Unbounded Distributed Broadcast Encryption and Registered ABE from Succinct LWE

Hoeteck Wee
NTT Research
wee@di.ens.fr

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

Abstract

We construct distributed broadcast encryption and registered attribute-based encryption (ABE) that support an arbitrary polynomial of users from the succinct LWE assumption. Specifically, if we take λ to be the security parameter and N to be the number of users, we obtain the following:

- We obtain a distributed broadcast encryption scheme where the size of the public parameters, user public/secret keys, and ciphertexts are optimal (i.e., have size $\mathsf{poly}(\lambda, \log N)$). Security relies on the $\mathsf{poly}(\lambda, \log N)$-succinct LWE assumption. Previously, this was only known from indistinguishability obfuscation or witness encryption. All constructions that did not rely on these general tools could only support an a priori bounded number of users.
- We obtain a key-policy registered ABE scheme that supports arbitrary bounded-depth Boolean circuit policies from the $\mathsf{poly}(\lambda, d, \log N)$-succinct LWE assumption in the random oracle model, where d is the depth of the circuit computing the policy. The public parameters, user public/secret keys, and ciphertexts have size $\mathsf{poly}(\lambda, d, \log N)$, which are optimal up to the $\mathsf{poly}(d)$ factor. This is the first registered ABE scheme with nearly-optimal parameters. All previous schemes (including constructions based on indistinguishability obfuscation, witness encryption, or evasive LWE) either have ciphertexts that scale with the policy size and attribute length, or can only support a bounded number of users (with long public parameters and public keys that scale with the number of users).

1 Introduction

In recent years, registration-based cryptography [GHMR18] has emerged as a popular paradigm for realizing advanced cryptographic primitives without a trusted authority. In this work, we study two trustless cryptographic primitives: distributed broadcast encryption and registered attribute-based encryption (registered ABE):

- **Distributed broadcast encryption:** In broadcast encryption [FN93], a sender can encrypt a message to an arbitrary set of recipients with a ciphertext whose size scales sublinearly with the number of recipients. Traditionally, broadcast encryption requires a central authority who issues decryption keys to each recipient in the system. Distributed broadcast encryption [WQZD10, BZ14] removes the central authority. Instead, each receiver independently generates their own public/secret key and publishes it in a public-key directory. Thereafter, a sender can encrypt a message to an arbitrary set of public keys with a ciphertext whose size scales sublinearly with the number of recipients.
- **Registered ABE:** Attribute-based encryption [SW05, GPSW06] is a generalization of public-key encryption that provides fine-grained access control to encrypted data. In key-policy ABE, decryption keys are associated with a policy f while ciphertexts are associated with an attribute x. Decryption recovers the associated message m only if the decryption policy is satisfied (e.g., if $f(x) = 0$). Like the case with broadcast encryption, the standard notion of an ABE scheme assumes the existence of a central trusted authority that issues keys to users. Registered ABE [HLWW23] removes this trusted authority. In this model, each user independently generates their own public/secret key and publishes it to a public-key directory. Then, there is a process that aggregates all of the individual public keys (with their associated decryption policies) into a succinct master public key

mpk. The aggregated master public key functions as the public key for an ABE scheme (i.e., one can encrypt a message with respect to any attribute x with the guarantee that only registered users associated with an accepting policy f can decrypt).

In both settings, the goal is to support the capabilities provided by broadcast encryption or attribute-based encryption without needing to assume the existence of a trusted authority. In the traditional centralized versions of these primitives, there is a central point of failure. If the adversary compromises the central authority, they immediately compromise the security of every user in the system. Registration-based cryptography ensures that individual users retain control of their secret keys and that trust is not concentrated in any single entity.

Bounded vs. unbounded. In the last few years, there has been a flurry of activity in constructing distributed broadcast encryption [WQZD10, BZ14, KMW23, FWW23, GKPW24, CW24, CHW25] and registered ABE [HLWW23, FWW23, ZZGQ23, AT24, GLWW24, CHW25, ZZC+25] from different cryptographic assumptions. Thus far, these works can be categorized into two main categories:

- **Unbounded constructions:** The first class of constructions are those that support an arbitrary polynomial of users. Namely, these are schemes where any number of users can join the system by posting their public key to the public-key directory. This is the most natural formulation of these primitives (and an implicit requirement in the original work introducing registration-based cryptography [GHMR18]). Constructing schemes that support an arbitrary number of users has thus far relied on strong tools such as indistinguishability obfuscation [BZ14, HLWW23], witness encryption for NP [FWW23], or strong non-falsifiable assumptions [ZZC+25].
- **Bounded constructions:** A second line of work has focused on direct algebraic constructions of trustless cryptographic primitives that do not need general-purpose tools like indistinguishability obfuscation or witness encryption. These constructions often have the advantage of being simpler, concretely-efficient, and in the case of lattice-based schemes, plausibly post-quantum secure. However, with the exception of [ZZC+25], which is based on private-coin evasive LWE, all of the constructions in this family [WQZD10, HLWW23, KMW23, ZZGQ23, AT24, GLWW24, CW24, GKPW24, CHW25] make a significant compromise by assuming there is an a priori *maximum* number of users that the scheme can support. Moreover, in these schemes, the parameters as well as the size of each user's public key grows linearly (or worse) with the maximum number of users. For schemes with large number of users, the size of these parameters introduces significant overhead.

A major challenge in the study of trustless cryptography is developing new techniques that support an arbitrary number of users from simple (and falsifiable) assumptions (and without relying on heavy machinery such as program obfuscation or witness encryption).

1.1 Our Results

In this work, we develop new techniques for constructing distributed broadcast encryption and (key-policy) registered ABE for general policies that support an *a priori* unbounded number of users. Security of both constructions relies on the succinct LWE assumption [Wee24], and our registered ABE scheme additionally relies on the random oracle model. The succinct LWE assumption is a simple and falsifiable lattice assumption (which is also implied by public-coin evasive LWE). We now provide a brief comparison of our results with those from prior work (see also Tables 1 and 2).

Distributed broadcast encryption. Our first result is a distributed broadcast encryption scheme that supports an arbitrary polynomial number of users from the $\mathsf{poly}(\lambda, \log N)$-succinct LWE assumption, where λ is the security parameter and N is the total number of users. The size of the public parameters, the user public/secret keys, as well as the ciphertext in our scheme are all $\mathsf{poly}(\lambda, \log N)$. Previously, this was only known from indistinguishability obfuscation [BZ14] or from plain witness encryption [FWW23]. Compared to these approaches, our algebraic approach is conceptually simpler. For instance, the [FWW23] construction relies on witness encryption together with a function-binding hash function; this latter primitive in turn relies on fully homomorphic encryption. In contrast, our approach for distributed broadcast encryption does not need any kind of homomorphic computation machinery.

Scheme	Assumption	$\lvert\mathsf{pp}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	TP	PQ	AD
[BZ14]	$i\mathcal{O}$ + one-way function	–	1	1	1	✓	✗	✓[*]
[FWW23]	witness encryption + LWE	1	1	1	1	✓	✓	✓[*]
[WQZD10]	bilinear Diffie-Hellman exponent	N	N^2	N	1	✓	✗	✗
[KMW23]	bilinear Diffie-Hellman exponent	N	N	1	1	✗	✗	✓[*]
[GKPW24]	generic bilinear group	N	N	1	1	✗	✗	✗
[CW24]	$\mathsf{poly}(\lambda, N)$-succinct LWE	N^2	N	1	1	✗	✓	✗
[CHW25]	$\mathsf{poly}(\lambda, N)$-succinct LWE[†]	N^2	N	1	1	✗	✓	✓
This work	$\mathsf{poly}(\lambda, \log N)$-succinct LWE	1	1	1	1	✗[‡]	✓	✗

[*] These schemes were originally shown to satisfy *semi-static* security. Adaptive security can then be obtained by using the Gentry-Waters compiler [GW09] in the random oracle model or the more recent Hsieh-Waters-Wu compiler [HWW25], which gives adaptive security in the plain model.

[†] Security of this scheme additionally relies on the random oracle model.

[‡] We can obtain a variant of this construction with transparent setup by using the recently-introduced decomposed LWE assumption [AMR25]. We refer to Remark 4.13 for more details.

Table 1: Comparison with previous distributed broadcast encryption schemes. For each scheme, we report the size of the public parameters pp, the user public key pk, the user secret key sk, and the ciphertext ct as a function of the number of users N. For ease of comparison, we suppress $\mathsf{poly}(\lambda, \log N)$ factors, where λ is the security parameter. For each scheme, we also indicate whether the public parameters pp (if present) can be sampled with a *transparent* setup (TP), whether it is plausibly post-quantum secure (PQ), and whether it is proven to be adaptively secure (AD). We write $i\mathcal{O}$ to denote indistinguishability obfuscation [BGI$^+$01].

Compared to direct algebraic constructions based on bilinear maps [WQZD10, KMW23, GKPW24] or succinct LWE [CW24, CHW25], our scheme is the only one that supports an unbounded number of users. In previous schemes, both the size of the public parameters and the size of an individual user's public key scale with the number of users N. As we discuss more in Section 2.1, all of these schemes were bounded because each user's public key needed to include a "cross-term" for every other user in the system. A key technical contribution of this work is a new technique for efficiently deriving the cross-terms from a succinct commitment. This enables algebraic schemes that support an unbounded number of users. We provide a more detailed comparison in Table 1.

Key-policy registered ABE. Our second construction is a key-policy registered ABE scheme for general policies (modeled as bounded-depth Boolean circuits) from the succinct LWE assumption in the random oracle model. Our scheme supports an arbitrary number of users and moreover, has succinct ciphertexts (of size $\mathsf{poly}(\lambda, d, \log N)$, where d is the depth of the circuit and N is the number of users). Notably, the size of the ciphertext does not scale with the size of the policy circuit or with the length of the attribute. Prior to this work, the only scheme with this level of ciphertext succinctness is the scheme from [CHW25] from succinct LWE (in the random oracle model), but that scheme only supports a bounded number of users (and requires long public parameters and user public keys). The other constructions from obfuscation [HLWW23], witness encryption [FWW23], or private-coin evasive LWE [ZZC$^+$25] all have ciphertexts whose size scales linearly with the attribute length. Thus, we obtain the first unbounded registered ABE scheme with nearly-optimal ciphertext size (i.e., optimal up to the $\mathsf{poly}(d)$ factor). We provide a more detailed comparison in Table 2.

Concurrent work. In a concurrent and independent work, Abram, Malavolta, and Roy [AMR25] introduced the decomposed LWE assumption, a weaker variant of the succinct LWE assumption, and show (among other things) how to obtain a registered ABE scheme that supports general circuit policies in the plain model. We provide a brief comparison of our two schemes:

- **Bounded vs. unbounded:** The [AMR25] registered ABE scheme relies on a structured reference string that contains N secret keys for a (centralized) ABE scheme, where N is a bound on the number of parties. As a result,

Scheme	**Assumption**	$\lvert\mathsf{pp}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	**TP**	**PQ**	**AD**
[HLWW23]	$i\mathcal{O}$ + SSB hash function	1	1	1	$\mathsf{poly}(\lvert C\rvert, \lvert\mathbf{x}\rvert)$	✓	✗	✓
[FWW23]	witness encryption + LWE	1	1	1	$\mathsf{poly}(\lvert C\rvert, \lvert\mathbf{x}\rvert)$	✓	✓	✗
[ZZC+25]	private-coin evasive LWE	$\mathsf{poly}(d)$	$\mathsf{poly}(d)$	$\mathsf{poly}(d)$	$\mathsf{poly}(d, \lvert\mathbf{x}\rvert)$	✓	✓	✗
[CHW25]	$\mathsf{poly}(\lambda, d, N)$-succinct LWE*	$N^2 \cdot \mathsf{poly}(d)$	$N \cdot \mathsf{poly}(d)$	$\mathsf{poly}(d)$	$\mathsf{poly}(d)$	✗	✓	✗
This work	$\mathsf{poly}(\lambda, d, \log N)$-succinct LWE*	$\mathsf{poly}(d)$	$\mathsf{poly}(d)$	$\mathsf{poly}(d)$	$\mathsf{poly}(d)$	✗‡	✓	✗

*Security of these schemes are in the random oracle model.

‡We can obtain a variant of this construction with transparent setup by using the recently-introduced decomposed LWE assumption [AMR25]. We refer to Remark 5.31 for more details.

Table 2: Comparison with previous (key-policy) registered ABE schemes that support general circuit policies. For each scheme, we report the size of the public parameters pp, the user public key pk, the user secret key sk, and the ciphertext ct as a function of the number of users N, the attribute $\mathbf{x}$, and the policy circuit C. We write d to denote the depth of the circuit C. We assume the decryption algorithm is provided the description of the policy circuit C as well as the attribute $\lvert\mathbf{x}\rvert$ as input, so these do not necessarily have to be encoded as part of the secret key or the ciphertext. For ease of comparison, we suppress $\mathsf{poly}(\lambda, \log N)$ factors, where λ is the security parameter. For each scheme, we also indicate whether the public parameters pp can be sampled with a *transparent* setup (TP), whether it is plausibly post-quantum secure (PQ), and whether it is proven to be adaptively secure (AD). We write $i\mathcal{O}$ to denote indistinguishability obfuscation [BGI+01] and "SSB hash function" to refer to a somewhere-statistically-binding hash function [HW15].

their scheme only supports an a priori bounded number of parties. Our scheme requires a structured reference string whose size scales *logarithmically* with N, and thus, can support an arbitrary polynomial number of parties. Supporting an unbounded polynomial number of parties is the central goal of this work. As noted in Remark 5.31, we can also obtain a version of our scheme with a transparent setup (where the public parameters now consist of a uniform random string) by using the decomposed LWE assumption from [AMR25].

- **Security definition:** The work of [AMR25] analyze security of their scheme under a "very selective" model where the adversary has to choose its public keys before seeing the CRS and the honest parties' keys. In our setting, we consider the standard security definition for registered ABE where the adversary can register malicious keys of its choosing (*after* seeing the CRS and the honest parties' keys). The standard security model for registered ABE captures adversarial strategies such as rogue-key attacks (see [RY07, BDN18] for examples of rogue-key attacks in the context of aggregate signatures). Since the broader goal of registration-based cryptography is to allow users to choose their own keys, it seems unreasonable to remove this capability from the adversary. As discussed in more detail in [CHW25, §1.2], we are not aware of any generic techniques that lifts a registered ABE scheme secure in the very selective model to the standard security definition for registered ABE.

