $A$ if $R$ is short and $AR = G$, where $G = I_n \otimes g^T$ is the gadget matrix and $g^T = [1, 2, \ldots, 2^{\log q}]$.

\(^3\)Note though that if the commitment is honestly-generated, then same-set binding implies different-set binding; see Remark 5.13.

\(^4\)We note that these bounds match the base construction of Peikert et al. [PPS21]. While [PPS21, Figure 1] reports that their scheme has $O(\log \ell)$-size openings (ignoring the security parameter $\lambda$), the construction itself [PPS21, Construction 1] has $O(\log^* \ell)$-size openings.
A general framework for constructing vector commitments. We begin by describing a general framework for constructing lattice-based vector commitments that encapsulates the recent schemes from [PPS21, ACL^*22]:

- The common reference string (CRS) specifies a collection of \( \ell \) matrices \( A_1, \ldots, A_{\ell} \in \mathbb{Z}_q^{m \times m} \) and \( \ell \) vectors \( t_1, \ldots, t_{\ell} \in \mathbb{Z}_q^m \) along with some auxiliary input \( \text{aux} := \{ A_i^{-1}(t_j) \}_{i \neq j} \).

- The commitment to a vector \( x = (x_1, \ldots, x_{\ell}) \in \{0, 1\}^\ell \) is a vector \( c \leftarrow \sum_{i \in [\ell]} x_i t_i \in \mathbb{Z}_q^m \).

- An opening to index \( i \in [\ell] \) and value \( x_i \in \{0, 1\} \) is a short vector \( v_i \in \mathbb{Z}_q^m \) such that
  \[
  c = A_i v_i + x_i t_i. \tag{1.1}
  \]
  The honest opening is computed as \( v_i \leftarrow \sum_{j \neq i} x_j A_j^{-1}(t_j) \).

Correctness follows by inspection:

\[
A_i v_i + x_i t_i = \sum_{j \neq i} x_j A_i A_j^{-1}(t_j) + x_i t_i = \sum_{i \in [\ell]} x_i t_i = c.
\]

For binding, we require that it is hard to find a short vector \( z \in \mathbb{Z}_q^m \) such that \( A_i z = t_i \) for any \( i \in [\ell] \) given the components in the CRS. Next, we explain how the schemes PPS from [PPS21] and MatrixACLMT from [ACL^*22] fall into this framework.

- In PPS\(_1\), the matrices \( A_i \mapsto \mathbb{Z}_q^{n \times m} \) and vectors \( t_i \mapsto \mathbb{Z}_q^n \) are independent and uniformly random for all \( i \in [\ell] \). Binding in turn is based on the standard SIS assumption.

- In MatrixACLMT, they sample uniformly random vectors \( u_i \mapsto \mathbb{Z}_q^n \), a matrix \( A \mapsto \mathbb{Z}_q^{n \times m} \), and invertible matrices \( W_i \mapsto \mathbb{Z}_q^{n \times n} \) for each \( i \in [\ell] \). Then, they set \( A_i \leftarrow W_i A, t_i \leftarrow W_i u_i \). In this case, \( A_i^{-1}(t_j) = A^{-1}(W_j^{-1}W_i u_i) \). Binding is based on a new assumption which stipulates that it is hard to find a short vector \( z \in \mathbb{Z}_q^m \) where \( Az = u_i \) for any \( i \in [\ell] \) given the CRS. The authors of [ACL^*22] then show how to leverage the extra structure arising from the correlated \( A_i \)'s to obtain a functional commitment scheme for constant-degree polynomials as well as a preprocessing succinct non-interactive argument (SNARG) for NP.

Before describing our approach, we describe two limitations of these instantiations:

- **Small message space**: In both the PPS\(_1\) and the MatrixACLMT instantiations of this framework, both correctness and security require that the entries of the vector \( x = (x_1, \ldots, x_{\ell}) \) be small. This is because the verification relation is checking that the opening \( v_i = \sum_{j \neq i} x_j A_j^{-1}(t_j) \) is small. Thus, correctness requires that each \( x_j \) be small. Moreover, in the proof of binding, the reduction algorithm takes a commitment \( c \) along with two openings \( (x_i, v_i), (x'_i, v'_i) \) to derive a solution to SIS or a related problem. The existing reductions require that the difference \( (x_i - x'_i) \) be small (in order to derive a short solution).

- **Uniform target vectors.** In both the PPS\(_1\) and MatrixACLMT constructions, the target vectors \( t_i \) are essentially random vectors. This is important for ensuring that \( A_i^{-1}(t_j) \) does not leak a trapdoor for \( A_i \), which would immediately break binding. Using structured target vectors could enable additional functionality. For instance, in Remark 6.1, we show that instantiating MatrixACLMT with structured targets can be used to support functional openings. Unfortunately, this instantiation also leaks a trapdoor for \( A_i \), and is insecure.

The approach we take in this work avoids these limitations and allows us to construct vector commitments with a large message space as well as support new capabilities like polynomial and functional openings.

---

By PPS\(_1\), we refer to the the base scheme from [PPS21, Construction 1]; they also present a second tree-based scheme that uses PPS\(_1\) as a building block.

The authors of [ACL^*22] describe their scheme in the ring setting. We write MatrixACLMT to denote one possible translation from the ring setting to the integer setting. Note that there are other ways to translate their scheme to the integer setting such as sampling \( W_i \mapsto \mathbb{Z}_q^{m \times m} \) and then setting \( A_i \leftarrow AW_i \).
Our approach. We consider the same verification relation \( c = A_i v_i + x_i t_i \) from Eq. (1.1), but take a completely different approach for computing the commitment \( c \) and the openings \( v_i \): we sample a random tuple \((v_1, \ldots, v_\ell, c)\) that simultaneously satisfies the verification relation for all \( i \in [\ell] \). As in the previous verification relation, we require that the openings \( v_1, \ldots, v_\ell \) are short. The commitment \( c \) can have large entries. In our particular setting, we write \( c = G \hat{c} \) where \( \hat{c} \in \mathbb{Z}_q^\ell \) is a short vector. Using the gadget matrix \( G \) will be important in the security analysis. Then, Eq. (1.1) corresponds to the relation \( G \hat{c} = A_i v_i + x_i t_i \), or equivalently, \( A_i v_i - G \hat{c} = -x_i t_i \) for all \( i \in [\ell] \). We can express these \( \ell \) relations as a linear system:

\[
\begin{bmatrix}
A_1 & -G \\
\vdots & \vdots \\
A_\ell & -G \\
\end{bmatrix}
\begin{bmatrix}
v_1 \\
\vdots \\
v_\ell \\
\end{bmatrix}
= 
\begin{bmatrix}
-x_1 t_1 \\
\vdots \\
-x_\ell t_\ell \\
\end{bmatrix}
\tag{1.2}
\]

Our goal now is to sample a random short tuple \((v_1, \ldots, v_\ell, \hat{c})\) that satisfies Eq. (1.2). This can be done by giving out a random trapdoor for the matrix \( B_\ell \):

\[
B_\ell :=
\begin{bmatrix}
A_1 & -G \\
\vdots & \vdots \\
A_\ell & -G \\
\end{bmatrix}
\tag{1.3}
\]

Using \( B_\ell \), we can sample a random short preimage \((v_1, \ldots, v_\ell, \hat{c})\) satisfying Eq. (1.2). This yields the commitment \( c = G \hat{c} \) and the openings \( v_1, \ldots, v_\ell \). In our construction, we set the target vector \( t_i \) to the first basis vector \( e_1 = [1, 0, \ldots, 0]^T \) for all \( i \in [\ell] \). We now make the following observations:

- Binding: To argue that the scheme is binding, we require that it is hard to find a short vector \( z \) where \( A_i z = 0 \) for any \( i \in [\ell] \) even given the (related) matrix \( B_\ell \) and a trapdoor for \( B_\ell \). Here, \( A_\ell \) denotes \( A_i \) with the first row removed. We refer to assumptions of this type as "basis-augmented SIS" (BASIS) assumptions (Assumption 3.2). As we sketch below (and show formally in Theorem 3.4), when \( A_1, \ldots, A_\ell \in \mathbb{Z}_q^{n \times m} \) are random, this instantiation of the BASIS assumption holds under the standard SIS assumption. We refer to this instance of the BASIS assumption with random matrices as BASIS\text{rand}. Now, to argue binding, we observe that an adversary that breaks binding is able to come up with an index \( i \in [\ell] \), short vectors \( v, v' \in \mathbb{Z}_q^\ell \) and values \( x, x' \in \mathbb{Z}_q \) such that \( c = A_i v + x e_1 = A_i v' + x' e_1 \). This means that

\[
A_i (v - v') = (x' - x) e_1.
\]

As long as \( x' - x \neq 0 \), \( v - v' \neq 0 \), and so \( v - v' \) is a SIS solution to \( A_i \). Observe that this analysis does not impose any restriction on the magnitude of \( x' - x \). This means our construction naturally supports committing to arbitrary vectors over \( \mathbb{Z}_q^\ell \) as opposed to vectors with small entries.\(^7\) We give the formal reduction to the BASIS\text{rand} assumption in Theorem 3.11.

- Private openings. A vector commitment scheme supports private openings if the commitment \( c \) and any collections of openings \( \{(i, x_i, v_i)\}_{i \in S} \) completely hide the values \( x_j \) for \( j \notin S \). Since we sample the commitment \( c \) and the openings \( v_i \) jointly in our approach, it is straightforward to argue (by appealing to properties of discrete Gaussians; see Lemma 2.7) that the commitment \( c \) is statistically close to uniform over \( \mathbb{Z}_q^n \) and each opening \( v_i \) is statistically close to the distribution \( A_i^{-1} (c - x_i t_i) \). Thus our scheme provides statistically private openings out of the box.

Taken together, this yields a vector commitment from standard SIS that supports statistically private openings and commitments to arbitrary vectors over \( \mathbb{Z}_q^\ell \). We give the full description and analysis in Section 3.

\(^7\)As discussed earlier, previous vector commitments [PPS21, ACL22] based on SIS or its generalizations needed to assume small inputs for both correctness and security.
Reducing BASIS\textsubscript{rand} to standard SIS. As described above, the binding property of our vector commitment relies on the BASIS assumption, which says that SIS with respect to \( A_i \) (i.e., \( A_i \) with the first row removed) is hard even given the related matrix \( B_i \) from Eq. (1.3) and a trapdoor for \( B_i \). As we show in Theorem 3.4, when the matrices \( A_1, \ldots, A_f \) are uniform and independent, this assumption (BASIS\textsubscript{rand}) reduces to the standard SIS assumption in a straightforward way. Here, we provide a sketch of the reduction. For ease of exposition, we show that SIS with respect to \( A_i \) (as opposed to \( A_1 \)) is hard given a trapdoor for \( B_i \). We also describe the approach for the case \( i = 1 \), and refer to Theorem 3.4 for the full analysis.

The idea is simple: we set \( A_1 \) to be the SIS challenge and sample matrices \( A_2, \ldots, A_f \) together with trapdoors \( R_2, \ldots, R_f \) (i.e., \( A_i R_i = G \)). Let \( B_i \) be \( B_i \) with the first column block removed (i.e., the column block containing \( A_1 \)). Then, using \( R_2, \ldots, R_f \) we can construct a trapdoor \( \tilde{B}_i \) for \( B_i \) (i.e., \( B_i \tilde{R}_i = G_{n\ell} = I_{n\ell} \otimes g^\ell \)):

\[
\tilde{B}_i = \begin{bmatrix}
0 & \cdots & 0 & -G \\
A_2 & \cdots & \cdots & -G \\
\vdots & \ddots & \ddots & \vdots \\
A_f & \cdots & \cdots & -G \\
\end{bmatrix} \quad \text{and} \quad \tilde{R}_i = \begin{bmatrix}
-R_2 \\
-R_3 \\
\vdots \\
-R_f \\
-I & 0 & \cdots & 0 \\
\end{bmatrix}
\]

Using standard trapdoor extension techniques [ABB10a, ABB10b, CHKP10, MP12], we can extend \( \tilde{R}_i \) to obtain a trapdoor \( R_i \) for \( B_i \). This yields a BASIS\textsubscript{rand} instance (i.e., comprised of the matrix \( A_1 \), the matrix \( B_i \), and the trapdoor \( R_i \)). Thus, an adversary that breaks the BASIS\textsubscript{rand} assumption implies an adversary that breaks SIS (with comparable parameters). We give the formal analysis in Theorem 3.4.

Functional commitments using structured A\textsubscript{i}. Instantiating our framework with uniform \( A_i \in \mathbb{Z}_q^{n\times m} \) (as in PPS\textsubscript{1}), we obtain a vector commitment scheme with private openings and supporting large messages from the standard SIS assumption. If we instead use a structured set of matrices \( A_i \) as in MatrixACLMT, we obtain functional commitments, polynomial commitments, and aggregatable commitments.

We start by describing our functional commitment scheme. Our starting point is to consider the main verification relation from Eq. (1.1) and generalize it in two ways: (1) we replace the matrices \( A_1, \ldots, A_f \in \mathbb{Z}_q^{n\times m} \) with structured matrices; and (2) we consider a matrix extension of the verification relation. In particular, we first sample \( A \in \mathbb{Z}_q^{n\times m} \). Then, for each \( i \in [f] \), we sample an invertible matrix \( W_i \in \mathbb{Z}_q^{m\times n} \) and set \( A_i \leftarrow W_i A \). We now consider a matrix analog of the verification relation from Eq. (1.1) where each target vector \( t_i \) is replaced with the matrix \( W_i G \) (this choice will be helpful for supporting functional openings). Our matrix verification relation is now

\[
C = A_i V_i + x_i W_i G. \tag{1.4}
\]

Our goal now is to sample a tuple \((V_1, \ldots, V_f, C)\) that satisfy Eq. (1.4) for all \( i \in [f] \) and where \( V_1, \ldots, V_f \in \mathbb{Z}_q^{m\times m} \) are short. As before, the commitment \( C \) can be large and we specifically define it to be \( C = G\hat{C} \), where \( \hat{C} \in \mathbb{Z}_q^{m\times m} \) is short. This way, we can sample \( \hat{C} \) using an analogous trapdoor sampling procedure as before. Specifically, the trapdoor for the same matrix \( B_i \) from Eq. (1.3) allows us to jointly sample short openings \( V_1, \ldots, V_f \) along with a matrix \( \hat{C} \) that satisfy Eq. (1.4):

\[
B_i \begin{bmatrix} V_1 \\ \vdots \\ V_f \\ \hat{C} \end{bmatrix} = \begin{bmatrix} A_1 & -G \\ \vdots & \ddots & \ddots & \vdots \\ A_f & -G \end{bmatrix} \begin{bmatrix} V_1 \\ \vdots \\ V_f \\ \hat{C} \end{bmatrix} = \begin{bmatrix} -x_i W_i G \\ \vdots \\ -x_f W_f G \end{bmatrix}. \tag{1.5}
\]

By construction, for all \( i \in [f] \), we have that \( A_i V_i - G\hat{C} = -x_i W_i G \), or equivalently, \( C = G\hat{C} = A_i V_i + x_i W_i G \) and Eq. (1.4) holds. We now show that this directly extends to yield a succinct functional commitment. Since \( A_i = W_i A \) and \( W_i \) is invertible, we can rewrite Eq. (1.4) as

\[
W_i^{-1} C = A V_i + x_i G,
\]

where \( V_i \) is short. Readers familiar with the homomorphic encryption scheme of Gentry et al. [GSW13] or the homomorphic signature scheme of Gorbunov et al. [GVW15] may recognize that \( W_i^{-1} C \) is an encryption of \( x_i \) under
randomness $V_i$ or that $V_i$ is a signature on $x_i$ under the verification key $W_i^{-1}C$. Thus, we can use the same lattice-based homomorphic evaluation machinery [GSW13, BGG*14] to homomorphically compute an opening to $f(x)$ for an arbitrary Boolean circuit $f : \{0, 1\}^f \to \{0, 1\}$.

In slightly more detail, let $\tilde{C} = [W_1^{-1}C | \cdots | W_f^{-1}C]$ and $\tilde{V} = [V_1 | \cdots | V_f]$. Then,

$$A\tilde{V} = A[V_1 | \cdots | V_f] = [W_1^{-1}C - x_1G | \cdots | W_f^{-1}C - x_fG] = \tilde{C} - x^T \otimes G.$$

Using the homomorphic evaluation techniques from [GSW13, BGG*14], there exists a short matrix $H_{\tilde{C}, f, x}$ that depends on $\tilde{C}, f$, and $x$ such that

$$(\tilde{C} - x^T \otimes G) \cdot H_{\tilde{C}, f, x} = \tilde{C}_f - f(x) \cdot G,$$

where $\tilde{C}_f$ is a matrix that can be efficiently computed from $\tilde{C}$ and $f$. To open $C$ to a function $f$, the user computes $\tilde{V}_f \leftarrow \tilde{V} \cdot H_{\tilde{C}, f, x}$. To verify a candidate value $y \in \{0, 1\}$ with respect to a function $f$ and commitment $C$, the verifier first computes $\tilde{C}_f$ from $(C, W_1, \ldots, W_f, f)$ and then checks that $\tilde{V}_f$ is short and moreover,

$$A\tilde{V}_f = \tilde{C}_f - y \cdot G.$$

For correctness, observe that

$$A\tilde{V}_f = A\tilde{V}H_{\tilde{C}, f, x} = (\tilde{C} - x^T \otimes G) \cdot H_{\tilde{C}, f, x} = \tilde{C}_f - f(x) \cdot G.$$

For binding, we require that SIS is hard with respect to $A$ even given the matrix $B_f$ and a trapdoor for $B_f$. We refer to this instance of the BASIS assumption with structured $A_i$’s as $\text{BASIS}_{\text{struct}}$. Since the matrices $A_i$ that comprise $B_f$ are now correlated, we do not know how to reduce $\text{BASIS}_{\text{struct}}$ to the standard SIS assumption. Nonetheless, $\text{BASIS}_{\text{struct}}$ is a falsifiable assumption under which we obtain a succinct functional commitment for all bounded-depth Boolean circuits. This is the first succinct functional commitment for general circuits from a falsifiable assumption. We provide the full description in Section 4 and a comparison to previous succinct functional commitments in Table 1.

**Functional commitments with private openings.** Using the approach from [GVW15] for constructing context-hiding homomorphic signatures [GVW15], we can easily extend our functional commitment scheme above to support private openings (i.e., where the commitment $C$ and the opening $\tilde{V}_f$ reveals nothing more about the input $x$ other than the value $f(x)$). We sketch the approach here. Let $C$ be a commitment to $x$ and let $\tilde{V}_f = \tilde{V} \cdot H_{\tilde{C}, f, x}$ be the opening computed as described above. Then, define the matrix $D_f$ to be

$$D_f = [A \mid \tilde{C}_f + (f(x) - 1) \cdot G] = [A \mid A\tilde{V}_f + (2f(x) - 1) \cdot G].$$

Since $\tilde{V}_f$ is short and $2f(x) - 1 \in \{-1, 1\}$, the matrix $[A \tilde{V}_f]$ is a trapdoor for $D_f$. We now include a random target $u \in \mathbb{Z}_q^n$ as part of the CRS, and define the opening to be a random short vector $v_f$ where $D_f v_f = u$. The honest prover samples $v_f$ using the trapdoor for $D_f$ (derived from $\tilde{V}_f$). To check an opening $v_f$ is a valid opening to a value $y \in \{0, 1\}$ with respect to a function $f$ and commitment $C$, the verifier computes $\tilde{C}_f$ from $(C, W_1, \ldots, W_f, f)$ as before, defines the matrix $D_f = [A \mid \tilde{C}_f + (y - 1) \cdot G]$, and finally, checks that $D_f v_f = u$. To argue that $v_f$ hides all information about $x$ other than what is revealed by $f(x)$, observe that the matrix $D_f$ depends only on $f(x)$ and not $x$. Thus, given a trapdoor for $A$ (which can be extended into a trapdoor for $D_f$ for all $f$), and the value $f(x)$, we can sample a short vector $v_f$ such that $D_f v_f = u$ whose distribution is statistically close to the real opening. This latter procedure only depends on $f(x)$ and not $x$, so hiding follows. While this construction is hiding, we do not know how to show that it is binding; however, it does satisfy the weaker notion of target binding where binding holds for all honestly-generated commitments. We provide the full details and analysis in Section 4.2.

**Polynomial commitments.** We can obtain a polynomial commitment over $\mathbb{Z}_q$ via a simple adaptation of our functional commitment. The starting point is to construct a functional commitment scheme for linear functions on
Table 1: Summary of succinct functional commitments. For each scheme, we report the size of the CRS as a function of the security parameter $\lambda$ and the input length $\ell$. We say that a scheme supports “fast verification” if after an input-independent preprocessing step, the verification running time is sublinear in $\ell$. In all schemes, the size of the commitment and the openings are polylogarithmic in the input length $\ell$.

$\mathbb{Z}_q^\ell$ (as opposed to a function on the binary domain $\{0, 1\}^\ell$). We first consider linear functions with small coefficients. Let $z \in \{0, 1\}^\ell$ be a vector and define the linear function $f_z(x) := z^T x$. We use the same commitment and opening structure as in our functional commitment. Namely, a commitment $C$ and the openings $V_1, \ldots, V_{\ell}$ for an input $x$ satisfy $AV_i = W_i^{-1} C - x_i G$, where $x_i \in \mathbb{Z}_q$ now. Observe that

$$\sum_{i \in [\ell]} z_i AV_i = \sum_{i \in [\ell]} z_i W_i^{-1} C - \sum_{i \in [\ell]} z_i x_i G = \sum_{i \in [\ell]} z_i W_i^{-1} C - (z^T x) \cdot G.$$

Thus, $V_z = \sum_{i \in [\ell]} z_i V_i$ is an opening to the function $f_z$. Here, we need $z \in \{0, 1\}^\ell$ to be short so $V_z$ is short. To extend to arbitrary linear functions over $\mathbb{Z}_q^\ell$ (rather than short ones), we rely on standard binary decomposition and blow up the vector dimension by a factor of $O(\log q)$. Namely, to commit to a vector $x$, the user now commits to $x \otimes g^z$, and to open to a linear function $f_z$ where $z \in \mathbb{Z}_q^\ell$, the user constructs an opening with respect to $f_z^{-1}(z)$. This yields a functional commitment scheme for linear functions over $\mathbb{Z}_q^\ell$.

As observed by Libert et al. [LRY16], a functional commitment scheme for linear functions over $\mathbb{Z}_q^\ell$ directly implies a polynomial commitment over $\mathbb{Z}_q$ for polynomials of degree up to $d = \ell - 1$. Namely, a commitment to a polynomial $f \in \mathbb{Z}_q[x]$ of degree $d$ is a vector commitment to the coefficients of $f$. To open the commitment to a point $x \in \mathbb{Z}_q$, the user constructs a linear opening with respect to the evaluation vector $[1, x, x^2, \ldots, x^d]$. For this to work, it is critical that our functional commitment for linear functions over $\mathbb{Z}_q^\ell$ supports committing to and opening to arbitrary $\mathbb{Z}_q$ values (and not just small values). Thus, we obtain a polynomial commitment scheme where the commitment size and the opening size is $\text{poly}(\lambda, \log d)$. We provide the full details in Section 4.1.

### Aggregatable commitments

Another application of using structured matrices $A_i$ is it immediately gives an aggregatable commitment. As before, we instantiate our framework with $A_i = W_i A$. We also sample target vectors $u_1, \ldots, u_\ell \stackrel{\$}{\in} \mathbb{Z}_q^n$ and include them as part of the CRS. To commit to an input $x \in \mathbb{Z}_q^n$, we sample $(v_1, \ldots, v_{\ell}, c)$ where

\[
B_\ell = \begin{bmatrix} v_1 \\ \vdots \\ v_{\ell} \\ c \end{bmatrix} = \begin{bmatrix} A_1 & -G \\ \vdots & \vdots \\ A_\ell & -G \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_{\ell} \\ c \end{bmatrix} = \begin{bmatrix} -x_1 W_i u_1 \\ \vdots \\ -x_\ell W_i u_{\ell} \end{bmatrix}.
\]
Let \( c = \mathbf{G} \mathbf{e} \). Then, for all \( i \in [t] \), \( A_i \mathbf{v}_i - c = -x_i \mathbf{W}_i \mathbf{u}_i \), or equivalently, \( \mathbf{W}_i^{-1} c = A \mathbf{v}_i + x_i \mathbf{u}_i \). Observe now that this scheme immediately supports aggregation: for any set \( S \subseteq [t] \),

\[
\sum_{i \in S} \mathbf{W}_i^{-1} c = A \sum_{i \in S} \mathbf{v}_i - \sum_{i \in S} x_i \mathbf{u}_i.
\]

Thus, \( \sum_{i \in S} \mathbf{v}_i \) is an opening to all of the indices in \( S \). We show in Section 5 that under the same BASIS\text{struct} assumption (i.e., SIS is hard with respect to \( A \) given \( B_i \) and a trapdoor for \( B_i \)), this scheme satisfies “same-set binding.” This means no efficient adversary can open a commitment \( c \) to different sets of values \( \{(i, x_i)\}_{i \in S} \) and \( \{(i, x'_i)\}_{i \in T} \) for the same set \( S \). Unlike our vector commitment construction, the security of our aggregatable construction only holds when the input vector \( \mathbf{x} \) is short (i.e., our reduction to the BASIS\text{struct} assumption in Theorem 5.6 constructs an SIS solution whose norm scales with the magnitude of the opened values).

Our aggregatable vector commitment scheme does not satisfy the stronger notion of “different-set binding.” This means an efficient adversary may be able to construct a commitment \( c \) along with valid openings \( \{(i, x_i)\}_{i \in S} \) and \( \{(i, x'_i)\}_{i \in T} \) to (distinct) sets \( S \) and \( T \), respectively, such that \( x_i = 0 \) and \( x'_i = 1 \). Indeed in Remark 5.12, we describe an explicit attack where an adversary can use the trapdoor for \( B_i \) to (heuristically) obtain a trapdoor for the matrix \( [\mathbf{W}_i^{-1} A | \mathbf{W}_i^{-1} A] \) whenever \( S \neq T \) and where \( \mathbf{W}_S = \sum_{i \in S} \mathbf{W}_i^{-1} \) and \( \mathbf{W}_T = \sum_{i \in T} \mathbf{W}_i^{-1} \). Knowledge of this trapdoor allows an adversary to construct a valid opening to \( \{(i, x_i)\}_{i \in S} \) and \( \{(i, x'_i)\}_{i \in T} \) for any choice of \( x_i, x'_i \).

Conceptually, our approach for constructing an aggregatable vector commitment scheme is to replace the fixed target value \( x_i \mathbf{e}_1 \) from our basic vector commitment with random linear combinations of \( \{x_i\}_{i \in S} \) (where the coefficients of the random linear combination are the vectors \( \{u_i\}_{i \in S} \)). A similar approach was used for aggregating pairing-based signatures in [BDN18] and for aggregating openings (to constant-degree polynomials) in [ACL+22].

**Aggregating functional commitments.** The same aggregation technique applies to our succinct functional commitment scheme. Recall the functional commitment verification relation from Eq. (1.6): \( \mathbf{A} \mathbf{V}_f = \hat{C}_f - y \cdot \mathbf{G} \). Here \( \mathbf{V}_f \) is the opening, \( \hat{C}_f \) is a function of the commitment \( C \) and the function \( f \), and \( y \) is the value. To support aggregating up to \( t \) openings, we include random vectors \( \mathbf{u}_1, \ldots, \mathbf{u}_t \leftarrow \mathbb{Z}_q^n \) in the CRS. Then, given a collection of openings \( (f_1, y_1, \mathbf{V}_1), \ldots, (f_t, y_t, \mathbf{V}_t) \) where the functions \( f_1, \ldots, f_t \) are sorted in lexicographic order, we define the aggregate opening to be \( \mathbf{v} = \sum_{i \in [t]} \mathbf{V}_i \cdot \mathbf{G}^{-1}(\mathbf{u}_i) \). The new verification relation is then

\[
\sum_{i \in [t]} \mathbf{C}_f \cdot \mathbf{G}^{-1}(\mathbf{u}_i) = A \mathbf{v} - \sum_{i \in [t]} y_i \mathbf{u}_i.
\]

Similar to the case with aggregatable vector commitments, we can argue “same-function binding,” where no efficient adversary can open a commitment \( C \) to two different sets of values \( (y_1, \ldots, y_t) \neq (y'_1, \ldots, y'_t) \) with respect to the same set of functions \( (f_1, \ldots, f_t) \). We provide the full analysis in Section 5.1.

**Understanding the BASIS assumption.** The BASIS assumptions we introduce in this work enable a number of new constructions of vector commitments and their generalizations. While the basic version BASIS\text{and} that suffices for vector commitments can be reduced to the standard SIS assumption (Theorem 3.4), the more general version BASIS\text{struct} with structured matrices does not. Nonetheless, the BASIS\text{struct} assumption is still falsifiable and thus, yields the first succinct functional commitments and polynomial commitments from falsifiable lattice assumptions. We invite cryptanalysis of our new family of SIS assumptions and are also optimistic that the assumption as well as our general methodology will be helpful for realizing new lattice-based cryptographic primitives.

In Section 6, we compare the BASIS\text{struct} assumption to similar assumptions made in [ACL+22]. We show a close connection between the two families of assumptions. We can also view the BASIS assumptions as a new type of “\( q \)-type” assumption in the lattice-based setting (where the size of the assumption grows with the input dimension).

### 1.3 Related Work and Concurrent Work

In this section, we describe some closely-related notions of functional commitments as well as a comparison to two concurrent construction of functional commitments for Boolean circuits [BCFL22, dCP23].
Interactive functional commitments. Functional commitments have also been extensively studied in the interactive model. In these settings, there is typically an interactive opening procedure between the committer and the verifier. Ishai et al. [IKO07] introduced interactive functional commitments for linear function, and subsequently, Bitansky and Chiesa [BC12] extended it to interactive functional commitments. In both cases, these were used to construct (interactive) succinct arguments without relying on probabilistically-checkable proofs (PCPs). Alternatively, using PCPs or their generalization to interactive oracle proofs [BCS16], we can also construct a functional commitment from any collision-resistant hash function via Kilian’s interactive succinct argument [Kil92], which can then be made non-interactive in the random oracle model [Mic00]. Our focus in this work is on non-interactive vector and functional commitments in the plain model (i.e., without random oracles).

Homomorphic commitments. Homomorphic commitments [GSW13, GVW15] also allow a user to commit to an input $x$ and later on, open up the commitment to an arbitrary function evaluation $f(x)$. The key difference is that there are no succinctness requirements on a homomorphic commitment. In particular, the size of the commitment in previous constructions [GSW13, GVW15] scaled linearly with the input length. On the flip side, homomorphic commitments can be constructed from standard lattice assumptions.

Concurrent works. Recently, two concurrent works [dCP23, BCFL22] introduced new constructions of functional commitments. The authors of [dCP23] introduce a vector commitment with a short CRS (i.e., logarithmic in the input dimension) from the standard SIS assumption. In their construction, the CRS is a uniform random string rather than a structured reference string. They then generalize their construction to obtain a “dual” functional commitment scheme for (bounded-depth) Boolean circuits from the standard SIS assumption. Their functional commitment does not support succinct openings. Namely, the size of the opening in their construction scales either with the length of

$\begin{array}{|c|c|c|c|c|c|c|c|c|c|}
\hline
\text{Scheme} & |\text{crs}| & |\sigma| & |\pi| & T_{\text{Commit}} & T_{\text{Open}} & T_{\text{Verify}} & \text{Assumption} & \text{FV} & \text{TR} & \text{AS} \\
\hline
\text{[dCP23]} & |y| & 1 & |\pi| & |F| & |\sigma| & |\pi| & \text{SIS} & ✗ & ✗ & ✗ \checkmark \\
\text{[BCFL22]} & w^5 & 1 & 1 & |x| & |F| & |\sigma| & twin-k-M-ISIS & ✗ & ✗ & ✗ \\
\hline
\text{Construction 4.2} & |x|^2 & 1 & 1 & |x| & |F| & |\sigma| & BASIS_{\text{struct}} & ✗ & ✗ & ✗ \\
\hline
\end{array}$

*Can be made adaptively-secure via complexity leveraging (and in some cases, relying on sub-exponential hardness of SIS). The use of complexity leveraging increases all parameter sizes by $|y|^{1/\varepsilon}$ for a constant $\varepsilon > 0$.
†For comparison purposes, we consider a variant with a deterministic commitment and opening procedure where Commit runs in quasi-linear time. We refer to Remark 4.12 for more details.

Table 2: Comparison to concurrent works [BCFL22, dCP23] on lattice-based succinct functional commitments for general Boolean circuits. To establish a common basis for comparison between schemes, we fix a bivariate function $F(x, y)$ with a single output. We consider the setting where the user commits to input $x$ and opens to the value $F(x, y)$. In [BCFL22] and in our construction, the user commits to $x$ and opens to the function $F_y(\cdot) := F(\cdot, y)$. In the “dual” notion of [dCP23], the user would commit to the function $F_{\pi}(\cdot) := F(x, \cdot)$ and open to the value $y$. We write $w$ to denote the width, $d$ to denote the depth, and $|F|$ to denote the size of the Boolean circuit computing $F$. Our construction and [dCP23] impose an a priori bound on $d$ whereas [BCFL22] imposes an a priori bound on $w$. For each scheme, we report the size of the common reference string $crs$, the size of the commitment $\sigma$, the size of the opening $\pi$, and the running times $T_{\text{Commit}}, T_{\text{Open}}, T_{\text{Verify}}$ of the commit, opening, and verification algorithms, respectively. For simplicity, we suppress poly($\lambda, d$) terms as well as polylogarithmic terms in $|x|, |y|$, and $|F|$. We say a scheme supports “fast verification” (FV) if after an input-independent preprocessing step, the verification time is sublinear in $|x|, |y|$, and $|F|$. We also indicate whether the scheme has a transparent setup (TR) and whether it is adaptively secure (AS). We say the commitment is adaptively secure if in the binding security game, the adversary can choose $x, y$ after seeing the common reference string, and that it is selectively secure if the adversary has to choose either $x$ or $y$ beforehand.

In a “dual” functional commitment scheme, the commitment is to a function $f$ and the opening is to a point $x$. In the version we are considering, the commitment is to an input $x$ and the opening is to a function $f$. In the case of functional commitments for circuits, we can translate between these two notions using universal circuits, but this transformation will incur an a priori bound on the description length of the function.
the input $x$ (when committing to a function $f$ and opening to an input $x$) or the description length of the function $f$ (when committing to an input $x$ and opening to a function $f$). In this work, we consider the setting where both the commitment and the opening are succinct (i.e., polylogarithmic in both $|x|$ and $|f|$). However, we rely on a stronger (but still falsifiable) version of the SIS assumption.

The authors of [BCFL22] provide new pairing-based and lattice-based constructions of functional commitments (with succinct commitments and openings) for arithmetic circuits of a priori bounded width. Like our scheme, the size of the openings in their lattice-based instantiation grows with the depth of the circuit. Their scheme also relies on new, non-standard, but falsifiable generalizations of SIS (specifically, variants of the [ACL*22] family of assumptions used to obtain lattice-based preprocessing succinct arguments). The functional commitments we introduce in this work support arbitrary Boolean circuits (without a width constraint), and similarly, the sizes of the commitment and the openings scale with the depth of the computation. The structured version of our basis-augmented SIS assumption ($\text{BASIS}_{\text{strm}}$) is conceptually similar to the [ACL*22] family of assumptions (the latter imposes even more structure), and we provide a more detailed discussion in Section 6. We provide a comparison with the lattice-based functional commitments from [dCP23, BCFL22] in Table 2.

2 Preliminaries

We write $\lambda$ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. For integers $a, b \in \mathbb{N}$, we write $[a, b]$ to denote the set $\{a, a + 1, \ldots, b\}$. For a positive integer $q \in \mathbb{N}$, we write $\mathbb{Z}_q$ to denote the integers modulo $q$. We use bold uppercase letters to denote matrices (e.g., $A, B$) and bold lowercase letters to denote vectors (e.g., $u, v$). We use non-boldface letters to refer to their components: $v = (v_1, \ldots, v_n)$. For matrices $A_1, \ldots, A_t \in \mathbb{Z}_q^{n \times m}$, we write $\text{diag}(A_1, \ldots, A_t) \in \mathbb{Z}_q^{m \times nt}$ to denote the block diagonal matrix with blocks $A_1, \ldots, A_t$ along the main diagonal (and 0 elsewhere).

We write $\text{poly}(\lambda)$ to denote a fixed function that is $O(\lambda^c)$ for some $c \in \mathbb{N}$ and $\text{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. For functions $f = f(\lambda), g = g(\lambda)$, we write $g \geq O(f)$ to denote that there exists a fixed function $f' : \mathbb{N} \to \mathbb{N}$ such that $g(\lambda) > f'(\lambda)$ for all $\lambda \in \mathbb{N}$. We say an event occurs with overwhelming probability if its complement occurs with negligible probability. An algorithm is efficient if it runs in probabilistic polynomial time in its input length. We say that two families of distributions $\mathcal{D}_1 = \{\mathcal{D}_{\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if no efficient algorithm can distinguish them with non-negligible probability, and we say they are statistically indistinguishable if the statistical distance $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ is bounded by a negligible function $\text{negl}(\lambda)$.

Min-entropy. We recall some basic definitions on min-entropy. Our definitions are taken from [DRS04, GVW15]. For a (discrete) random variable $X$, we write $H_\infty(X) = -\log(\max_x \Pr[X = x])$ to denote its min-entropy. For two (possibly correlated) discrete random variables $X$ and $Y$, we define the average min-entropy of $X$ given $Y$ to be $H_\infty(X | Y) = -\log(\sum_y \max_x \Pr[X = x | Y = y])$. The optimal probability of an unbounded adversary guessing $X$ given the correlated value $Y$ is $2^{-H_\infty(X | Y)}$.

2.1 Lattice Preliminaries

In this section, we recall some standard notions from lattice-based cryptography. Unless otherwise noted, we use the $\ell_\infty$-norm for vectors and matrices. For a vector $u$, we write $\|u\| := \max_i |u_i|$, and for a matrix $A$, we write $\|A\| = \max_{i,j} |A_{i,j}|$. For $k \in \mathbb{N}$, we write $I_k \in \mathbb{Z}_q^{k \times k}$ to denote the $k$-by-$k$ identity matrix. We now recall the short integer solution (SIS) problem [Ajt96] as well as its inhomogeneous variant:

Assumption 2.1 (Short Integer Solution [Ajt96]). Let $\lambda$ be a security parameter and $n = n(\lambda), m = m(\lambda), q = q(\lambda)$, and $\beta = \beta(\lambda)$ be lattice parameters. We say the short integer solution problem $\text{SIS}_{n, m, q, \beta}$ holds if for all efficient adversaries $\mathcal{A}$, \[ \Pr \left[ A x = 0 \quad \text{and} \quad 0 < \|x\| \leq \beta \quad : \quad \begin{bmatrix} A \in \mathbb{Z}_q^{n \times m}, \\ x \leftarrow \mathcal{A}(1^\lambda, A) \end{bmatrix} \right] = \text{negl}(\lambda). \]

We also define the inhomogeneous SIS (ISIS) assumption where the target $0$ in the above assumption is replaced by a uniform random vector $y \sim \mathbb{Z}_q^n$. When $m = \text{poly}(n, \log q)$ and for sufficiently-large $q \geq \beta \cdot \text{poly}(n)$, hardness of the
SIS and ISIS problems can be based on approximating certain worst-case lattice problems on \(n\)-dimensional lattices to within a \(\beta \cdot \text{poly}(n)\) factor [Ajt96, MR04, GPV08].

**Lattices.** Let \(B \in \mathbb{R}^{n \times n}\) be a full-rank matrix (over \(\mathbb{R}\)). Then, the \(n\)-dimensional lattice \(L\) generated by \(B\) is \(L = L(B) = B \cdot Z^n = \{Bz : z \in Z^n\}\). For a vector \(c \in \mathbb{R}^n\) and a lattice \(L \subset \mathbb{R}^n\), we write \(c + L = \{c + x : x \in L\}\) to denote the coset of \(L\) associated with \(c\). For a lattice \(L\), we define the dual lattice \(L^* = \{w \in \mathbb{R}^n : \forall x \in L, w^T x \in \mathbb{Z}\}\). Finally, if \(A \in \mathbb{Z}_q^{n \times m}\) for integers \(n, m, q\), we define \(L^+(A) = \{x \in \mathbb{Z}_q^m : Ax \equiv 0 \mod q\}\). We will also use the leftover hash lemma and a simple corollary of it:

**Lemma 2.2** (Leftover Hash Lemma [HILL99]). Let \(n, m, q\) be lattice parameters and suppose \(m \geq 2n \log q\). Then, the statistical distance between the following distributions is at most \(2^{-\alpha n}\):

\[
\{(A, Ax) : A \in \mathbb{Z}_q^{n \times m}, x \in \{0, 1\}^m \} \quad \text{and} \quad \{(A, y) : A \in \mathbb{Z}_q^{n \times m}, y \in \mathbb{Z}_q^m\}.
\]

**Corollary 2.3** (Column-Space of Random Matrix [GPV08, Lemma 5.1]). Let \(n, q\) be lattice parameters and take any \(m \geq 2n \log q\). Then, for all but a \(q^{-n}\) fraction of matrices \(A \in \mathbb{Z}_q^{n \times m}\), the columns of \(A\) generate \(\mathbb{Z}_q^n\).

**Discrete Gaussians over lattices.** For a Gaussian width parameter \(s \in \mathbb{R}^+\) and a dimension \(n \in \mathbb{N}\), we write \(\rho_s : \mathbb{R}^n \rightarrow \mathbb{R}\) to denote the Gaussian function \(\rho_s(x) := \exp(-\pi \|x\|^2 / s^2)\). Typically, the input dimension will be implicit. For a lattice coset \(c + L\), we write \(\rho_s(c + L) = \sum_{x \in L} \rho_s(c + x)\) to denote the Gaussian mass associated with the coset. We write \(D_{2n,s} \equiv D_{2n,s}^2\) to denote the discrete Gaussian distribution over \(\mathbb{Z}^n\) with parameter \(s \in \mathbb{R}^+\); namely, \(D_{2n,s}(x) := \rho_s(x)/\rho_s(\mathbb{Z}^n)\). For a matrix \(A \in \mathbb{Z}_q^{n \times t}\), and a vector \(v \in \mathbb{Z}_q^t\), we write \(A^{-1}(v)\) to denote the random variable \(x \leftarrow D_{t,s}(\mathbb{Z}_q^t)\) conditioned on \(Ax = v \mod q\). We extend \(A^{-1}\) to matrices by applying \(A^{-1}\) to each column of the input.

**The smoothing parameter.** We also recall the notion of the smoothing parameter introduced by Micciancio and Regev [MR04]. For an \(n\)-dimensional lattice \(L\) and a positive real number \(\epsilon > 0\), the smoothing parameter \(\eta_\epsilon(L)\) is the smallest real value \(s > 0\) such that \(\rho_{s/\eta_\epsilon}(L^*) \leq 1 + \epsilon\). We now state some lemmas on the smoothing parameter:

**Lemma 2.4** (Smoothing Parameter [MR04, Lemma 4.4, implicit]). Let \(L\) be an \(n\)-dimensional lattice. Then, for all \(\epsilon \in (0, 1), s \geq \eta_\epsilon(L), \) and all cosets \(c + L, \rho_s(c + L) \in \left[\frac{1}{1 + \epsilon}, 1\right] \cdot \rho_s(L)\).

**Lemma 2.5** (Smoothing Parameter for \(L^+(A)\)). Let \(n, m, q\) be lattice parameters where \(m \geq 2n \log q\) and \(q\) is prime. Let \(t = \text{poly}(n)\) be a dimension. Then, there exists a negligible function \(\epsilon(n) = \text{negl}(n)\) such that for all \(A_1, \ldots, A_t \in \mathbb{Z}_q^{n \times m}\),

\[
\Pr[\eta_\epsilon(L^+(\text{diag}(A_1, \ldots, A_t))) \leq \log(tm) : A_1 \leftarrow \mathbb{Z}_q^{n \times m}] \geq 1 - q^{-n}.
\]

**Proof.** The proof follows from the same argument as [GPV08, Lemma 5.3]. Let \(Z = \{x \in \mathbb{Z}^m : \|x\| \leq q/4\}\) be the set of integer-valued vectors with norm at most \(q/4\). Let \(Z_1 = \{x \in \mathbb{Z}^m : \|x\| \leq q/4\}\). Take any nonzero \(0 \neq s \in \mathbb{Z}_q^n\).

Then,

\[
\Pr[A_1 \bmod q \in Z_1 : A_1 \leftarrow \mathbb{Z}_q^{n \times m}] \leq \frac{(q/2)^m}{q^m} = 2^{-m} \leq q^{-2n}.
\]

By a union bound,

\[
\Pr[\text{there exists } 0 \neq s \in \mathbb{Z}_q^n \text{ where } A_1 \bmod q \in Z_1 : A_1 \leftarrow \mathbb{Z}_q^{n \times m}] \leq q^{-n}.
\]

Correspondingly, since \(A = \text{diag}(A_1, \ldots, A_t)\), this means that

\[
\Pr[\text{there exists } 0 \neq s \in \mathbb{Z}_q^{nt} \text{ where } A \bmod q \in Z : A \leftarrow \mathbb{Z}_q^{nt \times m}] \leq q^{-n}.
\]

(2.1)

Let \(L(A) = \{y \in \mathbb{Z}^{tm} : y = A^\top x \mod q \text{ for some } x \in \mathbb{Z}_q^{nt}\}\). Eq. (2.1) shows that with probability \(1 - q^{-n}\) over the choice of \(A_1\), the length of the shortest vector \(\lambda_1^+(L(A))\) in \(L(A)\) satisfies \(\lambda_1^+(L(A)) \geq q/4\). Next, we note that
Thus, we can write \( \eta_c(L^+(A)) \leq \frac{\omega(\sqrt{\log(\ell m)})}{\lambda_c^A((L^+(A))^\perp)} \leq \frac{\omega(\sqrt{\log(\ell m)})}{1/q \cdot \lambda_c^A((L^+(A))^\perp)} \leq 4 \cdot \omega(\sqrt{\log(\ell m)}) \).

Taking the \( \omega(\sqrt{\log(\ell m)}) \) function to be \( \log(\ell m)/4 \) yields the claim. \( \square \)

**Lemma 2.6** (Gaussian Tail Bound [MP12, Lemma 2.6, adapted]). Let \( B \in \mathbb{R}^{n \times n} \) be a full-rank basis and let \( L = L(B) \) be the \( n \)-dimensional lattice generated by \( B \). Suppose \( s \geq \eta_c(L) \) for some \( \epsilon \in (0,1) \). Then for all \( c \in \mathbb{R}^n \),

\[
\Pr \left[ \| v \| > \sqrt{n}s : v \leftarrow D_{c+L,s} \right] \leq 2^{-n} \cdot \frac{1+\epsilon}{1-\epsilon}.
\]

Our analysis will make use of the following lemma on the marginal distributions of Gaussian-distributed preimages. The proof of this is essentially implicit in the preimage sampling algorithm of Gentry et al. [GPV08], but we provide a full proof here for completeness:

**Lemma 2.7** (Discrete Gaussian Preimages). Let \( n, q \) be lattice parameters and take any \( m \geq 2n \log q \). Take matrices \( A \in \mathbb{Z}_q^{n \times m} \) and \( B \in \mathbb{Z}_q^{n \times t} \) where \( t = \text{poly}(n, \log q) \). Suppose the columns of \( A \) generate \( \mathbb{Z}_q^n \). Let \( C = [A \mid B] \in \mathbb{Z}_q^{n \times (m+t)} \). Then for all target vectors \( t \in \mathbb{Z}_q^n \) and all width parameters \( s \geq \eta_c(L^+(A)) \) for some \( \epsilon(n) = \text{negl}(n) \), the statistical distance between the following two distributions is \( \text{negl}(n) \):

\[
\{ v : v \leftarrow C_s^{-1}(t) \} \quad \text{and} \quad \{ \frac{v_1}{v_2} : v_2 \leftarrow D^L_{Z_q,v_1} v_1 \leftarrow A_s^{-1}(t-Bv_2) \}.
\]

**Proof.** Consider \( v \leftarrow C_s^{-1}(t) \). In the following, we parse \( v = \frac{v_1}{v_2} \). Consider the distribution \( D \) of \( v \):

\[
D(v) = \frac{\rho_s(v_1)}{\sum_{v' : C_v^t = v} \rho_s(v')} = \frac{\rho_s(v_1) \rho_s(v_2)}{\sum_{v_2' \in \mathbb{Z}_q^n} \sum_{v_1' : Av_1' = -Bv_2'} \rho_s(v_1') \rho_s(v_2')}
\]

Since the columns of \( A \) generate \( \mathbb{Z}_q^n \) for every \( v_2' \in \mathbb{Z}_q^t \), there exists some \( u_{v_2} \in \mathbb{Z}_q^m \) such that \( Au_{v_2} = t - Bv_2' \). Let \( Q \) be the denominator in Eq. \( 2.2 \). Then,

\[
Q = \sum_{v_2' \in \mathbb{Z}_q^t} \sum_{v_1' : Av_1' = -Bv_2'} \rho_s(v_1') \rho_s(v_2') = \sum_{v_2' \in \mathbb{Z}_q^t} \rho_s(v_2') \rho_s(u_{v_2} + L^+(A)).
\]

Since \( s \geq \eta_c(L^+(A)) \), Lemma 2.4 says that for all \( v_2' \in \mathbb{Z}_q^t \),

\[
\rho_s(u_{v_2} + L^+(A)) \in [\frac{1-\epsilon}{1+\epsilon}] \cdot \rho_s(L^+(A)).
\]

Thus, we can write \( Q = \delta_Q \cdot \rho_s(Z_q^t) \rho_s(L^+(A)) \) for some \( \delta_Q \in [\frac{1-\epsilon}{1+\epsilon},1] \).

- Consider first the marginal distribution \( D_2 \) of \( v_2' \):

\[
D_2(v_2') = \sum_{v_1 : Av_1 = -Bv_2} D \left( \frac{v_1}{v_2} \right) = \sum_{v_1 : Av_1 = -Bv_2} \frac{\rho_s(v_1) \rho_s(v_2)}{\delta_Q \cdot \rho_s(Z_q^t) \rho_s(L^+(A))} = \delta_Q \cdot \frac{\rho_s(u_{v_2} + L^+(A)) \rho_s(v_2)}{\delta_Q \cdot \rho_s(Z_q^t) \rho_s(L^+(A))} = \delta_Q \cdot \rho_s(u_{v_2} + L^+(A)).
\]

where \( u_{v_2} \in \mathbb{Z}_q^m \) is an arbitrary vector that satisfies \( Au_{v_2} = t - Bv_2 \) and \( \delta \in \left[ \frac{1-\epsilon}{1+\epsilon},1 \right] \) by Lemma 2.4. Since \( \epsilon = \text{negl}(n) \), this means that \( \delta/\delta_Q \in [1 - \text{negl}(n), 1 + \text{negl}(n)] \). Thus, for all \( v_2' \in \mathbb{Z}_q^t \), \( D_2(v_2') \in [1 - \text{negl}(n), 1 + \text{negl}(n)] \cdot D_{Z_q,v_2} \), so the statistical distance between \( D_2 \) and \( D_{Z_q,v_2} \) is bounded by a negligible function \( \text{negl}(n) \).

- Consider now the conditional distribution \( D_1 \) of \( v_1 \) given \( v_2' \):

\[
D_1(v_1 \mid v_2') = \frac{D \left( \frac{v_1}{v_2} \right)}{D_{Z_q,v_2}(v_2')} = \frac{\rho_s(v_1) \rho_s(v_2)}{\delta_Q \cdot \rho_s(Z_q^t) \rho_s(L^+(A))} \cdot \frac{\rho_s(Z_q^t)}{\rho_s(v_2)} = \frac{\delta' \rho_s(v_1)}{\delta_Q \rho_s(u_{v_2} + L^+(A))},
\]

where \( u_{v_2} \in \mathbb{Z}_q^m \) is an arbitrary vector satisfying \( Au_{v_2} = t - Bv_2 \) and \( \delta' \in \left[ \frac{1-\epsilon}{1+\epsilon},1 \right] \) by Lemma 2.4. By the same argument as above, in this is \( \text{negl}(n) \)-close to the distribution \( A_s^{-1}(t-Bv_2) \) and the claim follows. \( \square \)
The gadget matrix. We recall the definition of the gadget matrix [MP12]. For positive integers $n, q \in \mathbb{N}$, let $G_n = I_n \otimes g^* \in \mathbb{Z}_q^{n \times m}$ be the gadget matrix where $g^* = [1, 2, \ldots, 2^{\lceil \log q \rceil}]$ and $m' = n(\lfloor \log q \rfloor + 1)$. The inverse function $G_n^{-1} : \mathbb{Z}_q^{n \times t} \rightarrow \mathbb{Z}_q^{m \times t}$ expands each entry $x \in \mathbb{Z}_q$ into a column of size $\lceil \log q \rceil + 1$ consisting of the bits in the binary representation of $x$. By construction, for every matrix $A \in \mathbb{Z}_q^{n \times t}$, it follows that $G_n \cdot G_n^{-1}(A) = A \mod q$. When the lattice dimension $n$ is clear, we omit the subscript and simply write $G$ and $G^{-1}(-)$ to denote $G_n$ and $G_n^{-1}(-)$. We also write $g^{-1}(-)$ to denote the 1-dimensional operator $G_1^{-1}(-)$.

Gadget trapdoors. Our constructions will use the gadget trapdoors introduced by Micciancio and Peikert [MP12], which builds on a long sequence of works on constructing lattice trapdoors [Ajt96, GPV08, AP09, ABB10a, ABB10b, CHKP10].

Theorem 2.8 (Gadget Trapdoor [MP12, adapted]). Let $n, m, q$ be lattice parameters. Let $m' = n(\lceil \log q \rceil + 1)$. Then there exist efficient algorithms (TrapGen, SamplePre) with the following syntax:

- **Trapdoor generation**: The matrix $A$ output by TrapGen$(1^n, q, m)$ is statistically close to uniform. Specifically, if $(A, R) \leftarrow \text{TrapGen}(1^n, q, m)$ and $A' = R \cdot A \mod q$, then $\Delta(A, A') \leq 2^{-n}$. Moreover, $AR = G$ and $\|R\| = 1$.

- **Preimage sampling**: For all matrices $R \in \mathbb{Z}_q^{m \times m'}$, parameters $s > 0$, and all target vectors $v \in \mathbb{Z}_q^n$ in the column span of $A$, the output $u \leftarrow \text{SamplePre}(A, R, v, s)$ of SamplePre satisfies $Au = v$.

- **Preimage distribution**: Suppose $R$ is a gadget trapdoor for $A \in \mathbb{Z}_q^{n \times m}$ (i.e., $AR = G$). Then, for all $s \geq \sqrt{m n^2 \cdot \|R\| \cdot \omega(\sqrt{\log n})}$, and all target vectors $v \in \mathbb{Z}_q^n$, the statistical distance between the following distributions is at most $2^{-n}$:

\[
\{ u \leftarrow \text{SamplePre}(A, R, v, s) \} \quad \text{and} \quad \{ u \leftarrow A_s^{-1}(v) \}.
\]

More generally, the above properties hold if $AR = HG$ for some invertible matrix $H \in \mathbb{Z}_q^{n \times n}$. In this case, we refer to $H$ as the "tag."

For a matrix $V \in \mathbb{Z}_q^{n \times t}$, we define $\text{SamplePre}(A, R, V, s)$ to be the algorithm that outputs the matrix where the $i$th column is $\text{SamplePre}(A, R, v_i, s)$ and $v_i$ denotes the $i$th column of $V$.

Ajtai trapdoors. When analyzing our new hardness assumptions, it will also be convenient to use the more traditional lattice trapdoors introduced by Ajtai [Ajt96] and subsequently expanded by a number of subsequent works [GPV08, AP09, ABB10b, ABB10a, CHKP10, LW15]. We recall the main property we need:

Definition 2.9 (Ajtai Trapdoor [Ajt96]). Let $n, m, q$ be lattice parameters and $A \in \mathbb{Z}_q^{n \times m}$ be a matrix. We say that a matrix $R \in \mathbb{Z}_q^{m \times m'}$ is an Ajtai-trapdoor$^g$ for $A$ if $AR = 0 \mod q$ and $R$ is full rank over $\mathbb{R}$. We write $\hat{R} \in \mathbb{R}^{m \times m'}$ to denote the Gram-Schmidt orthogonalization of the columns of $R$ (from left to right).

Theorem 2.10 (Preimage Sampling [GPV08]). There exists an efficient algorithm $\text{SamplePre}'(A, R, v, s)$ that takes as input a matrix $A \in \mathbb{Z}_q^{n \times m}$, an Ajtai-trapdoor $R \in \mathbb{Z}_q^{m \times m'}$ for $A$, a target vector $v \in \mathbb{Z}_q^n$ in the column-span of $A$, and a Gaussian width parameter $s$, and outputs a vector $u \in \mathbb{Z}_q^n$ such that $Au = v$. Moreover, if $s \geq \|R\| \cdot \omega(\sqrt{m \log m})$, then the statistical distance between the following distributions is negl$(n)$:

\[
\{ u \leftarrow \text{SamplePre}'(A, R, v, s) \} \quad \text{and} \quad \{ u \leftarrow A_s^{-1}(v) \}.
\]

$^g$Technically, $R$ is a trapdoor for the lattice $\mathcal{L}^*(A)$, but for simplicity of description, we simply refer to $R$ as a trapdoor for the matrix $A$. 

15
**Homomorphic evaluation.** Our construction of succinct functional commitments will rely on the lattice homomorphic evaluation procedure developed in [GSW13, BGG+14]. Our presentation is adapted from that in [BV15, BCTW16, BTVW17].

**Theorem 2.11 (Homomorphic Encodings [GSW13, BGG+14]).** Let $\lambda$ be a security parameter and $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$ be lattice parameters. Let $m' = n(\lceil \log q \rceil + 1)$. Let $\ell = \ell(\lambda)$ be an input length. Let $\mathcal{F} = \{ F_\ell \}_{\lambda \in \mathbb{N}}$ be a family of functions $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ that can be computed by a Boolean circuit of depth at most $d = d(\lambda)$. Then, there exist a pair of efficient algorithms $(\text{EvalF}, \text{EvalFX})$ with the following properties:

- $\text{EvalF}(A, f) \rightarrow A_f$: On input a matrix $A \in \mathbb{Z}_q^{n \times \ell m'}$ and a function $f \in \mathcal{F}$, the input-independent evaluation algorithm outputs a matrix $A_f \in \mathbb{Z}_q^{n \times m'}$.

- $\text{EvalFX}(A, f, x) \rightarrow H_{A_f, x}$: On input a matrix $A \in \mathbb{Z}_q^{n \times \ell m'}$, a function $f \in \mathcal{F}$, and an input $x \in \{0, 1\}^\ell$, the input-dependent evaluation algorithm outputs a matrix $H_{A_f, x} \in \mathbb{Z}_q^{\ell m' \times m'}$.

Moreover for all security parameters $\lambda \in \mathbb{N}$, matrices $A \in \mathbb{Z}_q^{n \times \ell m'}$, all functions $f \in \mathcal{F}$, and all inputs $x \in \{0, 1\}^\ell$, the matrices $A_f \leftarrow \text{EvalF}(A, f)$ and $H_{A_f, x} \leftarrow \text{EvalFX}(A, f, x)$ satisfy the following properties:

- $\|H_{A_f, x}\| \leq (n \log q)^O(d)$.

- $(A - x^i \otimes G) \cdot H_{A_f, x} = A_f - f(x) \cdot G$.

### 3 Vector Commitments with Private Opening from SIS

In this section, we show how to construct a vector commitment with private openings from the standard SIS assumption. We start by recalling the definition of a vector commitment:

**Definition 3.1 (Vector Commitment).** A vector commitment scheme with succinct local openings over a message space $M$ consists of a tuple of efficient algorithms $\Pi_{VC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify})$ with the following properties:

- $\text{Setup}(1^\lambda, 1^\ell) \rightarrow \text{crs}$: On input the security parameter $\lambda$ and the vector length $\ell$, the setup algorithm outputs a common reference string $\text{crs}$.

- $\text{Commit} (\text{crs}, x) \rightarrow (\sigma, \text{st})$: On input the common reference string $\text{crs}$ and a vector $x \in M^\ell$, the commit algorithm outputs a commitment $\sigma$ and a state $\text{st}$.

- $\text{Open} (\text{st}, i) \rightarrow \pi$: On input a commitment state $\text{st}$ and an index $i \in [\ell]$, the open algorithm outputs an opening $\pi$.

- $\text{Verify} (\text{crs}, \sigma, i, x_i, \pi) \rightarrow \{0, 1\}$: On input the common reference string $\text{crs}$, a commitment $\sigma$, an index $i$, a message $x_i \in M$, and an opening $\pi$, the verification algorithm outputs a bit $b \in \{0, 1\}$.