- **Hardness assumption:** Security of the [AMR25] construction relies on the decomposed LWE assumption in the plain model whereas our registered ABE scheme relies on the succinct LWE assumption in the random oracle model. The work [AMR25] show that the decomposed LWE assumption is a weaker assumption than succinct LWE. As we note in Remarks 4.13 and 5.31, we can also modify our scheme to obtain security based on the decomposed LWE assumption (and in fact, the resulting scheme would have the added benefit of having a transparent setup). In either case, security of our registered ABE scheme would still rely on the random oracle. Like [CHW25], we rely on the random oracle to handle the ability of the adversary to register malicious keys.

2 Technical Overview

In the following, let n, m, q be lattice parameters where $m = O(n \log q)$. We use curly underlines to suppress low-norm errors. Namely, we write $\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{A}}$ to denote $\mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}$, where $\mathbf{e}$ is a low-norm error vector. For a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a target vector $\mathbf{z} \in \mathbb{Z}_q^n$, we write $\mathbf{y} \longleftarrow \mathbf{B}^{-1}(\mathbf{z})$ to denote sampling $\mathbf{y} \in \mathbb{Z}_q^m$ from a discrete Gaussian distribution conditioned on $\mathbf{B}\mathbf{y} = \mathbf{z}$. We write $\mathbf{G}$ to denote the standard gadget matrix [MP12].

ℓ-succinct LWE. Our constructions rely on the succinct LWE assumption introduced by Wee [Wee24]. For a parameter ℓ, the ℓ-succinct LWE assumption asserts that the following distributions are computationally indistinguishable:

$$(\mathbf{B}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}}\,,\ \mathbf{W}\,,\ \mathbf{T}) \quad \text{and} \quad (\mathbf{B}\,,\ \mathbf{u}^{\mathsf{T}}\,,\ \mathbf{W}\,,\ \mathbf{T}),$$

where $\mathbf{B} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{\textsf{R}} \mathbb{Z}_q^n$, $\mathbf{W} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{\ell n\times m}$, $\mathbf{T}$ is a random Gaussian matrix where $[\mathbf{I}_\ell \otimes \mathbf{B} \mid \mathbf{W}]\mathbf{T} = \mathbf{I}_\ell \otimes \mathbf{G}$, and $\mathbf{I}_\ell$ is the identity matrix of dimension ℓ. Equivalently, ℓ-succinct LWE asserts that LWE is hard with respect to the (random) matrix $\mathbf{B}$ given a trapdoor for the related matrix $[\mathbf{I}_\ell \otimes \mathbf{B} \mid \mathbf{W}]$. In the following, we will often refer to $(\mathbf{B}, \mathbf{W}, \mathbf{T})$ as the public parameters for a succinct LWE instance of dimension ℓ.

Matrix commitment scheme. The key ingredient we use underlying our distributed broadcast encryption and registered ABE schemes is the recent matrix commitment scheme by Wee [Wee25]. Specifically, the work of [Wee25] shows that given the public parameters $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ for a succinct LWE instance of dimension $2m^2$ (i.e., $\mathbf{W} \in \mathbb{Z}_q^{2m^2 n\times m}$ and $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]\mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$) and any matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times N}$, there is an efficient and deterministic algorithm to compute a commitment $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ and a low-norm opening $\mathbf{Z}^{m\times N}$ such that

$$\mathbf{C} \cdot \mathbf{V}_N = \mathbf{M} - \mathbf{B} \cdot \mathbf{Z} \in \mathbb{Z}_q^{n\times N}, \tag{2.1}$$

where $\mathbf{V}_N \in \mathbb{Z}_q^{m\times N}$ is a fixed low-norm verification matrix that is publicly derived from $\mathsf{pp}_{\mathsf{com}}$ and the width N. Wee used the matrix commitment scheme to construct key-policy and ciphertext-policy ABE schemes for bounded-depth Boolean circuits from the $\mathsf{poly}(\lambda, d)$-succinct LWE assumption, where d denotes the depth of the Boolean circuit.

2.1 Distributed Broadcast Encryption

We now show how to use matrix commitments to construct a distributed broadcast encryption scheme. In distributed broadcast encryption, each public key is associated with an index $i \in \mathbb{N}$, and one can encrypt to any set of public keys, provided that each key has a different index. When $N = 2^\lambda$, we can interpret each index as an identity (or a hash of an identity) and we say the scheme supports an arbitrary or unbounded polynomial number of users. The main technical challenges for constructing a distributed broadcast encryption that support an unbounded number of users are twofold:

- First, the size of all of the scheme components (i.e., the public parameters pp, the user public key pk, the user secret key sk, and the ciphertext ct) must all be bounded by $\mathsf{poly}(\lambda, \log N)$,
- Second, the running time of key-generation needs to be bounded by $\mathsf{poly}(\lambda, \log N)$, and that of encryption and decryption must be bounded by $|S| \cdot \mathsf{poly}(\lambda, \log N)$, where S is the broadcast set.

As a warm-up, we first describe a scheme with short parameters (i.e., a scheme that addresses the first challenge), but where the running time of key-generation time, encryption, and decryption is *slow* (i.e., scaling with $\mathsf{poly}(\lambda, N)$).

- **Public parameters:** The public parameters for the distributed broadcast encryption scheme $\mathsf{pp} = (\mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$ consists of the public parameters $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ for a succinct LWE instance of dimension $2m^2$, a matrix $\mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}$, and a vector $\mathbf{p} \xleftarrow{\textsf{R}} \mathbb{Z}_q^n$. The vector $\mathbf{p}$ can be viewed as a public key for a dual Regev encryption scheme and the matrix $\mathbf{A}$ is used to program the challenge set into the public parameters in the security analysis. Let $\mathbf{V}_N \in \mathbb{Z}_q^{m\times N}$ be the low-norm verification matrix associated with $\mathsf{pp}_{\mathsf{com}}$ and matrices with width N. Let $\mathbf{V}_N = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N]$ where $\mathbf{v}_i \in \mathbb{Z}_q^m$ is the i^{th} column of $\mathbf{V}_N$.
- **User key-generation:** To generate a public key for a slot $i \in [N]$, the user samples $\mathbf{r}_i \xleftarrow{\textsf{R}} \{0,1\}^m$. The public key is $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_i$ and the secret key is $\mathbf{r}_i$.
- **Encryption:** Let $S \subseteq [N]$ be a set of indices and let $\{(j, \mathbf{t}_j)\}_{j\in S}$ be a set of public keys. To encrypt a message $\mu \in \{0,1\}$ to this set of public keys, the encrypter proceeds as follows:

  - For each $j \in S$, let $\mathbf{C}_j \in \mathbb{Z}_q^{n \times m}$ be the commitment to $\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j$ where $\mathbf{u}_j \in \{0,1\}^N$ is the j^{th} standard basis vector. Let $\mathbf{Z}_j \in \mathbb{Z}_q^{m \times N}$ be the respective opening. By Eq. (2.1), this means

    $$\mathbf{C}_j \mathbf{V}_N = (\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j) - \mathbf{B}\mathbf{Z}_j.$$

    If we parse $\mathbf{Z}_j = [\mathbf{z}_{j,1} \mid \cdots \mid \mathbf{z}_{j,N}]$, then this means

    $$\mathbf{C}_j \mathbf{v}_j = \mathbf{t}_j - \mathbf{B}\mathbf{z}_{j,j} \quad \text{and} \quad \forall i \neq j : \mathbf{C}_j \mathbf{v}_i = -\mathbf{B}\mathbf{z}_{j,i}. \tag{2.2}$$

    We often refer to $\mathbf{z}_{j,i}$ for $j \neq i$ as a cross-term since it recodes from $\mathbf{B}$ to the product $\mathbf{C}_j \mathbf{v}_i$ of user i's public key $\mathbf{C}_i$ with user j's decryption component $\mathbf{v}_j$.

  - The encrypter now samples an LWE secret $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and outputs the ciphertext

    $$\mathsf{ct} = \left(\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}}\ ,\ \utilde{\mathbf{s}^{\mathsf{T}}\big(\mathbf{A} + \textstyle\sum_{j \in S} \mathbf{C}_j\big)}\ ,\ \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor} \right)$$

- **Decryption:** Take any user $i \in S$. By Eq. (2.2), we have

  $$\utilde{\mathbf{s}^{\mathsf{T}}\big(\mathbf{A} + \textstyle\sum_{j \in S} \mathbf{C}_j\big)} \cdot \mathbf{v}_i \approx \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{v}_i + \textstyle\sum_{j \in S} \mathbf{s}^{\mathsf{T}}\mathbf{C}_j\mathbf{v}_i = \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{v}_i + \mathbf{s}^{\mathsf{T}}\mathbf{t}_i - \textstyle\sum_{j \in S} \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{z}_{j,i}.$$

  Observe now that given the users' public keys $\{(j, \mathbf{t}_j)\}_{j \in S}$, the decrypter can compute $\mathbf{z}_{j,i}$ itself ($\mathbf{z}_{j,i}$ is the i^{th} column of the opening $\mathbf{Z}_j$ to the matrix $\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j$). Next, using the fact that $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_i$, and knowledge of the secret key $\mathbf{r}_i$, the decrypter computes

  $$\utilde{\mathbf{s}^{\mathsf{T}}\big(\mathbf{A} + \textstyle\sum_{j \in S} \mathbf{C}_j\big)} \cdot \mathbf{v}_i - \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}}\big(\mathbf{r}_i - \textstyle\sum_{j \in S} \mathbf{z}_{j,i}\big) \approx \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{v}_i + \mathbf{s}^{\mathsf{T}}(\mathbf{B}\mathbf{r}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_i) - \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{r}_i = \mathbf{s}^{\mathsf{T}}\mathbf{p}.$$

  Subtracting this from $\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p} + b \cdot \lfloor q/2 \rfloor}$ and rounding recovers the message μ.

By construction, the size of the public parameters pp, the users' public/secret keys $(\mathsf{pk}, \mathsf{sk})$, and the size of the ciphertext ct are $\mathsf{poly}(\lambda, \log N)$. However, the running time of key generation, encryption, and decryption is $\mathsf{poly}(\lambda, N)$ because they compute commitments and/or openings to matrices of width N. When N is super-polynomial, this means key-generation, encryption, and decryption are no longer efficient algorithms. In order to support an arbitrary number of users (or alternatively, an identity-based distributed broadcast encryption scheme) with $N = 2^{\lambda}$, we need to reduce the running times of these algorithms to $\mathsf{poly}(\lambda, \log N)$. We discuss this below (after sketching the security analysis).

Arguing security. We consider selective security where the adversary declares the set $S \subseteq [N]$ of challenge indices at the beginning of the security game. To prove security from succinct LWE, we need to show how to simulate the challenge ciphertext given $(\mathbf{B}, \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}}, \mathbf{W}, \mathbf{T})$. We do so by programming the matrix $\mathbf{A}$ and the public keys pk_i for the users $i \in S$. Here, we crucially exploit the fact that the challenge ciphertext does not depend on the public keys outside S (which are chosen adversarially). As an aside, we note that unlike standard broadcast encryption, we do not need to explicitly simulate secret keys for users outside S, since malicious users outside the set S can register any (possibly malformed) public key. The reduction has the following high-level procedure:

- For all $j \in S$, the challenger samples the public key as $\mathbf{t}_i \xleftarrow{\text{R}} \mathbb{Z}_q^n$. By the leftover hash lemma, the honestly-generated public keys $\mathbf{t}_j = \mathbf{B}\mathbf{r}_j + \mathbf{p} - \mathbf{A}\mathbf{v}_j$ are statistically indistinguishable from a uniform random vector $\mathbf{t}_j \xleftarrow{\text{R}} \mathbb{Z}_q^n$. Importantly, the public keys are now independent of $\mathbf{A}$, and thus, the challenger can now sample the public keys $\mathbf{t}_j$ *before* it samples the matrix $\mathbf{A}$.

- When setting the public parameters, the challenger embeds the challenge set into the public parameters. Namely, after sampling the public keys $\mathbf{t}_j \xleftarrow{\text{R}} \mathbb{Z}_q^n$ for the honest users $j \in S$, the challenger now sets $\mathbf{A} = \mathbf{B}\mathbf{R}_{\mathsf{A}} - \sum_{j \in S} \mathbf{C}_j$, where $\mathbf{R}_{\mathsf{A}} \xleftarrow{\text{R}} \{0,1\}^{m \times m}$ and $\mathbf{C}_j$ is a commitment to the vector $\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j$. It also sets $\mathbf{p} = \mathbf{B}\mathbf{r}_{\mathsf{p}}$ where $\mathbf{r}_{\mathsf{p}} \xleftarrow{\text{R}} \{0,1\}^m$. Again by the leftover hash lemma, the distribution of $\mathbf{A}$ and $\mathbf{p}$ are statistically close to uniform, which coincides with their distribution in the real scheme.

- With the above modifications, the challenge ciphertext can now be written as

$$\mathsf{ct} = \left(\utilde{\mathbf{s}^\mathsf{T}\mathbf{B}}\,,\ \utilde{\mathbf{s}^\mathsf{T}(\mathbf{A} + \textstyle\sum_{j \in S} \mathbf{C}_j)}\,,\ \utilde{\mathbf{s}^\mathsf{T}\mathbf{p} + \mu \cdot \lfloor q/2 \rceil}\right) \approx \left(\utilde{\mathbf{s}^\mathsf{T}\mathbf{B}}\,,\ \utilde{\mathbf{s}^\mathsf{T}\mathbf{B}} \cdot \mathbf{R}_\mathbf{A}\,,\ \utilde{\mathbf{s}^\mathsf{T}\mathbf{B}} \cdot \mathbf{r}_\mathbf{p} + \mu \cdot \lfloor q/2 \rceil\right),$$

  which can be simulated from $\utilde{\mathbf{s}^\mathsf{T}\mathbf{B}}$. By the $2m^2$-succinct LWE assumption, $\utilde{\mathbf{s}^\mathsf{T}\mathbf{B}}$ is pseudorandom given $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. This implies the ciphertext is pseudorandom and security holds.

We refer to Section 4 for the formal description of the construction and security analysis.