We now define several standard properties on vector commitment schemes:

- **Correctness:** For all security parameters $\lambda$, vector dimensions $\ell$, and inputs $x = (x_1, \ldots, x_\ell) \in M^\ell$,

$$\Pr \left[ \text{Verify}(\text{crs}, \sigma, i, m_i, \pi) = 1 : (\sigma, \text{st}) \leftarrow \text{Commit}(\text{crs}, x); \pi \leftarrow \text{Open}(\text{st}, i) \right] = 1 - \operatorname{negl}(\lambda).$$

- **Succinctness:** The vector commitment scheme is succinct if there exists a universal polynomial $\operatorname{poly}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\sigma| = \operatorname{poly}(\lambda, \log \ell)$ and $|\pi| = \operatorname{poly}(\lambda, \log \ell)$ in the correctness definition.
Assumption 3.3 (Assumption 3.2) Assume two concrete instantiations of the \( \alpha \) and \( \beta \) adversaries \( A \). Let \( s \) be lattice parameters. Let \( \sigma \) be the standard SIS assumption. For instance, our first construction of a vector commitments with \( B \) trapdoor for a matrix \( A \). In this section, we introduce the family of SIS assumptions that we use to build our vector commitment schemes. At

\[
\begin{align*}
\Pr & \left[ \text{Verify}(\sigma, i, x_i, \pi) = 1 \right. \\
& \left. \text{and } x_i \neq x'_i \text{ and } \text{Verify}(\sigma, i, x'_i, \pi') = 1 \right] \\
\Leftrightarrow & \sigma \gets \text{Setup}(1^\lambda, 1^\ell); \\
& \sigma, i, (x_i, \pi), (x'_i, \pi') \gets \mathcal{A}(1^\lambda, 1^\ell, \text{crs}) \\
\Rightarrow & \negl(\lambda).
\end{align*}
\]

• **Private openings:** For a vector dimension \( \ell \), an adversary \( \mathcal{A} \), and a simulator \( S = (S_0, S_1) \), we define two distributions \( \text{Real}_{\mathcal{A}}(1^\lambda, 1^\ell) \) and \( \text{Ideal}_{\mathcal{A}, S}(1^\lambda, 1^\ell) \) as follows:

<table>
<thead>
<tr>
<th>\text{Real}_{\mathcal{A}}(1^\lambda)</th>
<th>\text{Ideal}_{\mathcal{A}, S}(1^\lambda)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Give ( crs \gets \text{Setup}(1^\lambda, 1^\ell) ) to ( \mathcal{A} ).</td>
<td>1. Sample ( (crs, \sigma, s) \gets S_0(1^\lambda, 1^\ell) ) and give ( crs ) to ( \mathcal{A} ).</td>
</tr>
<tr>
<td>2. Algorithm ( \mathcal{A} ) outputs an input ( x \in M' ).</td>
<td>2. Algorithm ( \mathcal{A} ) outputs an input ( x \in M' ).</td>
</tr>
<tr>
<td>3. Compute ( \sigma, s ) ← Commit(crs, x) and give ( \sigma ) to ( \mathcal{A} ).</td>
<td>3. Give ( \sigma ) to ( \mathcal{A} ).</td>
</tr>
<tr>
<td>4. Algorithm ( \mathcal{A} ) outputs an index ( i \in [\ell] ).</td>
<td>4. Algorithm ( \mathcal{A} ) outputs an index ( i \in [\ell] ).</td>
</tr>
<tr>
<td>5. Compute ( S_i(st, i, x_i) ) and give ( S_i ) to ( \mathcal{A} ).</td>
<td>5. Compute ( S_i(st, i, x_i) ) and give ( S_i ) to ( \mathcal{A} ).</td>
</tr>
<tr>
<td>6. Algorithm ( \mathcal{A} ) outputs a bit ( b \in {0, 1} ) which is the output of the experiment.</td>
<td>6. Algorithm ( \mathcal{A} ) outputs a bit ( b \in {0, 1} ) which is the output of the experiment.</td>
</tr>
</tbody>
</table>

We say that the vector commitment scheme has statistically (resp., computationally) private openings if for all polynomials \( \ell = \ell(\lambda) \) and adversaries \( \mathcal{A} \) (resp., efficient adversaries \( \mathcal{A} \)), there exists an efficient simulator \( S = (S_0, S_1) \) such that \( \text{Real}_{\mathcal{A}}(1^\lambda, 1^\ell) \) and \( \text{Ideal}_{\mathcal{A}, S}(1^\lambda, 1^\ell) \) are statistically (resp., computationally) indistinguishable.

### 3.1 The Basis-Augmented SIS (BASIS) Assumption

In this section, we introduce the family of SIS assumptions that we use to build our vector commitment schemes. At a high level, our assumptions assert that the SIS problem is hard with respect to a random matrix \( A \) even given a trapdoor for a matrix \( A \) that is correlated with \( A \). We refer to our family of assumptions as the “basis-augmented SIS” (BASIS) assumption. As we discuss below (Theorem 3.4), some versions of the BASIS assumption can be reduced to the standard SIS assumption. For instance, our first construction of a vector commitments with *private* openings (Construction 3.9) relies on a version that reduces to the standard SIS assumption.

**Assumption 3.2 (BASIS Assumption).** Let \( \lambda \) be a security parameter and \( n = n(\lambda), m = m(\lambda), q = q(\lambda), \) and \( \beta = \beta(\lambda) \) be lattice parameters. Let \( s = s(\lambda) \) be a Gaussian width parameter. Let \( \text{Samp} \) be an efficient sampling algorithm that takes as input a security parameter \( \lambda \) and a matrix \( A \in \mathbb{Z}_q^{n \times m} \) and outputs a matrix \( B \in \mathbb{Z}_q^{\ell \times m} \) along with auxiliary input \( aux \). We say that the basis-augmented SIS (BASIS) assumption holds with respect to \( \text{Samp} \) if for all efficient adversaries \( \mathcal{A} \),

\[
\Pr \left[ Ax = 0 \text{ and } 0 < ||x|| \leq \beta : \frac{A \leftarrow \mathbb{Z}_q^{n \times m}}{\text{Samp}(1^\lambda, A), T \leftarrow B_{\mathcal{A}}^{-1}(G_{\mathcal{A}}); \ x \leftarrow \mathcal{A}(1^\lambda, A, B, T, aux)} \right] = \negl(\lambda).
\]

In other words, we require that SIS is hard with respect to \( A \) even given a trapdoor \( T \) for the related matrix \( B \).

**Assumption 3.3 (BASIS Assumption Instantiations).** Let \( \lambda \) be a security parameter and \( n = n(\lambda), m = m(\lambda), q = q(\lambda), \) and \( \beta = \beta(\lambda) \) be lattice parameters. Let \( s = s(\lambda) \) be a Gaussian width parameter and \( \ell = \ell(\lambda) \) be a dimension. We consider two concrete instantiations of the BASIS assumption:

- **BASIS\_rand:** The sampling algorithm \( \text{Samp}(1^\lambda, A) \) samples \( i^* \sim [\ell], A_i \sim \mathbb{Z}_q^{(n+1) \times m} \) for all \( i \neq i^* \), \( a \sim \mathbb{Z}_q^m \), sets \( A_{i^*} \leftarrow \begin{bmatrix} A_1 & \cdots & -G_{n+1} \end{bmatrix} \), and outputs

\[
B_{i^*} = \begin{bmatrix} A_1 & \cdots & -G_{n+1} \\
\vdots & \ddots & \vdots \\
A_i & \cdots & -G_{n+1} \end{bmatrix}
\]

and \( aux = i^* \).

We refer to this assumption as “the BASIS assumption with random matrices.”
• BASIS\textsubscript{struct}: The sampling algorithm Samp(1^A, A) samples \( W_i \in \mathbb{Z}_q \times n \) for all \( i \in [\ell] \) and outputs

\[
B_\ell = \begin{bmatrix}
W_1 A & \cdots & -G_n \\
\vdots & \ddots & \vdots \\
W_\ell A & \cdots & -G_n 
\end{bmatrix}
\quad \text{and} \quad \text{aux} = (W_1, \ldots, W_\ell).
\] (3.1)

This is essentially BASIS\textsubscript{rand} with \textit{structured} matrices \( A_1, \ldots, A_\ell \). We refer to this assumption as “the BASIS assumption with structured matrices.”

Each of the above assumptions is parameterized by the tuple of parameters \((n, m, q, \beta, s, \ell)\). Strictly speaking, in both cases above, the auxiliary information aux can be efficiently computed directly from \( A \) and \( B_\ell \), and thus, can be safely omitted. We include them here for notational convenience.

**Hardness of BASIS\textsubscript{rand}**. We start by showing that the BASIS assumption with random matrices (i.e., BASIS\textsubscript{rand} in Assumption 3.3) holds under the standard SIS assumption.

**Theorem 3.4** (Hardness of BASIS\textsubscript{rand}). Let \( \lambda \) be a security parameter and \( n = n(\lambda), m = m(\lambda), q = q(\lambda), \beta = \beta(\lambda) \) be lattice parameters. Take any polynomial \( \ell = \ell(\lambda) \) and suppose \( n \geq \lambda, m \geq O(n \log q), \) and \( s \geq O(\ell m \log(n \ell)) \). Then, under the SIS\textsubscript{n,m,q,\beta} assumption, the BASIS\textsubscript{rand} assumption with parameters \((n, m, q, \beta, s, \ell)\) holds.

**Proof.** Take any polynomial \( \ell = \ell(\lambda) \). We proceed via a hybrid argument:

• Hyb\textsubscript{0}: This is the BASIS\textsubscript{rand} security game:
  - The challenger samples \( A \in \mathbb{Z}_q^{n \times m} \) and \((B_\ell, \text{aux}) \leftarrow \text{Samp}(1^A, A)\). In this case, the sampling algorithm samples \( i^* \in [\ell] \). It then samples \( A_i \in \mathbb{Z}_q^{m \times A_i}, A_i \leftarrow [a_i^{n \times m}] \), and \( A_i \in \mathbb{Z}_q^{n \times (n+1) \times m} \) for each \( i \neq i^* \).
  - The challenger gives \( A, B_\ell = \text{diag}(A_1, \ldots, A_\ell) - 1^\ell \otimes G_{n+1} \in \mathbb{Z}_q^{(n+1) \times (m+sm')}, T \leftarrow (B_\ell)^{-1}(G_{n+1}1^\ell) \) and \( \text{aux} = i^* \) to \( \mathcal{A} \), where \( m' = (n+1)(\lceil \log q \rceil + 1) \).
  - The output of the experiment is 1 if \( 0 < |x| \leq \beta \) and \( A \), \( x = 0 \).

• Hyb\textsubscript{i}: Same as Hyb\textsubscript{0} except the challenger samples \((A_i, R_i) \leftarrow \text{TrapGen}(1^{n+1}, q, m)\) for each \( i \neq i^* \).

• Hyb\textsubscript{i}: Same as Hyb\textsubscript{1} except after constructing \( B_\ell \), the challenger constructs a matrix \( R \in \mathbb{Z}_q^{(m+sm') \times m} \) as follows:

\[
R = \begin{bmatrix}
R_1 & -R_1 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & 0 \\
-1_{r+1} & 0 & \cdots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & -1_{r'} 
\end{bmatrix} \in \mathbb{Z}_q^{(m+sm') \times m'}.
\] (3.2)

It then samples \( T \leftarrow \text{SamplePre}(B_\ell, R, G_{(n+1)\ell}, s) \).

For an adversary \( \mathcal{A} \), we write Hyb\textsubscript{i}(\( \mathcal{A} \)) to denote the output distribution of an execution of experiment Hyb\textsubscript{i} with adversary \( \mathcal{A} \). We now analyze each pair of adjacent hybrids.

**Lemma 3.5.** Suppose \( n \geq \lambda \) and \( m \geq O(n \log q) \). Then, for all adversaries \( \mathcal{A} \), it holds that \( \text{Hyb}_0(\mathcal{A}) \approx \text{Hyb}_1(\mathcal{A}) \).

**Proof.** The only difference between Hyb\textsubscript{0} and Hyb\textsubscript{1} is the distribution of \( A_i \) for \( i \neq i^* \). In Hyb\textsubscript{0}, \( A_i \in \mathbb{Z}_q^{n \times (n+1) \times m} \) while in Hyb\textsubscript{1}, it is sampled using TrapGen\((1^{n+1}, q, m)\). By Theorem 2.8, these two distributions are statistically indistinguishable when \( m \geq O((n+1) \log q) = O(n \log q) \). Since \( \ell = \text{poly}(\lambda) \), the lemma follows by a hybrid argument. \( \square \)
Lemma 3.6. Suppose \( n \geq \lambda, m \geq O(n \log q), \) and \( s \geq O(tm \log(nt)) \). Then, for all adversaries \( \mathcal{A} \), \( \text{Hyb}_1(\mathcal{A}) \approx \text{Hyb}_2(\mathcal{A}) \).

Proof. The only difference between \( \text{Hyb}_1 \) and \( \text{Hyb}_2 \) is the distribution of the trapdoor \( T \). Consider the distribution in \( \text{Hyb}_2 \). By Theorem 2.8, we have that \( A_i R_i = G_{n+1} \in \mathbb{Z}^{(n+1) \times m} \) for all \( i \neq i' \) and moreover \( ||R_i|| = 1 \). Thus, by construction, \( B_i R = G_{t(n+1)} \), and \( ||R|| = 1 \). Suppose \( s \geq \sqrt{(tm + m')\ell m'} ||R|| \cdot \omega(\log(nt)) = O(tm \log(nt)) \) for \( m \geq m' = O(n \log q) \). Again by Theorem 2.8, the distribution of \( T \leftarrow \text{SamplePre}(B_i, R, G_{(n+1)t}, s) \) is statistically close \( T \leftarrow (B_i)^{-1}(G_{(n+1)t}) \), which is the distribution in \( \text{Hyb}_1 \).

Lemma 3.7. Under the \( \text{SIS}_{n,m,q,\beta} \) assumption, for all efficient adversaries \( \mathcal{A} \), \( \Pr[\text{Hyb}_2(\mathcal{A}) = 1] = \text{negl}(\lambda) \).

Proof. Suppose there exists an efficient adversary \( \mathcal{A} \) where \( \Pr[\text{Hyb}_2(\mathcal{A}) = 1] = \epsilon \) for some non-negligible \( \epsilon \). We use \( \mathcal{A} \) to construct an adversary \( \mathcal{B} \) for the SIS assumption:

1. At the beginning of the game, algorithm \( \mathcal{B} \) obtains a SIS challenge \( A \in \mathbb{Z}_q^{n \times m} \).
2. Algorithm \( \mathcal{B} \) samples an index \( i^* \leftarrow [t] \). It also samples a row \( a \leftarrow \mathbb{Z}_q^m \) and sets \( A_i \leftarrow \left[ \begin{array}{c} a \end{array} \right] \). For all \( i \neq i^* \), it samples \( (A_i, R_i) \leftarrow \text{TrapGen}(1^* m, m, q) \). It constructs the matrix \( B_i = \text{diag}(A_i, \ldots, A_i) \) and the trapdoor \( R \) according to Eq. (3.2). Finally, algorithm \( \mathcal{B} \) computes \( T \leftarrow \text{SamplePre}(B_i, R, G_{(n+1)t}, s) \) and gives \( A, B_i, T, \) and auxiliary data \( i^* \) to \( \mathcal{A} \).
3. After algorithm \( \mathcal{A} \) outputs a vector \( x \in \mathbb{Z}_q^m \), algorithm \( \mathcal{B} \) outputs the same vector \( x \).

By construction, algorithm \( \mathcal{B} \) perfectly simulates an execution of \( \text{Hyb}_2 \) for algorithm \( \mathcal{A} \), so with probability \( \epsilon \), algorithm \( \mathcal{A} \) outputs \( x \) where \( Ax = 0 \) and \( 0 < ||x|| \leq \beta \). Thus, this is a valid SIS solution, so algorithm \( \mathcal{B} \) breaks SIS with the same advantage \( \epsilon \).

Combining Lemmas 3.5 to 3.7, we have that for all efficient adversaries \( \mathcal{A} \), \( \Pr[\text{Hyb}_2(\mathcal{A}) = 1] = \text{negl}(\lambda) \).

Hardness of BASISstruct. While we are able to reduce the hardness of the BASIS assumption with random matrices (BASISrand) to the standard SIS assumption, we do not know of an analogous reduction for the BASIS assumption with structured matrices (BASISstruct). In Section 6, we compare our BASISstruct assumption to the conceptually similar \( k-R \)-SIS assumptions introduced in [ACL'22] for constructing lattice-based vector commitment and preprocessing SNARGs. In addition, we also describe an alternative view of the BASISstruct assumption, and suggest a set of candidate parameter instantiations (Remark 6.2).

Remark 3.8 (An Alternative Formulation of BASISstruct). When the matrices \( W_1, \ldots, W_l \) in the BASISstruct assumption are invertible (e.g., this holds with overwhelming probability when \( q \) is prime), we can alternatively interpret the BASISstruct assumption as saying that SIS is hard with respect to \( A \) given a trapdoor for the matrix

\[
\hat{B}_l = \begin{bmatrix} I_l \otimes A & \hat{W}G \end{bmatrix} \quad \text{where} \quad \hat{W} = \begin{bmatrix} \hat{W}_1 \\ \vdots \\ \hat{W}_l \end{bmatrix} \in \mathbb{Z}_q^{n \times n} \quad \text{and} \quad \hat{W}_i \leftarrow \mathbb{Z}_q^{n \times n}.
\]

When \( W_1, \ldots, W_l \) are random invertible matrices, we can set \( \hat{W}_i := -W_i^{-1} \). In this case, we can write \( \hat{B}_l = (\hat{W}^*)^{-1}\hat{B}_l \), where \( \hat{B}_l \) is the matrix in Eq. (3.1) of the BASISstruct assumption and \( \hat{W}^* = \text{diag}(\hat{W}_1, \ldots, \hat{W}_l) \). Then, if \( T \) is a trapdoor for the matrix \( B_l \) (i.e., \( B_l T = G_{nt} \)), then \( T \) is also a trapdoor for matrix \( \hat{B}_l \) with tag \( \hat{W}^* \) since \( B_l T = \hat{W}^* B_l T = \hat{W}^* G_{nt} \).
3.2 Vector Commitments with Private Opening from SIS

We now show how to construct a vector commitment scheme with statistically private openings from the BASIS\textsubscript{rand} assumption. By Theorem 3.4, we can in turn base hardness on the standard SIS assumption (with polynomial modulus).

Construction 3.9 (Vector Commitments from SIS). Let $\lambda$ be a security parameter and $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$ be lattice parameters. Let $m' = n(\lceil \log q \rceil + 1)$ and $B = B(\lambda)$ be a bound. Let $s_0 = s_0(\lambda), s_1 = s_1(\lambda)$ be Gaussian width parameters. Let $f$ be the vector dimension. We construct a vector commitment scheme $\Pi_{VC} = (\text{Setup, Commit, Open, Verify})$ for $\mathbb{Z}_q^n$ as follows:

- **Setup($1^\lambda, 1^f$):** On input the security parameter $\lambda$ and the vector dimension $f$, the setup algorithm samples $(A_i, R_i) \leftarrow \text{TrapGen}(1^n, q, m)$ for each $i \in [f]$. Then, it constructs matrices $B_f$ and $R$ where

$$B_f = \begin{bmatrix} A_1 & -G \\ \vdots & \vdots \\ A_f & -G \end{bmatrix} \in \mathbb{Z}_q^{nf \times (tm + m')} \quad \text{and} \quad R = \begin{bmatrix} \text{diag}(R_1, \ldots, R_f) \\ 0^{m \times tm} \end{bmatrix} \in \mathbb{Z}_q^{(tm + m') \times tm}.$$  

(3.3)

Finally, the setup algorithm samples $T \leftarrow \text{SamplePre}(B_f, R, \text{G}_{\text{ns}}, s_0)$ and outputs the common reference string $\text{crs} = (A_1, \ldots, A_f, T)$.

- **Commit($\text{crs}, x$):** On input the common reference string $\text{crs} = (A_1, \ldots, A_f, T)$ and a vector $x \in \mathbb{Z}_q^f$, the commit algorithm constructs $B_f$ from $A_1, \ldots, A_f$ according to Eq. (3.3) and then uses $T$ to sample

$$\begin{bmatrix} v_1 \\ \vdots \\ v_f \\ \hat{c} \end{bmatrix} \leftarrow \text{SamplePre}(B_f, T, -x \otimes e_1, s_1).$$

(3.4)

where $e_1 = [1, 0, \ldots, 0]^T \in \mathbb{Z}_q^m$ is the first standard basis vector. It computes $c \leftarrow \text{G} \hat{c} \in \mathbb{Z}_q^n$ and outputs the commitment $\sigma = c$ and the state $\text{st} = (v_1, \ldots, v_f)$.

- **Open($\text{st}, i$):** On input the state $\text{st} = (v_1, \ldots, v_f)$ and the index $i \in [f]$, the opening algorithm outputs $v_i$.

- **Verify($\text{crs}, \sigma, i, x_i, \pi$):** On input the common reference string $\text{crs} = (A_1, \ldots, A_f, T)$, a commitment $\sigma = c \in \mathbb{Z}_q^n$, an index $i \in [f]$, a message $x_i \in \mathbb{Z}_q$, and an opening $\pi = v_i$, the verification algorithm outputs 1 if

$$||v_i|| \leq B \quad \text{and} \quad c = A_i v_i + x_i e_1.$$

Theorem 3.10 (Correctness). Suppose $n \geq \lambda$, $m \geq O(n \log q)$, $s_0 \geq O(tm \log(nt))$, $s_1 \geq O(\ell^{3/2} m^{3/2} \log(nt) \cdot s_0)$, and $B \geq \sqrt{tm + m'} \cdot s_1$. Then Construction 3.9 is correct.

Proof. Take any polynomial $t = t(\lambda)$, any vector $x \in \mathbb{Z}_q^f$, and any index $i \in [f]$. Let $\text{crs} = (A_1, \ldots, A_f, T) \leftarrow \text{Setup}(1^\lambda, 1^f)$. Let $(\sigma, \text{st}) \leftarrow \text{Commit}(\text{crs}, x)$ where $\sigma = c = \text{G} \hat{c} \in \mathbb{Z}_q^n$ and $\text{st} = (v_1, \ldots, v_f)$. Let $\pi = v_i \leftarrow \text{Open}(\text{st}, i)$, and consider Verify($\text{crs}, \sigma, i, x_i, \pi$):

- Let $B_f \in \mathbb{Z}_q^{n \times (tm + m')}$ and $R \in \mathbb{Z}_q^{(tm + m') \times tm}$ be the matrices from Eq. (3.3). By construction, $B_f R = \text{G}_{\text{ns}}$ and $\|R\| = 1$. Suppose $m \geq m' = O(n \log q)$ and $s_0 \geq \sqrt{(tm + m') m' \|R\| \cdot \omega(\sqrt{\log(nt)})} = O(tm \log(nt))$.

By Theorem 2.8, this means $B_f T = \text{G}_{\text{ns}}$, and moreover, by Lemmas 2.5 and 2.6, $\|T\| \leq \sqrt{tm + m'} s_0$ with probability $1 - \text{negl}(n) = 1 - \text{negl}(\lambda)$.

- Suppose $s_1 \geq \sqrt{(tm + m') m' \|T\| \cdot \omega(\sqrt{\log(nt)})} = O(t^{3/2} m^{3/2} \log(nt) \cdot s_0)$. Then, by Theorem 2.8 and by construction of $v_1, \ldots, v_f, \hat{c}$ (see Eq. (3.4)), we have that $A_i v_i - c = A_i v_i - \text{G} \hat{c} = -x_i e_1$. In addition, by Lemmas 2.5 and 2.6, we have that $\|v_i\| \leq \sqrt{tm + m's_1} \leq B$, and the verification algorithm accepts with overwhelming probability. \qed
Theorem 3.11 (Binding). Take any polynomial $t = t(\lambda)$ and suppose $n \geq \lambda$, $m \geq O(n \log q)$, and $s_0 \geq O(tm \log nt)$. Then, under the BASIS$_{\text{rand}}$ assumption with parameters $(n-1, m, q, 2B, s_0, t)$, Construction 3.9 is computationally binding.

Proof. Take any polynomial $t = t(\lambda)$. We now define a sequence of hybrid experiments.

- **Hyb$_0$:** This is the real binding experiment:
  - The challenger starts by sampling $(A_i, R_i) \leftarrow \text{TrapGen}(1^n, q, m)$ for each $i \in [t]$. It then constructs $R$ according to Eq. (3.3) and sets $B_i = [\text{diag}(A_1, \ldots, A_t) | -1^t \otimes G]$. It samples $T \leftarrow \text{SamplePre}(B_t, R, G_{nt}, s_0)$. Then, the challenger gives $\text{crs} = (A_1, \ldots, A_t, T)$ to $\mathcal{A}$.
  - Algorithm $\mathcal{A}$ then outputs a commitment $c \in \mathbb{Z}_q^n$, an index $i \in [t]$, and openings $(x, v)$, $(x', v')$.
  - The output of the experiment is 1 if $x \neq x'$ and 
    \[
    \|v\| \leq B \quad \text{and} \quad \|v'\| \leq B \quad \text{and} \quad c = A_iv + xe_1 \quad \text{and} \quad c = A_iv' + x'e_1.
    \]

- **Hyb$_1$:** Same as Hyb$_0$ except at the beginning of the game, the challenger samples an index $i^* \overset{\text{R}}{\leftarrow} [t]$. The output of the experiment is 1 if the conditions in Hyb$_0$ hold and $i = i^*$.

- **Hyb$_2$:** Same as Hyb$_1$ except the challenger samples $T \leftarrow (B_t)_{s_0^{-1}}(G_{nt})$.

- **Hyb$_3$:** Same as Hyb$_2$ except the challenger samples $A_i \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}$ for all $i \in [t]$.

For an adversary $\mathcal{A}$, we write $\text{Hyb}_1(\mathcal{A})$ to denote the output distribution of an execution of experiment $\text{Hyb}_1$ with adversary $\mathcal{A}$. We now analyze each pair of adjacent hybrids.

Lemma 3.12. For all adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_0(\mathcal{A}) = 1] = t \cdot \Pr[\text{Hyb}_1(\mathcal{A}) = 1]$.

Proof. This follows by definition. The index $i^*$ is sampled independently of the adversary’s view, so $\Pr[i^* = i] = 1/t$, and the claim follows. \hfill $\square$

Lemma 3.13. Suppose $n \geq \lambda$, $m \geq O(n \log q)$, $s_0 \geq O(tm \log nt))$. Then, for all adversaries $\mathcal{A}$, $\text{Hyb}_1(\mathcal{A}) \approx \text{Hyb}_2(\mathcal{A})$.

Proof. The only difference between $\text{Hyb}_1$ and $\text{Hyb}_2$ is the distribution of $T$. Consider the distribution of $T$ in $\text{Hyb}_1$. Here, $\text{B}_iR = G_{nt}$ by construction and $\|R_i\| = 1$ since $\|R_i\| = 1$ for all $i \in [t]$ by Theorem 2.8. Suppose $m \geq m' = O(n \log q)$ and $s_0 \geq \sqrt{\frac{t}{2} + m'} \log(m' \|R_i\|) = O(tm \log nt))$. Again by Theorem 2.8, the distribution of $T \leftarrow \text{SamplePre}(\text{B}_t, R, G_{nt}, s_0)$ is statistically close to the distribution of $(\text{B}_t)_{s_0^{-1}}(G_{nt})$. This is the distribution of $T$ in $\text{Hyb}_2$ and the claim follows. \hfill $\square$

Lemma 3.14. Suppose $n \geq \lambda$ and $m \geq O(n \log q)$. Then, for all adversaries $\mathcal{A}$, $\text{Hyb}_2(\mathcal{A}) \approx \text{Hyb}_3(\mathcal{A})$.

Proof. The only difference between $\text{Hyb}_2$ and $\text{Hyb}_3$ is the distribution of $A_i$. In $\text{Hyb}_2$, the challenger samples $(A_i, R_i) \overset{\text{R}}{\leftarrow} \text{TrapGen}(1^n, q, m)$. By Theorem 2.8, for sufficiently large $m \geq O(n \log q)$, the distribution of $A_i$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$. This coincides with the distribution of $A_i$ in $\text{Hyb}_3$ and the claim follows by a hybrid argument since $t = \text{poly}(\lambda)$. \hfill $\square$

Lemma 3.15. Under the BASIS$_{\text{rand}}$ assumption with parameters $(n-1, m, q, 2B, s_0, t)$, for all efficient adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_3(\mathcal{A}) = 1] = \text{negl}(\lambda)$.

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ where $\Pr[\text{Hyb}_3(\mathcal{A}) = 1] = \epsilon$ for some non-negligible $\epsilon$. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the BASIS$_{\text{rand}}$ assumption:

1. At the beginning of the game, algorithm $\mathcal{B}$ obtains the challenge $A \in \mathbb{Z}_q^{(n-1) \times m}$, $B_t \in \mathbb{Z}_q^{nt \times (tm+mt')}$, $T \in \mathbb{Z}_q^{(tm+mt') \times tm'}$ and aux = $i^*$.

2. Algorithm $\mathcal{B}$ parses $B_t = [\text{diag}(A_1, \ldots, A_t) | -1^t \otimes G]$ and defines $\text{crs} = (A_1, \ldots, A_t, T)$. Algorithm $\mathcal{B}$ gives $\text{crs} = (A_1, \ldots, A_t, T)$ to algorithm $\mathcal{A}$. 

21
3. After algorithm $\mathcal{A}$ outputs a commitment $\hat{c} \in \mathbb{Z}_q^{t}$, the index $i \in [t]$, and openings $(x, v), (x', v')$, algorithm $\mathcal{B}$ first checks that $i = i'$. If so, then it outputs $v - v'$. Otherwise, it outputs $\bot$.

By construction, algorithm $\mathcal{B}$ perfectly simulates an execution of $\text{Hyb}_3$ for algorithm $\mathcal{A}$, so with probability $\epsilon$, algorithm $\mathcal{A}$ outputs $c, i, (x, v), (x', v')$ where

$$i = i' \quad \text{and} \quad x \neq x' \quad \text{and} \quad \|v\| \leq B \quad \text{and} \quad \|v'\| \leq B \quad \text{and} \quad c = A_{i'}v + xe_1 \quad \text{and} \quad c = A_i v' + x'e_1.$$

This means that $|v - v'| \leq 2B$, and moreover, $A_{i'}(v - v') = (x' - x)e_1$. Since $x \neq x'$, this means that $v - v' \neq 0$. Next, by definition of $A_{i'}$ in the BASIS$_{\text{rand}}$ assumption, we have

$$A_{i'}(v - v') = \begin{bmatrix} a^T & 0 \end{bmatrix} A \begin{bmatrix} v - v' \end{bmatrix} = \begin{bmatrix} x' - x \end{bmatrix} 0^{n-1}.$$

Thus, $A(v - v') = 0$ and $v - v'$ is a valid solution to the BASIS$_{\text{rand}}$ assumption. Correspondingly, algorithm $\mathcal{B}$ breaks the BASIS$_{\text{rand}}$ assumption with advantage $\epsilon$. $\square$

By Lemmas 3.12 to 3.14, we have that for all adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_0(\mathcal{A}) = 1] \leq \ell - (\Pr[\text{Hyb}_3(\mathcal{A}) = 1] + \text{negl}(\ell))$. Since $\ell = \text{poly}(\lambda)$, we can conclude via Lemma 3.15 that for all efficient adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_0(\mathcal{A}) = 1] \leq \text{negl}(\lambda)$. $\square$

**Theorem 3.16 (Hiding).** Suppose $n \geq \lambda$, $m \geq O(n \log q)$, $q$ is prime, $s_0 \geq O(tm \log(nt))$, and $s_1 \geq O(\ell^{3/2}m^{3/2} \log(nt) \cdot s_0)$. Then, Construction 3.9 has statistically private openings.

**Proof.** We construct an efficient simulator $\mathcal{S} = (S_0, S_1)$ as follows:

- $S_0(1^\lambda, 1^t)$: On input the security parameter $\lambda$ and the vector dimension $t$, the simulator samples $(A_i, R_i) \leftarrow \text{TrapGen}(1^t, q, m)$ for each $i \in [t]$. Let $\hat{A} = \text{diag}(A_1, \ldots, A_t)$. Then, construct the matrix $B_t = [\hat{A} | -1^t \otimes G]$ along with the trapdoor $R$ according to Eq. (3.3). It then computes $T \leftarrow \text{SamplePre}(B_t, R, G_{nt}, s_0)$ and sets the common reference string to be $\text{crs} = (A_1, \ldots, A_t, T)$. Next, it samples $\hat{c} \leftarrow D_{A, s_1}^m$ and outputs $\text{crs}, \sigma = c = \hat{G}c$, and $\text{st}_S = (A_1, \ldots, A_t, R_1, \ldots, R_t, c)$.

- $S_1(\text{st}_S, i, x_1)$: On input the simulation state $\text{st}_S = (A_1, \ldots, A_t, R_1, \ldots, R_t, c)$, an index $i \in [t]$, and a value $x_i \in \mathbb{Z}_q$, the simulator samples $v_i \leftarrow \text{SamplePre}(A_i, R_i, -x_i e_1 + c, s_1)$. It outputs the opening $\pi_i = v_i$.

To complete the proof, it suffices to argue that the real distribution and the simulated distributions are statistically indistinguishable. First, the simulator samples $\text{crs}$ using the same procedure as Setup, so it suffices to analyze the commitment and the opening. Take any vector $x \in \mathbb{Z}_q^m$ and let $(\sigma, \text{st}) \leftarrow \text{Commit}(\text{crs}, x)$ where $\sigma = c = \hat{G}c$, and $\text{st} = (v_1, \ldots, v_t)$ are the openings. We consider their joint distribution in the real and simulated space:

- Let $B_t \in \mathbb{Z}_q^{m \times (tm + m^t)}$ be the matrix from Eq. (3.3). Let $T \leftarrow \text{SamplePre}(B_t, R, G_{nt}, s_0)$ be the trapdoor sampled in the real scheme. As in the proof of Theorem 3.10, we can appeal to Theorem 2.8 to conclude that with overwhelming probability, $B_t T = G_{nt}$ and $\|T\| \leq \sqrt{tm + m^t s_0}$. In the real scheme, the vector $\hat{c}$ and the openings $v_1, \ldots, v_t$ are sampled according to $\text{SamplePre}(B_t, T, x \otimes e_1, s_1)$ according to Eq. (3.4). Suppose $m \geq m^t = O(n \log q)$ and

$$s_1 \geq \sqrt{(tm + m^t)tm'} \cdot \omega(\hat{G})(\hat{c}) = O(\ell^{3/2}m^{3/2} \log(nt) \cdot s_0).$$

By Theorem 2.8, the joint distribution of $(v_1, \ldots, v_t, \hat{c})$ is statistically close to $(B_t)^{-1}_s(x \otimes e_1)$.

- Let $\hat{A} = \text{diag}(A_1, \ldots, A_t)$. Then, $B_t = [\hat{A} | -1^t \otimes G]$. With overwhelming probability over the choice of $A_1, \ldots, A_t$, the columns of $\hat{A}$ generate $\mathbb{Z}_q^m$ by Corollary 2.3 and $\eta_i(\hat{A}) \leq \log(tm)$ by Lemma 2.5 for a negligible function $\epsilon = \text{negl}(n)$. Since $s_1 \geq \log(tm)$, by Lemma 2.7, the distribution of $(v_1, \ldots, v_t, \hat{c}) \leftarrow (B_t)^{-1}_s(x \otimes e_1)$ is statistically close to the distribution

$$\hat{c} \leftarrow D_{\mathbb{Z}_q^{m'_s}, (v_1, \ldots, v_t)} \leftarrow \hat{A}^{-1}_s\{-(x \otimes e_1) + (1^t \otimes \hat{G}c)\}.$$

- Since $\hat{A} = \text{diag}(A_1, \ldots, A_t)$, this means that each $v_i$ is distributed according to $(A_i)^{-1}_s(-x_i e_1 + \hat{G}c)$. By Theorem 2.8, this is statistically close to the distribution of $\text{SamplePre}(A_i, R_i, -x_i e_1 + \hat{G}c, s_1)$ (since $s_1 \geq \sqrt{mm'} \cdot \omega(\hat{G})(\hat{c}) = O(m \log n)$), which precisely coincides with the simulated distribution. $\square$
Parameter instantiation. Let $\lambda$ be a security parameter and $\ell$ be the vector dimension. We can instantiate the lattice parameters in Construction 3.9 to satisfy Theorems 3.10, 3.11, and 3.16:

- We set the lattice dimension $n = \lambda$ and $m = O(n \log q)$.
- We set $s_0 = O(\ell m \log(nt))$ and $s_1 = O(\ell^{3/2} m^{3/2} \log(nt) \cdot s_0) = O(\ell^{5/2} m^{5/2} \log^2(nt))$.
- We set the bound $B = \sqrt{tm + m'} \cdot s_1 = O(\ell^3 m^3 \log^2(nt)) = O(\ell^3 n^3 \log^2(nt) \log^3 q)$.
- Finally, we set the modulus $q = B \cdot \text{poly}(n)$ so that the SIS $n-1, m, q, 2B$ assumption holds. By Theorem 3.4, this implies the BASIS$_\text{rand}$ with parameters $(n - 1, m, q, 2B, s_0, \ell)$. In this case, $\log q = O(\log \lambda + \log \ell)$.

With this setting of parameters, we obtain a commitment scheme over $\mathbb{Z}_q^\ell$ with the following parameter sizes:

- **Commitment size:** A commitment $\sigma$ to a vector $x \in \mathbb{Z}_q^\ell$ consists of a vector $\sigma = c \in \mathbb{Z}_q^n$, so $|\sigma| = O(n \log q) = O(\lambda \cdot (\log \lambda + \log \ell))$.

- **Opening size:** An opening $\pi$ to index $i$ consists of a vector $\pi = v_i \in \mathbb{Z}_q^\ell$, where $\|v_i\| \leq B$. Thus, the length of $v_i$ is bounded by $O(m \log B)$, or equivalently, $|\pi| = O(\lambda \cdot (\log^2 \lambda + \log^2 \ell))$.

- **CRS size:** The CRS consists of $(A_i, \ldots, A_t, T)$, where $A_i \in \mathbb{Z}_q^n$ and $T \in \mathbb{Z}_q^{(\ell m + m') \times \ell m'}$. Thus, the total size of the CRS is $|\text{crs}| = \ell nm \log q + (\ell m + m') (\ell m') \log q = \ell^2 \cdot \text{poly}(\lambda, \log \ell)$.

Thus, Construction 3.9 is a succinct vector commitment scheme, and we obtain the following corollary:

**Corollary 3.17 (Vector Commitments with Private Openings from SIS).** Let $\lambda$ be a security parameter. Then, for all polynomials $t = t(\lambda)$, under the SIS assumption with a polynomial norm bound $\beta = \text{poly}(\lambda, \ell)$ and a polynomial modulus $q = \text{poly}(\lambda, \ell)$, there exists a vector commitment scheme over $\mathbb{Z}_q^\ell$ that is computationally binding and has statistically private openings. The size of a commitment to a vector $x \in \mathbb{Z}_q^\ell$ has size $O(\lambda(\log \lambda + \log \ell))$ and the openings have size $O(\lambda(\log^2 \lambda + \log^2 \ell))$. The size of the CRS is $\ell^2 \cdot \text{poly}(\lambda, \log \ell)$.

**Extensions: linear homomorphism and updatability.** Similar to the non-private scheme of Peikert et al. [PPS21], our vector commitment scheme is linearly homomorphic and supports stateless updates. We provide more details below:

**Remark 3.18 (Linear Homomorphism).** Like the vector commitment scheme of Peikert et al. [PPS21], Construction 3.9 has a linear verification procedure. This means that if $c$ is a commitment to a vector $x$ (with openings $v_1, \ldots, v_t$) and $c'$ is a commitment to a vector $x'$ (with opening $v'_1, \ldots, v'_t$), then $c + c'$ is a commitment to the vector $x + x' \in \mathbb{Z}_q^\ell$ with openings $v_1 + v'_1, \ldots, v_t + v'_t$. In particular, observe that for every $i \in [t]$, $c + c' = A_i(v_i + v'_i) + (x_i + x'_i)e_i$.

Note that the norm of the openings $\|v_i + v'_i\| \leq \|v_i\| + \|v'_i\| \leq 2B$ can now be as high as $2B$ rather than $B$. More generally, given a collection of $t$ vector commitments $c_1, \ldots, c_t$ to vectors $x_1, \ldots, x_t$, and a set of small coefficients $\alpha_1, \ldots, \alpha_t$, then $\sum_{i \in [t]} \alpha_i c_i$ is a commitment to $\sum_{i \in [t]} \alpha_i x_i$, with openings of size at most $\sum_{i \in [t]} \|\alpha_i\| \cdot B$. By setting the verification bound accordingly, the scheme supports a bounded number of linear operations on committed vectors. Note that we can also set the modulus $q$ and the norm bound $B$ to be super-polynomial in $\lambda$ (e.g., $2^{|\text{poly}(\lambda)|}$); then, the scheme supports an arbitrary polynomial number of homomorphic operations (with small coefficients). Security in this case relies on solving worst-case lattice problems to a super-polynomial approximation factor.

23
Remark 3.19 (Stateless Updates). Let $\sigma$ be a commitment to a vector $x \in \mathbb{Z}_q^l$ (with openings $\pi_1, \ldots, \pi_\ell$). Suppose we want to update $\sigma$ to a commitment $\sigma'$ of a vector $x' \in \mathbb{Z}_q^l$ that differs from $x$ at just a single index $i \in [\ell]$ (i.e., $x_j = x'_j$ for all $j \neq i$). A vector commitment scheme supports stateless updates [CPSZ18, PPS21] if it is possible to compute the new commitment $\sigma'$ from $\sigma$ given only the original commitment $\sigma$, the index $i$, the original value $x_i$, and the new value $x'_i$ (or even just the difference $x'_i - x_i \in \mathbb{Z}_q$). Notably, the update algorithm does not know $x_j$ for $j \neq i$. The update algorithm outputs a new commitment $\sigma'$ along with a set of opening updates $\delta'_1, \ldots, \delta'_\ell$. For all indices $i \in [\ell]$, it should be possible to compute an opening $\pi'_i$ to $\sigma'$ given the original opening $\pi_i$ and the opening update $\delta'_i$.

We can leverage the linear verification property (Remark 3.18) to support stateless updates. Suppose $c$ is a vector commitment to a vector $x \in \mathbb{Z}_q^l$ (with openings $v_1, \ldots, v_\ell$). To update $c$ to a commitment on $x'$, we first compute a commitment $c'$ to the difference $x' - x$. Let $v'_1, \ldots, v'_\ell$ be the openings to $x'_1 - x_1, \ldots, x'_\ell - x_\ell$. By linearity, $c + c'$ is a commitment to $x + (x' - x) = x'$ with openings given by $v_1 + v_1', \ldots, v_\ell + v_\ell'$. Observe that the update procedure only requires knowledge of the difference $x' - x$. Since Construction 3.9 supports a bounded number of homomorphic operations, it also supports a bounded number of updates. Namely, to support up to $k$ updates, we set the norm bound on the opening to be $kB$, and correspondingly, the size of the commitment and the openings scale with $\text{poly}(\log k)$. Similar to Remark 3.18, we can also set the norm bound and the modulus to be super-polynomial in order to support an arbitrary polynomial number of updates.

4 Succinct Functional Commitments for Circuits

In this section, we show how to obtain a succinct functional commitment for general circuits from the BASIS\textsubscript{struct} assumption. We consider schemes where the parameters scale with the depth of the Boolean circuit. We start with the formal definition:

Theorem 4.1 (Succinct Functional Commitment). Let $\lambda$ be a security parameter. Let $\mathcal{F} = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f \colon \{0, 1\}^\ell \rightarrow \{0, 1\}$ on inputs of length $\ell = (\ell(\lambda))$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. A succinct functional commitment for $\mathcal{F}$ is a tuple of efficient algorithms $\Pi_{\text{FC}} = (\text{Setup, Commit, Eval, Verify})$ with the following properties:

- **Setup($1^\lambda, 1^\ell, 1^d$) $\rightarrow$ crs:** On input the security parameter $\lambda$, the input length $\ell$, and the bound on the circuit depth $d$, the setup algorithm outputs a common reference string $\text{crs}$.

- **Commit($\text{crs, x}$) $\rightarrow$ ($\sigma, \text{st}$):** On input the common reference string $\text{crs}$ and an input $x \in \{0, 1\}^\ell$, the commitment algorithm outputs a commitment $\sigma$ and a state $\text{st}$.

- **Eval($\text{st, f}$) $\rightarrow$ $\pi_f$:** On input a commitment state $\text{st}$ and a function $f \in \mathcal{F}$, the evaluation algorithm outputs an opening $\pi_f$.

- **Verify($\text{crs, f, y, \pi}$) $\rightarrow$ $\{0, 1\}$:** On input the common reference string $\text{crs}$, a commitment $\sigma$, a function $f \in \mathcal{F}$, a value $y \in \{0, 1\}$, and an opening $\pi$, the verification algorithm outputs a bit $b \in \{0, 1\}$.

We now define several correctness and security properties on the functional commitment scheme:

- **Correctness:** For all security parameters $\lambda$, all functions $f \in \mathcal{F}$, and all inputs $x \in \{0, 1\}^\ell$,

$\Pr[\text{Verify}(\text{crs, f, y, \pi_f}) = 1 : (\sigma, \text{st}) \leftarrow \text{Commit(\text{crs, x})}; \pi_f \leftarrow \text{Eval(\text{st, f})}] = 1 - \text{negl}(\lambda).$

- **Succinctness:** The functional commitment scheme is succinct if there exists a universal polynomial $\text{poly}(\cdot, \cdot, \cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\sigma| = \text{poly}(\lambda, d, \log \ell)$ and $|\pi_f| = \text{poly}(\lambda, d, \log \ell)$ in the correctness definition.\footnote{We could consider an even stronger notion of succinctness where the size of the commitment and the opening depends polylogarithmically on the size of the Boolean circuits computing $\mathcal{F}$. However, like existing (non-succinct) lattice-based homomorphic commitments and signatures [GJVW18], the size of the commitment and openings in our construction scale with the depth of the computation.}
• **Binding:** We say $\Pi_{\text{FC}}$ satisfies statistical (resp., computational) binding if for all adversaries $\mathcal{A}$ (resp., efficient adversaries $\mathcal{A}$),

$$\Pr[\text{Verify} \left( \text{crs}, f, 0, \pi_0 \right) = 1 = \text{Verify} \left( \text{crs}, f, 1, \pi_1 \right) : \text{crs} \leftarrow \text{Setup} \left( 1^\lambda, 1^f, 1^{d\ell} \right); \left( \sigma, f, \pi_0, \pi_1 \right) \leftarrow \mathcal{A} \left( 1^\lambda, 1^f, 1^{d\ell}, \text{crs} \right)] = \negl(\lambda).$$

• **Private openings:** For an adversary $\mathcal{A}$ and a simulator $\mathcal{S} = (S_0, S_1)$, we start by defining two distributions $\text{Real}_{\mathcal{A}}(1^\lambda, 1^f, 1^{d\ell})$ and $\text{Ideal}_{\mathcal{A}, \mathcal{S}}(1^\lambda, 1^f, 1^{d\ell})$:

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Description</th>
</tr>
</thead>
</table>
| $\text{Real}_{\mathcal{A}}(1^\lambda, 1^f, 1^{d\ell})$ | 1. Give crs $\leftarrow$ Setup$(1^\lambda, 1^f, 1^{d\ell})$ to $\mathcal{A}$.  
2. Algorithm $\mathcal{A}$ outputs an input $x \in \{0, 1\}^f$.  
3. Compute $(\sigma, st) \leftarrow \text{Commit}$(crs, x) and give $\sigma$ to $\mathcal{A}$.  
4. Algorithm $\mathcal{A}$ outputs a function $f \in \mathcal{F}_f$.  
5. Give $\pi_f \leftarrow \text{Eval}(st, f)$ to $\mathcal{A}$.  
6. Algorithm $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ which is the output of the experiment. |
| $\text{Ideal}_{\mathcal{A}, \mathcal{S}}(1^\lambda, 1^f, 1^{d\ell})$ | 1. Sample $(\sigma, st) \leftarrow S_0(1^\lambda, 1^f, 1^{d\ell})$ and give crs to $\mathcal{A}$.  
2. Algorithm $\mathcal{A}$ outputs an input $x \in \{0, 1\}^f$.  
3. Give $\sigma$ to $\mathcal{A}$.  
4. Algorithm $\mathcal{A}$ outputs a function $f \in \mathcal{F}_f$.  
5. Compute $\pi_f \leftarrow S_1(st, f, f(x))$ and give $\pi_f$ to $\mathcal{A}$.  
6. Algorithm $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$ which is the output of the experiment. |

We say that $\Pi_{\text{FC}}$ has statistical (resp., computational) private openings if for all adversaries $\mathcal{A}$ (resp., efficient adversaries $\mathcal{A}$), there exists an efficient simulator $\mathcal{S} = (S_0, S_1)$ such that $\text{Real}_{\mathcal{A}}(1^\lambda, 1^f, 1^{d\ell})$ and $\text{Ideal}_{\mathcal{A}, \mathcal{S}}(1^\lambda, 1^f, 1^{d\ell})$ are statistically (resp., computationally) indistinguishable.

**Construction 4.2 (Succinct Functional Commitment).** Let $\lambda$ be a security parameter and $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$ be lattice parameters where $q$ is prime. Let $m' = n(\log q) + 1$ and $B = B(\lambda)$ be a bound. Let $s_0 = s_0(\lambda)$, $s_1 = s_1(\lambda)$ be Gaussian width parameters. Let $\mathcal{F} = \{ \mathcal{F}_i \}_{i \in \mathbb{N}}$ be a family of Boolean valued functions $f : \{0, 1\}^f \rightarrow \{0, 1\}$ where each function $f : \{0, 1\}^f \rightarrow \{0, 1\}$ is a function on inputs of length $\ell = \ell(\lambda)$ and which can be computed by a Boolean circuit of depth at most $d = d(\lambda)$. We construct a functional commitment $\Pi_{\text{FC}} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify})$ for $\mathcal{F}$ as follows:

- **Setup$(1^\lambda, 1^f, 1^{d\ell})$:** On input the security parameter $\lambda$, the input length $\ell$, and the bound $d$ on the circuit depth, the setup algorithm samples $(A, R) \leftarrow \text{TrapGen}(n, q, m)$ and for each $i \in \ell$, samples an invertible matrix $W_i \in \mathbb{Z}_{q}^{n \times m}$. Next, it computes $\hat{R}_i \leftarrow RG^{-1}(W_i^{-1}G) \in \mathbb{Z}_{q}^{nm' \times m'}$ for each $i \in \ell$ and constructs matrices $B_\ell$ and $R$ as follows:

\[
B_\ell = \begin{bmatrix}
W_1 A & \cdots & -G \\
\vdots & \ddots & \vdots \\
W_\ell A & -G
\end{bmatrix} \in \mathbb{Z}_{q}^{n \ell \times (m + m')} \quad \text{and} \quad \hat{R} = \begin{bmatrix}
\text{diag}(\hat{R}_1, \ldots, \hat{R}_\ell) \\
0^{m' \times (m + m')}
\end{bmatrix} \in \mathbb{Z}_{q}^{(m + m') \times \ell m'}. \tag{4.1}
\]

Finally, the setup algorithm samples $T \leftarrow \text{SamplePre}(B_\ell, \hat{R}, G_{n\ell}, s_0)$ and outputs the common reference string $\text{crs} = (A, W_1, \ldots, W_\ell, T)$.

- **Commit$(\text{crs}, x)$:** On input the common reference string $\text{crs} = (A, W_1, \ldots, W_\ell, T)$ and a vector $x \in \{0, 1\}^f$, the commit algorithm constructs $B_\ell$ from $A, W_1, \ldots, W_\ell$ according to Eq. (4.1). It then constructs a target matrix

\[
U_x = \begin{bmatrix}
-x_1 W_1 G \\
\vdots \\
-x_\ell W_\ell G
\end{bmatrix} \in \mathbb{Z}_{q}^{n \ell \times m'}. \tag{4.2}
\]

It then uses $T$ to sample a preimage

\[
\begin{bmatrix}
V_1 \\
\vdots \\
V_\ell \\
\hat{C}
\end{bmatrix} \leftarrow \text{SamplePre}(B_\ell, T, U_x, s_1). \tag{4.3}
\]

It outputs the commitment $\sigma = C = G\hat{C} \in \mathbb{Z}_{q}^{n \times m'}$ and the state $\text{st} = (x, C, V_1, \ldots, V_\ell)$.  

25
• Eval(crs, st, f): On input the common reference string \(\text{crs} = (A, W_1, \ldots, W_t, T)\), a commitment state \(s = (x, C, \tilde{x}_1, \ldots, \tilde{x}_t)\), and a function \(f : \{0, 1\}^t \rightarrow \{0, 1\}\), the evaluation algorithm sets \(\tilde{C} \leftarrow [W_i^{-1}C | \cdots | W_t^{-1}C]\), computes \(H_{\tilde{c},f,x} \leftarrow \text{EvalFX}(\tilde{C}, f, x)\), and outputs the opening \(\pi_f = V_f \leftarrow [V_1 | \cdots | V_t] \cdot H_{\tilde{c},f,x}\).

• Verify(crs, f, y, π): On input the common reference string \(\text{crs} = (A, W_1, \ldots, W_t, T)\), a commitment \(\sigma = C = G\tilde{C} = Z_n^{n \times m'}\) and state \(s = (x, C, \tilde{x}_1, \ldots, \tilde{x}_t)\), the verification algorithm sets \(\tilde{C} \leftarrow [W_i^{-1}C | \cdots | W_t^{-1}C]\), computes \(\tilde{C}_f \leftarrow \text{EvalF}(\tilde{C}, f)\) and outputs 1 if

\[
\| \tilde{V}_f \| \leq B \quad \text{and} \quad AV_f = \tilde{C}_f - yG.
\] (4.4)

**Theorem 4.3** (Correctness). Suppose \(n \geq \lambda, m \geq O(n \log q), s_0 \geq O(\log^2 m)\) and \(B \geq O(n \log^2 m)\). Then, Construction 4.2 is correct.

**Proof.** Take a security parameter \(\lambda\), a function \(f \in \mathcal{F}_n\), and an input \(x \in \{0, 1\}^t\). Let \(\text{crs} = (A, W_1, \ldots, W_t, T) \leftarrow \text{Setup}(\lambda, \lambda', \lambda'')\) and \((\sigma, st) \leftarrow \text{Commit}(\text{crs}, x)\) where \(\sigma = C = G\tilde{C} = Z_n^{n \times m'}\) and state \(s = (x, C, \tilde{x}_1, \ldots, \tilde{x}_t)\). Let \(\pi = V_f \leftarrow \text{Eval}(\text{crs}, st, f)\) and consider Verify(crs, f, y, π):

- Let \(B_f \in Z_q^{n \times (m + m')}\) and \(R \in Z_q^{n \times (m + m')}\) be the matrices from Eq. (4.1). For each \(i \in [t]\), we have that \(B_i = G\tilde{R}\) and \(\|R\| = \max_{i \in [t]} \|\tilde{R}_i\| \leq m' = O(n \log q)\).

- Suppose \(m \geq m' = O(n \log q)\). By Theorem 2.8 and Lemmas 2.4 and 2.6, for sufficiently-large \(m \geq O(n \log q)\),

\[
s_0 \geq \sqrt{(\log(\|\tilde{R}\|) \cdot \log(\|\tilde{R}\|))} = O(m' \log(n)) \quad \text{with overwhelming probability.}
\]

- Suppose \(s_1 \geq \sqrt{(\log(\|\tilde{R}\|) \cdot \log(\|\tilde{R}\|))} = O(m' \log(n))\). By Theorem 2.8 and by construction of \((V_1, \ldots, V_t, C)\) in Eq. (4.3), we have

\[
W_iAV_i - C = W_iAV_i - G\tilde{C} = -\tilde{x}_iW_iG.
\]

Rearranging, we obtain \(AV_i = W_i^{-1}C - x_iG\). Let \(\tilde{C} = [W_i^{-1}C | \cdots | W_t^{-1}C]\) and \(\tilde{V} = [V_1 | \cdots | V_t]\). Then,

\[
\tilde{C} - x^T \otimes G = A[V_1 | \cdots | V_t] = AV.
\] (4.5)

Let \(B_0 = \sqrt{\log m} \cdot s_1\) be the "initial" noise bound. By Lemmas 2.4 and 2.6, \(\|V_i\| \leq \sqrt{\log m} \cdot s_1 = B_0\) and so \(\|\tilde{V}\| \leq B_0\).

- By construction of Eval, we have that \(V_f = \tilde{V} \cdot H_{\tilde{c},f,x}\) where \(H_{\tilde{c},f,x} \leftarrow \text{EvalFX}(\tilde{C}, f, x)\). By Theorem 2.11,

\[
\|H_{\tilde{c},f,x}\| \leq (\log q)\cdot \sigma, \quad \text{so} \quad \|V_f\| \leq m' \cdot B_0 \cdot (\log q)\cdot \sigma \leq s_1 \cdot \sqrt{\log m} \cdot (\log q)\cdot \sigma.
\]

- Again appealing to Theorem 2.11 and Eq. (4.5), we can write

\[
AV_f = AVH_{\tilde{c},f,x} = (\tilde{C} - x^T \otimes G) \cdot H_{\tilde{c},f,x} = \tilde{C}_f - f(x) \cdot G,
\]

where \(\tilde{C} = \text{EvalF}(\tilde{C}, f)\). Correspondingly, Verify(crs, f, y, π) outputs 1. \(\square\)

**Theorem 4.4** (Binding). Suppose \(n \geq \lambda, m \geq O(n \log q), s_0 \geq O(m^2 \log(n))\), and \(\beta \geq 2Bm\sqrt{m'}\log n\). Then, under the BASISstructure assumption with parameters \((n, m, q, \beta, s_0, t)\), Construction 4.2 is computationally binding.

**Proof.** We proceed via a hybrid argument:

- Hybrid: This is the real binding experiment.
As shown in the proof of Theorem 4.3, probability is a trapdoor for $A$

By construction, algorithm $B$.

Suppose Lemma 4.7.

Proof. The only difference between Lemma 4.6. By Theorem 2.8 the distribution of $A$

Suppose $A$ to construct an adversary $B$. The CRS is now sampled independently of $R$.

• $Hyb_1$: Same as $Hyb_0$ except after constructing the matrix $B_t \in \mathbb{Z}_q^{n \times (tm + m')}$ according to Eq. (4.1), the challenger samples $T \leftarrow (B_t)^{-1}(G_{nt})$ without using the trapdoor $R$. The CRS is now sampled independently of $R$.

• $Hyb_2$: Same as $Hyb_1$ except the challenger samples $A \leftarrow \mathbb{Z}_q^{n \times m}$.

For an adversary $A$, we write $Hyb_1(A)$ to denote the output of an execution of $Hyb_1$ with adversary $A$. We now show that each adjacent pair of experiments are computationally indistinguishable.

**Lemma 4.5.** Suppose $n \geq \lambda$, $m \geq O(n \log q)$, and $s_0 \geq O(tm^2 \log(nt))$. Then, for all adversaries $A$, $Hyb_0(A) \approx Hyb_1(A)$.

Proof. The only difference between $Hyb_0$ and $Hyb_1$ is the distribution of $T$. In $Hyb_0$, $T \leftarrow \text{SamplePre}(B_t, \tilde{R}, G_{nt}, s_0)$. As shown in the proof of Theorem 4.3, $B_t \tilde{R} = G_{nt}$ and $||R|| \leq m' = O(n \log q)$. Suppose $m \geq m' = O(n \log q)$ and $s_0 \geq \sqrt{(tm + m')tm'}||R|| \cdot \omega(\sqrt{\log(nt)}) = O(tm^2 \log(nt))$.

By Theorem 2.8 the distribution of $T$ is statistically close to $T \leftarrow (B_t)^{-1}(G_{nt})$, which is the distribution in $Hyb_1$. \(\square\)

**Lemma 4.6.** Suppose $n \geq \lambda$ and $m \geq O(n \log q)$. Then, for all adversaries $A$, $Hyb_1(A) \approx Hyb_2(A)$.

Proof. The only difference between $Hyb_2$ and $Hyb_1$ is the distribution of $A$. In $Hyb_1$, the challenger samples $(A, R) \leftarrow \text{TranGen}(1^n, q, m)$. By Theorem 2.8, the distribution of $A$ is statistically close to $A \leftarrow \mathbb{Z}_q^{n \times m}$. \(\square\)

**Lemma 4.7.** Suppose $\beta \geq 2Bm \sqrt{tm} \log n$. Under the BASIS$_{struct}$ assumption with parameters $(n, m, q, \beta, s_0, \ell)$, for all efficient adversaries $A$, $\Pr[Hyb_2(A) = 1] = \epsilon$ for some non-negligible $\epsilon$. We use $A$ to construct an adversary $B$ for the BASIS$_{struct}$ assumption:

1. Algorithm $B$ receives a challenge $A \in \mathbb{Z}_q^{n \times m}$, $B_t \in \mathbb{Z}_q^{n \times (tm + m')}$, $T \in \mathbb{Z}_q^{(tm + m') \times tm'}$, and $aux = (W_1, \ldots, W_t)$.

   Algorithm $B$ gives $\text{crs} = (A, W_1, \ldots, W_t, T)$ to $A$.

2. Algorithm $A$ outputs a commitment $C \in \mathbb{Z}_q^{n \times m'}$, a function $f \in \mathcal{F}_\lambda$, and openings $V_0, V_1 \in \mathbb{Z}_q^{m \times m'}$.

3. Algorithm $B$ outputs $x \leftarrow \text{SamplePre}(A, V_0 - V_1, 0, s')$ where $s' = 2Bm \sqrt{tm} \log n$.

   By construction, algorithm $B$ perfectly simulates the common reference string according to the specification of $Hyb_2$. Thus, with probability $\epsilon$, $||V_0||, ||V_1|| \leq B$, $AV_0 = \tilde{C}_f$, $AV_1 = \tilde{C}_f - G$. This means that $A(V_0 - V_1) = G$, so $V_0 - V_1$ is a trapdoor for $A$. By Theorem 2.8, the distribution of $x$ is statistically close to $A^{\cdot T}(0)$. Thus, $x$ is non-zero with probability $1 - negl(n)$. Finally, by Lemma 2.6, $||x|| \leq \sqrt{m\beta}$, and the claim holds. \(\square\)

Combining Lemmas 4.5 to 4.7, the functional commitment scheme is computationally binding. \(\square\)
Parameter instantiation. Let $\lambda$ be a security parameter and $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ on inputs of length $\ell = \ell(\lambda)$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. We instantiate the lattice parameters in Construction 4.2 as follows to satisfy Theorems 4.3 and 4.4:

- Let $\epsilon > 0$ be a constant. We set the lattice dimension $n = d^{1/\epsilon} \cdot \text{poly}(\lambda)$ and $m = O(n \log q)$.
- We set $s_0 = O(\ell m^2 \log(\ell n))$ and
  $$s_1 = O(\ell^{3/2} m^{3/2} n \log(\ell m) \cdot s_0) = O(\ell^{5/2} m^{7/2} n \log^2(\ell m)) = O(\ell^{5/2} n^{7/2} \log^2(n) \log^{7/2} q).$$
- We set the bound $B = s_1 \cdot \sqrt{\ell m + m^2} \cdot (n \log q)^{O(d)} = \ell^3 \log^2 \ell \cdot (n \log q)^{O(d)}$.
- We set the modulus $q$ so that the BASIS$_{\text{struct}}$ assumption holds with parameters $(n, m, q, \beta, s_0, \ell)$, where
  $$\beta = 2Bm\sqrt{m'} \log n = \ell^3 \log^2 \ell \cdot (n \log q)^{O(d)} = 2^{O(d)} = 2^{O(\ell n')},$$
  where we write $\tilde{O}(\cdot)$ to suppress polylogarithmic factors in $\lambda, d, \ell$. Note that this also requires that SIS$_{n,m,q,\beta}$ hold. For instance, we set $q = \beta \cdot \text{poly}(n)$. Then, $\log q = \text{poly}(d, \log \lambda, \log \ell)$. Note that the underlying SIS assumption now relies on a sub-exponential noise bound.