Local openings. To support $N = 2^\lambda$ (i.e., an arbitrary polynomial number of users), we show how to implement the key-generation, encryption, and decryption algorithms in $\mathsf{poly}(\lambda, \log N)$ time. The modification is purely algorithmic and the modifications have no effect on the correctness or security analysis. Our approach relies on two key observations:

- The above scheme only commits to *sparse* matrices (of the form $\mathbf{u}_i^\mathsf{T} \otimes \mathbf{t}_i$). While this matrix has N columns, only a single column is non-zero.
- The key-generation, encryption, and decryption algorithms only needs local access to the verification matrix $\mathbf{V}_N$ and the openings $\mathbf{Z}_j$. In fact, each algorithm only reads a *single* column of the verification matrix $\mathbf{V}_L$ or the opening matrix $\mathbf{Z}_j$.

In Appendix A, we show that the [Wee25] matrix commitment scheme satisfies the following two properties:

- The commitment $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ to a sparse matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times N}$ with K non-zero columns can be computed in time $\mathsf{poly}(m, \log q, \log N, K)$.
- There is an algorithm running in time $\mathsf{poly}(m, \log q, \log N)$ for computing the i^{th} column of the verification matrix $\mathbf{V}_N$ and an algorithm running in time $\mathsf{poly}(m, \log q, \log N, K)$ for computing the i^{th} column of the opening $\mathbf{Z}_j$ to a matrix $\mathbf{M}$ with K non-zero columns.

In some sense, the matrix commitment scheme from [Wee25] has a Merkle-tree-like structure where a commitment to a matrix $\mathbf{M} = [\mathbf{M}_\mathrm{L} \mid \mathbf{M}_\mathrm{R}]$ is derived by first committing to its left half $\mathbf{M}_\mathrm{L}$ and its right half $\mathbf{M}_\mathrm{R}$, and finally committing to the resulting commitments $[\mathbf{C}_\mathrm{L} \mid \mathbf{C}_\mathrm{R}]$. We can efficiently commit and provide local openings for (exponentially-long) sparse vectors with a Merkle tree, and the same is true for the matrix commitment scheme of [Wee25]. This yields a distributed broadcast encryption scheme that supports $N = 2^\lambda$ users.

Comparison with [CW24]. The structure of our distributed broadcast encryption scheme shares some similarities with that from [CW24]. In our notation, the public keys in [CW24] consists of a (random) matrix $\mathbf{C}_i$ together with a collection of low-norm cross-terms $\mathbf{z}_{i,j}$ where $\mathbf{B}\mathbf{z}_{i,j} = -\mathbf{C}_i\mathbf{v}_j$. The secret key is a low-norm vector $\mathbf{z}_{i,i}$ where $\mathbf{B}\mathbf{z}_{i,i} = \mathbf{p} + \mathbf{A}\mathbf{v}_i - \mathbf{C}_i\mathbf{v}_i$. The public parameters include the matrices $\mathbf{A}, \mathbf{B}$, the target vector $\mathbf{p}$, the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_N$, along with a (sufficiently-large) succinct LWE trapdoor that is used to sample public keys $\mathbf{C}_i$, cross-terms $\mathbf{z}_{i,j}$, and secret keys $\mathbf{z}_{i,i}$. From a structural perspective, the public/secret keys between our scheme and the [CW24] scheme are very similar (and likewise for the ciphertexts). The key difference is the following:

- In [CW24], the key-generation algorithm jointly *samples* the public-key matrix $\mathbf{C}_i$ together with the cross-terms $\mathbf{z}_{i,j}$. This leads to a scheme with $O(N)$-size public keys. Notably, there is no compact description of the cross-terms needed for decryption (i.e., the terms $\mathbf{z}_{j,i}$ that recode from $\mathbf{B}$ to $\mathbf{C}_j\mathbf{v}_i$). Note that publishing the randomness used to sample $\mathbf{C}_i$ and $\mathbf{z}_{i,j}$ for $j \neq i$ is not sufficient since the randomness would also leak the secret key $\mathbf{z}_{i,i}$.
- In our scheme, the public key is simply a vector $\mathbf{t}_i$, and the associated public-key matrix $\mathbf{C}_i$ and the cross-terms $\mathbf{z}_{i,j}$ are all *deterministically* derived from $\mathbf{t}_i$ using the matrix commitment scheme. The secret key is the randomness used to sample $\mathbf{t}_i$. Because the matrix $\mathbf{C}_i$ and the cross-terms have a *compact* description, our scheme supports an unbounded number of users. A key technical contribution of this work is showing that the structure of the [Wee25] matrix commitments enables us to compress cross-terms (or alternatively, derive cross-terms from a public procedure).

We note here that the need to include a cross-term for every other user is a standard feature in nearly all constructions of distributed broadcast encryption and registered ABE that do not go through obfuscation or witness encryptions [WQZD10, HLWW23, KMW23, ZZGQ23, AT24, GLWW24, CW24, GKPW24, CHW25]. This is also the main reason these schemes cannot handle an arbitrary number of users. While we still rely on the same type of cross-term cancellation in this work, our use of matrix commitments enables a new compact description for these cross terms. This enables schemes that support an arbitrary number of users.

Comparison with [FWW23]. Our distributed broadcast encryption construction also shares some high-level similarities with the [FWW23] construction based on witness encryption. Both constructions embed a succinct commitment (alternatively, a hash) of the public keys of the users in the broadcast set S in the ciphertext, and moreover, in the security analysis, both reductions modify the distribution of the public keys for the users in S. In a bit more detail:

- In [FWW23], the commitment to the public keys is the instance used in the witness encryption ciphertext. In their setting, the commitment is a (function-binding) hash of the users' public keys. In our scheme, the commitment to the public keys is $\sum_{j \in S} \mathbf{C}_j$ and we embed it as an LWE sample $\utilde{\mathbf{s}^{\mathsf{T}}\big(\mathbf{A} + \sum_{j \in S} \mathbf{C}_j\big)}$ in the ciphertext. In both schemes, the commitment has a tree-like structure, and decryption relies on a local opening to the user's public key, which can be derived given just the public keys for the set S (along with the public parameters).

- In order to invoke semantic security of the witness encryption scheme in the [FWW23] security proof, they not only modify the distribution of the public keys in S, and also rely on the commitment satisfying a "function binding" property (which can be based on LWE). In contrast, our security proof is simpler and follows by directly programming the set S into the public parameters (e.g., setting $\mathbf{A} = -\sum_{j \in S} \mathbf{C}_j$, after modifying $\mathbf{t}_j$, and thus $\mathbf{C}_j$, to be independent of $\mathbf{A}$). This partitioning strategy (for arguing selective security) is a standard approach for analyzing the security of (distributed) broadcast encryption schemes (c.f., [BGW05, KMW23, CW24]). On the flip side, we note that the proof strategy in [FWW23] based on function-binding hash functions shows their scheme to satisfy *semi-static* security; this can in turn be lifted to full adaptive security via the [GW09, HWW25] transformations.

2.2 Key-Policy Registered ABE

We can combine our techniques for distributed broadcast encryption with ideas from the recent work of [CHW25] to obtain a key-policy registered ABE scheme with succinct ciphertexts in the random oracle model. In some sense, the work of [CHW25] starts with the distributed broadcast encryption scheme from [CW24] and shows how to extend it to a registered ABE scheme. In this work, we start from our new distributed broadcast encryption scheme (from Section 2.1) and show how to apply the [CHW25] techniques to lift the scheme to a registered ABE scheme. Since our underlying distributed broadcast encryption scheme supports an unbounded number of users, our registered ABE scheme also achieves this property. The previous construction of [CHW25] only supports a bounded number of users. Our techniques yield the first registered ABE scheme that can simultaneously support an arbitrary polynomial number of users and which has succinct ciphertexts (see Table 2). Such a scheme was not previously known even from witness encryption or indistinguishability obfuscation. Here, we provide a brief overview of how we augment our distributed broadcast encryption scheme to obtain a (key-policy) registered ABE scheme.

Lattice-based homomorphic evaluation. Our construction relies on the classic homomorphic evaluation machinery from [GSW13, BGG⁺14]. Specifically, given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, a Boolean function $f \colon \{0,1\}^\ell \to \{0,1\}$, and an input $\mathbf{x} \in \{0,1\}^\ell$, there exists a low-norm matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}}$ such that

$$(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{A},f,\mathbf{x}} = \mathbf{A}_f - f(\mathbf{x}) \cdot \mathbf{G}, \tag{2.3}$$

where $\mathbf{A}_f$ is a matrix that only depends on $\mathbf{A}$ and f.

Registered key-policy ABE. We now describe the general structure of our key-policy registered ABE scheme. Specifically, we describe a "slotted" registered ABE scheme where each key is associated with an index $i \in [N]$, and instead of users joining the system dynamically, there is instead an aggregation algorithm that takes as input a collection of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ along with their associated policies $f_1, \ldots, f_N$ and aggregates them together into a master public key mpk and a set of helper decryption keys $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_N$ for the N users. To decrypt, the user combines their secret key and their helper decryption key with the ciphertext. The work of [HLWW23] show that the slotted primitive generically implies standard the usual notion of registered ABE (that supports dynamic registrations) with only $\mathsf{poly}(\lambda, \log N)$ overhead. In this work, we focus exclusively on the simpler slotted primitive.

For ease of exposition, we describe our construction with long ciphertexts (that scale with the attribute length). We can then apply the ciphertext compression approach from [Wee24, Wee25] to obtain a registered ABE scheme with succinct ciphertexts. Essentially, instead of encrypting with respect to $\mathbf{x}^{\mathsf{T}} \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times \ell m}$ (as in the classic [BGG+14] ABE scheme), we instead encrypt to a [Wee25] commitment $\mathbf{C}_{\mathbf{x}} \in \mathbb{Z}_q^{n \times m}$ to $\mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}$. We start with a basic version of the scheme:

- **Public parameters:** The public parameters $\mathsf{pp} = (\mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p}, \{\mathbf{d}_i\}_{i \in [N]})$ have essentially the same structure as that for our distributed broadcast encryption scheme. Here $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ are the public parameters for a succinct LWE instance of dimension $2m^2$, $\mathbf{A} \xleftarrow{\textsc{r}} \mathbb{Z}_q^{n \times \ell m}$ is the matrix used to embed the attribute, $\mathbf{p} \xleftarrow{\textsc{r}} \mathbb{Z}_q^m$ is a dual Regev public key (for encoding the message), and $\mathbf{d}_1, \ldots, \mathbf{d}_N \xleftarrow{\textsc{r}} \mathbb{Z}_q^n$ are vectors used for noise smudging (in the security analysis). As described, the size of the CRS scales linearly with N, but since the $\mathbf{d}_i$ vectors are uniformly random, we can compress them by working in the random oracle model and setting $\mathbf{d}_i = H_1(i)$, where H_1 is modeled as a random oracle.

- **User key-generation:** To generate a public key for a slot $i \in [N]$ and a function f, the user samples $\mathbf{r}_i \xleftarrow{\textsc{r}} \{0,1\}^m$ and sets the secret key to be $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) \in \mathbb{Z}_q^n$. The secret key is the randomness $\mathbf{r}_i$.

- **Key aggregation:** Given a collection of public keys $\mathbf{t}_1, \ldots, \mathbf{t}_N$ for functions $f_1, \ldots, f_N$, the aggregation algorithm computes commitments $\mathbf{C}_i \in \mathbb{Z}_q^{n \times m}$ and openings $\mathbf{Z}_i \in \mathbb{Z}_q^{m \times L}$ to $\mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i$ where $\mathbf{u}_i \in \{0,1\}^N$ is the i^{th} unit vector. It parses $\mathbf{Z}_i = [\mathbf{z}_{i,1} \mid \cdots \mid \mathbf{z}_{i,N}]$. The master public key mpk and helper decryption key hsk_i for user i are defined to be
$$\mathsf{mpk} = \widehat{\mathbf{C}} = \sum_{j \in [N]} \mathbf{C}_j \quad \text{and} \quad \mathsf{hsk}_i = \hat{\mathbf{z}}_i = \sum_{j \in [N]} \mathbf{z}_{j,i}. \tag{2.4}$$

- **Encryption:** To encrypt a message $\mu \in \{0,1\}$ with attribute $\mathbf{x} \in \{0,1\}^\ell$, the encryption algorithm samples $\mathbf{s} \xleftarrow{\textsc{r}} \mathbb{Z}_q^n$ and outputs the ciphertext
$$\mathsf{ct} = \Big(\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}}}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p} + \lfloor q/2 \rfloor \cdot \mu} \Big). \tag{2.5}$$

- **Decryption:** To decrypt using a secret key $\mathbf{r}_i$ for slot i and an associated function f_i where $f_i(\mathbf{x}) = 0$, the decrypter first computes
$$\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathbf{A},f,\mathbf{x}} \approx \mathbf{s}^{\mathsf{T}}(\mathbf{A}_f - f(\mathbf{x}) \cdot \mathbf{G}) = \mathbf{s}^{\mathsf{T}}\mathbf{A}_f$$
by Eq. (2.3). Let $\mathbf{V}_N \in \mathbb{Z}_q^{m \times N}$ be the low-norm verification matrix associated with $\mathsf{pp}_{\mathsf{com}}$ and matrices of width N. Write $\mathbf{V} = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N]$. Then, using the fact that $\mathbf{C}_i$ is a commitment to $\mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i$ and $\mathbf{Z}_i$ is the associated opening, we appeal to Eqs. (2.1) and (2.4) to write
$$\utilde{\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}}} \cdot \mathbf{v}_i \approx \mathbf{s}^{\mathsf{T}} \sum_{j \in [N]} \mathbf{C}_j \mathbf{v}_i = \mathbf{s}^{\mathsf{T}}\mathbf{t}_i - \mathbf{s}^{\mathsf{T}} \sum_{j \in [N]} \mathbf{B}\mathbf{z}_{j,i} = \mathbf{s}^{\mathsf{T}}\mathbf{t}_i - \mathbf{s}^{\mathsf{T}}\mathbf{B}\hat{\mathbf{z}}.$$
Finally, using the fact that $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i)$, the decrypter can use its secret key $\mathbf{r}_i$ to compute
$$\begin{aligned} &\utilde{\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}}} \cdot \mathbf{v}_i - \utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathbf{A},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{d}_i) - \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}} \cdot (\mathbf{r}_i - \hat{\mathbf{z}}) \\ &\quad \approx \mathbf{s}^{\mathsf{T}}(\mathbf{B}\mathbf{r}_i + \mathbf{p} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i)) - \mathbf{s}^{\mathsf{T}}\mathbf{B}\hat{\mathbf{z}} - \mathbf{s}^{\mathsf{T}}\mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{r}_i + \mathbf{s}^{\mathsf{T}}\mathbf{B}\hat{\mathbf{z}} \\ &\quad = \mathbf{s}^{\mathsf{T}}\mathbf{p}. \end{aligned}$$
Taking the difference with $\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p} + \lfloor q/2 \rfloor \cdot \mu}$ and rounding now recovers the message μ.