With this setting of parameters, we obtain a functional commitment scheme for $F$ with the following parameter sizes:

- **Commitment size:** A commitment $\sigma$ to an input $x \in \{0, 1\}^\ell$ consists of a matrix $\sigma = C \in \mathbb{Z}_q^{n \times m'}$ so
  $$|\sigma| = nm' \log q = O(n^2 \log^2 q) = \text{poly}(\lambda, d, \log \ell).$$

- **Opening size:** An opening $\pi$ to a function $f$ consists of a matrix $\pi = V_f \in \mathbb{Z}_q^{m \times m'}$. Then,
  $$|\pi| = mm' \log q = \text{poly}(\lambda, d, \log \ell).$$

In Remark 4.9, we describe a simple approach to compress the opening to a vector instead of a matrix.

- **CRS size:** The CRS consists of $(A, W_1, \ldots, W_f, T)$, where $A \in \mathbb{Z}_q^{n \times m}$, $W_i \in \mathbb{Z}_q^{n \times n}$, and $T \in \mathbb{Z}_q^{\ell \cdot (m + m') \times m'}$. Thus, the total size of the CRS is
  $$|\text{crs}| = nm \log q + \ell n^2 \log q + (\ell m + m')(\ell m') \log q = \ell^2 \cdot \text{poly}(\lambda, d, \log \ell).$$

Thus, Construction 4.2 is a succinct functional commitment scheme for bounded-depth circuits. We summarize the instantiation in the following corollary:

**Corollary 4.8 (Succinct Vector Commitment from BASIS$_{\text{struct}}$).** Let $\lambda$ be a security parameter, and let $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ on inputs of length $\ell = \ell(\lambda)$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. Under the BASIS$_{\text{struct}}$ assumption with a norm bound $\beta = 2^{O(d)}$ and modulus $q = 2^{O(d)}$, there exists a computationally-binding succinct functional commitment scheme for $F$. Both the size of the commitment and the opening are $\text{poly}(\lambda, d, \log \ell)$, and the CRS has size $\ell^2 \cdot \text{poly}(\lambda, d, \log \ell)$. Here, $\tilde{O}(\cdot)$ suppresses polylogarithmic factors in $\lambda, d, \ell$.

**Remark 4.9 (Reducing the Opening Size).** An opening $\pi_f$ to a function $f$ in Construction 4.2 consists of a matrix $\pi_f = V_f \in \mathbb{Z}_q^{m \times m'}$ where $m, m' = O(n \log q)$. It is easy to adapt Construction 4.2 to obtain slightly shorter openings (i.e., $\pi_f = V_f \in \mathbb{Z}_q^{n \times n}$). The idea is simple: we publish a random target vector $u \in \mathbb{Z}_q^n$ in the CRS and define the new opening to be $\pi_f \leftarrow V_f G^{-1}(u)$, where $V_f$ is the original opening from Construction 4.2. The updated verification relation then checks that $||V_f||$ is small and that $A v_f = CG^{-1}(u) - y \cdot u$. We also use this approach to aggregate openings in Section 5.1 (see Construction 5.16). However, giving out the "matrix" opening is convenient when specializing our construction to obtain polynomial commitments (Section 4.1).
Remark 4.10 (Comparison with [ACL+22]). The authors of [ACL+22] showed how to construct a functional commitment for constant-degree polynomials where the size of the CRS scales exponentially with the degree of the polynomial. Our functional commitment scheme (Construction 4.2) supports arbitrary Boolean circuits of bounded depth, and the size of our CRS scales polynomially with the depth of the circuit family. Moreover, security of our construction can be reduced to a function-independent assumption (the BASISstruct assumption) whereas the scheme in [ACL+22] relied on a function-dependent assumption. We compare the two assumptions in more detail in Section 6.

An advantage of the [ACL+22] construction is that if supports fast verification with preprocessing. Namely, in their scheme, the verifier can precompute a verification key for a function $f$, and subsequently, verify openings with respect to $f$ in time that is polylogarithmic in the running time of $f$. In contrast, with our scheme, the verifier has to first homomorphically compute $f$ on the commitment in order to verify. In Section 4.1, we show that for the special case of linear functions, we can adapt Construction 4.2 to support fast verification in the preprocessing model.

Remark 4.11 (A Candidate SNARG with Expensive Verification). The authors of [ACL+22] show how to boost their functional commitment for quadratic polynomials to a preprocessing SNARG for NP as follows:

1. First, [ACL+22] applying “sparsification” to their functional commitment scheme. Over the integers, one analog of sparsification is to require the adversary to output a short $V$ such that $AV = DC$, where $D \in \mathbb{Z}_q^{2m \times n}$, where $\mathbb{A} \in \mathbb{Z}_q^{2m \times 2m \log q}$ is a sparsification matrix. The CRS includes short preimages of $\mathbb{A}$ to enable sampling of $V$.

2. Next, [ACL+22] introduce a knowledge assumption that says that the only way an adversary can produce $C$ and $V$ by computing a short linear combination of the preimages in the CRS (where the coefficients correspond to the committed vector $x$).

3. To support openings to multiple quadratic polynomials with a succinct opening (i.e., sublinear in the number of openings), [ACL+22] introduces a novel SIS-based technique.

Taken together, [ACL+22] show how to obtain an “extractable” commitment scheme, a notion that is equivalent to a succinct argument of knowledge for satisfiability of quadratic systems. This yields a publicly-verifiable preprocessing SNARG for NP since satisfiability of degree-2 polynomials is NP-complete. Specifically, a proof for a statement $x$ consists of a commitment $\sigma$ to a satisfying witness $w$ and an opening $\pi$ of $\sigma$ to a satisfying assignment to the quadratic constraint system representing the NP relation. By relying on preprocessing (Remark 4.10), the [ACL+22] SNARG has short proofs and fast verification.

We can apply an analogous approach to our functional commitment scheme (Construction 4.2) to obtain a candidate SNARG for NP; our SNARG would have short proofs but an expensive verification step (since our functional commitment does not support fast verification in the preprocessing model). We also note that even without sparsification, our construction is still a candidate SNARG: we do not know how to prove soundness of our construction, but at the same time, are not aware of any attacks either. An attack on our candidate SNARG (without sparsification) would be interesting, and we invite cryptanalysis of our candidate.

Remark 4.12 (Deterministic Commitments). Here, we describe a variant of Construction 4.2 with a deterministic commitment algorithm. The advantage of this approach is that computing the commitment $\sigma = C$ now runs in time $t \cdot \text{poly}(\lambda, d, \log t)$ time; with preimage sampling, the same algorithm requires time that scales quadratically with the dimension $t$. The drawback of using a deterministic commitment procedure is that the scheme no longer extends to support private openings (see Section 4.2). To support deterministic commitments, we proceed as follows:

- In Setup, construct the matrix $B_t$ exactly as in Eq. (4.1), and define the target matrix

$$
\tilde{U} = \begin{bmatrix} -W_tG & \cdots & \cdots & -W_tG \end{bmatrix} \in \mathbb{Z}_q^{m t \times m'},
$$

Then, the Setup algorithm constructs the matrix $T = \begin{bmatrix} T_{\text{open}} \\ T_{\text{com}} \end{bmatrix}$ as

$$
\begin{bmatrix} T_{\text{open}} \\ T_{\text{com}} \end{bmatrix} \leftarrow \text{SamplePre} \left( B_t, \tilde{R}, \tilde{U}, s_1 \right),
$$
where $T_{\text{open}} \in \mathbb{Z}_q^{n \times tm'}$ and $T_{\text{com}} \in \mathbb{Z}_q^{n \times tm}$.

- To commit to a vector $x \in \{0, 1\}^\ell$, the Commit algorithm computes $\hat{C} = T_{\text{com}}(x \otimes I_{m'})$ and sets $\sigma = C = G\hat{C}$.
- To evaluate a function $f$ on a commitment to $x$, the Eval algorithm first computes
  \[
  \begin{bmatrix}
  V_1 \\
  \vdots \\
  V_\ell
  \end{bmatrix} \leftarrow T_{\text{open}}(x \otimes I_{m'})
  \]
  and then proceeds exactly as in Construction 4.2.

To see that this preserves correctness, observe that
\[
B_\ell \cdot \begin{bmatrix}
V_1 \\
\vdots \\
V_\ell
\end{bmatrix} = B_\ell T(x \otimes I_{m'}) = \tilde{U}(x \otimes I_{m'}) = \begin{bmatrix}
-x_1 W_1 G \\
\vdots \\
-x_\ell W_\ell G
\end{bmatrix} = U_x.
\]

Thus, $V_1, \ldots, V_\ell, \hat{C}$ is a short solution to the same linear system as in Eq. (4.3). Correctness follows by the same analysis as in Theorem 4.3 (with slightly larger bounds). Security follows similarly to the proof of Theorem 4.4 (since $T_{\text{com}}, T_{\text{open}}$ can be derived from the CRS components in Construction 4.2). Observe that with this deterministic commitment and opening procedures and using the same parameter instantiation as in Corollary 4.8, the running time $\text{Commit}$ is $\ell \cdot \text{poly}(\lambda, d, \log \ell)$ and the running time of $\text{Eval}$ (on function $f$) is $|f| \cdot \text{poly}(\lambda, d, \log \ell)$.

**Remark 4.13 (An Alternative Formulation of Construction 4.2).** We can obtain an alternative formulation of Construction 4.2 that does not require explicit matrix inverses by starting with the alternative version of BASIS struct in Remark 3.8. Specifically, we define the matrix
\[
\tilde{B}_\ell = [I_\ell \otimes A | \tilde{W} G]
\]
where $\tilde{W} = \begin{bmatrix}
\tilde{W}_1 \\
\vdots \\
\tilde{W}_\ell
\end{bmatrix}$ and $\tilde{W}_i \in \mathbb{Z}_q^{nxn}$.

Let $T$ be a trapdoor for $\tilde{B}_\ell$. The CRS is then $(A, \tilde{W}_1, \ldots, \tilde{W}_\ell, T)$. To commit to an input $x$, the prover now computes
\[
\begin{bmatrix}
V_1 \\
\vdots \\
V_\ell
\end{bmatrix} \leftarrow \text{SamplePre}(\tilde{B}_\ell, T, -x \otimes G, s_1).
\]

By construction of $\tilde{B}_\ell$ and again setting $C = G\hat{C}$, we have for all $i \in [\ell]$,
\[
AV_i + \tilde{W}_i C = -x_i G,
\]
or equivalently, $AV_i = -\tilde{W}_i C - x_i G$. Letting $\tilde{C} = [-\tilde{W}_1 C | \cdots | -\tilde{W}_\ell C]$ and $\tilde{V} = [\tilde{V}_1 | \cdots | \tilde{V}_\ell]$, we can again write
\[
A\tilde{V} = \tilde{C} - x^T \otimes G,
\]
from which correctness follows as in the proof of Theorem 4.3. The binding analysis follows the same structure as the proof of Theorem 4.4, except we rely on the variant of BASIS struct where $T$ is a trapdoor for $\tilde{B}_\ell$. For the particular case where $q$ is prime (i.e., the setting where $\tilde{W}_1, \ldots, \tilde{W}_\ell$ are invertible with overwhelming probability), these two variants of BASIS struct are equivalent (see Remark 3.8).
4.1 Opening to Linear Functions and Applications to Polynomial Commitments

An appealing property of the succinct functional commitment by Albrecht et al. [ACL*22] and the non-succinct construction of Gorbunov et al. [GVW15] is they support fast verification after an initial slow preprocessing step. In these schemes, there is a preprocessing algorithm that takes the CRS and the description of a function \( f \), and outputs a short verification key \( vk_f \). Later on, given a commitment \( \sigma \), an opening \( \pi_f \), and the precomputed verification key \( vk_f \), the verification algorithm runs in time that is sublinear in the time it takes to compute \( f \). In contrast, our succinct functional commitment (Construction 4.2) has short commitments and openings, but the verification algorithm runs in time proportional to \( f \) (in order to homomorphically compute \( C_f \) from \( C \)).

Here, we describe a simple adaptation of Construction 4.2 for the setting of linear functions that supports fast verification in the preprocessing model. Our construction naturally supports linear functions over \( \mathbb{Z}_q \geq z \rightarrow \pi \). Let \( \mathcal{F} = \{ f \} \rightarrow \mathbb{Z}_q \). We construct a functional commitment \( \Pi_f \) = (Setup, Commit, Eval, Verify) for \( \mathcal{F} = \{ f \} \rightarrow \mathbb{Z}_q \) as follows:

- **Setup**(\( 1^\lambda, 1^\ell \)): Same as Construction 4.2.

- **Commit(crs, x)**: Same as Construction 4.2 (with \( x \in \mathbb{Z}_q^\ell \)).

- **Eval(crs, st, f_z)**: On input the common reference string \( crs = (A, W_1, \ldots, W_{\ell}, T) \), a commitment state \( st = (x, C, V_1, \ldots, V_\ell) \), and a function \( f_z \) where \( z \in \{0,1\}^\ell \), the evaluation algorithm outputs \( V_z = \sum_{i=1}^{\ell} z_i V_i \).

- **Verify(crs, \sigma, f_z, y, \pi)**: On input the common reference string \( crs = (A, W_1, \ldots, W_{\ell}, T) \), a commitment \( \sigma = C \in \mathbb{Z}_q^{n \times m'} \), a function \( f_z : \mathbb{Z}_q^{\ell} \rightarrow \mathbb{Z}_q \) where \( z \in \{0,1\}^\ell \), a value \( y \in \mathbb{Z}_q \), and an opening \( \pi = V_z \in \mathbb{Z}_q^{m \times m'} \), the verification algorithm outputs 1 if

\[
\|V_z\| \leq B \quad \text{and} \quad AV_z = \sum_{i=1}^{\ell} z_i W_i^{-1} C - y G.
\]

**Remark 4.15** (Verification in the Preprocessing Model). As described, the running time of the verification algorithm Construction 4.14 is linear in the vector dimension \( \ell \). However, observe that the verifier can precompute the matrix \( W_z := \sum_{i=1}^{\ell} z_i W_i^{-1} \), which only depends on the CRS and the function \( z \in \{0,1\}^\ell \). Given the precomputed verification key \( W_z \), the verification relation becomes:

\[
\|V_z\| \leq B \quad \text{and} \quad AV_z = W_z C - y G.
\]

This yields a verification algorithm whose running time is \( \text{poly}(\lambda, \log \ell) \).

**Theorem 4.16** (Correctness). Suppose \( n \geq \lambda, m \geq O(n \log q) \), \( s_0 \geq O(\ell m^2 \log(n\ell)) \), \( s_1 \geq O(\ell^{3/2} m^{3/2} \log(n\ell) \cdot s_0) \) and \( B \geq \ell \sqrt{m + m'} \cdot s_1 \). Then, Construction 4.14 is correct.

**Proof.** This follows by a similar argument as the proof of Theorem 4.3. Take a security parameter \( \lambda \), a vector \( z \in \{0,1\}^\ell \), and an input \( x \in \mathbb{Z}_q^\ell \). Let \( crs = (A, W_1, \ldots, W_{\ell}, T) \leftrightarrow Setup(1^\lambda, 1^\ell) \) and \( (\sigma, st) \leftrightarrow \text{Commit(crs, x)} \) where \( \sigma = C = G \hat{C} = \mathbb{Z}_q^{n \times m'} \) and \( st = (x, C, V_1, \ldots, V_\ell) \). Let \( \pi = V_z \leftrightarrow \text{Eval(crs, st, f_z)} \) and consider Verify(crs, \( \sigma, f_z, z^T \cdot x, \pi \)):

- By the analysis in the proof of Theorem 4.3, we have \( AV_i = W_i^{-1} C - x_i G \) where \( \|V_i\| \leq \sqrt{m + m'} s_1 \).
• By construction of Eval, we have \( V_z = \sum_{i \in [t]} z_i V_i \). Since \( z_i \in \{0, 1\} \), this means \( \|V_z\| \leq t \sqrt{m + m's_1} \leq B \). Moreover,
\[
AV_z = \sum_{i \in [t]} z_i AV_i = \sum_{i \in [t]} z_i W_i^{-1}C - \sum_{i \in [t]} z_i x_i G = \sum_{i \in [t]} z_i W_i^{-1}C - (z'x)_i G,
\]
and the verification algorithm accepts.

\[\square\]

**Theorem 4.17** (Binding). Suppose \( n \geq 1, m \geq O(n \log q), s_0 \geq O(\log n t) \), \( q \) is prime, and \( \beta \geq 2B \sqrt{m't} \log n \). Then, under the BASIS\text{struct} assumption with parameters \((n, m, q, \beta, s_0, t)\), Construction 4.14 is computationally binding.

**Proof.** We use the same sequence of hybrid experiments as in the proof of Theorem 4.4.

- **Hyb\(_0\):** This is the real binding experiment.
- **Hyb\(_1\):** Same as Hyb\(_0\) except the challenger samples \( T \leftarrow (B_t)_{t=1}^n (G_{nt}) \) when setting up the CRS.
- **Hyb\(_2\):** Same as Hyb\(_1\) except the challenger samples \( A \leftarrow \mathbb{F}_{q^m}^{n \times m} \) in the CRS.

For the given choice of parameters, and for all adversaries \( \mathcal{A} \), we have that \( \text{Hyb\(_0\)}(\mathcal{A}) \approx \text{Hyb\(_1\)}(\mathcal{A}) \) and \( \text{Hyb\(_1\)}(\mathcal{A}) \approx \text{Hyb\(_2\)}(\mathcal{A}) \) using the same argument as in the proofs of Lemmas 4.5 and 4.6. It suffices to consider the output distribution in Hyb\(_2\):

**Lemma 4.18.** Suppose \( q \) is prime and \( \beta \geq 2B \sqrt{m't} \log n \). Under the BASIS\text{struct} assumption with parameters \((n, m, q, \beta, s_0, t)\), for all efficient adversaries \( \mathcal{A} \), \( \Pr[\text{Hyb\(_2\)}(\mathcal{A}) = 1] = \varepsilon \) for some non-negligible \( \varepsilon \). We use \( \mathcal{A} \) to construct an adversary \( \mathcal{B} \) for the BASIS\text{struct} assumption in a similar manner as the proof of Lemma 4.7:

1. Algorithm \( \mathcal{B} \) receives a challenge \( A \in \mathbb{Z}_q^{n \times m}, B_t \in \mathbb{Z}_q^{n \times (tm + m')}, T \in \mathbb{Z}_q^{(tm + m') \times m'} \), and aux = \((W_1, \ldots, W_t)\). Algorithm \( \mathcal{B} \) gives \( \text{crs} = (A, W_1, \ldots, W_t, T) \) to \( \mathcal{A} \).
2. Algorithm \( \mathcal{A} \) outputs a commitment \( C \in \mathbb{Z}_q^{n \times m'} \), a function \( f_z : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q \), where \( z \in \{0, 1\}^t \), and openings \((y_1, V_1), (y_2, V_2)\), where \( y_1 \neq y_2 \in \mathbb{Z}_q \) and \( V_1, V_2 \in \mathbb{Z}_q^{m \times m'} \).
3. Algorithm \( \mathcal{B} \) outputs \( x \leftarrow \text{SamplePre}(A, V_1 - V_2, 0, s') \), where \( s' = 2B \sqrt{m't} \log n \)

By construction, algorithm \( \mathcal{B} \) perfectly simulates the common reference string according to the specification of Hyb\(_2\). Thus, with probability \( \varepsilon \), the following conditions hold: \( \|V_1\|, \|V_2\| \leq B \), and \( A(V_1 - V_2) = (y_2 - y_1)G \). Thus, \( V_1 - V_2 \) is a trapdoor for \( A \) with tag \((y_2 - y_1)I_n \). Since \( q \) is prime and \( y_2 - y_1 \neq 0 \), the matrix \((y_2 - y_1)I_n \) is invertible. Thus, by Theorem 2.8, the distribution of \( x \) is statistically close to \( A^{−1}(0) \). Thus, \( x \) is non-zero with overwhelming probability. Finally, by Lemma 2.6 \( |x|| \leq \sqrt{m's} = \beta \), and the claim holds.

\[\square\]

Computational binding now follows by a hybrid argument.

**Remark 4.19** (Extending to General \( \mathbb{Z}_q \)-Linear Functions). While Construction 4.14 supports commitments to arbitrary vectors over \( \mathbb{Z}_q \), we still require that the functions \( f_z \) have small coefficients. We can support arbitrary linear functions on \( \mathbb{Z}_q^t \) by blowing up the vector dimension by a factor \( k = [\log q] + 1 = O(\log q) \) factor:

- The CRS is a CRS for a vector commitment scheme on dimension \( tk = O(t \log q) \): \( \text{crs} \leftarrow \text{Setup}(1^t, 1^{tk}) \).
- A commitment to a vector \( x \in \mathbb{Z}_q^t \) is a commitment to the vector \( x \otimes g \in \mathbb{Z}_q^{tk} \) in the underlying scheme: \((\sigma, st) \leftarrow \text{Commit}(\text{crs}, x \otimes g) \).
- An opening to the linear function \( f_z : \mathbb{Z}_q^t \rightarrow \mathbb{Z}_q \) for an arbitrary \( z \in \mathbb{Z}_q^t \) is an opening with respect to the linear function \( f_{g^{-1}(z)} \) in the underlying scheme: \( \pi \leftarrow \text{Eval}(\text{crs}, st, f_{g^{-1}(z)}) \).
By construction, \( g^{-1}(z) \in \{0, 1\}^d \). For correctness, observe that \((x \otimes g)^{-1}(z) = x'z = z'x \). Thus, with \( O(\log q) \) overhead, Construction 4.14 gives a vector commitment scheme for arbitrary linear functions over \( \mathbb{Z}_q^t \). We no longer impose any size restrictions on the coefficients of the linear function or on the vector values.

**Remark 4.20 (Polynomial Commitments).** In a polynomial commitment [KZG10], a committer can commit to a polynomial \( h \in \mathbb{Z}_q[x] \) over \( \mathbb{Z}_q \) and subsequently open to evaluations \( h(x) \) of the polynomial on arbitrary points \( x \in \mathbb{Z}_q \) of the committer’s choosing. The succinctness requirement is that the commitment and the openings are both short (sublinear in the degree of the polynomial). By the approach of Libert et al. [LRY16], our functional commitment for linear functions over \( \mathbb{Z}_q \) directly implies a polynomial commitment. Namely, to commit to a degree-\( d \) polynomial \( h(x) = \sum_{i=0}^d h_i x^i \), the committer constructs a commitment to the vector of coefficients \( \mathbf{h} = [h_0, \ldots, h_d] \). To open to a point \( x \in \mathbb{Z}_q \), the committer constructs an opening to the function \( f_x \) where \( x = [1, x, x^2, \ldots, x^d] \). By construction \( x'\mathbf{h} = h(x) \). When instantiated with our vector commitment scheme for linear functions (Construction 4.14 and Remark 4.19), the size of the commitment and the size of the opening are both \( \text{poly}(\lambda, \log d) \), which satisfies the efficiency requirements for a polynomial commitment. Note though that this construction does not provide hiding.

We also note that a similar transformation from vector commitments supporting linear functions to polynomial commitments does not apply if we start with the scheme of Albrecht et al. [ACL*22]. The [ACL*22] scheme only supports committing to vectors with small values and opening to linear functions with small coefficients (with magnitude smaller than \( q \)). Our transformation relies on opening to a linear function with coefficients \((1, x, \ldots, x^d)\). If \( q > x^d \), then the bit-length of \( q \) scales with the degree of the polynomial and the resulting scheme is no longer succinct. Note that the binary-decomposition approach from Remark 4.19 is not applicable in this case since it scales up the magnitude of the vector components by a factor proportional to \( x^d \).

**Parameter instantiation.** Let \( \lambda \) be a security parameter, \( q = q(\lambda) \) be the modulus, \( t = t(\lambda) \) be a vector dimension, and \( \mathcal{F} = \{ f_i \}_{i \in \mathbb{N}} \) be a family of linear functions over \( \mathbb{Z}_q^t \). We instantiate the lattice parameters in Construction 4.14 as follows to satisfy Theorems 4.16 and 4.17:

- We set the lattice dimension \( n = \text{poly}(\lambda) \) and \( m = O(n \log q) \).
- We set \( s_0 = O(tm^2 \log(nt)) \) and \( s_1 = O(t^{3/2}m^{3/2} \log(nt) \cdot s_0) = O(t^{5/2}m^{7/2} \log^2 (nt)) \).
- We set the bound \( B = t \sqrt{m} + m^{s_1} = O(t^4 m^4 \log^2(nt)) = O(t^4 n^4 \log^2(nt) \log^4 q) \).
- We set the modulus \( q \) so that the BASIS\text{struct} assumption holds with parameters \((n, m, q, \beta, s_0, \ell)\), where \( \beta \geq 2Bm^2 \log n = \text{poly}(\lambda, \ell) \). Note that this also requires that SIS\text{struct} hold. For instance, we set \( q = \beta \cdot \text{poly}(n) \). Then, \( \log q = \text{poly}(\log \lambda, \log \ell) \).

Taken together and in conjunction with Remarks 4.15, 4.19, and 4.20, we obtain the following corollary:

**Corollary 4.21 (Polynomial Commitments from BASIS\text{struct}).** Let \( \lambda \) be a security parameter. Then, for all polynomials \( \ell = t(\lambda) \), and under the BASIS\text{struct} assumption with a norm bound \( \beta = \text{poly}(\lambda, \ell) \) and modulus \( q = \text{poly}(\lambda, \ell) \), we have the following vector commitment schemes:

- A vector commitment scheme over \( \mathbb{Z}_q^t \) that supports opening to arbitrary linear functions \( x \mapsto z'x \) for all \( z \in \mathbb{Z}_q^t \) with commitment and opening size \( \text{poly}(\lambda, \log \ell) \).
- A polynomial commitment scheme over \( \mathbb{Z}_q \) for polynomials of degree up to \( \ell - 1 \) with commitment and opening size \( \text{poly}(\lambda, \log \ell) \).

Both schemes are computationally binding and support fast verification in the preprocessing model where the linear function \( f \) or the point \( x \in \mathbb{Z}_q \) are known in advance (i.e., with preprocessing, the running time of Verify is \( \text{poly}(\lambda, \log \ell) \)). Without preprocessing, the running time of Verify is \( \text{poly}(\lambda, \ell) \). The size of the CRS in both constructions is \( \ell^2 \cdot \text{poly}(\lambda, \log \ell) \).
4.2 Supporting Private Openings

In this section, we show how to extend our functional commitment scheme (Construction 4.2) to support (statistically) private openings. Recall from Definition 4.1 that a functional commitment supports private opening if the commitment \(\sigma\) to an input \(x\) together with an opening \(\pi_f\) with respect to a function \(f\) leaks no additional information about \(x\) other than the value \(f(x)\). In the context of homomorphic signatures [GVW15], this property is called context hiding. We show that the same approach used to achieve context hiding in the setting of homomorphic signatures applies to our setting and yields a succinct functional commitment that supports private openings. However, the transformation does not preserve the binding property on the functional commitments scheme. Nonetheless, we can still show that the scheme satisfies a weaker notion of binding called target binding, which says that any honestly-generated commitment on \(x\) can only be opened to \(f(x)\) for any function \(f\). We start by defining this notion and then give our construction below.

Definition 4.22 (Target Binding). Let \(\lambda\) be a security parameter and let \(\Pi_{FC} = (\text{Setup}, \text{Commit}, \text{Eval}, \text{Verify})\) be a succinct functional commitment for a family of functions \(F = \{F_\lambda\}_{\lambda \in \mathbb{N}}\) on inputs of length \(t = t(\lambda)\) and computable by Boolean circuits of depth at most \(d = d(\lambda)\). For a security parameter \(\lambda\), we define the target binding game between an adversary \(\mathcal{A}\) and a challenger:

1. The challenger starts by sampling \(\text{crs} \leftarrow \text{Setup}(1^\lambda, 1^t, 1^d)\) and gives \(\text{crs}\) to \(\mathcal{A}\).
2. Algorithm \(\mathcal{A}\) outputs an input \(x \in \{0, 1\}^t\).
3. Algorithm \(\mathcal{B}\) computes \(\sigma \leftarrow \text{Commit}(\text{crs}, x)\) and gives \(\sigma\) to \(\mathcal{A}\).
4. Algorithm \(\mathcal{A}\) outputs a function \(f \in F_\lambda\) and an opening \(\pi\).

We say that \(\Pi_{FC}\) satisfies statistical (resp., computational) target binding if for all adversaries \(\mathcal{A}\) (resp., efficient adversaries \(\mathcal{A}\)),

\[
\Pr[\text{Verify}(\text{crs}, \sigma, f, 1 - f(x), \pi) = 1] = \text{negl}(\lambda)
\]

in the target binding game.

Construction 4.23 (Succinct Functional Commitment with Private Opening). Let \(\lambda\) be a security parameter and \(F = \{F_\lambda\}_{\lambda \in \mathbb{N}}\) be a family of functions \(f : \{0, 1\}^t \rightarrow \{0, 1\}\) where each function \(f : \{0, 1\}^t \rightarrow \{0, 1\}\) is a function on inputs of length \(t = t(\lambda)\) and which can be computed by a Boolean circuit of depth at most \(d = d(\lambda)\). Let \(\Pi_{FC} = (\text{Setup}, \text{Commit}, \text{Eval}, \text{Verify})\) be the succinct functional commitment scheme from Construction 4.2 for \(F\). Let \(n, m, q, m^t, B, s_0, s_1\) be the scheme parameters from Construction 4.2. Let \(s_2 = s_2(\lambda)\) be an additional Gaussian width parameter. To construct a functional commitment scheme with private openings for \(F\), we modify the Setup, Eval, and Verify algorithms in Construction 4.2 as follows (the Commit algorithm is unchanged):

- **Setup(1^\lambda, 1^t, 1^d)**: Sample \(A, W_1, \ldots, W_{t}, T\) according to the specification of Setup(1^\lambda, 1^t, 1^d) in Construction 4.2. Then, sample \(u \overset{\$}{\leftarrow} \mathbb{Z}_q\), and output the common reference string \(\text{crs} = (A, W_1, \ldots, W_t, T, u)\).