Arguing security. We consider attribute-selective security where the adversary commits to the challenge attribute $\mathbf{x}$ at the beginning of the security game. To prove security from the succinct LWE assumption, we again need to show how to simulate the challenge ciphertext (Eq. (2.5)) from $(\mathbf{B}, \utilde{\mathbf{s}^\top \mathbf{B}}, \mathbf{W}, \mathbf{T})$. Since the adversary commits to the challenge attribute $\mathbf{x}$, we can use the strategy from [BGG+14] and program $\mathbf{x}$ into the matrix $\mathbf{A}$ (e.g., by setting $\mathbf{A} = \mathbf{B}\mathbf{R}_{\mathbf{A}} + \mathbf{x}^\top \otimes \mathbf{G}$). Now the reduction can simulate the attribute-embedding component in the challenge ciphertext as

$$\utilde{\mathbf{s}^\top(\mathbf{A} - \mathbf{x}^\top \otimes \mathbf{G})} \approx \mathbf{s}^\top \mathbf{B}\mathbf{R}_{\mathbf{A}} \approx \utilde{\mathbf{s}^\top \mathbf{B}} \cdot \mathbf{R}_{\mathbf{A}}.$$

Similarly, the challenger sets $\mathbf{p} = \mathbf{A}\mathbf{r}_{\mathbf{p}}$ for a low-norm $\mathbf{p}$. Then, it can simulate the message-embedding component of the challenger ciphertext as

$$\utilde{\mathbf{s}^\top \mathbf{p} + \lfloor q/2 \rfloor \cdot \mu} \approx \mathbf{s}^\top \mathbf{B}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot \mu \approx \utilde{\mathbf{s}^\top \mathbf{B}} \cdot \mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot \mu.$$

The difficult term is simulating $\utilde{\mathbf{s}^\top \widehat{\mathbf{C}}}$. Here, $\widehat{\mathbf{C}} = \sum_{i \in [N]} \mathbf{C}_i$, where $\mathbf{C}_i$ is the commitment to the public key of user i. In registered ABE, these keys can be chosen *maliciously*. This is a major distinction between registered ABE and distributed broadcast encryption. In distributed broadcast encryption, the keys that influence the challenge ciphertext are honestly generated, so the reduction algorithm can program them so as to be able to simulate the challenge ciphertext. This is not the case in registered ABE. We solve this problem by using the ciphertext re-randomization technique from [CHW25]. The idea is to start with the following *randomized* aggregation algorithm:

- The aggregation algorithm samples $\mathbf{C}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ together with low-norm vectors $\mathbf{z}_{0,i} \in \mathbb{Z}_q^m$ where $\mathbf{C}_0 \mathbf{v}_i = -\mathbf{B}\mathbf{z}_{0,i}$ for all $i \in [N]$. We can view $\mathbf{C}$ as a *random* [Wee25] commitment to the all-zeroes matrix $\mathbf{0}^{n \times N}$ with opening $\mathbf{Z}_0 = [\mathbf{z}_{0,1} \mid \cdots \mid \mathbf{z}_{0,N}]$. We show that there is an efficient algorithm to sample $\mathbf{C}_0$ together with $\mathbf{Z}_0$ using the succinct LWE trapdoor (see Section 5.1 and Theorem 5.9).
- The aggregated master public key is then $\mathsf{mpk} = \widehat{\mathbf{C}} = \mathbf{C}_0 + \sum_{j \in [N} \mathbf{C}_j$ and the corresponding helper decryption key for user i is $\mathsf{hsk}_i = \hat{\mathbf{z}}_i = \mathbf{z}_{0,i} + \sum_{j \in [N]} \mathbf{z}_{j,i}$.

It is not hard to see that this modification still preserves correctness of the scheme. Now, in the security proof, the reduction algorithm will set $\mathbf{C}_0 = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} - \sum_{j \in [N]} \mathbf{C}_j$, where $\mathbf{R}_{\widehat{\mathbf{C}}}$ is a low-norm matrix sampled by the reduction. This allows the reduction to simulate the challenge ciphertext component as

$$\utilde{\mathbf{s}^\top \widehat{\mathbf{C}}} \approx \mathbf{s}^\top \left(\mathbf{C}_0 + \sum_{j \in [N]} \mathbf{C}_j \right) = \mathbf{s}^\top \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} \approx \utilde{\mathbf{s}^\top \mathbf{B}} \cdot \mathbf{R}_{\widehat{\mathbf{C}}}.$$

Of course, there is still the caveat that the reduction algorithm now needs to simulate the low-norm vectors $\mathbf{z}_{0,i}$ where $\mathbf{C}_0 \mathbf{v}_i = -\mathbf{B}\mathbf{z}_{0,i}$. To do so, the reduction algorithm will first set $\mathbf{d}_i = \mathbf{B}\mathbf{r}_{\mathbf{d}_i}$ for a low-norm $\mathbf{r}_{\mathbf{d}_i}$ known to the reduction. Now, we consider two possibilities depending on whether a key for a slot i was honestly-generated (by the reduction algorithm) or chosen maliciously by the adversary:

- Suppose a slot i is associated with an honestly-generated key. Using again the fact that for all $j \neq i$, $\mathbf{C}_j \mathbf{v}_i = -\mathbf{B}\mathbf{z}_{j,i}$ and $\mathbf{C}_i \mathbf{v}_i = \mathbf{t}_i - \mathbf{B}\mathbf{z}_{i,i}$, we have

  $$\mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} \mathbf{v}_i + \textstyle\sum_{i \in [N]} \mathbf{B}\mathbf{z}_{j,i} = (\mathbf{C}_0 + \sum_{j \in [N]} \mathbf{C}_j)\mathbf{v}_i + \sum_{j \in [N]} \mathbf{B}\mathbf{z}_{j,i} = \mathbf{C}_0 \mathbf{v}_i + \mathbf{t}_i. \tag{2.6}$$

  When generating the keys for the honest users, the reduction algorithm programs the public key to be $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{d}_i = \mathbf{B}\mathbf{r}_i + \mathbf{B}\mathbf{r}_{\mathbf{d}_i}$. Combined with Eq. (2.6), this means

  $$\mathbf{B} \cdot \underbrace{\left(\mathbf{r}_{\mathbf{d}_i} + \mathbf{r}_i - \mathbf{R}_{\widehat{\mathbf{C}}} \mathbf{v}_i - \textstyle\sum_{j \in [N]} \mathbf{z}_{j,i}\right)}_{\mathbf{z}_{0,i}} = \mathbf{t}_i - \mathbf{C}_0 \mathbf{v}_i - \mathbf{t}_i = -\mathbf{C}_0 \mathbf{v}_i.$$

- Suppose a slot i is associated with an adversarially-chosen key $\mathbf{t}_i$. In this case, the associated policy f_i is not satisfied by the challenge attribute $\mathbf{x}$ (i.e., $f(\mathbf{x}) = 1$). To construct $\mathbf{z}_{0,i}$ in this case, the reduction algorithm will need to know a short $\mathbf{r}_i$ such that $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} + \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i)$. As in [CHW25], we facilitate this by requiring each public key include a non-interactive zero-knowledge (NIZK) proof of knowledge of the associated secret key $\mathbf{r}_i$. The reduction algorithm in turn extracts the associated secret key from each public key chosen by the adversary. In this case, since $f(\mathbf{x}) = 1$,

$$\mathbf{B}\mathbf{R}_{\mathbf{A}} \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) = (\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) = (\mathbf{A}_{f_i} - \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{d}_j) = \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{B}\mathbf{r}_{\mathbf{d}_i}.$$

This means

$$\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} + \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i) = \mathbf{B}\mathbf{r}_i + \mathbf{B}\mathbf{r}_{\mathbf{p}} + \mathbf{B}\mathbf{R}_{\mathbf{A}}\mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{B}\mathbf{r}_{\mathbf{d}_i}.$$

By Eq. (2.6), we have

$$\mathbf{B} \cdot \underbrace{\left(\mathbf{r}_{\mathbf{d}_i} + \mathbf{r}_i + \mathbf{r}_{\mathbf{p}} + \mathbf{R}_{\mathbf{A}}\mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \textstyle\sum_{j \in [N]} \mathbf{z}_{j,i}\right)}_{\mathbf{z}_{0,i}} = \mathbf{t}_i - \mathbf{C}_0\mathbf{v}_i - \mathbf{t}_i = -\mathbf{C}_0\mathbf{v}_i$$

In both cases, the reduction algorithm is able to construct a low-norm $\mathbf{z}_{0,i}$ such that $\mathbf{B}\mathbf{z}_{0,i} = -\mathbf{C}_0\mathbf{v}_i$. Thus, using this procedure, the reduction algorithm can program $\mathbf{C}_0 = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} - \sum_{j \in [N]} \mathbf{C}_j$ and simulate the challenge ciphertext. There are two remaining issues we need to address:

- First, in registered ABE, we require aggregation to be deterministic. Following [CHW25], we can derandomize the aggregation algorithm by having the real scheme sample $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$ using the random oracle (by hashing the input to the aggregation algorithm). To implement this strategy, we require an "explainable" sampling procedure for sampling $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$. Namely, given any (correctly-distributed) tuple $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$, there is an efficient algorithm that outputs a set of random coins to the sampling algorithm that would produce the tuple. In Section 5.1 (Theorem 5.9), we show that the explainable discrete Gaussian preimage sampler from [CHW25] can be used in conjunction with the succinct LWE trapdoor to build such a sampler. To argue that the reduction's strategy for constructing $\mathbf{z}_{0,i}$ follows the *same* distribution as that output by the sampler, we use a similar noise smudging argument as in [CHW25]. Specifically, the $\mathbf{r}_{\mathbf{d}_i}$ component in each $\mathbf{z}_{0,i}$ is sampled from a sufficiently-wide (but still low-norm) Gaussian distribution so as to drown out the remaining low-order terms. We give the details of the explainable sampler in Section 5.2 (see the proof of Theorem 5.14).

- Second, to achieve succinct ciphertexts that are independent of the attribute length, we use the compression technique from [Wee24, Wee25]. Specifically, let $\mathbf{C}_{\mathbf{x}}$ be a commitment to $\mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}$, and let $\mathbf{Z}_{\mathbf{x}}$ be the associated opening. Let $\mathbf{V}_{\ell m}$ be the verification matrix associated with $\mathsf{pp}_{\mathsf{com}}$ and matrices of length ℓm. Then Eq. (2.1) says

  $$\mathbf{C}_{\mathbf{x}}\mathbf{V}_{\ell m} = \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G} - \mathbf{B}\mathbf{Z}_{\mathbf{x}}.$$

  Sample $\mathbf{B}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ and define $\mathbf{A} = -\mathbf{B}_0\mathbf{V}_{\ell m}$. The attribute-embedding component $\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})}$ in the ciphertext is now replaced by $\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}})}$. The observation now is that

  $$[\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}} \mid \utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}})}] \cdot \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} \approx -\mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{Z}_{\mathbf{x}} - \mathbf{s}^{\mathsf{T}}\mathbf{B}_0\mathbf{V}_{\ell m} - \mathbf{s}^{\mathsf{T}}\mathbf{C}_{\mathbf{x}}\mathbf{V}_{\ell m} = \mathbf{s}^{\mathsf{T}}(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}).$$

  Thus $\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}})} \in \mathbb{Z}_q^m$ can be viewed as a *compressed* representation of $\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})}$.

We give the full details in Construction 5.11. Taken together, we obtain a key-policy registered ABE scheme with succinct ciphertexts and which supports an unbounded number of users from the $2m^2$-succinct LWE assumption.

3 Preliminaries

Throughout this work, we write $\lambda \in \mathbb{N}$ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, \ldots, n\}$. We write $\mathsf{poly}(\lambda)$ to denote a function that is bounded by a fixed polynomial in λ and $\mathsf{negl}(\lambda)$ to denote a function that is negligible in λ (i.e., $f(\lambda) = \mathsf{negl}(\lambda)$ if $f = o(\lambda^{-c})$ for all constants $c \in \mathbb{N}$). For a finite set

S, we write $x \xleftarrow{R} S$ to denote a uniform random draw from S. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote a sample from $\mathcal{D}$. When $\mathcal{A}$ is a deterministic algorithm, we write $x = \mathcal{A}(\cdot)$ to denote assigning x to the output of $\mathcal{A}$. We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We say that two distribution ensembles $\mathcal{D}_0 = \{\mathcal{D}_{0,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{A}(1^\lambda, x) : x \leftarrow \mathcal{D}_{0,\lambda}] - \Pr[\mathcal{A}(1^\lambda, x) : x \leftarrow \mathcal{D}_{1,\lambda}]| = \mathsf{negl}(\lambda).$$

We say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are statistically indistinguishable if the statistical distance between $\mathcal{D}_{0,\lambda}$ and $\mathcal{D}_{1,\lambda}$ is bounded by a negligible function $\mathsf{negl}(\lambda)$, and that they are identical if the statistical distance is identically 0.

3.1 Lattice Preliminaries

We use bold uppercase letters (e.g., $\mathbf{B}, \mathbf{B}$) for matrices and bold lowercase letters (e.g., $\mathbf{u}, \mathbf{v}$) for vectors. We use non-boldface letters for their components (e.g., $\mathbf{v} = [v_1, \ldots, v_n]$). For a matrix $\mathbf{V} \in \mathbb{Z}^{n \times n'}$ over the integers, we write $\|\mathbf{V}\|$ to denote the maximum absolute value of its entries. When $\mathbf{V} \in \mathbb{Z}_q^{n \times n'}$, we write $\|\mathbf{V}\| := \|\mathbf{V}_\mathbb{Z}\|$, where $\mathbf{V}_\mathbb{Z} \in \mathbb{Z}^{n \times n'}$ is the matrix obtained by replacing each component of $\mathbf{V}$ with its integer representative in the interval $(-q/2, q/2)$. We write $\mathbf{A} \otimes \mathbf{B}$ to denote the Kronecker product between matrices $\mathbf{A}$ and $\mathbf{B}$. For matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ with compatible dimensions, we have

$$(\mathbf{A} \otimes \mathbf{C})(\mathbf{B} \otimes \mathbf{D}) = \mathbf{AB} \otimes \mathbf{CD}. \tag{3.1}$$

For a matrix $\mathbf{A}$, we write $\mathrm{vec}(\mathbf{A})$ to denote the vector obtained by concatenating the columns of $\mathbf{A}$. We use the following identity:

$$\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^\mathsf{T} \otimes \mathbf{A}) \cdot \mathrm{vec}(\mathbf{B}). \tag{3.2}$$

Next, we recall the (generalized) leftover hash lemma from [DORS08]. We state the specific formulation from [ABB10]:

Lemma 3.1 (Generalized Leftover Hash Lemma [ABB10, Lemma 13, adapted])**.** *Let n, m, q be integers such that $m \geq 2n \log q$ and $q > 2$ is prime. Then, for all fixed vectors $\mathbf{e} \in \mathbb{Z}_q^m$ and all $k = \mathsf{poly}(n)$, the statistical distance between the following distributions is $\mathsf{negl}(n)$:*

- $(\mathbf{A}, \mathbf{AR}, \mathbf{e}^\mathsf{T}\mathbf{R})$ *where* $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}, \mathbf{R} \xleftarrow{R} \{0,1\}^{m \times k}$.
- $(\mathbf{A}, \mathbf{U}, \mathbf{e}^\mathsf{T}\mathbf{R})$ *where* $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}, \mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times k}, \mathbf{R} \xleftarrow{R} \{0,1\}^{m \times k}$.

Discrete Gaussians. We write $D_{\mathbb{Z},\sigma}$ to denote the discrete Gaussian distribution over $\mathbb{Z}$ with width parameter $\sigma > 0$. For a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a target $\mathbf{Z} \in \mathbb{Z}_q^{n \times n'}$ in the image of $\mathbf{B}$, we write $\mathbf{B}_\sigma^{-1}(\mathbf{Z})$ to denote the random variable $\mathbf{Y} \leftarrow D_{\mathbb{Z},\sigma}^{m \times n'}$ conditioned on $\mathbf{BY} = \mathbf{Z} \bmod q$. For positive integers $n, q \in \mathbb{N}$ and $m \geq n \lceil \log q \rceil$, we write $\mathbf{G}_n := \mathbf{I}_n \otimes \mathbf{g}^\mathsf{T} \in \mathbb{Z}_q^{n \times m}$ to denote the gadget matrix [MP12], where $\mathbf{I}_n$ is the identity matrix of dimension n, $\mathbf{g}^\mathsf{T} = [1, 2, \ldots, 2^{\lceil \log q \rceil - 1}, 0, \ldots, 0] \in \mathbb{Z}^m$. We write $\mathbf{G}_n^{-1}(\cdot) \colon \mathbb{Z}_q^n \to \mathbb{Z}_q^m$ to denote the standard deterministic entry-wise bit decomposition (and padding with 0s if $m \geq n \cdot \lceil \log q \rceil$). When the context is clear, we omit the subscript n and simply write $\mathbf{G}$ and $\mathbf{G}^{-1}(\cdot)$. We now recall some useful properties on the discrete Gaussian distribution that we will use:

Lemma 3.2 (Gaussian Tail Bound [MP12, Lemma 2.6, adapted])**.** *Let n, m, q be lattice parameters where $m \geq 2n \log q$. For all but a $\mathsf{negl}(n)$-fraction of matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, all $\sigma \geq \log m$, and all vectors $\mathbf{y} \in \mathbb{Z}_q^n$ in the column space of $\mathbf{B}$,*

$$\Pr[\|\mathbf{u}\| \geq \sqrt{m}\sigma : \mathbf{u} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{y})] \leq O(2^{-m}).$$

In addition, for all $\lambda \in \mathbb{N}$,

$$\Pr[|x| \geq \sqrt{\lambda}\sigma : x \leftarrow D_{\mathbb{Z},\sigma}] \leq 2^{-\lambda}.$$

Lemma 3.3 (Gaussian Preimages [GPV08])**.** *Let n, m, q be lattice parameters where $m \geq 2n \log q$ and q is prime. Let $\sigma \geq \log m$. There exist a negligible function $\mathsf{negl}(\cdot)$ such that for all but a $\mathsf{negl}(n)$-fraction of matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, the statistical distance between the following distributions is at most $\mathsf{negl}(n)$:*

$$\{(\mathbf{y}, \mathbf{By}) : \mathbf{y} \leftarrow D_{\mathbb{Z},\sigma}^m\} \quad \textit{and} \quad \{(\mathbf{y}, \mathbf{z}) : \mathbf{z} \xleftarrow{R} \mathbb{Z}_q^n, \mathbf{y} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{z})\}.$$

Moreover, this property holds when $\sigma \geq \log m$ and $\mathbf{B} = \mathbf{G}_n$.

Lemma 3.4 (Marginal of Gaussian Preimages [WW23, Corollary 2.11, adapted])**.** *Let n, m, q be lattice parameters where $m \geq 2n \log q$ and q is prime. Let $\ell, k = \mathsf{poly}(n, \log q)$. There exist a negligible function $\mathsf{negl}(\cdot)$ such that for all but a q^{-n}-fraction of matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, all matrices $\mathbf{W} \in \mathbb{Z}_q^{n\ell \times k}$ and matrices $\mathbf{C} = [\mathbf{I}_\ell \otimes \mathbf{B} \mid \mathbf{W}]$, all target vectors $\mathbf{y} \in \mathbb{Z}_q^{n\ell}$, and all width parameters $\sigma \geq 4 \log(\ell m)$, the statistical distance between the following distributions is at most $\mathsf{negl}(n)$:*

$$\left\{\mathbf{v} : \mathbf{v} \leftarrow \mathbf{C}_\sigma^{-1}(\mathbf{y})\right\} \quad \textit{and} \quad \left\{\begin{bmatrix}\mathbf{v}_1 \\ \mathbf{v}_2\end{bmatrix} : \begin{array}{c} \mathbf{v}_2 \leftarrow D_{\mathbb{Z},\sigma}^k \\ \mathbf{v}_1 \leftarrow (\mathbf{I}_\ell \otimes \mathbf{B})_\sigma^{-1}(\mathbf{y} - \mathbf{B}\mathbf{v}_2) \end{array}\right\}.$$

Gadget trapdoors. We now recall the notion of a gadget trapdoor:

Lemma 3.5 (Gadget Trapdoor [Ajt96, GPV08, MP12])**.** *Let n, m, q be lattice parameters with $m \geq 3n \log q$. There exists efficient algorithms* $(\mathsf{TrapGen}, \mathsf{SamplePre})$ *with the following syntax:*

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{B}, \mathbf{T})$*: On input the lattice dimension n, the width m, and the modulus q, the trapdoor-generation algorithm outputs a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$.*
- $\mathsf{SamplePre}(\mathbf{B}, \mathbf{T}, \mathbf{Z}, \sigma) \to \mathbf{Y}$*: On input a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, a trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$, a target matrix $\mathbf{Z} \in \mathbb{Z}_q^{n \times n'}$, and a width parameter $\sigma > 0$, the preimage-sampling algorithm outputs $\mathbf{Y} \in \mathbb{Z}_q^{m \times n'}$.*

Moreover, the above algorithms satisfy the following properties:

- **Trapdoor distribution:** *If $(\mathbf{B}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, then the distribution of $\mathbf{B}$ is $\mathsf{negl}(n)$-close to the uniform distribution over $\mathbb{Z}_q^{n \times m}$. Moreover, $\mathbf{B}\mathbf{T} = \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ and $\|\mathbf{T}\| = 1$.*
- **Preimage sampling:** *For all matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$, width parameters $\sigma > 0$, and all $\mathbf{Z} \in \mathbb{Z}_q^{n \times n'}$ in the image of $\mathbf{B}$, the output $\mathbf{Y} \leftarrow \mathsf{SamplePre}(\mathbf{B}, \mathbf{T}, \mathbf{Z}, \sigma)$ satisfies $\mathbf{B}\mathbf{Y} = \mathbf{Z}$.*
- **Preimage distribution:** *There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$ where $\mathbf{B}\mathbf{T} = \mathbf{G}$, all $\sigma \geq m\|\mathbf{T}\| \log n$, and all targets $\mathbf{Z} \in \mathbb{Z}_q^{n \times n'}$, the output of $\mathsf{SamplePre}(\mathbf{B}, \mathbf{T}, \mathbf{Z}, \sigma)$ is $\mathsf{negl}(n)$-close to the distribution $\mathbf{B}_\sigma^{-1}(\mathbf{Z})$.*

Homomorphic computation. Our construction of registered ABE will rely on the lattice homomorphic evaluation machinery from [GSW13, BGG+14].

Theorem 3.6 (Homomorphic Encodings [GSW13, BGG+14])**.** *Let λ be a security parameter and n, m, q be lattice parameters where $m \geq 2n \log q$. Let $\mathcal{F}_{\ell,d}$ be a family of functions $f \colon \{0,1\}^\ell \to \{0,1\}$ that can be computed by a Boolean circuit of depth d. There exists a pair of efficient algorithms* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *with the following properties:*

- $\mathsf{EvalF}(\mathbf{A}, f) \to \mathbf{A}_f$*: On input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$ and a function $f \in \mathcal{F}_{\ell,d}$, output a matrix $\mathbf{A}_f \in \mathbb{Z}_q^{n \times m}$.*
- $\mathsf{EvalFX}(\mathbf{A}, f, \mathbf{x}) \to \mathbf{H}_{\mathbf{A},f,\mathbf{x}}$*: On input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, a function $f \in \mathcal{F}_{\ell,d}$, and an input $\mathbf{x} \in \{0,1\}^\ell$, output a matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \in \mathbb{Z}_q^{\ell m \times m}$.*

For all matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, functions $f \in \mathcal{F}_{\ell,d}$, and inputs $\mathbf{x} \in \{0,1\}^\ell$, the matrices $\mathbf{A}_f \leftarrow \mathsf{EvalF}(\mathbf{A}, f)$ and $\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \leftarrow \mathsf{EvalFX}(\mathbf{A}, f, \mathbf{x})$ satisfy the following properties:

- $(\mathbf{A} - \mathbf{x}^\mathsf{T} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{A},f,\mathbf{x}} = \mathbf{A}_f - f(\mathbf{x}) \cdot \mathbf{G}$.
- $\left\|\mathbf{H}_{\mathbf{A},f,\mathbf{x}}\right\| \leq m^{O(d)}$.

Learning with errors. The learning with errors (LWE) assumption [Reg05] with parameters (n, m, q, χ) states that

$$(\mathbf{B}, \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}) \overset{c}{\approx} (\mathbf{B}, \mathbf{c}^\mathsf{T}),$$

where $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m$, and $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$.

ℓ-succinct LWE. The ℓ-succinct LWE assumption [Wee24] states that $(\mathbf{B}, \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}})$ is pseudorandom even given a trapdoor for the matrix $[\mathbf{I}_\ell \otimes \mathbf{B} \mid \mathbf{W}]$, where $\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell n \times m}$. We give the formal statement below:

Assumption 3.7 (ℓ-Succinct LWE [Wee24])**.** Let λ be a security parameter and (n, m, q, χ) be LWE parameters. The (ℓ, σ)-succinct LWE assumption states that

$$(\mathbf{B}, \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}, \mathbf{W}, \mathbf{T}) \overset{c}{\approx} (\mathbf{B}, \mathbf{c}^{\mathsf{T}}, \mathbf{W}, \mathbf{T}),$$

where $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m$, $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$, $\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell n\times m}$, and $\mathbf{T} \leftarrow [\mathbf{I}_\ell \otimes \mathbf{B} \mid \mathbf{W}]_\sigma^{-1}(\mathbf{I}_\ell \otimes \mathbf{G})$. We abbreviate the assumption to ℓ-succinct LWE when $\sigma = \mathsf{poly}(\lambda, \ell, m)$.

3.2 Matrix Commitments

Our constructions rely on the matrix commitments recently introduced in the work of Wee [Wee25]. Specifically, a commitment to a matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times L}$ with respect to $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$ such that

$$\mathbf{C} \cdot \mathbf{V}_L = \mathbf{M} - \mathbf{B} \cdot \mathbf{Z}, \tag{3.3}$$

where $\mathbf{V}_L \in \mathbb{Z}_q^{m\times L}$ is a *fixed* verification matrix that only depends on the dimension L and $\mathbf{Z} \in \mathbb{Z}_q^{m\times L}$ is a short opening. The work of [Wee25] shows how to sample a commitment $\mathbf{C}$ to an arbitrary matrix $\mathbf{M} \in \mathbb{Z}_q^{n\times L}$ (for *any* L) given $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{W} \in \mathbb{Z}_q^{2m^2 n\times m}$, $\mathbf{T} \in \mathbb{Z}_q^{(2m^2+1)m\times 2m^3}$, and $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}] \cdot \mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$. In this context, $\mathbf{T}$ is a trapdoor for a succinct LWE instance with dimension $2m^2$. We now give the formal statement, and for completeness, include the description of the algorithms from [Wee25, §3.2] in Appendix A:

Lemma 3.8 (Matrix Commitment [Wee25, adapted])**.** *Let n, m, q be lattice parameters with $m \geq 2n \log q$. There exist a triple of efficient and* deterministic *algorithms* $(\mathsf{Com}^{\mathsf{mx}}, \mathsf{Ver}^{\mathsf{mx}}, \mathsf{Open}^{\mathsf{mx}})$ *with the following syntax:*

- $\mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}) \to \mathbf{C}$*: On input public parameters* pp *and a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n\times L}$*, output a matrix* $\mathbf{C} \in \mathbb{Z}_q^{n\times m}$.
- $\mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^L) \to \mathbf{V}_L$*: On input public parameters* pp *and the length parameter* L*, output a matrix* $\mathbf{V}_L \in \mathbb{Z}_q^{m\times L\lceil \log q\rceil}$.
- $\mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}) \to \mathbf{Z}$*: On input public parameters* pp *and a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n\times L}$*, output a matrix* $\mathbf{Z} \in \mathbb{Z}_q^{m\times L\lceil \log q\rceil}$.