- **Eval(crs, st, f)**: On input the common reference string \(\text{crs} = (A, W_1, \ldots, W_t, T, u)\), a commitment state \(st = (x, C, \Gamma_{t1}, \ldots, \Gamma_t)\), a function \(f : \{0, 1\}^t \rightarrow \{0, 1\}\), the evaluation algorithm proceeds as follows:
  1. Compute \(\hat{C} \leftarrow [W_1^{-1}C | \cdots | W_t^{-1}C] \in \mathbb{Z}_q^{n \times m^t}\), \(H_{\hat{C}, f, x} \leftarrow \text{EvalFX}(\hat{C}, f, x)\), and \(V_f \leftarrow [V_1 | \cdots | V_t] \cdot H_{\hat{C}, f, x}^t\).
  2. Let \(\hat{C}_f \leftarrow \text{EvalF}(\hat{C}, f)\) and define the matrix \(D_f = [A | \hat{C}_f + (f(x) - 1) \cdot G] \in \mathbb{Z}_q^{n \times (m + m^t)}\). Let \(R_f = [-v_f | v_f]\).

- **Verify(crs, \sigma, f, y, \pi)**: On input the common reference string \(\text{crs} = (A, W_1, \ldots, W_t, T, u)\), a commitment \(\sigma = C \in \mathbb{Z}_q^{n \times m^t}\), a function \(f : \{0, 1\}^t \rightarrow \{0, 1\}\), a value \(y \in \{0, 1\}\), and an opening \(\pi = v_f \in \mathbb{Z}_q^{m + m^t}\), the verification algorithm computes \(\tilde{C} \leftarrow [W_1^{-1}C | \cdots | W_t^{-1}C] \in \mathbb{Z}_q^{n \times m^t}\). Let \(\hat{C}_f \leftarrow \text{EvalF}(\hat{C}, f)\) and output 1 if
  \[
  ||v_f|| \leq B \quad \text{and} \quad [A | \hat{C}_f + (y - 1) \cdot G]v_f = u.
  \]
Theorem 4.24 (Correctness). Suppose \( n \geq \lambda, m \geq O(n \log q), s_0 \geq O(t m^2 \log(q)), s_1 \geq O(t^{3/2} m^{3/2} \log(q) \cdot s_0), s_2 \geq s_1 \cdot m^{3/2} t^{1/2} \cdot (n \log q)^O(d), \) and \( B \approx s_2 \cdot \sqrt{m + m'} \). Then, Construction 4.23 is correct.

Proof. Take a security parameter \( \lambda \), a function \( f \in \mathcal{F}_{\lambda} \), and an input \( x \in \{0, 1\}^f \). Let \( \text{crs} = (A, W_1, \ldots, W_r, T, u) \leftarrow \text{Setup}(1^\lambda, t^f, 1^\delta) \) and \( (\sigma, st) \leftarrow \text{Commit}(\text{crs}, x) \) where \( \sigma = C = GC = \mathbb{Z}^{q \times m} \) and \( st = (x, C, V_1, \ldots, V_r) \). Let \( \pi = v_f \leftarrow \text{Eval}(\text{crs}, st, f) \) and consider \( \text{Verify}(\text{crs}, \sigma, f, f(x), \pi) \). By the same argument as in the proof of Theorem 4.3, the following hold:

- Let \( \tilde{C} = [W_i^{-1} C \mid \cdots \mid W_t^{-1} C] \) and \( \tilde{V} = [V_1 \mid \cdots \mid V_t] \). Then, \( \tilde{C} - x^* \otimes G = \tilde{AV} \), and \( ||\tilde{V}|| \leq B_0 = \sqrt{m + m'} s_1 \).
- Let \( H_{\tilde{C}, f, x} \leftarrow \text{EvalFX}(\tilde{C}, f, x) \) and \( v_f \leftarrow \tilde{V} H_{\tilde{C}, f, x} \). By Theorem 2.11, \( ||v_f|| \leq s_1 \cdot \sqrt{m} \cdot (n \log q)^O(d) \).

where \( \tilde{C}_f \leftarrow \text{EvalF}(\tilde{C}, f) \).

Let \( D_f = [A | \tilde{C}_f + (f(x) - 1) \cdot G] \in \mathbb{Z}_q^{n \times (m + m')} \) and \( R_f = \begin{bmatrix} -v_f \ t \end{bmatrix} \in \mathbb{Z}_q^{(m + m') \times m'} \). Then, \( ||R_f|| = ||v_f|| \leq s_1 \cdot \sqrt{m} \cdot (n \log q)^O(d) \) and

\[ D_f R_f = -A V_f + \tilde{C}_f + (f(x) - 1) \cdot G = (2 f(x) - 1) \cdot G \in \{G, -G\} \]

Thus, \( R_f \) is a gadget trapdoor for \( D_f \) (with tag \( I_0 \) or \( -I_n \), depending on the value of \( f(x) \)). Suppose that \( m \geq m' = O(n \log q) \) and

\[ s_2 \geq \sqrt{(m + m') m'} \cdot ||R_f|| \cdot \omega(\sqrt{\log n}) = s_1 \cdot m^{3/2} t^{1/2} \cdot (n \log q)^O(d) \]

Since \( v_f \leftarrow \text{SamplePre}(D_f, R_f, u, s_2) \) we appeal to Theorem 2.8 to conclude that \( D_f v_f = u \), and moreover, by Lemmas 2.4 and 2.7, \( ||v_f|| \leq \sqrt{m + m'} s_2 \leq B \). The verification algorithm accepts. \( \Box \)

Theorem 4.25 (Target Binding). Suppose the conditions on \( n, m, s_0, s_1 \) in Theorem 4.24 hold \( (n \geq \lambda, m \geq O(n \log q), s_0 \geq O(t m^2 \log(q)), s_1 \geq O(t^{3/2} m^{3/2} \log(q)) \cdot s_0) \). Then, under the BASIS\textsubscript{struct} assumption with parameters \( (n, m, q, \beta, s_0, \ell) \), where \( \beta = s_1 \cdot m^{3/2} t^{1/2} \cdot B \cdot (n \log q)^O(d) \), Construction 4.23 satisfies computational target binding.

Proof. We use a similar hybrid structure as in the proof of Theorem 4.4.

- Hyb_\text{b}:
  - The challenger starts by sampling \( (A, R) \leftarrow \text{TrapGen}(1^n, q, m) \) and \( W_i \leftarrow \mathbb{Z}_q^{n \times n} \) for each \( i \in \{1, \ldots, t\} \). It sets \( \hat{R}_i \leftarrow RG^{-1}(W_i^{-1} G) \) and constructs \( B_f \) and \( \hat{R} \) according to Eq. (4.1). It samples \( T \leftarrow \text{SamplePre}(B_f, \hat{R}, G_{nt}, s_0) \) and \( u \leftarrow \mathbb{Z}_q^{nt} \). It gives \( \text{crs} = (A, W_1, \ldots, W_t, T, u) \) to \( \mathcal{A} \).
  - Algorithm \( \mathcal{A} \) chooses an input vector \( x \in \{0, 1\}^f \).
  - The challenger gives \( (\sigma, st) \leftarrow \text{Commit}(\text{crs}, x) \) to \( \mathcal{A} \) where \( \sigma = C \in \mathbb{Z}_q^{nt} \) and \( st = (x, C, V_1, \ldots, V_r) \).
  - At the end of the experiment, the adversary outputs a function \( f \in \mathcal{F}_{\lambda} \) and an opening \( \pi = v_f \). The output of the experiment is 1 if \( ||v_f|| \leq B \) and \( [A | \tilde{C}_f - f(x) \cdot G] v_f = u \) where \( \tilde{C}_f \leftarrow \text{EvalF}(\tilde{C}, f) \) and \( \tilde{C} = [W_i^{-1} C \mid \cdots \mid W_t^{-1} C] \). Otherwise, the experiments outputs 0.

- Hyb_\text{b}:
  - Same as Hyb_\text{b} except after constructing the matrix \( B_f \in \mathbb{Z}_q^{nt \times (m + m')} \) according to Eq. (5.1), the challenger samples \( T \leftarrow (B_f)^{-1}(G_{nt}) \) without using the trapdoor \( R \). The CRS is now sampled independently of \( R \).

- Hyb_\text{b}:
  - Same as Hyb_\text{b} except the challenger samples \( A \leftarrow \hat{R} \) where \( \hat{R} \leftarrow \mathbb{Z}_q^{n \times m} \).

- Hyb_\text{b}:
  - Same as Hyb_\text{b} except the challenger samples \( u \leftarrow Ar \) where \( r \leftarrow \{0, 1\}^m \).

For an adversary \( \mathcal{A} \), we write \( \text{Hyb}_\text{b}(\mathcal{A}) \) to denote the output of an execution of \( \text{Hyb}_\text{b}(\mathcal{A}) \) with adversary \( \mathcal{A} \). We now show that each adjacent pair of experiments are computationally indistinguishable.
Lemma 4.26. Suppose $n \geq \lambda$, $m \geq O(n \log q)$, and $s_0 \geq O(tm^2 \log(nt))$. Then, for all adversaries $\mathcal{A}$, $\text{Hyb}_1(\mathcal{A}) \approx \text{Hyb}_2(\mathcal{A})$.

Proof. Identical to the proof of Lemma 4.5.

Lemma 4.27. Suppose $n \geq \lambda$ and $m = O(n \log q)$. Then, for all adversaries $\mathcal{A}$, $\text{Hyb}_1(\mathcal{A}) \approx \text{Hyb}_2(\mathcal{A})$.

Proof. Identical to the proof of Lemma 4.6.

Lemma 4.28. Suppose $n \geq \lambda$ and $m \geq 2n \log q$. Then, for all adversaries $\mathcal{A}$, $\text{Hyb}_2(\mathcal{A}) \approx \text{Hyb}_3(\mathcal{A})$.

Proof. The only difference between $\text{Hyb}_2$ and $\text{Hyb}_3$ is the distribution of $u$. In $\text{Hyb}_2$, $u \leftarrow \mathbb{Z}_q^n$ while in $\text{Hyb}_3$, $u \leftarrow \mathcal{A}r$ where $r \leftarrow \{0, 1\}^m$. Since $A \leftarrow \mathbb{Z}_q^n \times m$ and $m \geq 2n \log q$, the claim follows by the leftover hash lemma (Lemma 2.2).

Lemma 4.29. Suppose the conditions on $n, m, s_0, s_1$ in Theorem 4.24 hold and $m \geq n \log q + \lambda$. Let $\beta = s_1 \cdot m^{3/2} \ell^{1/2} \cdot B \cdot (n \log q)^{O(d)}$. Then, under the BASIS_{struct} assumption with parameters $(n, m, q, s_0, \ell)$, for all efficient adversaries $\mathcal{A}$, $\Pr[\text{Hyb}_3(\mathcal{A}) = 1] = \text{negl}(\lambda)$.

Proof. Suppose there exists an efficient adversary $\mathcal{A}$ where $\Pr[\text{Hyb}_3(\mathcal{A}) = 1] = \epsilon$ for some non-negligible $\epsilon$. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ for the BASIS_{struct} assumption:

1. Algorithm $\mathcal{B}$ receives a challenge $A \in \mathbb{Z}_q^{n \times m}$, $B_r \in \mathbb{Z}_q^{m \times (m + m')}$, and $T \in \mathbb{Z}_q^{(m + m') \times (m + m')}$, and $\text{aux} = (W_1, \ldots, W_{\ell})$.
2. Algorithm $\mathcal{B}$ samples $r \leftarrow \{0, 1\}^m$, computes $u \leftarrow \mathcal{A}r$, and gives $\text{crs} = (A, W_1, \ldots, W_{\ell}, T, u)$ to $\mathcal{A}$.
3. Algorithm $\mathcal{A}$ outputs a vector $x \in \mathbb{Z}_q^t$.
4. Algorithm $\mathcal{B}$ constructs the matrix $U_x \in \mathbb{Z}_q^{n \times m'}$ according to Eq. (4.2) and samples $(V_1, \ldots, V_r, \hat{C})$ according to Eq. (4.3). It gives $\sigma = C = G \hat{C} \in \mathbb{Z}_q^{n \times m'}$ and the state $= (x, C, V_1, \ldots, V_r)$ to $\mathcal{A}$.
5. Algorithm $\mathcal{A}$ outputs a function $f \in \mathcal{T}_\lambda$ and an opening $v_f \in \mathbb{Z}_q^{m + m'}$.
6. Algorithm $\mathcal{B}$ outputs the vector $z = [I_n \mid V_f]v_f - r$, where $V_f = [V_1 \mid \cdots \mid V_r] \cdot \hat{H}_{C, f, x}, \hat{H}_{C, f, x} \leftarrow \text{EvalFX}(\hat{C}, f, x)$, and $\hat{C} = [W_1^{1 \cdot C} \mid \cdots \mid W_{\ell}^{1 \cdot C}]$.

By construction, algorithm $\mathcal{B}$ perfectly simulates the common reference string $\text{crs}$ and the commitment $(\sigma, st)$ according to the specification of $\text{Hyb}_3$. Thus, with probability $\epsilon$, algorithm $\mathcal{A}$ outputs $(f, V_f)$ where $\|V_f\| \leq B$ and $[A \mid \hat{C}_f - f(x) \cdot G]V_f = u$. Since $(V_1, \ldots, V_r, \hat{C})$ are sampled using the Commit algorithm, we can appeal to the analysis of Theorem 4.24 and conclude that with overwhelming probability, $\|V_f\| \leq s_1 \cdot (n \log q)^{O(d)}$ and $AV_f = \hat{C}_f - f(x) \cdot G$ where $\hat{C} \leftarrow \text{EvalF}(\hat{C}, f)$. This means that

$$u = [A \mid \hat{C}_f - f(x) \cdot G]V_f = [A \mid AV_f]V_f = A[I_n \mid V_f]V_f.$$

Since $u = A_r$, this means that $Az = A[I_n \mid V_f]v_f - r = 0$. To conclude the proof, we argue that $0 < \|z\| \leq \beta$:

- Since $\|V_f\| \leq s_1 \cdot \sqrt{tm} \cdot (n \log q)^{O(d)}$, $\|V_f\| \leq B$, $\|r\| = 1$, and $m > m'$, we have that
  $$\|z\| \leq s_1 \cdot \sqrt{tm} \cdot (n \log q)^{O(d)} B(m + m') + 1 \leq s_1 \cdot m^{3/2} \ell^{1/2} \cdot B \cdot (n \log q)^{O(d)} = \beta.$$
- It suffices to show that $z \neq 0$, or equivalently, that $r \neq [I_n \mid V_f]v_f$. Here, we can appeal to the same entropy argument as in [GVW15, Theorem 3.1]. By construction, $V_f$ and $v_f$ are functions of $u \in \mathbb{Z}_q^n$ (and other quantities that are independent of $r$). By construction, $u$ contains at most $n \log q$ bits of information about $r$. This means that
  $$H_\infty(r \mid V_f, v_f) = H_\infty(r \mid u) \geq m - n \log q \geq \lambda.$$  
  This means that $\Pr[r = [I_n \mid V_f]v_f] \leq 2^{-\lambda}$, and so $\mathcal{B}$ breaks the BASIS_{struct} assumption with advantage $\epsilon - 2^{-\lambda}$. 

36
Theorem 4.30 (Private Opening). Suppose the conditions of Theorem 4.24 hold: namely, \( n \geq \lambda, m \geq O(n \log q), s_0 \geq O(tm^2 \log(nf)), s_1 \geq O(t^{1/2}m^{3/2} \log(nf) \cdot s_0), \) and \( s_2 \geq s_1 \cdot m^{3/2} t^{1/2} \cdot (n \log q)^O(d) \). Suppose also that \( q \) is prime. Then, Construction 4.23 provides statistical private openings.

Proof. We construct an efficient simulator \( S = (S_0, S_1) \) as follows:

- \( S_0(1^t, 1^f, 1^d) \): On input the security parameter \( \lambda \), the input dimension \( t \) and the depth bound \( d \), the simulator samples \( (A, R) \leftarrow \text{TrapGen}(1^t, q, m) \) and an invertible matrix \( W_i \leftarrow \mathbb{Z}_q^m \times n \) for each \( i \in [t] \). Then, it constructs the matrix \( B_i \) and trapdoor \( R \) according to Eq. (4.1). Next, it samples \( T \leftarrow \text{SamplePre}(B_t, R, G_{nm}, s_0) \) and \( u \leftarrow \mathbb{Z}_q^m \). It samples \( \tilde{C} \leftarrow D_{n \times m}' \), computes \( C \leftarrow G \tilde{C} \), and outputs \( \text{crs} = (A, W_1, \ldots, W_t, T, u), \) the commitment \( \sigma = C \), and the state \( st_S = (A, R, C, u) \).

- \( S_1(st_S, f, y) \): On input the simulation state \( st = (A, R, C, u) \), a function \( f \in \mathcal{F}_S \), and a value \( y \in \{0, 1\} \), the simulator first computes \( \tilde{C} \leftarrow [W_1^{-1}C | \cdots | W_t^{-1}C] \) and \( \hat{C} \leftarrow \text{EvalF}(\tilde{C}, f) \). It defines the matrix \( D_f = [A | \hat{C}_f + (y - 1)G] \) and \( R_f = [\tilde{R}] \) and outputs \( \text{v}_f \leftarrow \text{SamplePre}(D_f, R_f, u, s_2) \).

To complete the proof, it suffices to show that the real distribution and the simulated distribution are statistically indistinguishable. We proceed with a hybrid argument:

- **Hyb_0**: This is the real distribution:
  1. The challenger starts by sampling \( (A, R) \leftarrow \text{TrapGen}(1^t, q, m) \) and \( W_i \leftarrow \mathbb{Z}_q^m \times n \) for each \( i \in [t] \). It sets \( R_i \leftarrow RG^{-1}(W_i^{-1}G) \) and constructs \( B_i \) and \( R \) according to Eq. (4.1). It samples \( T \leftarrow \text{SamplePre}(B_t, R, G_{nm}, s_0) \) and \( u \leftarrow \mathbb{Z}_q^m \). It gives \( \text{crs} = (A, W_1, \ldots, W_t, T, u) \) to \( \mathcal{A} \).
  2. Algorithm \( \mathcal{A} \) chooses an input vector \( x \in \{0, 1\}^t \).
  3. The challenger constructs the target matrix \( U_x \) according to Eq. (4.2) and samples \( (V_1, \ldots, V_t, \tilde{C}) \) using \( \text{SamplePre}(B_t, T, U_x, s_1) \). It gives the commitment \( \sigma = C = G \tilde{C} \in \mathbb{Z}_q^m \times m' \) to \( \mathcal{A} \).
  4. Adversary \( \mathcal{A} \) outputs a function \( f \in \mathcal{F}_S \).
  5. The challenger now computes \( \tilde{C} \leftarrow [W_1^{-1}C | \cdots | W_t^{-1}C] \), \( H_{\tilde{C}, f, x} \leftarrow \text{EvalFX}(\tilde{C}, f, x) \), and \( V_f \leftarrow [V_1 | \cdots | V_t] \). Next, it computes \( \tilde{C}_f \leftarrow \text{EvalF}(\tilde{C}, f) \), \( D_f = [A | \tilde{C}_f + (f(x) - 1) \cdot G] \), and \( R_f = [\tilde{R}^f] \).
  6. Finally, algorithm \( \mathcal{A} \) outputs a bit \( b \in \{0, 1\} \), which is also the output of the experiment.

- **Hyb_1**: Same as Hyb_0, except when simulating the opening, the challenger instead sets \( R_f = [\tilde{R}] \) and samples \( v_f \leftarrow \text{SamplePre}(D_f, R_f, u, s_2) \).

- **Hyb_2**: Same as Hyb_1 except the challenger samples \( \tilde{C} \leftarrow D_{n \times s_2}' \). This is the simulated distribution.

For an adversary \( \mathcal{A} \), we write \( \text{Hyb}_b(\mathcal{A}) \) to denote the output of an execution of \( \text{Hyb}_b(\mathcal{A}) \) with adversary \( \mathcal{A} \). We now show that each adjacent pair of experiments are statistically indistinguishable.

Lemma 4.31. Suppose the conditions of Theorem 4.24 hold. Then, for all adversaries \( \mathcal{A} \), \( \text{Hyb}_0(\mathcal{A}) \approx \text{Hyb}_1(\mathcal{A}) \).

Proof. The only difference between the Hyb_0 and Hyb_1 is the distribution of \( v_f \):

- Consider the distribution of \( v_f \) in \( \text{Hyb}_0 \). By the same analysis as in the proof of Theorem 4.24, \( D_f R_f \in \{G, -G\} \) where \( \|R_f\| \leq s_1 \cdot \sqrt{tm} \cdot (n \log q)^{O(d)} \). Moreover, if \( s_2 \geq \sqrt{(m + m')^2 \|R_f\| \cdot \omega(\sqrt{\log n})} = s_1 \cdot m^{3/2} t^{1/2} \cdot (n \log q)^{O(d)} \), then by Theorem 2.8, the distribution of \( v_f \leftarrow \text{SamplePre}(D_f, R_f, u, s_2) \) is statistically close to the distribution \( v_f \leftarrow (D_f s_2^{-1}(u)) \).

- In \( \text{Hyb}_1 \), \( D_f R_f = AR = G \) and \( \|R_f\| = \|R\| = 1 \). As long as \( s_2 \geq \sqrt{(m + m')^2 \|R_f\| \cdot \omega(\sqrt{\log n})} = O(m \log n) \), the distribution of \( v_f \leftarrow \text{SamplePre}(D_f, R_f, u, s_2) \) is statistically close to the distribution of \( v_f \leftarrow (D_f s_2^{-1}(u)) \).
In both cases, the distribution of $v_f$ is statistically close to $(D_f)_{s_2}^{-1}(u)$ and the claim holds. \hfill \Box

**Lemma 4.32.** Suppose the conditions of Theorem 4.24 hold and $q$ is prime. Then for all adversaries $\mathcal{A}$, $\text{Hyb}_1(\mathcal{A}) \simeq$ $\text{Hyb}_2(\mathcal{A})$.

**Proof.** The only difference between $\text{Hyb}_1$ and $\text{Hyb}_2$ is the distribution of $\hat{C}$:

- In $\text{Hyb}_1$, the challenger samples

\[
\begin{bmatrix}
V_1 \\
\vdots \\
V_\ell \\
\hat{C}
\end{bmatrix} \leftarrow \text{SamplePre}(B_\ell, T, U_x, s_1),
\]

where $B_\ell$ is the matrix from Eq. (4.1). By the same analysis as in the proof of Theorem 4.3, we have that $B_\ell T = G$ and $||T|| \leq \sqrt{\ell m} + m's_0$. If

\[
s_1 \geq \sqrt{(\ell m + m')\ell m'} ||T|| \cdot o(\sqrt{\log(\ell m)}) = O(\ell^{1/2}m^{3/2}\log(\ell m) \cdot s_0),
\]

we can appeal to Theorem 2.8 to conclude that the distribution of $(V_1, \ldots, V_\ell, \hat{C})$ is statistically close to $(B_\ell)_{s_2}^{-1}(U_x)$.

Next, let $\hat{A} = \text{diag}(W_1A, \ldots, W_\ell A)$. Then, $B_\ell = [\hat{A} \mid -1^\ell \otimes G]$. By Corollary 2.3, with probability $1 - q^{-\omega}$ over the choice of $A$, the columns of $A$ generate $Z_q^n$. Since $W_1, \ldots, W_\ell$ are invertible, this means the columns of $\hat{A}$ generate $Z_q^{\ell m'}$. Again, since $W_1$ is invertible, the distribution of $W_1A$ is uniform over $Z_q^{n \times m}$, so by Lemma 2.5, $\eta_\ell(\hat{A}) \leq \log(\ell m)$ for a negligible function $\varepsilon = \text{negl}(n)$. Since $s_1 \geq \log(\ell m)$, we appeal to Lemma 2.7 to conclude that the distribution of $(V_1, \ldots, V_\ell, \hat{C}) \leftarrow (B_\ell)_{s_2}^{-1}(U_x)$ is statistically close to the distribution

\[
\left\{ \hat{C} \leftarrow D_{s_3}^{m' \times m'}, (V_1, \ldots, V_\ell) \leftarrow (\hat{A})_{s_2}^{-1}(U_x + (1^\ell \otimes \hat{G})) \right\}.
\]

- In $\text{Hyb}_2$, the challenger samples $\hat{C} \leftarrow D_{s_3}^{m' \times m'}$.

In both experiments, the marginal distribution of $\hat{C}$ is statistically close to the distribution $D_{s_3}^{m' \times m'}$. \hfill \Box

Statistical private openings now follows from Lemmas 4.31 and 4.32.

**Parameter instantiation.** Let $\lambda$ be a security parameter and $\mathcal{F} = \{\mathcal{F}_h\}_{h \in \mathbb{N}}$ be a family of functions $f : \{0, 1\}^f \rightarrow \{0, 1\}$ on inputs of length $\ell = \ell(\lambda)$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. We instantiate the lattice parameters in Construction 4.23 as follows to satisfy Theorems 4.24, 4.25, and 4.30. The parameter instantiations essentially match those of the non-private construction (Construction 4.2):

- Let $\varepsilon > 0$ be a constant. We set the lattice dimension $n = d^{1/\varepsilon} \cdot \text{poly}(\lambda)$ and $m = O(n \log q)$.

- We set $s_0 = O(\ell m^2 \log(n \ell))$, $s_1 = O(\ell^{3/2}m^{3/2}\log(n \ell) \cdot s_0) = O(\ell^{5/2}m^{7/2}\log^2(n \ell))$, and $s_2 \geq s_1 \cdot m^{3/2}\ell^{1/2} \cdot (n \log q)^{O(d)} = \ell^{3} \log^{2} \ell \cdot (n \log q)^{O(d)}$.

- We set $B = s_2 \cdot \sqrt{m + m'} = \ell^{3} \log^{2} \ell \cdot (n \log q)^{O(d)}$.

- We choose the modulus $q$ so that the BASIS\textsubscript{struct} assumption holds with parameters $(n, m, q, \beta, s_0, \ell)$, where

\[
\beta = s_1 \cdot m^{3/2}\ell^{1/2} \cdot B \cdot (n \log q)^{O(d)} = 2^{O(d)} = 2^{O(n')},
\]

where we write $\tilde{O}(\cdot)$ to suppress polylogarithmic factors in $\lambda, d, \ell$. Note that this also requires that SIS\textsubscript{n,m,q,\beta} hold. For instance, we set $q = \beta \cdot \text{poly}(n)$. Then, $\log q = \text{poly}(d, \log \lambda, \log \ell)$. Note that the underlying SIS assumption relies on a sub-exponential noise bound.
With this setting of parameters, we obtain a functional commitment scheme with private openings for $F$ with similar properties as the non-hiding variant (Corollary 4.8):

**Corollary 4.33** (Succinct Functional Commitment with Private Opening from $\text{BASIS}_{\text{struct}}$). Let $\lambda$ be a security parameter, and let $F = \{ F_j \}_{j \in \mathbb{N}}$ be a family of functions $f : \{0, 1\}^t \rightarrow \{0, 1\}$ on inputs of length $t = t(\lambda)$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. Under the $\text{BASIS}_{\text{struct}}$ assumption with a norm bound $\beta = 2^{O(d)}$ and modulus $q = 2^{O(d)}$, there exists a succinct functional commitment scheme for $F$ that satisfies computational target binding and has statistically private openings. Both the size of the commitment and the opening are $\text{poly}(\lambda, d, \log t)$ and the CRS has size $t^2 \cdot \text{poly}(\lambda, d, \log t)$. Here, $O(\cdot)$ suppresses polylogarithmic factors in $\lambda$, $d$, and $t$.

## 5 Aggregatable Vector and Functional Commitments

In this section, we describe a variant of our SIS-based vector commitment (Construction 3.9) that supports aggregation. The same techniques also apply to our succinct functional commitment scheme (Construction 4.2) and yields an aggregatable vector commitment scheme (see Section 5.1). In an aggregatable commitment, one can take a collection of commitments where we allow the adversary to output two different commitments where we allow the adversary to output two different commitments and assign a tuple of efficient algorithms $\Pi$ (Aggregatable Vector Commitment)

**Definition 5.1** (Aggregatable Vector Commitment). An aggregatable vector commitment over a message space $M$ is a tuple of efficient algorithms $\Pi_{\text{AVC}} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify}, \text{Aggregate}, \text{VerifyAgg})$ where $(\text{Setup}, \text{Commit}, \text{Open}, \text{Verify})$ is a standard vector commitment scheme and the additional functions $(\text{Aggregate}, \text{VerifyAgg})$ satisfy the following properties:

- **Aggregate** ($\text{crs}, \sigma, \{(i, x_i, \pi_i)\}_{i \in S} \rightarrow \pi_S$): On input the common reference string $\text{crs}$, a commitment $\sigma$, a set of openings $\pi_i$ to $x_i \in M$ to indices $i \in S$, the aggregation algorithm outputs an aggregated opening $\pi_S$.

- **VerifyAgg** ($\text{crs}, \sigma, S, \{(i, x_i)\}_{i \in S}, \pi \rightarrow \{0, 1\}$): On input the common reference string $\text{crs}$, a commitment $\sigma$, a set $S \subseteq [\ell]$, and values $x_i \in M$ for $i \in S$, the aggregate verification algorithm outputs a bit $b \in \{0, 1\}$.

In addition to the basic properties of a vector commitment scheme, an aggregatable vector commitment scheme should satisfy the following additional properties:

- **Correctness of aggregation**: For all security parameters $\lambda \in \mathbb{N}$, vector lengths $\ell \in \mathbb{N}$, and sampling $\text{crs} \leftarrow \text{Setup}(1^\lambda, 1^\ell)$, and for all sets $S \subseteq [\ell]$, commitments $\sigma$, and openings $\{(i, x_i, \pi_i)\}_{i \in S}$ where $\text{Verify}(\text{crs}, \sigma, i, x_i, \pi_i) = 1$ for all $i \in S$, it holds that
  \[ \Pr[\text{VerifyAgg}(\text{crs}, \sigma, \{(i, x_i)\}_{i \in S}, \pi') : \pi' \leftarrow \text{Aggregate}(\text{crs}, \sigma, \{(i, x_i, \pi_i)\}_{i \in S})] = 1 - \text{negl}(\lambda), \]

  where the probability is taken over the randomness of Setup.

- **Succinctness**: The aggregatable vector commitment scheme is succinct if there exists a universal polynomial $\text{poly}(\cdot, \cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\pi'| = \text{poly}(\lambda, \log \ell)$ in the correctness of aggregation definition.