Moreover, for all $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ *where* $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{W} \in \mathbb{Z}_q^{2m^2 n\times m}$, $\mathbf{T} \in \mathbb{Z}_q^{(2m^2+1)m\times 2m^3}$, $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}] \cdot \mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$*, all parameters* $L \in \mathbb{N}$*, all matrices* $\mathbf{M} \in \mathbb{Z}_q^{n\times L}$*, and setting* $\mathbf{C} = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{x})$, $\mathbf{V}_L = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^L)$*, and* $\mathbf{Z} = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{x})$,

$$\mathbf{C} \cdot \mathbf{V}_L = \mathbf{M} \cdot \mathbf{G}_L - \mathbf{B} \cdot \mathbf{Z} \quad \textit{and} \quad \|\mathbf{V}_L\| \leq O(\|\mathbf{T}\| \cdot m^4 \log q) \quad \textit{and} \quad \|\mathbf{Z}\| \leq O(\|\mathbf{T}\| \cdot m^7 \log q \log L).$$

Remark 3.9 (Committing to $\mathbf{M}$)**.** The commitment relation $(\mathbf{C} \cdot \mathbf{V}_L = \mathbf{M} \cdot \mathbf{G}_L - \mathbf{B} \cdot \mathbf{Z})$ in Lemma 3.8 does not completely match Eq. (3.3), but this is easy to fix by simply multiplying by $\mathbf{G}_L^{-1}(\mathbf{I}_L)$. In this case, we have

$$\mathbf{C} \cdot \underbrace{\mathbf{V}_L \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L)}_{\tilde{\mathbf{V}}_L} = \mathbf{M} \cdot \mathbf{G}_L \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L) - \mathbf{B} \cdot \mathbf{Z} \cdot \mathbf{G}^{-1}(\mathbf{I}_L) = \mathbf{M} - \mathbf{B} \cdot \underbrace{\mathbf{Z} \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L)}_{\tilde{\mathbf{Z}}_L}.$$

Since the columns of $\mathbf{G}_L^{-1}(\mathbf{I}_L)$ have Hamming weight 1, the matrices $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{Z}}$ are submatrices of $\mathbf{V}$ and $\mathbf{Z}$, respectively. This means $\|\tilde{\mathbf{V}}\| \leq \|\mathbf{V}\|$ and $\|\tilde{\mathbf{Z}}\| \leq \|\mathbf{Z}\|$. Since Eq. (3.3) is more convenient to work with in our constructions, we define algorithms $(\mathsf{Com}^{\mathsf{mat}}, \mathsf{Ver}^{\mathsf{mat}}, \mathsf{Open}^{\mathsf{mat}})$ as follows:

- $\mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$: Output $\mathbf{C} = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}) \in \mathbb{Z}_q^{n\times m}$.
- $\mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}, 1^L)$: Compute $\mathbf{V}'_L = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^L)$ and output $\mathbf{V}_L = \mathbf{V}'_L \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L) \in \mathbb{Z}_q^{m\times L}$.
- $\mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$: Compute $\mathbf{Z}'_L = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$ and output $\mathbf{Z} = \mathbf{Z}'_L \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L) \in \mathbb{Z}_q^{m\times L}$.

Moreover, for all $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ where $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{W} \in \mathbb{Z}_q^{2m^2 n\times m}$, $\mathbf{T} \in \mathbb{Z}_q^{(2m^2+1)m\times 2m^3}$, $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}] \cdot \mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$, all parameters $L \in \mathbb{N}$, all matrices $\mathbf{M} \in \mathbb{Z}_q^{n\times L}$, and setting $\mathbf{C} = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{x})$, $\mathbf{V}_L = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^L)$, and $\mathbf{Z} = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{x})$,

$$\mathbf{C} \cdot \mathbf{V}_L = \mathbf{M} - \mathbf{B} \cdot \mathbf{Z} \quad \text{and} \quad \|\mathbf{V}_L\| \leq O(\|\mathbf{T}\| \cdot m^4 \log q) \quad \text{and} \quad \|\mathbf{Z}\| \leq O(\|\mathbf{T}\| \cdot m^7 \log q \log L).$$

Committing to sparse matrices. Our applications to distributed broadcast encryption and registered ABE relies on the ability for users to commit to *sparse* matrices $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ where L is exponential (e.g., $L = 2^\lambda$), but where $\mathbf{M}$ only contains a polynomial number of non-zero columns. This property allows us to construct schemes that support an a priori *unbounded* number of users. When committing to exponentially-wide matrices, the verification matrix $\mathbf{V}_L$ and the opening $\mathbf{Z}$ are also exponentially wide. Our applications also require an algorithm that provides *local* access to the columns of $\mathbf{V}_L$ and $\mathbf{Z}$. Namely, there is an efficient algorithm that runs in time $\mathsf{poly}(m, \log q, \log L)$ which can compute the i^{th} column of the verification matrix $\mathbf{V}_L$. Similarly, there is an algorithm that runs in time $\mathsf{poly}(K, m, \log q, \log L)$ that computes the i^{th} column of the opening matrix $\mathbf{Z}$, where K is the number of non-zero columns in the matrix $\mathbf{M}$. It is straightforward to adapt the construction from [Wee25, §3.2] to support these properties. We summarize the properties we need in the following lemma, and provide a formal proof of it in Appendix A.1.

Lemma 3.10 (Committing to Sparse Matrices and Computing Local Openings)**.** *There exists a tuple of efficient algorithms* $(\mathsf{ComSparse}^{\mathsf{mat}}, \mathsf{VerLocal}^{\mathsf{mat}}, \mathsf{OpenLocal}^{\mathsf{mat}})$ *with the following syntax and properties:*

- $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}) \to \mathbf{C}$*: On input the public parameters* pp *(with lattice parameters n, m, q) and a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$, $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$ *outputs* $\mathbf{C} = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$ *in time* $\mathsf{poly}(K, m, \log q, \log L)$*, where K is the number of non-zero columns in* $\mathbf{M}$.

- $\mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}, L, i) \to \mathbf{v}_{L,i}$*: On input the public parameters* pp *(with lattice parameters n, m, q), the length parameter L (in* binary*) and a column index $i \in [L]$,* $\mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}, L, i)$ *outputs the i^{th} column* $\mathbf{v}_{L,i} \in \mathbb{Z}_q^m$ *of the matrix* $\mathbf{V}_L = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}, 1^L)$ *in time* $\mathsf{poly}(m, \log q, \log L)$.

- $\mathsf{OpenLocal}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}, i) \to \mathbf{z}_i$*: On input the public parameters* pp *(with lattice parameters n, m, q), a sparse matrix* $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$*, and an index $i \in [L]$,* $\mathsf{OpenLocal}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}, i)$ *outputs the i^{th} column* $\mathbf{z}_i \in \mathbb{Z}_q^m$ *of* $\mathbf{Z}_i = \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$ *in time* $\mathsf{poly}(K, m, \log q, \log L)$*, where K is the number of non-zero columns in* $\mathbf{M}$.

4 Distributed Broadcast Encryption

In this section, we show how to construct an distributed broadcast encryption [WQZD10, BZ14] scheme from the succinct LWE assumption. In distributed broadcast encryption, users choose their own public and secret keys and post their public keys to a public-key directory. One can then encrypt a message to an arbitrary subset of public keys with a ciphertext whose size scales polylogarithmically with the number of users in the broadcast set. Previous distributed broadcast encryption schemes based on bilinear maps [KMW23, FWW23] or succinct LWE [CW24] could only support a bounded number of users, where the size of the public parameters and individual public keys scale linearly (or worse) with the bound on the number of users. We give the first distributed broadcast encryption scheme from succinct LWE that supports an arbitrary polynomial number of users. Previous schemes that could support an unbounded number of users relied either on indistinguishability obfuscation [BZ14] or witness encryption [FWW23].

Distributed broadcast encryption. We begin by recalling the notion of distributed broadcast encryption. Our definition is adapted from the works of [BZ14, KMW23].[1]

Definition 4.1 (Distributed Broadcast Encryption [BZ14, KMW23])**.** Let λ be a security parameter. A distributed broadcast encryption scheme Π_{DBE} is a tuple of efficient algorithms $\Pi_{\mathsf{DBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, N) \to \mathsf{pp}$: On input the security parameter λ and the number of users N (in *binary*), the setup algorithm outputs the public parameters pp. We assume that pp contains 1^λ and N.

- $\mathsf{KeyGen}(\mathsf{pp}, i) \to (\mathsf{pk}_i, \mathsf{sk}_i)$: On input the public parameters pp and an index $i \in [N]$, the key-generation algorithm outputs a public key pk_i and secret key sk_i.

[1] Some previous lattice-based schemes [CW24, CHW25] also required an $\mathsf{IsValid}$ algorithm that is used to (publicly) decide whether a user public key is well-formed or not, and only required correctness/security to hold when encrypting to well-formed keys. Our distributed broadcast encryption construction (Construction 4.2) will not require this property, so for ease of exposition, we omit this algorithm from the formal syntax. The syntax we use matches that from [BZ14, KMW23].

- $\mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mu) \to \mathsf{ct}$: On input the public parameters pp, a collection of public keys pk_i and a message $\mu \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mathsf{ct}, (i^*, \mathsf{sk}_{i^*})) \to \mu$: On input the public parameters pp, a collection of public keys pk_i, a ciphertext ct, and a secret key sk_{i^*} for an index i^*, the decryption algorithm outputs a message $\mu \in \{0, 1\}$.

We require that Π_{DBE} satisfy the following properties:

- **Correctness:** For all parameters $\lambda, N \in \mathbb{N}$, all pp in the support of $\mathsf{Setup}(1^\lambda, N)$, all sets $S \subseteq [N]$, all indices $i^* \in S$, all public keys pk_i for $i \in S \setminus \{i^*\}$, and all messages $\mu \in \{0, 1\}$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mathsf{ct}, (i^*, \mathsf{sk}_{i^*})) = \mu : \begin{array}{c} (\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i^*) \\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mu) \end{array}\right] = 1.$$

- **Selective security:** For a security parameter λ, a bound N, and a bit $b \in \{0, 1\}$, we define the selective security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - On input the security parameter 1^λ and the bound N, the adversary outputs the challenge set $S^* \subseteq [N]$.
  - The challenger samples $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$ and $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$ for $i \in S^*$. Finally, it computes $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i \in S^*}, b, S^*)$ and sends $(\mathsf{pp}, \{\mathsf{pk}_i\}_{i \in S^*}, \mathsf{ct}_b)$ to $\mathcal{A}$.
  - At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$, which is the output of the experiment.

  We say the distributed broadcast encryption scheme is selectively secure if for all efficient adversaries $\mathcal{A}$ and all bounds $N \leq 2^\lambda$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda)$$

  in the selective security game.

- **Succinctness:** There exists a fixed polynomial $\mathsf{poly}(\cdot, \cdot)$ such that for all $\lambda, N \in \mathbb{N}$, all subsets $S \subseteq [N]$, all public parameters pp in the support of $\mathsf{Setup}(1^\lambda, N)$, all public keys pk_i, all messages $\mu \in \{0, 1\}$, and all ciphertexts ct in the support of $\mathsf{Encrypt}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i \in S}, \mu, S)$, it holds that $|\mathsf{ct}| \leq \mathsf{poly}(\lambda, \log N)$.

Construction 4.2 (Distributed Broadcast Encryption). Let λ be a security parameter and (n, m, q, χ) be LWE parameters (that can be functions of λ and N). Let σ be a Gaussian width parameter. We construct a distributed broadcast encryption scheme as follows:

- $\mathsf{Setup}(1^\lambda, N)$: On input the security parameter λ and the number of users N, sample

$$\begin{aligned} (\mathbf{B}, \mathbf{T}_{\mathbf{B}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{2m^2 n \times m} \\ \mathbf{T} &\leftarrow \mathsf{SamplePre}\left([\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}], \begin{bmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}} \\ \mathbf{0} \end{bmatrix}, \mathbf{I}_{2m^2} \otimes \mathbf{G}, \sigma\right) \\ \mathbf{A} &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n. \end{aligned}$$

  If $\|\mathbf{T}\| > \sqrt{m}\sigma$, then set $\mathbf{T} = \begin{bmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}} \\ \mathbf{0} \end{bmatrix}$. Let $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and output the public parameters $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$.

- $\mathsf{KeyGen}(\mathsf{pp}, i)$: On input the public parameters $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$ and an index $i \in [N]$, sample

$$\begin{aligned} \mathbf{r}_i &\xleftarrow{\mathrm{R}} \{0, 1\}^m \\ \mathbf{v}_i &= \mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, N, i) \end{aligned}$$

  Output the public key $\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_i \in \mathbb{Z}_q^n$ and the secret key $\mathsf{sk} = \mathbf{r}_i$.

- $\mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mu)$: On input the public parameters $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, the public keys $\mathsf{pk}_i = \mathbf{t}_i \in \mathbb{Z}_q^{n \times m}$ and a message $\mu \in \{0, 1\}$, sample the following:

$$\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \,,\; \mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m \,,\; \mathbf{D}_1 \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m} \,,\; \mathbf{d}_2 \xleftarrow{\mathrm{R}} \{0,1\}^m .$$

If $\|\mathbf{e}\| > \sqrt{m}\chi$, then set $\mathbf{e} = \mathbf{0}^m$. Then, for all $i \in S$, compute $\mathbf{C}_i = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i)$, where $\mathbf{u}_i \in \{0,1\}^N$ is the i^{th} unit vector. Output the ciphertext

$$\mathsf{ct} = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\; \mathbf{s}^{\mathsf{T}} \left(\mathbf{A} + \sum_{i \in S} \mathbf{C}_i \right) + \mathbf{e}^{\mathsf{T}}\mathbf{D}_1 \,,\; \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot \mu \right)$$

- $\mathsf{Decrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mathsf{ct}, (i^*, \mathsf{sk}_{i^*}))$: On input the public parameters $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, the public keys $\mathsf{pk}_i = \mathbf{t}_i \in \mathbb{Z}_q^{n \times m}$, the ciphertext $\mathsf{ct} = (\mathbf{c}_1, \mathbf{c}_2, c_3)$, and a secret key $\mathsf{sk}_{i^*} = \mathbf{r}_{i^*}$ for an index $i^* \in S$, the decryption algorithm first computes

$$\begin{aligned} \mathbf{v}_{i^*} &= \mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, N, i^*) \\ \forall i \in S : \mathbf{z}_{i,i^*} &= \mathsf{OpenLocal}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i, i^*). \end{aligned}$$

Finally, compute

$$\tilde{\mu} = c_3 + \mathbf{c}_1^{\mathsf{T}}\mathbf{r}_{i^*} - \mathbf{c}_2^{\mathsf{T}}\mathbf{v}_{i^*} - \sum_{i \in S} \mathbf{c}_1^{\mathsf{T}}\mathbf{z}_{i,i^*}$$

and output 0 if $-q/4 < \xi < q/4$ and 1 otherwise.

Theorem 4.3 (Correctness). *Suppose $q > N \cdot O(m^9 \chi \sigma \log q \log N)$. Then, Construction 4.2 is correct.*

Proof. Take any $\lambda, N \in \mathbb{N}$, any pp in the support of $\mathsf{Setup}(1^\lambda, N)$, any set $S \subseteq [N]$, any index $i^* \in S$, any collection of public keys $\{\mathsf{pk}_i\}_{i \in S \setminus \{i^*\}}$, and any message $\mu \in \{0, 1\}$. Let $(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i^*)$ and $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mu)$. Consider $\mathsf{Decrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, \mathsf{ct}, (i^*, \mathsf{sk}_{i^*}))$:

- Let $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. By construction of Setup, $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]\mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$ and $\|\mathbf{T}\| \le \sqrt{m}\sigma$.
- By definition, $\mathsf{pk}_{i^*} = \mathbf{t}_{i^*} = \mathbf{B}\mathbf{r}_{i^*} + \mathbf{p} - \mathbf{A}\mathbf{v}_{i^*}$ where $\mathbf{r}_{i^*} \in \{0,1\}^m$ and $\mathbf{v}_{i^*} = \mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, N, i^*)$.
- Let $\mathbf{s}, \mathbf{e}, \mathbf{D}_1, \mathbf{d}_2$ be the components sampled by $\mathsf{Encrypt}$. By definition, $\|\mathbf{e}\| \le \sqrt{m}\chi$ and $\|\mathbf{D}_1\|, \|\mathbf{d}_2\| \le 1$. The ciphertext can be written as

$$\mathsf{ct} = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, c_3) = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\; \mathbf{s}^{\mathsf{T}} \left(\mathbf{A} + \sum_{i \in S} \mathbf{C}_i \right) + \mathbf{e}^{\mathsf{T}}\mathbf{D}_1 \,,\; \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot \mu \right).$$

- Let $\mathbf{v}_{i^*}$ and $\mathbf{z}_{i,i^*}$ be the vectors computed by $\mathsf{Decrypt}$. For each $i \in S$, let $\mathbf{C}_i = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i)$. By Remark 3.9 and Lemma 3.10, we have for all $i \in S$

$$\mathbf{C}_i\mathbf{v}_{i^*} = \begin{cases} \mathbf{t}_{i^*} - \mathbf{B}\mathbf{z}_{i^*,i^*} & i = i^* \\ -\mathbf{B}\mathbf{z}_{i,i^*} & i \neq i^*. \end{cases}$$

Thus, we can now write

$$\begin{aligned} \mathbf{c}_2^{\mathsf{T}}\mathbf{v}_{i^*} + \sum_{i \in S} \mathbf{c}_1^{\mathsf{T}}\mathbf{z}_{i,i^*} &= \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{v}_{i^*} + \sum_{i \in S} \mathbf{s}^{\mathsf{T}}\mathbf{C}_i\mathbf{v}_{i^*} + \sum_{i \in S} \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{z}_{i,i^*} + \mathbf{e}^{\mathsf{T}}\mathbf{D}_1\mathbf{v}_{i^*} + \sum_{i \in S} \mathbf{e}^{\mathsf{T}}\mathbf{z}_{i,i^*} \\ &= \mathbf{s}^{\mathsf{T}}(\mathbf{A}\mathbf{v}_{i^*} + \mathbf{t}_{i^*}) + \tilde{e}_1 \\ &= \mathbf{s}^{\mathsf{T}}(\mathbf{p} + \mathbf{B}\mathbf{r}_{i^*}) + \tilde{e}_1. \end{aligned}$$

where $\tilde{e}_1 = \mathbf{e}^{\mathsf{T}}\mathbf{D}_1\mathbf{v}_{i^*} + \sum_{i\in S}\mathbf{e}^{\mathsf{T}}\mathbf{z}_{i,i^*}$. Then

$$\begin{aligned}\tilde{\mu} &= c_3 + \mathbf{c}_1^{\mathsf{T}}\mathbf{r}_{i^*} - \mathbf{c}_2^{\mathsf{T}}\mathbf{v}_{i^*} - \sum_{i\in S}\mathbf{c}_1^{\mathsf{T}}\mathbf{z}_{i,i^*} \\ &= \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2\rfloor\cdot\mu + \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{r}_{i^*} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{i^*} - \mathbf{s}^{\mathsf{T}}(\mathbf{p} + \mathbf{B}\mathbf{r}_{i^*}) - \tilde{e}_1 \\ &= \lfloor q/2\rfloor\cdot\mu + \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{i^*} - \tilde{e}_1,\end{aligned}$$

Let $\tilde{e} = \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{i^*} - \tilde{e}_1$. As long as $|\tilde{e}| < q/4$, correctness holds.

- We now bound $|\tilde{e}|$. First, for all $i \in S$, by Remark 3.9 and Lemma 3.10,

$$\begin{aligned}\|\mathbf{v}_{i^*}\| &\le O(\|\mathbf{T}\|\cdot m^4\log q) \le O(m^{9/2}\sigma\log q) \\ \left\|\mathbf{z}_{i,i^*}\right\| &\le O(\|\mathbf{T}\|\cdot m^7\log q\log N) \le O(m^{15/2}\sigma\log q\log N).\end{aligned}$$

First, we bound $|\tilde{e}_1|$. Since $\|\mathbf{e}\| \le \sqrt{m}\chi$, we have

$$\begin{aligned}|\tilde{e}_1| \le \left\|\mathbf{e}^{\mathsf{T}}\mathbf{D}_1\mathbf{v}_{i^*}\right\| + \sum_{i\in S}\left\|\mathbf{e}^{\mathsf{T}}\mathbf{z}_{i,i^*}\right\| &\le O(m^7\chi\sigma\log q) + N\cdot O(m^9\chi\sigma\log q\log N) \\ &= N\cdot O(m^9\chi\sigma\log q\log N).\end{aligned}$$

Next,

$$|\tilde{e}| \le |\mathbf{e}^{\mathsf{T}}\mathbf{d}_2| + |\mathbf{e}^{\mathsf{T}}\mathbf{r}_{i^*}| + |\tilde{e}_1| \le N\cdot O(m^9\chi\sigma\log q\log N).$$

Setting $q > N\cdot O(m^9\chi\sigma\log q\log N)$ suffices for correctness. □

Theorem 4.4 (Selective Security). *Suppose $n \ge \lambda$, $m \ge 3n\log q$, and $\sigma \ge O(m^3\log m)$. Then, under the $(2m^2, \sigma)$-succinct LWE assumption with parameters (n, m, q, χ), Construction 4.2 is selectively secure.*

Proof. Take any $N \le 2^\lambda$ and any efficient adversary $\mathcal{A}$ for the selective security game. We now define a sequence of hybrid experiments:

- $\mathsf{Hyb}_0^{(b)}$: This is the selective security experiment where the challenger encrypts the bit $b \in \{0,1\}$. Specifically, the game proceeds as follows:
  - On input the security parameter 1^λ and the bound N, algorithm $\mathcal{A}$ outputs the challenge set $S^* \subseteq [N]$.
  - The challenger then samples $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, N)$ by computing

$$\begin{aligned}(\mathbf{B}, \mathbf{T}_{\mathbf{B}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{2m^2n\times m} \\ \mathbf{T} \leftarrow \mathsf{SamplePre}([\mathbf{I}_{2m^2}\otimes\mathbf{B} \mid \mathbf{W}], &\left[\begin{smallmatrix}\mathbf{I}_{2m^2}\otimes\mathbf{T}_{\mathbf{B}} \\ \mathbf{0}\end{smallmatrix}\right], \mathbf{I}_{2m^2}\otimes\mathbf{G}, \sigma) \\ \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}, \mathbf{p} &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^n.\end{aligned}$$

    If $\|\mathbf{T}\| > \sqrt{m}\sigma$, then it sets $\mathbf{T} = \left[\begin{smallmatrix}\mathbf{I}_{2m^2}\otimes\mathbf{T}_{\mathbf{B}} \\ \mathbf{0}\end{smallmatrix}\right]$. The challenger sets $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$.
  - Next, for each $i \in S^*$, the challenger samples a public key by computing $\mathsf{pk}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$. Specifically, for each $i \in S^*$, the challenger samples

$$\begin{aligned}\mathbf{r}_i &\xleftarrow{\mathrm{R}} \{0,1\}^m \\ \mathbf{v}_i &= \mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, N, i)\end{aligned}$$

    and sets $\mathsf{pk}_i = \mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_i \in \mathbb{Z}_q^{n\times m}$.
  - Finally, the challenger constructs the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i\in[N]}, b)$. Concretely, the challenger starts by sampling

$$\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n,\ \mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m,\ \mathbf{D}_1 \xleftarrow{\mathrm{R}} \{0,1\}^{m\times m},\ \mathbf{d}_2 \xleftarrow{\mathrm{R}} \{0,1\}^m.$$

If $\|\mathbf{e}\| > \sqrt{m}\chi$, then the challenger sets $\mathbf{e} = \mathbf{0}^m$. Then, for all $i \in S$, the challenger computes $\mathbf{C}_i = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i)$. It defines the challenge ciphertext ct^* to be

$$\mathsf{ct}^* = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\; \mathbf{s}^{\mathsf{T}} \left(\mathbf{A} + \sum_{i \in S^*} \mathbf{C}_i \right) + \mathbf{e}^{\mathsf{T}}\mathbf{D}_1 \,,\; \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot b \right).$$

- The challenger gives $(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S^*}, \mathsf{ct}^*)$ to $\mathcal{A}$.
- At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

- $\mathsf{Hyb}_1^{(b)}$: Same as $\mathsf{Hyb}_0^{(b)}$, except when sampling the public parameters, the challenger samples

$$\mathbf{T} \xleftarrow{\mathrm{R}} [\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]_\sigma^{-1}(\mathbf{I}_{2m^2} \otimes \mathbf{G}) \quad \text{and} \quad \mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}.$$

- $\mathsf{Hyb}_2^{(b)}$: Same as $\mathsf{Hyb}_1^{(b)}$, except when sampling the public parameters, the challenger no longer checks the condition $\|\mathbf{T}\| > \sqrt{m}\sigma$. Similarly, when constructing the challenge ciphertext ct^*, the challenger no longer checks if $\|\mathbf{e}\| > \sqrt{m}\chi$.

- $\mathsf{Hyb}_3^{(b)}$: Same as $\mathsf{Hyb}_2^{(b)}$, except the challenger samples $\mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ for all $i \in S^*$.

- $\mathsf{Hyb}_4^{(b)}$: Same as $\mathsf{Hyb}_3^{(b)}$, except the challenger now sets $\mathbf{A} = \mathbf{B}\mathbf{D}_1 - \sum_{i \in S^*} \mathbf{C}_i$ and $\mathbf{p} = \mathbf{B}\mathbf{d}_2$. Specifically, this experiment operates as follows:
  - On input the security parameter 1^λ, algorithm $\mathcal{A}$ outputs the bound N and a challenge set $S^* \subseteq [N]$.
  - The challenger samples

$$\begin{gathered} \mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{2m^2 n \times m} \\ \mathbf{T} \leftarrow [\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]_\sigma^{-1}(\mathbf{I}_{2m^2} \otimes \mathbf{G}) \\ \mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \text{ for all } i \in S^* \\ \mathbf{D}_1 \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}, \mathbf{d}_2 \xleftarrow{\mathrm{R}} \{0,1\}^m \end{gathered}$$

  The challenger sets $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. For each $i \in S^*$, the challenger sets $\mathsf{pk}_i = \mathbf{t}_i$ and also computes $\mathbf{C}_i = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i)$. The challenger sets $\mathbf{A} = \mathbf{B}\mathbf{D}_1 - \sum_{i \in S^*} \mathbf{C}_i$, $\mathbf{p} = \mathbf{B}\mathbf{d}_2$, and $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$.
  - To construct the challenge ciphertext, the challenger samples $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m$. It then defines the challenge ciphertext ct^* to be

$$\begin{aligned} \mathsf{ct}^* &= \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\; \mathbf{s}^{\mathsf{T}} \left(\mathbf{A} + \sum_{i \in S^*} \mathbf{C}_i \right) + \mathbf{e}^{\mathsf{T}}\mathbf{D}_1 \,,\; \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot b \right) \\ &= \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\; (\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}})\mathbf{D}_1 \,,\; (\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}})\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot b \right). \end{aligned}$$

  - The challenger gives $(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S^*}, \mathsf{ct}^*)$ to $\mathcal{A}$.
  - At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

- $\mathsf{Hyb}_5^{(b)}$: Same as $\mathsf{Hyb}_4^{(b)}$, except the challenger samples $\mathbf{c}_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$ and defines the challenge ciphertext to be

$$\mathsf{ct}^* = (\mathbf{c}_1^{\mathsf{T}} \,,\; \mathbf{c}_1^{\mathsf{T}}\mathbf{D}_1 \,,\; \mathbf{c}_1^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot b).$$

- $\mathsf{Hyb}_6^{(b)}$: Same as $\mathsf{Hyb}_5^{(b)}$, except the challenger samples $\mathbf{c}_2 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$ and $c_3 \xleftarrow{\mathrm{R}} \mathbb{Z}_q$. The challenger defines the challenge ciphertext to be $\mathsf{ct}^* = (\mathbf{c}_1^{\mathsf{T}} \,,\; \mathbf{c}_2^{\mathsf{T}} \,,\; c_3)$. Notably, in this experiment, the adversary's view is independent of the bit b.