- **Same-set binding**: We say $\Pi_{\text{AVC}}$ satisfies statistical (resp., computational) same-set binding if for all polynomials $\ell = t(\lambda)$ and all adversaries $\mathcal{A}$ (resp., efficient adversaries $\mathcal{A}$),

\[ \Pr \left[ \begin{array}{c} \text{VerifyAgg}(\text{crs}, \sigma, S, \{(i, x_i)\}_{i \in S}, \pi) = 1 \\ \text{and } x_i \neq x_i' \text{ for some } i \in S \\ \text{VerifyAgg}(\text{crs}, \sigma, S, \{(i, x_i')\}_{i \in S}, \pi') = 1 \end{array} \right] = \text{negl}(\lambda). \]

\[ \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^\ell); \\
\text{for some } S \subseteq [\ell], \mathcal{A}(1^\lambda, 1^\ell, \text{crs}) \]

**Remark 5.2** (Different-Set Binding). We can also define a stronger notion of security for aggregatable vector commitments where we allow the adversary to output two different sets $S$ and $T$ along with openings $\pi_S$ and $\pi_T$. 
to \{ (i, x_i) \}_{i \in S} and \{ (i, x_i') \}_{i \in T}, respectively, in the binding game. The adversary wins if there exists some index \( i \in S \cap T \) such that \( x_i \neq x_i' \). Namely, we say an aggregatable vector commitment scheme \( \Pi_{\text{VC}} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify}, \text{Aggregate}, \text{VerifyAgg}) \) satisfies statistical (resp., computational) different-set binding if for all adversaries (resp., efficient adversaries) \( A \),

\[
\Pr \left[ \begin{array}{l}
\text{VerifyAgg}(\text{crs}, \sigma, S, \{ (i, x_i) \}_{i \in S}, \pi_S) = 1 \\
\text{VerifyAgg}(\text{crs}, \sigma, T, \{ (i, x_i') \}_{i \in T}, \pi_T) = 1 \end{array} \right] = \negl(\lambda).
\]

Different-set binding effectively says that the commitment \( \sigma \) fully defines the value of \( x_i \) for all \( i \in [\ell] \). In contrast, if the vector commitment scheme only satisfies same-set binding, then an adversary may be able to open a commitment \( \sigma \) to \( x_i = 0 \) with respect to a set \( S \) and to \( x_i = 1 \) with respect to a different set \( T \). The vector commitments we construct in this work satisfy the weaker notion of same-set binding (Theorem 5.6), but do not satisfy different-set binding (Remark 5.12). Note though that same-set binding implies different-set binding in the setting where the commitments are generated honestly (Remark 5.13). It is an interesting open problem to construct a lattice-based vector commitment scheme that satisfies the stronger notion of different-set binding.

**Construction 5.3 (Aggregatable Vector Commitments).** Let \( \lambda \) be a security parameter and \( n = n(\lambda), m = m(\lambda), \) and \( q = q(\lambda) \) be lattice parameters. Let \( m' = n([\log q] + 1) \) and \( B = B(\lambda) \) be a bound. Let \( s_0 = s_0(\lambda), s_1 = s_1(\lambda) \) be Gaussian width parameters. Let \( \ell = \ell(\lambda) \) be the input dimension. Let \( p = p(\lambda) \) be the message-space modulus. We construct an aggregatable vector commitment scheme \( \Pi_{\text{VC}} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify}) \) for \( \mathbb{Z}_q^\ell \), as follows:

- **Setup(\( 1^\lambda, 1^\ell \)):** On input the security parameter \( \lambda \) and the input length \( \ell \), the setup algorithm samples \((A, R) \leftarrow \text{TrapGen}(1^n, q, m) \). Then, for each \( i \in [\ell] \), it samples an invertible matrix \( W_i \in \mathbb{Z}_q^{n \times n} \) and a target vector \( u_i \in \mathbb{Z}_q^n \). Next, it computes \( \tilde{R}_i = \text{RG}^{-1}(W_i^{-1}G) \in \mathbb{Z}_q^{m \times m'} \) for each \( i \in [\ell] \) and constructs \( B_i \) and \( R \) as follows:

\[
B_i = \begin{bmatrix}
W_i A & -G \\
& \vdots \\
& W_i A & -G
\end{bmatrix} \in \mathbb{Z}_q^{n \times (m + m')} \quad \text{and} \quad \tilde{R} = \begin{bmatrix}
\text{diag}(\tilde{R}_1, \ldots, \tilde{R}_\ell) \\
0_{n \times m'}
\end{bmatrix} \in \mathbb{Z}_q^{(m + m') \times \ell m'}.
\]

(5.1)

Finally, the setup algorithm samples \( T \leftarrow \text{SamplePre}(B_i, \tilde{R}, G_{n^\ell}, s_0) \) and outputs the common reference string \( \text{crs} = \{ A, \{ W_i \}_{i \in [\ell]}, \{ u_i \}_{i \in [\ell]}, T \} \).

- **Commit(\( \text{crs}, x \)):** On input the common reference string \( \text{crs} = \{ A, \{ W_i \}_{i \in [\ell]}, \{ u_i \}_{i \in [\ell]}, T \} \) and a vector \( x \in \mathbb{Z}_q^\ell \), the commit algorithm constructs \( B_i \) from \( A, W_1, \ldots, A_i \) according to Eq. (5.1). It then constructs the target vector

\[
\hat{u} = \begin{bmatrix}
-x_1 W_1 u_1 \\
\vdots \\
-x_\ell W_\ell u_\ell
\end{bmatrix}
\]

and uses \( T \) to sample a preimage

\[
\begin{bmatrix}
v_1 \\
\vdots \\
v_\ell \\
\hat{c}
\end{bmatrix} \leftarrow \text{SamplePre}(B_i, T, \hat{u}, s_1).
\]

(5.2)

It computes \( c \leftarrow G\hat{c} \in \mathbb{Z}_q^n \) and outputs the commitment \( \sigma = c \) and the state \( st = (v_1, \ldots, v_\ell) \).

- **Open(st, i):** On input the state \( st = (v_1, \ldots, v_\ell) \) and an index \( i \in [\ell] \), the opening algorithm outputs \( \pi = v_i \).

- **Verify(\( \text{crs}, i, x_i, \pi \)):** On input the common reference string \( \text{crs} = \{ A, \{ W_i \}_{i \in [\ell]}, \{ u_i \}_{i \in [\ell]}, T \} \), a commitment \( \sigma = c \), an index \( i \in [\ell] \), a value \( x_i \in \mathbb{Z}_p \), and an opening \( \pi = v \), the verification algorithm outputs \( 1 \) if

\[
\| v \|_2 \leq B \quad \text{and} \quad W_i^{-1} c = A v + x_i u_i.
\]
We now define the aggregation algorithm for openings and the verification algorithm for a set of indices $S \subseteq [\ell]$:

- **Aggregate**(crs, $\sigma$, $(i, x_i, \pi_i)_{i \in S}$): On input the common reference string $crs = (A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T)$, a commitment $\sigma = c$, and a set $S$ of openings $(i, x_i, \pi_i)$, where $\pi_i = v_i \in \mathbb{Z}_q^m$, the aggregation algorithm outputs $\pi = v = \sum_{i \in S} v_i$.

- **VerifyAgg**(crs, $\sigma$, $S$, $(i, x_i)_{i \in S}$, $\pi$): On input the common reference string $crs = (A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T)$, a commitment $\sigma = c$, a set of indices $S \subseteq [\ell]$, values $x_i \in \mathbb{Z}_q$ for each $i \in S$, and an opening $\pi = v$, the aggregate verification algorithm outputs 1 if

$$||v|| \leq |S| \cdot B \quad \text{and} \quad \sum_{i \in S} W_i^{-1}c = Av + \sum_{i \in S} x_i u_i.$$ 

**Theorem 5.4** (Correctness). Suppose $n \geq \lambda$, $m \geq O(n \log q)$, $s_0 \geq O(\ell m^2 \log(n\ell))$, $s_1 \geq O(\ell^{3/2} m^{3/2} \log(n\ell) \cdot s_0)$, and $B = \sqrt{\ell m + m^2 \cdot s_1}$. Then, Construction 5.3 is correct.

**Proof.** The proof is nearly identical to that of the proof of Theorem 3.10. Take any polynomial $\ell = \ell(\lambda)$, any vector $x \in \mathbb{Z}_q^m$, and any index $i \in [\ell]$. Let $crs = (A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$. Let $(\sigma, st) \leftarrow \text{Commit}(crs, x)$ where $\sigma = c \in \mathbb{G} \in \mathbb{Z}_q^m$ and $st = (v_1, \ldots, v_\ell)$. Let $\pi = v_i \leftarrow \text{Open}(st, i)$ and consider Verify(crs, $\sigma$, $i$, $x_i$, $\pi$):

- Let $B_i \in \mathbb{Z}_q^{(\ell m + m') \times \ell}$ and $\tilde{R} \in \mathbb{Z}_q^{(\ell m + m') \times \ell}$ be the matrices from Eq. (5.1). For each $i \in [\ell]$, we have that $\tilde{R}_i = R \tilde{G}^{-1}(W_i^{-1}G)$, so $W_i \tilde{AR}_i = W_i \text{ARG}^{-1}(W_i^{-1}G) = W_i \text{GG}^{-1}(W_i^{-1}G) = G$.

Thus, $B_i \tilde{R} = G_{nt}$ and $\|R\| = \max_{i \in [\ell]} \|\tilde{R}_i\| \leq m' = O(n \log q)$ since $\|R\| = 1 = \|\text{GG}^{-1}(W_i^{-1}G)\|$ for all $i \in [\ell]$. Suppose also that $m \geq m' = O(n \log q)$. By Theorem 2.8 and Lemmas 2.4 and 2.6, as long as

$$s_0 \geq \sqrt{(\ell m + m')m} \cdot \|R\| \cdot \omega(\sqrt{\log(n\ell)}) = O(\ell m^2 \log(n\ell)),$$

then $B_i T = G_{nt}$ and $\|T\| \leq \sqrt{\ell m + m' s_0}$ with overwhelming probability.

- Suppose $s_1 \geq \sqrt{(\ell m + m')m} \cdot \|T\| \cdot \omega(\sqrt{\log(n\ell)}) = O(\ell^{3/2} m^{3/2} \log(n\ell) \cdot s_0)$. Then, by Theorem 2.8 and by construction of $(v_1, \ldots, v_\ell, \tilde{c})$ in Eq. (5.2), $W_i \tilde{AV}_i - \tilde{G} \tilde{c} = -x_i W_i u_i$. Since $\tilde{c} = \tilde{G} \tilde{c}$, this means that $W_i^{-1}c = AV_i + x_i u_i$. Moreover, by Lemmas 2.4 and 2.6, $\|v_i\| \leq \sqrt{\ell m + m' s_0} \leq B$. Thus, Verify(crs, $\sigma$, $i$, $x_i$, $\pi$) outputs 1. □

**Theorem 5.5** (Correctness of Aggregation). **Construction 5.3 satisfies correctness of aggregation.**

**Proof.** Take any polynomial $\ell = \ell(\lambda)$ and let $crs = (A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T) \leftarrow \text{Setup}(1^\lambda, 1^\ell)$. Take any set $S \subseteq [\ell]$, commitment $\sigma = c \in \mathbb{G} \in \mathbb{Z}_q^n$, and openings $(i, x_i, \pi_i)_{i \in S}$ where $\pi_i = v_i$ and Verify(crs, $\sigma$, $i$, $x_i$, $\pi_i$) = 1. Let $v \leftarrow \text{Aggregate}(crs, \sigma, ((i, x_i, \pi_i)_{i \in S})$. Since $\pi_i$ is a valid opening to $\sigma$ on index $i$, we have that $\|v_i\| \leq B$ and moreover, $W_i^{-1}c = AV_i + x_i u_i$. By construction, $v = \sum_{i \in S} v_i$, so $||v|| \leq |S| \cdot B$. Moreover,

$$Av = \sum_{i \in S} Av_i = \sum_{i \in S} W_i^{-1}c - \sum_{i \in S} x_i u_i,$$

and the aggregate verification algorithm accepts. □

**Theorem 5.6** (Computational Same-Set Binding). **Let $\ell$ be the vector dimension. Suppose $n \geq \lambda$, $m \geq O(n \log q)$, $q$ is prime, and $s_0 \geq O(\ell m^2 \log(n\ell))$. Then, under the BASISstruct assumption with parameters $(n, m, q, \ell(2B + p), s_0, \ell)$, Construction 5.3 satisfies computational same-set binding.**

**Proof.** Let $\ell = \ell(\lambda)$ be a polynomial. We use a nearly identical set of hybrid experiments as in the proof of Theorem 4.4:

- **Hyb$_3$**: This is the real same-set binding experiment:
- The challenger starts by sampling \((A, R) \leftarrow \text{TrapGen}(1^n, q, m)\), \(W_i \xleftarrow{\$} \mathbb{Z}^{n \times n}_q\), and \(u_i \xleftarrow{\$} \mathbb{Z}_q^n\) for each \(i \in [\ell]\). It sets \(R_i \leftarrow \text{RG}^{-1}(W_i^{-1}G)\) and constructs \(B_i\) and \(R\) according to Eq. (5.1). It samples \(T \leftarrow \text{SamplePre}(B_i, R, G_{nt}, s_0)\) and gives \(\text{crs} = (A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T)\) to the adversary \(A\).

- Algorithm \(A\) outputs a commitment \(c \in \mathbb{Z}_q^n\) a set \(S \subseteq [\ell]\), values \(\{(i, x_i, x'_i)\}_{i \in S}\) and openings \(v, v'\).

- The output of the experiment is 1 if there exists \(i \in S\) such that \(x_i \neq x'_i\), \(\|v\|, \|v'|| \leq |S| \cdot B, Av = \sum_{i \in S} W_i^{-1}c - \sum_{i \in S} x_i u_i\), and \(Av' = \sum_{i \in S} W_i^{-1}c - \sum_{i \in S} x'_i u_i\).

- \(\text{Hyb}_1\): Same as \(\text{Hyb}_0\) except after constructing the matrix \(B_i \in \mathbb{Z}_q^{nt \times (tm + m')}\) according to Eq. (5.1), the challenger samples \(T \leftarrow (B_i)_{n \times (G_{nt})}\) without using the trapdoor \(R\). The CRS is now sampled independently of \(R\).

- \(\text{Hyb}_2\): Same as \(\text{Hyb}_1\) except the challenger samples \(A \xleftarrow{\$} \mathbb{Z}^{n \times m}_q\).

- \(\text{Hyb}_3\): Same as \(\text{Hyb}_2\) except for each \(i \in [\ell]\), it samples \(r_i \xleftarrow{\$} \{0, 1\}^m\) and sets \(u_i \leftarrow Ar_i\).

For an adversary \(A\), we write \(\text{Hyb}_i(A)\) to denote the output of an execution of \(\text{Hyb}_i\) with adversary \(A\). We now show that each adjacent pair of experiments are computationally indistinguishable.

**Lemma 5.7.** Suppose \(n \geq \lambda, m \geq O(n \log q), \) and \(s_0 \geq O(tm^2 \log(nt))\). Then, for all adversaries \(A\), \(\text{Hyb}_0(A) \approx \text{Hyb}_1(A)\).

**Proof.** Identical to the proof of Lemma 4.5 (since the structure of the CRS in Construction 5.3 is identical to that in Construction 4.2 except for the additional random vectors \(u_i\)). \(\square\)

**Lemma 5.8.** Suppose \(n \geq \lambda\) and \(m \geq O(n \log q)\). Then, for all adversaries \(A\), \(\text{Hyb}_1(A) \approx \text{Hyb}_2(A)\).

**Proof.** Identical to the proof of Lemma 4.6. \(\square\)

**Lemma 5.9.** Suppose \(n \geq \lambda\) and \(m \geq 2n \log q\). Then, for all adversaries \(A\) and all polynomials \(\ell = \ell(\lambda)\), \(\text{Hyb}_2(A) \approx \text{Hyb}_3(A)\).

**Proof.** The only difference between \(\text{Hyb}_2\) and \(\text{Hyb}_3\) is the distribution of \(u_i\). In \(\text{Hyb}_2\), \(u_i \xleftarrow{\$} \mathbb{Z}_q^n\) while in \(\text{Hyb}_3\), \(u \leftarrow A_r\) where \(r_i \xleftarrow{\$} \{0, 1\}^m\). Since \(A \xleftarrow{\$} \mathbb{Z}^{n \times m}_q\) and \(m \geq 2n \log q\), the distribution of \(u_i\) in the two distributions are statistically indistinguishable by the leftover hash lemma (Lemma 2.2). The claim now follows by a hybrid argument (since \(\ell = \ell(\lambda)\) is polynomially-bounded). \(\square\)

**Lemma 5.10.** Suppose \(q\) is prime and \(m \geq n \log q + \lambda\). Under the BASIS\text{struct} assumption with parameters \((n, m, q, \ell(2B + p), s_0, \ell)\), for all efficient adversaries \(A\), \(\Pr[\text{Hyb}_3(A) = 1] = \text{negl}(\lambda)\).

**Proof.** Suppose there exists an efficient algorithm \(A\) where \(\Pr[\text{Hyb}_3(A) = 1] = \varepsilon\) for some non-negligible \(\varepsilon\). We use \(A\) to construct an adversary \(B\) for the BASIS\text{struct} assumption.

1. At the beginning of the game, algorithm \(B\) receives a challenge \(A \in \mathbb{Z}_q^{n \times m}\), \(B_r \in \mathbb{Z}_q^{nt \times (tm + m')}\), \(T \in \mathbb{Z}_q^{(tm + m' \times tm)}\), and aux = \((W_1, \ldots, W_\ell)\).

2. For each \(i \in [\ell]\), algorithm \(B\) samples \(r_i \xleftarrow{\$} \{0, 1\}^m\) and sets \(u_i \leftarrow A_r\). It gives \(\text{crs} = (A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T)\) to \(A\).

3. Algorithm \(A\) outputs a commitment \(c \in \mathbb{Z}_q^n\), a set \(S \subseteq [\ell]\), openings \(\pi = v, \pi' = v'\), and values \(\{(i, x_i, x'_i)\}_{i \in S}\).

4. Algorithm \(B\) outputs \(z = v - v' + \sum_{i \in S} (x_i - x'_i)r_i\).
By construction, algorithm $B$ perfectly simulates the common reference string according to the specification of $\text{Hyb}_3$. Thus, with probability $\epsilon, \|v\|, \|v'\| \leq |SB| \leq tB$ and moreover, there exists $i \in S$ where $x_i \neq x'_i$ and
\[
\sum_{i \in S} W_i^{-1}c = Av + \sum_{i \in S} x_i u_i = Av' + \sum_{i \in S} x'_i u_i.
\]

We can thus write
\[
0 = A(v - v') + \sum_{i \in S} (x_i - x'_i) u_i = A(v - v') + \sum_{i \in S} (x_i - x'_i) Ar_i = Az.
\]

Moreover, $\|z\| \leq 2tB + p|S| \leq \ell(2B + p)$. It suffices to show that $z \neq 0$. Here, we use a min-entropy entropy argument similar to the proof of Lemma 4.29:

- If adversary $A$ is successful, there exists some $i \in S$ where $x_i \neq x'_i$.
- Let $z' = (x_i - x'_i)^{-1}(v' - v - \sum_{j \in S \setminus \{i\}} (x_j - x'_j)r_j) \in \mathbb{Z}_q^m$. If $z = 0$, then $r_i = z' \in \mathbb{Z}_q^m$. By construction, $v', v, \{x_i\}_{i \in S}, \{x'_i\}_{i \in S}$ are functions of $u_i \in \mathbb{Z}_q^n$ (and other quantities that are independent of $r_i$). By construction, each $u_i$ contains at most $n \log q$ bits of information about $r_i$. Moreover, $r_i$ is sampled independently of $r_j$ for all $j \neq i$.

This means that
\[
H_{\infty}(r_i | z') \geq H_{\infty}(r_i | u_i) \geq m - n \log q \geq \lambda.
\]

This means that $\Pr[r_i = z'] \leq 2^{-\lambda}$, so with overwhelming probability, $r_i \neq z'$, and so $z \neq 0$. In this case, algorithm $B$ breaks the $\text{BASIS}_{\text{struct}}$ assumption with advantage at least $\epsilon - 2^{-\lambda}$.

Combining Lemmas 5.7 to 5.10, same-set binding holds.

**Parameter instantiation.** Let $\lambda$ be a security parameter and $\ell$ be the vector dimension. We can instantiate the lattice parameters in Construction 5.3 to satisfy Theorems 5.4 and 5.6:

- We set the lattice dimension $n = \lambda$ and $m = O(n \log q)$.
- We set $s_0 = O(\ell m^2 \log(n\ell))$ and $s_1 = O(\ell^3 m^{3/2} \log(n\ell) \cdot s_0) = O(\ell^4 m^3 \log^2(n\ell))$.
- We set the bound $B = \sqrt{\ell m + m's_1} = O(\ell^3 m^4 \log^2(n\ell)) = O(\ell^4 n^4 \log^2(n\ell) \log^4 q)$.
- We choose the modulus $q$ so that the $\text{BASIS}_{\text{struct}}$ assumption holds with parameters $(n, m, q, \beta, s_0, \ell)$ where $\beta = \ell(2B + p) = O(\ell^4 n^4 \log^2(q) \log^4 q + \ell p)$.

Note that this also requires that $\text{SIS}_{n,m,q,\beta}$ hold. For instance, we can set $q = \beta \cdot \text{poly}(n)$. In this case, $\log q = O(\log \lambda + \log \ell + \log p)$.

With this setting of parameters, we obtain an aggregatable commitment scheme over $\mathbb{Z}_p^\ell$ with the following succinctness properties:

- **Commitment size:** A commitment $\sigma$ to a vector $x \in \mathbb{Z}_p^n$ consists of a vector $\sigma = c \in \mathbb{Z}_q^n$, so $|\sigma| = O(n \log q) = O(\lambda \cdot (\log \lambda + \log \ell + \log p))$.
- **Opening size:** An opening $\pi$ to a set $S$ consists of a vector $\pi = v_S \in \mathbb{Z}_q^m$. Thus, the length of $v_i$ is bounded by $O(m \log q)$, or equivalently, $|\pi| = O(\lambda \cdot (\log^2 \lambda + \log^2 \ell + \log^2 p))$.
- **CRS size:** The CRS consists of $(A, \{W_i\}_{i \in [\ell]}, \{u_i\}_{i \in [\ell]}, T)$, where $A \in \mathbb{Z}_q^{n \times m}$, $W_i \in \mathbb{Z}_q^{n \times n}$, $u_i \in \mathbb{Z}_q^n$, and $T \in \mathbb{Z}_{q^{(\ell m + m')} \times (\ell m')}$. Thus, the total size of the CRS is $|\text{crs}| = nm \log q + \ell (n^2 + n) \log q + (\ell m + m') (\ell m') \log q = \ell^2 \cdot \text{poly}(\lambda, \log \ell, \log p)$.
Thus, Construction 5.3 is a succinct aggregatable vector commitment scheme that satisfies same-set binding:

**Corollary 5.11 (Aggregatable Vector Commitment from BASIS\(_{\text{struct}}\).)** Let \( \lambda \) be a security parameter. Then, for all polynomials \( t = t(\lambda) \) and \( p = p(\lambda) \), under the BASIS\(_{\text{struct}}\) assumption with a norm bound \( \beta = \poly(\lambda, t, p) \) and modulus \( q = \poly(\lambda, t, p) \), there exists an aggregatable vector commitment scheme over \( \mathbb{Z}_p^t \) that satisfies computational same-set binding. A commitment to a vector \( x \in \mathbb{Z}_p^t \) has size \( O(\lambda(\log \lambda + \log t + \log p)) \) and the opening to any set of indices \( S \subseteq [t] \) has size \( O(\lambda(\log^2 \lambda + \log^2 t + \log^2 p)) \). The CRS has size \( t^2 \cdot \poly(\lambda, t, \log p) \).

**Different-set binding.** While Construction 5.3 is an aggregatable vector commitment scheme satisfying same-set binding, it does not satisfy different-set binding. We describe an attack in Remark 5.12. Here, we describe a general attack strategy:

- Let \( \text{crs} = (A, \{W_i\}_{i \in [t]}, \{u_i\}_{i \in [t]}, T) \leftarrow \text{Setup}(1^t, 1^t) \).

- Take any two sets \( S, T \subseteq [t] \) where \( S \neq T \). Let \( W_S = \sum_{i \in S} W_i^{-1} \) and \( W_T = \sum_{i \in T} W_i^{-1} \). The adversary’s goal is to compute a commitment \( c \in \mathbb{Z}_q^t \) and short openings \( v_S \in \mathbb{Z}_q^m, v_T \in \mathbb{Z}_q^m \) such that

\[
W_S^{-1}A + \sum_{i \in S} x_i W_S^{-1}u_i = W_T^{-1}A + \sum_{i \in T} y_i W_T^{-1}u_i,
\]

and there exists \( i \in S \cap T \) where \( x_i \neq y_i \). We can rewrite this as

\[
\begin{bmatrix} W_S^{-1}A | - W_T^{-1}A \end{bmatrix} \begin{bmatrix} v_S \backslash v_T \end{bmatrix} = \sum_{i \in T} y_i W_T^{-1}u_i - \sum_{i \in S} x_i W_S^{-1}u_i.
\]

If the adversary has a trapdoor for \( \begin{bmatrix} W_S^{-1}A | - W_T^{-1}A \end{bmatrix} \), then it can sample a short \( v_S, v_T \) to satisfy Eq. (5.4) for any choice of \( x_i \) and \( y_i \). Given \( v_S, v_T \), the adversary can then compute the commitment \( c \) according to Eq. (5.3), which breaks different-set binding.

It suffices now to show how to obtain a trapdoor for \( \begin{bmatrix} W_S^{-1}A | - W_T^{-1}A \end{bmatrix} \). Suppose the adversary obtains many short vectors \( v_{j,i} \in \mathbb{Z}_q^m \) that satisfy \( A v_{j,i} = W_i^{-1}t_j \) for some fixed set of vectors \( t_j \). Then,

\[
W_S^{-1}A \sum_{i \in S} v_{j,i} = W_S^{-1} \sum_{i \in S} W_i^{-1}t_j = W_S^{-1}W_S t_j = t_j
\]

\[
W_T^{-1}A \sum_{i \in T} v_{j,i} = W_T^{-1} \sum_{i \in T} W_i^{-1}t_j = W_T^{-1}W_T t_j = t_j.
\]

This means that

\[
\begin{bmatrix} W_S^{-1}A | - W_T^{-1}A \end{bmatrix} \begin{bmatrix} \sum_{i \in S} v_{j,i} \\ \sum_{i \in T} v_{j,i} \end{bmatrix} = t_j - t_j = 0.
\]

If the adversary can construct \( 2m \) vectors \( z_j \in \mathbb{Z}_q^{2m} \) that are linearly independent, this yields an Ajtai-trapdoor (Definition 2.9) for \( \begin{bmatrix} W_S^{-1}A | - W_T^{-1}A \end{bmatrix} \), which suffices to break the scheme via the above blueprint. We now show that we can use the trapdoor \( T \) in the common reference string of Construction 5.3 to obtain the short vectors \( z_j \):

- Let \( B_T \in \mathbb{Z}_q^{(m \times m') \times t \times t'} \) be the matrix from Eq. (5.1). By construction, \( T \) in the CRS is a trapdoor for \( B_T \).
• For each $j \in [2m]$, we can sample a (short) preimage in the kernel of $B_i$:

$$
\begin{bmatrix}
  v_{j,1} \\
  \vdots \\
  v_{j,t} \\
  \hat{c}_j
\end{bmatrix} \leftarrow \text{SamplePre}(B_i, T, 0, s_i).
$$

By construction of $B_i$, we have that for all $i \in [\ell]$, $W_i Av_{j,i} + G\hat{c}_j = 0$, or equivalently, $Av_{j,i} = -W_i^{-1}G\hat{c}_j$. We can set $t_j = -G\hat{c}_j$. By the structure of $B_i$ and Lemma 2.7, the distribution of $\hat{c}_j$ is statistically close to $D^*_{Z_{s_i}}$, and each $v_{j,i}$ is independent and distributed according to $A_{Z_{s_i}}^{-1}(W_i^{-1}G\hat{c}_j)$. The marginal distribution of each $v_{j,i}$ is then distributed according to $D^*_{Z_{s_i}}$. Moreover, the vectors $v_{j,i}$ and $v_{j',i}$ for $j \neq j'$ are independent.

• Heuristically, the vectors $z_j = \left[ \sum_{i \in S} v_{j,i} \atop \sum_{i \in T} v_{j,i} \right] \in \mathbb{Z}^{2m}$ are linearly independent for different $j$, and thus, can be used to construct a trapdoor for $[W_S^{-1}A | - W_T^{-1}A]$. Note that this step requires that $S \neq T$ (otherwise, the vectors $\{z_1, \ldots, z_{2m}\}$ lie in the column span of $[I_m \atop 0]$ and span a vector space of rank at most $m$).

We note that our attack strategy described above is general and having a trapdoor $T$ for $B_i$ is not essential to the attack. For instance, we can also consider a variant of the construction using an $[ACL^{*22}]$-style CRS distribution which contains vectors of the form $v_{j,i} = A^{-1}(W^{-1}t_j)$ for $i \neq j \in [\ell]$ (see Sections 1.2 and 6). Our general attack strategy above still applies in this setting, so long as $\ell - |S \cup T| \geq 2m$ (i.e., the CRS components allows the adversary to compute $z_j = \left[ \sum_{i \in S} v_{j,i} \atop \sum_{i \in T} v_{j,i} \right]$ for every $j \notin S \cup T$.

**Remark 5.13** (Different-Set Binding for Honestly-Generated Commitments). While Construction 5.3 does not satisfy different-set binding for maliciously-chosen commitments, it does satisfy different-set binding in the setting where the commitment is honestly-generated; this is analogous to the notion of target binding in the setting of functional commitments (Definition 4.22). In this setting, the adversary in the different-set binding game receives the CRS and chooses a vector $x \in \mathbb{Z}_q^\ell$. The challenger replies with a commitment $\sigma \leftarrow \text{Commit}(crs, x)$. At the end of the game, the adversary $\mathcal{A}$ wins if it outputs two sets $S, T \subseteq [\ell]$ along with openings $\pi_S$ and $\pi_T$ to values $\{(i, x_i)\}_{i \in S}$ and $\{(i, x'_i)\}_{i \in T}$, respectively. The adversary wins if $\pi_S$ and $\pi_T$ are valid openings and moreover, there exists $i \in S \cap T$ where $x_i = x'_i$. This binding notion is called weak binding by Gorbunov et al. [GRWZ20] and suffices for applications where the commitment is guaranteed to be well-formed (say, due to external protocol constraints or consensus mechanisms).

It is easy to see that the binding scheme implies different-set binding for honestly-generated commitments. Suppose an adversary is able to open $\sigma$ to $\{(i, x_i)\}_{i \in S}$ with a proof $\pi_S$ and to $\{(i, x'_i)\}_{i \in T}$ with a proof $\pi_T$ such that there exists $i \in S \cap T$ where $x_i = x'_i$. Let $\pi'_S$ and $\pi'_T$ be the honestly-computed openings of $\sigma$ to $S$ and $T$, respectively. Since $x_i = x'_i$, either $(\pi_S, \pi'_S)$ is a pair of inconsistent openings of $\sigma$ to $S$, or $(\pi_T, \pi'_T)$ is a pair of inconsistent openings of $\sigma$ to $T$. This breaks same-set binding of the underlying commitment scheme.

**Remark 5.14** (Aggregatable Commitments with an $[ACL^{*22}]$-Style CRS). We can also obtain a variant of Construction 5.3 using an $[ACL^{*22}]$-style CRS that does not include an explicit trapdoor for the matrix $B_i$, and instead, just contains a collection of short preimages. We provide a more detailed comparison of the two approaches in Section 6.

• The CRS consists of matrices $A \overset{\text{R}}{\overset{\quad \quad \quad m \times n}{\rightsquigarrow}} W_i \overset{\text{R}}{\overset{\quad \quad \quad \ell \times n}{\rightsquigarrow}}$ for each $i \in [\ell]$, vectors $u_i \overset{\text{R}}{\overset{\quad \quad \quad n}{\rightsquigarrow}} \mathbb{Z}_q^n$, and short vectors $z_{j,i} \in \mathbb{Z}_q^m$ for all $i \neq j$ where $Az_{j,i} = W_i^{-1}W_ju_j$.

• A commitment to a vector $x \in \mathbb{Z}_q^\ell$ is a vector $c = \sum_{i \in [\ell]} x_iW_iu_i \in \mathbb{Z}_q^n$. The opening to an index $i$ is the vector $v_i = \sum_{j \in \ell} x_jz_{j,i}$. The verification relation checks that $\|v_i\|$ is small and that

$$
W_i^{-1}c = x_iu_i + Av_i.
$$

Correctness follows from the fact that

$$
W_i^{-1}c = W_i^{-1} \sum_{i \in [\ell]} x_iW_iu_i = x_iu_i + \sum_{i \neq j} x_jW_i^{-1}W_ju_j = x_iu_i + \sum_{i \neq j} x_jAz_{j,i} = x_iu_i + Av_i.
$$

45
• Observe that Eq. (5.5) is the same verification relation from Construction 5.3. Thus, the same approach for aggregation applies.

Computational same-set binding in this case would rely on a variant of the ISIS assumption that asserts that given \( (A, \{ W_i \}_{i \in [\ell]}, \{ u_i \}_{i \in [\ell]}, \{ z_{ij} \}_{i, j \in [\ell]}) \), where \( z_{ij} \) are short vectors that satisfy \( Az_{ij} = W_i^{-1} W_j u_i \), no efficient adversary can compute a short \( v_i \) where \( Av_i = W_i^{-1} W_j u_i = u_i \). Note that this version of the construction still does not satisfy different-set binding. The vectors \( z_{ij} \) that are included as part of the CRS (which are necessary for correctness) still suffice to carry out the attack in Remark 5.12.

5.1 Aggregatable Functional Commitments

Our techniques for aggregating vector commitments also applies to aggregating openings for the functional commitment scheme from Section 4 (Construction 4.2). Recall that in the case of a functional commitment, an opening \( \pi \) is defined with respect to a function \( f \) and a value \( y \). Much like the setting with vector commitments, the goal here is to take a collection of \( t \) function-value-opening tuples \( \{(f_i, y_i, \pi_i)\}_{i \in [t]} \) and aggregate the openings into a single opening \( \pi \) whose size scales sublinearly with \( t \). We start by extending Definition 5.1 to the setting of functional commitments:

**Definition 5.15 (Aggregatable Functional Commitment).** Let \( \lambda \) be a security parameter, \( T = T(\lambda) \) be an aggregation bound, and \( \mathcal{F} = \{f_j\}_{j \in \mathbb{N}} \) be a family of functions \( f : \{0, 1\}^\ell \rightarrow \{0, 1\} \) on inputs of length \( \ell = \ell(\lambda) \) and depth at most \( d = d(\lambda) \). A \( T \)-aggregatable functional commitment for function family \( \mathcal{F} \) is a tuple of efficient algorithms \( \Pi_{\text{AFC}} = (\text{Setup}, \text{Commit}, \text{Eval}, \text{Verify}, \text{Aggregate}, \text{VerifyAgg}) \) where (\text{Setup}, \text{Commit}, \text{Eval}, \text{Verify}) is a functional commitment scheme over \( \mathcal{F} \) and the additional functions (\text{Aggregate}, \text{VerifyAgg}) satisfy the following properties:

- **Aggregates (\text{Aggregate}, \text{VerifyAgg}):** On input the common reference string \( \text{crs} \), a commitment \( \sigma \), a set of openings \( \pi_i \) for functions \( f_i \in \mathcal{F} \) and values \( y_i \), the aggregation algorithm outputs an aggregated opening \( \pi \).
- **VerifyAgg (\text{VerifyAgg}):** On input the common reference string \( \text{crs} \), a commitment \( \sigma \), a set of openings \( \pi_i \) for functions \( f_i \in \mathcal{F} \) and values \( y_i \), the aggregate verification algorithm outputs a bit \( b \in \{0, 1\} \).

In addition, we allow the Setup algorithm to take as input the aggregation bound parameter \( 1^T \) (encoded in unary). Next, in addition to the basic properties of a functional commitment scheme, an aggregatable functional commitment scheme should satisfy the following additional properties:

- **Correctness of aggregation:** For all aggregation bound parameters \( T = T(\lambda) \), there exists a negligible function \( \negl() \) such that for all security parameters \( \lambda \in \mathbb{N} \), and sampling \( \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^\ell, 1^d, 1^T) \), and for all \( t \leq T \), commitments \( \sigma \), and openings \( \{(f_i, y_i, \pi_i)\}_{i \in [t]} \) where \( \text{Verify}(\text{crs},\sigma,f_i,y_i,\pi_i)=1 \) for all \( i \in [t] \), it holds that

\[
\Pr[\text{VerifyAgg}(\text{crs}, \sigma, \{(f_i, y_i)\}_{i \in [t]}, \pi') : \pi' \leftarrow \text{Aggregate}(\text{crs}, \sigma, \{(f_i, y_i, \pi_i)\}_{i \in [t]})] = 1 - \negl(\lambda),
\]

where the probability is taken over the randomness of Setup.

- **Succinctness:** The aggregatable functional commitment scheme is succinct if there exists a universal polynomial \( \text{poly}(\cdot,\cdot,\cdot,\cdot) \) such that for all \( \lambda \in \mathbb{N} \), |\pi'| = \text{poly}(\lambda, d, \log \ell, \log T) in the correctness of aggregation definition.

- **Same-function binding:** We say \( \Pi_{\text{AFC}} \) satisfies statistical (resp., computational) same-function binding if for all polynomials \( T = T(\lambda) \) and all adversaries \( \mathcal{A} \) (resp., efficient adversaries \( \mathcal{A} \)),

\[
\Pr \left[ \begin{array}{l}
\text{VerifyAgg}(\text{crs}, \sigma, \{(f_i, y_i)\}_{i \in [t]}, \pi) = 1 \\
\text{and } y_i \neq y_i' \text{ for some } i \in [t] \text{ and} \\
\text{VerifyAgg}(\text{crs}, \sigma, \{(f_i, y_i')\}_{i \in [t]}, \pi') = 1
\end{array} \right] = \negl(\lambda).
\]

Aggregatable functional commitments. We now show how to adapt Construction 4.2 to support opening aggregation. Our construction can be viewed as a combination of Construction 4.2 and Construction 5.3:
Construction 5.16 (Aggregatable Functional Commitment). Let $\lambda$ be a security parameter and let $\mathcal{F} = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f : \{0, 1\}^f \to \{0, 1\}$ where each $f : \{0, 1\}^f \to \{0, 1\}$ is a function on inputs of length $f = \ell(\lambda)$ and which can be computed by a Boolean circuit of depth at most $d = d(\lambda)$. Let $n, m, q, m', B, s_0, s_1$ be the scheme parameters from Construction 4.2. Let $T = T(\lambda)$ be an arbitrary aggregation bound and $B_{agg} = B_{agg}(\lambda)$ be an aggregation norm bound. We now extend Construction 4.2 to obtain an aggregatable functional commitment $\Pi_{AVC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Verify}, \text{Aggregate}, \text{VerifyAgg})$ for $\mathcal{F}$:

- Setup($1^\lambda$, $1^T$, $1^d$, $1^T$): Sample $A, W_1, \ldots, W_t, T$ according to the specification of Setup($1^\lambda$, $1^T$, $1^d$) in Construction 4.2. Then, for each $i \in [T]$, sample $u_i \leftarrow \mathbb{Z}_q^n$, and output crs = ($A, W_1, \ldots, W_t, T, u_1, \ldots, u_T$).
- Commit(crs, x): Same as in Construction 4.2.
- Eval(crs, st, f): Same as in Construction 4.2.
- Verify(crs, f, y, π): Same as in Construction 4.2.
- Aggregate(crs, σ, {($f_i, y_i, π_i$)}$_{i \in [t]}$): On input the common reference string crs = ($A, W_1, \ldots, W_t, T, u_1, \ldots, u_T$), a commitment $σ = C \in \mathbb{Z}_q^{n \times m'}$, a collection of functions $f_1, \ldots, f_t \in \mathcal{F}$ (arranged in lexicographic order), values $y_i \in \{0, 1\}$, and openings $π_i = V_i \in \mathbb{Z}_q^{m \times m'}$, the aggregation algorithm outputs $π = v \leftarrow \sum_{i \in [t]} V_i \cdot G^{-1}(u_i) \in \mathbb{Z}_q^n$.
- VerifyAgg(crs, σ, {($f_i, y_i, π_i$)}$_{i \in [t]}$, π): On input the common reference string crs = ($A, W_1, \ldots, W_t, T, u_1, \ldots, u_T$), a commitment $σ = C \in \mathbb{Z}_q^{n \times m'}$, a collection of functions $f_1, \ldots, f_t \in \mathcal{F}$ (arranged in lexicographic order), values $y_i \in \{0, 1\}$, and an opening $π = v \in \mathbb{Z}_q^n$, the verification algorithm first sets $\hat{C} \leftarrow [W_1^{-1}C | \cdots | W_t^{-1}C]$. Then, for each $i \in [t]$, it computes $\hat{C}_i \leftarrow \text{EvalF}(\hat{C}, f_i)$. Finally, it outputs 1 if

$$\|v\| \leq B_{agg} \quad \text{and} \quad \text{Av} = \sum_{i \in [t]} \hat{C}_i G^{-1}(u_i) - \sum_{i \in [t]} y_i u_i.$$

Theorem 5.17 (Correctness of Aggregation). Suppose $B_{agg} \geq Tm'B$. Then, Construction 5.16 satisfies correctness of aggregation.

Proof. Take any $T = T(\lambda)$ and let crs = ($A, W_1, \ldots, W_t, T, u_1, \ldots, u_T$) ← Setup($1^\lambda$, $1^T$, $1^d$, $1^T$). Take any $t \leq T$, any commitment $σ = C \in \mathbb{Z}_q^{n \times m'}$, and set of openings ($f_i, y_i, π_i$) for $i \in [t]$, where $f_1, \ldots, f_t \in \mathcal{F}$ are in lexicographic order, and $π_i = V_i \in \mathbb{Z}_q^{m \times m'}$. Let $\hat{C} \leftarrow [W_1^{-1}C | \cdots | W_t^{-1}C]$ and $\hat{C}_i \leftarrow \text{EvalF}(\hat{C}, f_i)$ for each $i \in [t]$. Suppose Verify(crs, σ, $f_i$, $y_i$, π) = 1. By definition this means that

$$\|V_i\| \leq B \quad \text{and} \quad \text{Av}_i = \hat{C}_i - y_iG.$$

Suppose $v \leftarrow \text{Aggregate}(\text{crs}, C, \{(f_i, y_i, π_i)\}_{i \in [t]})$ and consider VerifyAgg(crs, C, {($f_i, y_i$)}$_{i \in [t]}$, v). By definition, $v = \sum_{i \in [t]} V_i G^{-1}(u_i)$. Since $\|V_i\| \leq B$, this means that $\|v\| \leq tm'B \leq Tm'B = B_{agg}$. Moreover,

$$\text{Av} = \sum_{i \in [t]} \text{Av}_i G^{-1}(u_i) = \sum_{i \in [t]} \hat{C}_i G^{-1}(u_i) - \sum_{i \in [t]} y_i u_i,$$

and the aggregate verification algorithm accepts. \qed

Theorem 5.18 (Computational Same-Function Binding). Suppose $n \geq \lambda, m \geq O(n \log q)$, and $s_0 \geq O(tm^2 \log(nt))$. Then, under the BASIS$_{\text{struct}}$ assumption with parameters $(n, m, q, 2B_{agg} + T, s_0, f)$, Construction 5.16 satisfies same-function binding.

Proof. The proof follows by a combination of the proofs of Theorem 4.4 and Theorem 5.6. Specifically, let $T = T(\lambda)$ be a polynomial. We now define a sequence of hybrid experiments:

- Hyb$_{\lambda}$: This is the same-function binding experiment:

11This ensures that there is a canonical ordering for all tuples of functions ($f_1, \ldots, f_t$).
The challenger starts by sampling \((A, R) \leftarrow \text{TrapGen}(1^n, q, m)\) and \(W_i \leftarrow \mathbb{Z}_q^{n \times m}\) for each \(i \in [\ell]\). It sets \(R_i \leftarrow \text{RG}^{-1}(W_i^{-1}G)\) and constructs \(B_i\) and \(R_i\) according to Eq. (4.1). It samples \(T \leftarrow \text{SamplePre}(B_i, R, G_{nt}, s_0)\), \(u_i \leftarrow \mathbb{Z}_q^n\) for each \(i \in [T]\), and gives \(\text{crs} = (A, W_1, \ldots, W_\ell, T, u_1, \ldots, u_T)\) to the adversary \(\mathcal{A}\).

- Algorithm \(\mathcal{A}\) outputs a commitment \(C \in \mathbb{Z}_q^{n \times m}\), a set of functions \(f_1, \ldots, f_\ell\) (arranged in lexicographic order), a set of values \(y_1, \ldots, y_\ell \in \{0, 1\}\), \(y'_1, \ldots, y'_\ell \in \{0, 1\}\), with \(t \leq T\), and openings \(v, v' \in \mathbb{Z}_q^m\).

- The output of the experiment is \(1\) if \(||v|| \cdot ||v'|| \leq B_{\text{agg}}\) and

\[
\text{Av} = \sum_{i \in [\ell]} \tilde{C}_i G^{-1}(u_i) - \sum_{i \in [\ell]} y_i u_i \quad \text{and} \quad \text{Av}' = \sum_{i \in [\ell]} \tilde{C}_i G^{-1}(u_i) - \sum_{i \in [\ell]} y'_i u_i,
\]

where \(\tilde{C}_i \leftarrow \text{EvalF}(\tilde{C}, f_i)\) and \(\tilde{C} = [W_1^{-1}C | \cdots | W_\ell^{-1}C]\). Otherwise, the experiments outputs 0.

- Hyb$_1$: Same as Hyb$_b$ except after constructing the matrix \(B_i \in \mathbb{Z}_q^{nt \times (tm + m')}\) according to Eq. (4.1), the challenger samples \(T \leftarrow (B_i)_{s_0}^{-1}(G_{nt})\) without using the trapdoor \(R\). The CRS is now sampled independently of \(R\).

- Hyb$_2$: Same as Hyb$_1$ except the challenger samples \(A \leftarrow \mathbb{Z}_q^{n \times m}\).

- Hyb$_3$: Same as Hyb$_2$ except for each \(i \in [\ell]\), it samples \(r_i \leftarrow \mathbb{Z}_q \times \{0, 1\}^m\) and sets \(u_i \leftarrow A r_i\).

For an adversary \(\mathcal{A}\), we write \(\text{Hyb}_b(\mathcal{A})\) to denote the output of an execution of Hyb$_b$ with adversary \(\mathcal{A}\). We now show that each adjacent pair of experiments are computationally indistinguishable.

**Lemma 5.19.** Suppose \(n \geq \lambda, m \geq O(n \log q)\), and \(s_0 \geq O(tm^2 \log(nt))\). Then, for all adversaries \(\mathcal{A}\), \(\text{Hyb}_b(\mathcal{A}) \approx \text{Hyb}_1(\mathcal{A})\).

**Proof.** Identical to the proof of Lemma 4.5. \(\square\)

**Lemma 5.20.** Suppose \(n \geq \lambda\) and \(m \geq O(n \log q)\). Then, for all adversaries \(\mathcal{A}\), \(\text{Hyb}_1(\mathcal{A}) \approx \text{Hyb}_2(\mathcal{A})\).

**Proof.** Identical to the proof of Lemma 4.6. \(\square\)

**Lemma 5.21.** Suppose \(n \geq \lambda\) and \(m \geq 2n \log q\). Then for all adversaries \(\mathcal{A}\) and all polynomials \(T = T(\lambda)\), \(\text{Hyb}_2(\mathcal{A}) \approx \text{Hyb}_2(\mathcal{A})\).

**Proof.** Identical to the proof of Lemma 5.9 (except using the fact that \(T\) is polynomially-bounded). \(\square\)

**Lemma 5.22.** Suppose \(m \geq n \log q + \lambda\). Under the BASISstruct assumption with parameters \((n, m, q, 2B_{\text{agg}} + T, s_0, \ell)\), for all efficient adversaries \(\mathcal{A}\), \(\Pr[\text{Hyb}_b(\mathcal{A}) = 1] = \negl(\lambda)\).

**Proof.** Suppose there exists an efficient algorithm \(\mathcal{B}\) where \(\Pr[\text{Hyb}_b(\mathcal{A}) = 1] = \epsilon\) for some non-negligible \(\epsilon\). We use \(\mathcal{A}\) to construct an efficient adversary \(\mathcal{B}\) for the BASISstruct assumption.

1. At the beginning of the game, algorithm \(\mathcal{B}\) receives a challenge \(A \in \mathbb{Z}_q^{n \times m}\), \(B_i \in \mathbb{Z}_q^{nt \times (tm + m')}\), \(T \in \mathbb{Z}_q^{(tm + m') \times tm'}\), and \(\text{aux} = (W_1, \ldots, W_\ell)\).

2. For each \(i \in [T]\), algorithm \(\mathcal{B}\) samples \(r_i \leftarrow \mathbb{Z}_q \times \{0, 1\}^m\) and sets \(u_i \leftarrow A r_i\). It gives \(\text{crs} = (A, W_1, \ldots, W_\ell, T, u_1, \ldots, u_T)\) to \(\mathcal{A}\).

3. Algorithm \(\mathcal{A}\) outputs a commitment \(C \in \mathbb{Z}_q^{n \times m'}\), a set of functions \(f_1, \ldots, f_\ell\) (arranged in lexicographic order), a set of values \(y_1, \ldots, y_\ell \in \{0, 1\}\), \(y'_1, \ldots, y'_\ell \in \{0, 1\}\), with \(t \leq T\), and openings \(v, v' \in \mathbb{Z}_q^m\).

4. Algorithm \(\mathcal{B}\) outputs \(z = v - v' + \sum_{i \in [\ell]} (y_i - y'_i) r_i\).
By construction, algorithm $B$ perfectly simulates the common reference string according to the specification of Hyb$_3$. Thus, with probability $\varepsilon$, $\|v\|, \|v'\| \leq B_{agg}, t \leq T$, and moreover, there exists $i \in [t]$ where $y_i \neq y'_i$ and

$$\sum_{i \in [t]} \tilde{C}_i G^{-1}(u_i) = Av + \sum_{i \in [t]} y_i u_i = A v' + \sum_{i \in [t]} y'_i u_i,$$

Rearranging terms, we now have

$$0 = A(v - v') + \sum_{i \in [t]} (y_i - y'_i) u_i = A(v - v') + \sum_{i \in [t]} (y_i - y'_i) A r_i = A z.$$

Moreover, $\|z\| \leq 2B_{agg} + t \leq 2B_{agg} + T$. It suffices to show that $z \neq 0$. Here, we use a min-entropy argument similar to the proof of Lemma 5.10:

- From above, there exists some $i \in [t]$ where $y_i \neq y'_i$ (and $y_i, y'_i \in \{0,1\}$ so $y_i - y'_i \in \{-1,1\}$).
- Let $z' = (y_i - y'_i)^{-1}(v' - v - \sum_{j \neq i} (y_j - y'_j) r_j) \in \mathbb{Z}_q^m$. If $z = 0$, then $r_i = z' \in \mathbb{Z}_q^m$. By construction, $v'$, $v$, $\{y_i\}_{i \in [t]}$, $\{y'_i\}_{i \in [t]}$ are functions of $u_i \in \mathbb{Z}_q^m$ (and other quantities that are independent of $r_i$). By construction, each $u_i$ contains at most $n \log q$ bits of information about $r_i$. Moreover, $r_i$ is sampled independently of $r_j$ for all $j \neq i$. This means that

$$H_{\infty}(r_i \mid z') \geq H_{\infty}(r_i \mid u_i) \geq m - n \log q \geq \lambda.$$

This means that $\Pr[r_i = z'] \leq 2^{-\lambda}$, so with overwhelming probability $z \neq 0$, and $B$ breaks the BASIS$_{struct}$ assumption with advantage at least $\varepsilon - 2^{-\lambda}$.

Same-function binding now follows by combining Lemmas 5.19 to 5.22.

Parameter instantiation. Let $\lambda$ be a security parameter and $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f : \{0,1\}^\ell \rightarrow \{0,1\}$ on inputs of length $\ell = \ell(\lambda)$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. Let $T = T(\lambda)$ be an aggregation bound. We can instantiate the lattice parameters in Construction 5.16 similar to those for Construction 4.2. We summarize the instantiation below:

- Let $\varepsilon > 0$ be a constant. We set the lattice dimension $n = d^{3/\varepsilon} \cdot \text{poly}(\lambda)$ and $m = O(n \log q)$.
- We set $s_0 = O(tm^2 \log(n\ell))$ and $s_1 = O(\ell^{3/2}m^{3/2} \log(n\ell) \cdot s_0) = O(\ell^{3/2}m^{7/2} \log^2(n\ell)) = O(\ell^{3/2}n^{7/2} \log^2(n\ell) \log^{7/2} q)$.
- We set the bound $B = s_1 \cdot \sqrt{tm + m^3} \cdot (n \log q)^{O(d)} = \ell^3 \log^2 \ell \cdot (n \log q)^{O(d)}$ and the aggregation bound $B_{agg} = TmB = T\ell^3 \log^2 \ell \cdot (n \log q)^{O(d)}$.
- We set the modulus $q$ so that the BASIS$_{struct}$ assumption holds with parameters $(n, m, q, \beta, s_0, \ell)$, where $\beta = 2B_{agg} + T = T \ell^3 \log^2 \ell \cdot (n \log q)^{O(d)} = 2^{O(d)} = 2^{O(n^\epsilon)}$,

where we write $\tilde{O}(\cdot)$ to suppress polylogarithmic factors in $\lambda, d, \ell, T$. Note that this also requires that $\text{SIS}_{n,m,q,\beta}$ hold. For instance, we set $q = \beta \cdot \text{poly}(n)$. Then, $\log q = \text{poly}(d, \log \lambda, \log \ell, \log T)$. Note that the underlying SIS assumption relies on a sub-exponential noise bound.

With this setting of parameters, we obtain an aggregatable functional commitment scheme for $\mathcal{F}$ with the following properties:

**Corollary 5.23 (Aggregatable Functional Commitment from BASIS$_{struct}$).** Let $\lambda$ be a security parameter, and let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f : \{0,1\}^\ell \rightarrow \{0,1\}$ on inputs of length $\ell = \ell(\lambda)$ and which can be computed by Boolean circuits of depth at most $d = d(\lambda)$. Let $T = T(\lambda)$ be an arbitrary polynomial. Under the BASIS$_{struct}$ assumption with a norm bound $\beta = 2^{O(d)}$ and modulus $q = 2^{O(d)}$, there exists a computationally-binding $T$-aggregatable functional commitment scheme for $\mathcal{F}$. The size of the commitment is $\text{poly}(\lambda, d, \log \ell, \log T)$, and an aggregate opening to a set of up to $T$ functions has size $\text{poly}(\lambda, d, \log \ell, \log T)$. The size of the CRS is $(\ell^3 + T) \cdot \text{poly}(\lambda, d, \log \ell, \log T)$. Here, $\tilde{O}(\cdot)$ suppresses polylogarithmic factors in $\lambda, d, \ell, T$.
We now compare the relative power of these two types of auxiliary information. We refer to the above auxiliary data with the aggregation bound $T$ as "Type I" auxiliary data and "Type II" auxiliary data, respectively.

The Albrecht et al. approach:

Our approach:

Next, we show that we can also use Type II auxiliary data to obtain a trapdoor for $\mathbf{B}_t$. We illustrate this with a concrete example. Suppose we want to obtain a trapdoor for the matrix $\mathbf{B}_k$ (where $k < t/m$):

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{A}_1 & -\mathbf{G} \\ \vdots & \vdots \\ \mathbf{A}_k & -\mathbf{G} \end{bmatrix} \in \mathbb{Z}_q^{km \times (km + m')}.$$
While asking for security given a full trapdoor for the related matrix \( B \) (Assumption 3.3) underlying our functional, polynomial, and aggregatable commitments. Consider a variant of the

To facilitate cryptanalysis of our new assumption, we provide an equivalent formulation of the

This distinction is important, and as we discuss in Remark 6.1, Type I auxiliary data seem essential to realizing the

about this is to appeal to the “evasive LWE” assumption [Wee22, Tsa22], which roughly says that if

\[
\text{It is easy to see that if LWE holds with public matrix } A
\]

One way to understand this assumption is to consider the analogous

The assumption in the LWE setting.

The assumption in the LWE setting. One way to understand this assumption is to consider the analogous assumption in the learning with errors (LWE) setting [Reg05]. Recall that the vanilla LWE assumption says that the distribution \( (A, s^T A + e^T) \) is pseudorandom when \( A \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}, s \overset{\$}{\leftarrow} \mathbb{Z}_q^n, \) and \( e \leftarrow \chi^m \), where \( \chi \) is an error distribution. It is easy to see that if LWE holds with public matrix \( A \), then SIS is also hard with respect to \( A \). Correspondingly, we can then ask whether LWE is hard with respect to \( A \) given \( A^{-1}(W_i^{-1}R) \) for each \( i \in [\ell] \). One way to reason about this is to appeal to the “evasive LWE” assumption [Wee22, Tsa22], which roughly says that if

\[
(A, B, s^T A + e^T, s^T B + e^T) \overset{\$}{\leftarrow} (A, B, u_1, u_2),
\]

\[
\text{In contrast, when the target vectors in the verification relation are uniform, it is straightforward to translate between the notions. For example, we refer to Remark 5.14 for an analog of the aggregatable commitment scheme (Construction 5.3) using Type II auxiliary data. In this case, both schemes satisfy same-set binding (assuming hardness of SIS/ISIS even given the auxiliary data), and neither satisfy different-set binding.
\]

Remark 6.1 (Structured Targets and Functional Commitments). The main verification relation of our functional commitment scheme (Construction 4.2) is

\[
C = A_i V_i + x_i G
\]

If we consider a Type II auxiliary data for this verification relation, the auxiliary data would contain \( A_i^{-1}(G) \), or equivalently, \( A^{-1}(W_i^{-1}G) \). However, \( A^{-1}(W_i^{-1}G) \) is a trapdoor for \( A \) (with tag \( W_i^{-1} \)), which trivially breaks security. In contrast, using Type I auxiliary data does not appear to yield a trapdoor for \( A \), and plausibly yields a succinct functional commitment scheme.

6.1 Another View of the BASISstruct Assumption

To facilitate cryptanalysis of our new assumption, we provide an equivalent formulation of the BASISstruct assumption (Assumption 3.3) underlying our functional, polynomial, and aggregatable commitments. Consider a variant of the BASISstruct assumption where \( T \) is an Ajtai trapdoor (Definition 2.9) for \( B \) (i.e., \( T \leftarrow B^{-1}(0^{m \times 2m}) \)). Note that we can efficiently convert between gadget trapdoors and Ajtai trapdoors, up to small polynomial losses in the quality of the trapdoor. It is easy to see that we can re-express \( B^{-1}(0^{m \times 2m}) \) as \( A^{-1}(W_i^{-1}R) \) for all \( i \in [\ell] \), and \( R \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times 2m} \). Therefore, the BASISstruct assumption is equivalent to:

\[
\text{SIS is hard with respect to } A \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m} \text{ given } A^{-1}(W_i^{-1}R) \text{ for all } i \in [\ell], \text{ where } W_i \overset{\$}{\leftarrow} \mathbb{Z}_q^{m \times n} \text{ and } R \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times 2m}.
\]

\[
\text{The assumption in the LWE setting. One way to understand this assumption is to consider the analogous assumption in the learning with errors (LWE) setting [Reg05]. Recall that the vanilla LWE assumption says that the distribution } (A, s^T A + e^T) \text{ is pseudorandom when } A \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}, s \overset{\$}{\leftarrow} \mathbb{Z}_q^n, \text{ and } e \leftarrow \chi^m, \text{ where } \chi \text{ is an error distribution. It is easy to see that if LWE holds with public matrix } A, \text{ then SIS is also hard with respect to } A. \text{ Correspondingly, we can then ask whether LWE is hard with respect to } A \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m} \text{ given } A^{-1}(W_i^{-1}R) \text{ for each } i \in [\ell]. \text{ One way to reason about this is to appeal to the “evasive LWE” assumption [Wee22, Tsa22], which roughly says that if}
\]

\[
(A, B, s^T A + e^T, s^T B + e^T) \overset{\$}{\leftarrow} (A, B, u_1, u_2),
\]

\[
\text{In contrast, when the target vectors in the verification relation are uniform, it is straightforward to translate between the notions. For example, we refer to Remark 5.14 for an analog of the aggregatable commitment scheme (Construction 5.3) using Type II auxiliary data. In this case, both schemes satisfy same-set binding (assuming hardness of SIS/ISIS even given the auxiliary data), and neither satisfy different-set binding.}
where $A \leftarrow \mathbb{Z}_q^{n \times m}$, $B \in \mathbb{Z}_q^{n \times m'}$, $s \leftarrow \mathbb{Z}_q^n$, $e_1 \leftarrow \chi^m$, $e_2 \leftarrow \chi^{m'}$, $u_1 \leftarrow \mathbb{Z}_q^m$, and $u_2 \leftarrow \mathbb{Z}_q^{m'}$, then

$$(A, B, s^T A + e_1^T, A^{-1}(B)) \approx (A, B, u_1, A^{-1}(B)).$$

Thus, under the evasive LWE assumption, if $s^T W^{-1} R + e_1^{-1}$ is pseudorandom for all $i \in [\ell]$, then LWE (and correspondingly, SIS) is hard with respect to $A$ even given $A^{-1}(W_i^{-1} R)$. This question itself is closely related to the question of building simpler PRFs from lattices (c.f., the discussion in [BPR12, §1.2, 1.3]). To the best of our knowledge, there are no known attacks on pseudorandomness of the above distribution.

**Remark 6.2** (Parameter Choices for the BASIS\textsubscript{struct} assumption). While hardness of the BASIS\textsubscript{rand} assumption can be based on the hardness of the standard SIS assumption, we do not know of an analogous reduction for the BASIS\textsubscript{struct} assumption. When setting parameters for the BASIS\textsubscript{struct} assumption, we use Theorem 3.4 as a guide and consider instantiations where $n \geq \lambda$, $m \geq O(n \log q)$ and $s \geq O(\ell m \log n) = \text{poly}(\lambda, \ell)$. Note that this means the quality of the basis decreases with the dimension. For this parameter setting, we are not aware of any concrete attacks on the BASIS\textsubscript{struct} assumption and conjecture that its security is comparable with the hardness of $\text{SIS}_{n,m,q,\beta}$, with a noise bound bound $\beta = \text{poly}(\lambda, \ell)$ that scales with the dimension of the vector (as in Theorem 3.4). In particular, the hardness of the SIS instance decreases with the dimension $\ell$ (similar to the case with $q$-type assumptions over groups [Che06]).

**Acknowledgments**

We thank Jonathan Bootle for helpful conversations about lattice-based SNARKs. We thank Dario Fiore and Chris Peikert for helpful discussions on the concurrent works. D. J. Wu is supported by NSF CNS-2151131, CNS-2140975, a Microsoft Research Faculty Fellowship, and a Google Research Scholar award.

**References**