We write $\mathsf{Hyb}_i^{(b)}(\mathcal{A})$ to denote the distribution of the output of $\mathsf{Hyb}_i^{(b)}$ with adversary $\mathcal{A}$. We now analyze each pair of adjacent distributions.

Lemma 4.5. *If $n \ge \lambda$, $m \ge 3n \log q$, and $\sigma \ge O(m^3 \log m)$, then for all $b \in \{0, 1\}$, $\mathsf{Hyb}_0^{(b)}$ and $\mathsf{Hyb}_1^{(b)}$ are statistically indistinguishable.*

Proof. By Lemma 3.5, $\|\mathbf{T}_{\mathbf{B}}\| = 1$. As long as $\sigma \ge (2m^2 + 1)m \|\mathbf{T}_{\mathbf{B}}\| \log(2m^2 n) = O(m^3 \log m)$, these two distributions are statistically indistinguishable by Lemma 3.5. □

Lemma 4.6. *If $n \ge \lambda$, $m \ge 2n \log q$, q is prime, and $\sigma \ge O(\log m)$, then for all $b \in \{0, 1\}$, $\mathsf{Hyb}_1^{(b)}$ and $\mathsf{Hyb}_2^{(b)}$ are statistically indistinguishable.*

Proof. By Lemmas 3.2 and 3.4, with overwhelming probability over the choice of $\mathbf{B} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, we have that $\|\mathbf{T}\| \le \sqrt{m}\sigma$. By Lemma 3.4, we have that $\|\mathbf{e}\| \le \sqrt{m}\chi$ with probability at least $1 - m \cdot O(2^{-m})$. Thus, the conditions the challenger checks in $\mathsf{Hyb}_1^{(b)}$ hold with negligible probability so the two experiments are statistically indistinguishable. □

Lemma 4.7. *If $n \ge \lambda$, $m \ge 2n \log q$, and $q > 2$ is prime, then for all $b \in \{0, 1\}$, $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$ are statistically indistinguishable by*

Proof. Specifically, by Lemma 3.1, the distributions $(\mathbf{B}, \mathbf{B}\mathbf{r})$ and $(\mathbf{B}, \mathbf{t})$ where $\mathbf{B} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{r} \xleftarrow{\text{R}} \{0, 1\}^{m \times m}$, and $\mathbf{t} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$ are statistically indistinguishable. Since $|S^*| = \mathsf{poly}(\lambda)$, the claim now follows by a standard hybrid argument. □

Lemma 4.8. *If $n \ge \lambda$, $m \ge 2n \log q$, and $q > 2$ is prime, then for all $b \in \{0, 1\}$, $\mathsf{Hyb}_3^{(b)}$ and $\mathsf{Hyb}_4^{(b)}$ are statistically indistinguishable.*

Proof. Specifically, invoking Lemma 3.1 with the error vector $\mathbf{e} \in \mathbb{Z}_q^m$ the challenger samples in $\mathsf{Hyb}_3^{(b)}$ and $\mathsf{Hyb}_4^{(b)}$, we can conclude that the distributions $(\mathbf{B}, \mathbf{B}\mathbf{D}_1, \mathbf{e}^{\mathsf{T}}\mathbf{D}_1)$ and $(\mathbf{B}, \mathbf{A}^*, \mathbf{e}_1^{\mathsf{T}}\mathbf{D}_1)$ are statistically indistinguishable when $\mathbf{B} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{D}_1 \xleftarrow{\text{R}} \{0, 1\}^{m \times m}$, $\mathbf{A}^* \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$. In $\mathsf{Hyb}_3^{(b)}$ and $\mathsf{Hyb}_4^{(b)}$, write $\mathbf{A} = \mathbf{A}^* - \sum_{i \in S^*} \mathbf{C}_i$. When $\mathbf{A}^* = \mathbf{B}\mathbf{D}_1$, we obtain the distribution of $\mathbf{A}$ in $\mathsf{Hyb}_3^{(b)}$ and when $\mathbf{A}^* \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, we obtain the distribution of $\mathbf{A}$ in $\mathsf{Hyb}_4^{(b)}$ (since $\mathbf{A}^*$ is sampled independently of all other quantities in the experiment). Similarly, the distributions $(\mathbf{B}, \mathbf{B}\mathbf{d}_2, \mathbf{e}^{\mathsf{T}}\mathbf{d}_2)$ and $(\mathbf{B}, \mathbf{p}, \mathbf{e}^{\mathsf{T}}\mathbf{d}_2)$ are statistically indistinguishable when $\mathbf{B} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{d}_2 \xleftarrow{\text{R}} \{0, 1\}^m$, $\mathbf{p} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and the claim holds. □

Lemma 4.9. *Under the $(2m^2, \sigma)$-succinct LWE assumption with lattice parameters (n, m, q, χ), for all $b \in \{0, 1\}$, $\mathsf{Hyb}_4^{(b)}$ and $\mathsf{Hyb}_5^{(b)}$ are computationally indistinguishable.*

Proof. Suppose $|\Pr[\mathsf{Hyb}_4^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_5^{(b)}(\mathcal{A}) = 1]| \ge \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the $(2m^2)$-succinct LWE assumption:

1. On input the succinct LWE challenge $(\mathbf{B}, \mathbf{c}_1, \mathbf{W}, \mathbf{T})$, algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ which outputs the bound $N = N(\lambda)$ and the challenge set $S^* \subseteq [N]$.

2. Algorithm $\mathcal{B}$ sets $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and samples $\mathbf{D}_1 \xleftarrow{\text{R}} \{0, 1\}^{m \times m}$ and $\mathbf{d}_2 \xleftarrow{\text{R}} \{0, 1\}^m$. Next, for each $i \in S^*$, algorithm $\mathcal{B}$ samples $\mathsf{pk}_i = \mathbf{P}_i \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$ and computes $\mathbf{C}_i = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{P}_i)$. Finally, algorithm $\mathcal{B}$ sets $\mathbf{A} = \mathbf{B}\mathbf{D}_1 - \sum_{i \in S^*} \mathbf{C}_i$, $\mathbf{p} = \mathbf{B}\mathbf{d}_2$, and $\mathsf{pp} = (N, \mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$.

3. Finally, algorithm $\mathcal{B}$ constructs the challenger ciphertext as $\mathsf{ct}^* = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_1^{\mathsf{T}}\mathbf{D}_1, \mathsf{ct}_1^{\mathsf{T}}\mathbf{d}_2 + \lfloor q/2 \rfloor \cdot b)$.

4. Algorithm $\mathcal{B}$ gives $\big(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S^*}, \mathsf{ct}^*\big)$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

By definition, the challenger samples $\mathbf{B} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{2m^2 n \times m}$, and $\mathbf{T} \leftarrow [\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]_\sigma^{-1}(\mathbf{I}_{2m^2} \otimes \mathbf{G})$. Thus, algorithm $\mathcal{B}$ perfectly simulates pp according to the distribution of $\mathsf{Hyb}_4^{(b)}$ and $\mathsf{Hyb}_5^{(b)}$. The remaining components in the public parameters and the public keys are sampled exactly as in $\mathsf{Hyb}_4^{(b)}$ and $\mathsf{Hyb}_5^{(b)}$. It suffices to consider the challenger ciphertext:

- If the challenger sets $\mathbf{c}_1^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}$ where $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $\mathbf{e} \xleftarrow{\mathrm{R}} D_{\mathbb{Z},\chi}^m$, then algorithm $\mathcal{B}$ perfectly simulates the distribution of $\mathsf{Hyb}_4^{(b)}$.
- If the challenger samples $\mathbf{c}_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$, then algorithm $\mathcal{B}$ perfectly simulates the distribution of $\mathsf{Hyb}_5^{(b)}$.

Thus, algorithm $\mathcal{B}$ breaks succinct LWE security with the same advantage ε. □

Lemma 4.10. *If $n \geq \lambda$, $m \geq 2n \log q$, and $q > 2$ is prime, then for all $b \in \{0,1\}$, $\mathsf{Hyb}_5^{(b)}$ and $\mathsf{Hyb}_6^{(b)}$ are statistically indistinguishable.*

Proof. Applying Lemma 3.1 with respect to the matrix $\begin{bmatrix} \mathbf{B} \\ \mathbf{c}_1^{\mathsf{T}} \end{bmatrix}$, the following distributions are statistically indistinguishable:

$$(\mathbf{B}, \mathbf{c}_1, \mathbf{B}\mathbf{D}_1, \mathbf{B}\mathbf{d}_2, \mathbf{c}_1^{\mathsf{T}}\mathbf{D}_1, \mathbf{c}_1^{\mathsf{T}}\mathbf{d}_2) \quad \text{and} \quad (\mathbf{B}, \mathbf{c}_1, \mathbf{A}^*, \mathbf{p}, \mathbf{c}_2, c_3),$$

when $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{c}_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$, $\mathbf{D}_1 \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}$, $\mathbf{d}_2 \xleftarrow{\mathrm{R}} \{0,1\}^m$, $\mathbf{A}^* \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{c}_2 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$, and $c_3 \xleftarrow{\mathrm{R}} \mathbb{Z}_q$. Applying Lemma 3.1 again with respect to the matrix $\mathbf{B}$, we conclude that the following two distributions are also statistically indistinguishable:

$$(\mathbf{B}, \mathbf{c}_1, \mathbf{A}^*, \mathbf{p}, \mathbf{c}_2, c_3) \quad \text{and} \quad (\mathbf{B}, \mathbf{c}_1, \mathbf{B}\mathbf{D}_1, \mathbf{B}\mathbf{d}_2, \mathbf{c}_2, c_3),$$

By a hybrid argument, we conclude that

$$(\mathbf{B}, \mathbf{c}_1, \mathbf{B}\mathbf{D}_1, \mathbf{B}\mathbf{d}_2, \mathbf{c}_1^{\mathsf{T}}\mathbf{D}_1, \mathbf{c}_1^{\mathsf{T}}\mathbf{d}_2) \quad \text{and} \quad (\mathbf{B}, \mathbf{c}_1, \mathbf{B}\mathbf{D}_1, \mathbf{B}\mathbf{d}_2, \mathbf{c}_2, c_3)$$

are statistically indistinguishable. The left distribution maps to $\mathsf{Hyb}_5^{(b)}$ while the right distribution maps to $\mathsf{Hyb}_6^{(b)}$. □

By construction, the challenger's behavior in $\mathsf{Hyb}_6^{(b)}$ is independent of the bit $b \in \{0,1\}$. Thus, $\mathsf{Hyb}_6^{(0)}(\mathcal{A}) \equiv \mathsf{Hyb}_6^{(1)}(\mathcal{A})$. The claim now follows by combining Lemmas 4.5 to 4.10. □

Theorem 4.11 (Succinctness)**.** *Suppose $m \log q = \mathrm{poly}(n, \log N)$. Then Construction 4.2 is succinct.*

Proof. A ciphertext in Construction 4.2 can be written as $(\mathbf{c}_1, \mathbf{c}_2, c_3)$ where $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{Z}_q^m$ and $c_3 \in \mathbb{Z}_q$. Thus, the total size of a ciphertext is $O(m \log q) = \mathrm{poly}(n, \log N)$, as required. □

Parameter instantiation. We now describe one instantiation of the parameters in Construction 4.2 to satisfy Theorems 4.3, 4.4, and 4.11. Let λ be the security parameter and N be the maximum number of users. Take any constant $0 < \varepsilon < 1$. We instantiate the LWE parameters (n, m, q, χ) and the width parameter σ where

$$\begin{aligned} n &= \lambda \log^{1/\varepsilon} N \\ m &= n \cdot \mathrm{poly}(\lambda, \log N) \\ \chi &= \mathrm{poly}(\lambda, \log N) \\ \sigma &= O(m^3 \log m) \\ q &= N \cdot \mathrm{poly}(\lambda, \log N) \end{aligned}$$

and the $(2m^2, \sigma)$-succinct LWE assumption holds with LWE parameters (n, m, q, χ). This corresponds to a succinct LWE instance where the modulus-to-noise ratio is $N \cdot \mathrm{poly}(\lambda, \log N) = 2^{\tilde{O}(n^{\varepsilon})}$. For this choice of parameters, we obtain a distributed broadcast encryption scheme with the following properties:

$$|\mathsf{pp}| = \mathrm{poly}(\lambda, \log N) \quad , \quad |\mathsf{pk}| = \mathrm{poly}(\lambda, \log N) \quad , \quad |\mathsf{sk}| = \mathrm{poly}(\lambda, \log N) \quad , \quad |\mathsf{ct}| = \mathrm{poly}(\lambda, \log N).$$

In particular, the size of the public parameters, the size of individual public/secret keys, and the size of the ciphertext are optimal . Moreover, in the case where $N = \mathrm{poly}(\lambda)$, then $q = \mathrm{poly}(\lambda)$. In this case, we can rely on succinct LWE with a *polynomial* modulus-to-noise-ratio. We summarize our instantiation with the following corollary:

Corollary 4.12 (Distributed Broadcast Encryption)**.** *Let λ be a security parameter and N be the number of users. Under the* $\mathsf{poly}(\lambda, \log N)$*-succinct LWE assumption with a sub-exponential modulus-to-noise ratio, there exists a distributed broadcast encryption scheme supporting up to N users where the size of the public parameters, the size of individual public/secret keys, and the size of the ciphertext is* $\mathsf{poly}(\lambda, \log N)$*. Moreover, when $N = \mathsf{poly}(\lambda)$, then security can be based on the* $\mathsf{poly}(\lambda, \log N)$*-succinct LWE assumption with a* polynomial *modulus-to-noise ratio.*

Remark 4.13 (Transparent Setup via Decomposed LWE)**.** Our distributed broadcast encryption scheme (Construction 4.2) makes black-box use of the [Wee25] matrix commitment scheme. Moreover, the cryptographic assumption in the security proof (Theorem 4.4) can equivalently be restated as asking that

$$\left(\mathsf{pp}_{\mathsf{com}}, \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}\right) \overset{c}{\approx} \left(\mathsf{pp}_{\mathsf{com}}, \mathbf{c}^{\mathsf{T}}\right),$$

where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ is the public parameters for the matrix commitment scheme, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m$, and $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$. In [Wee25, Appendix C], Wee shows an alternative way to realize a matrix commitment scheme from the decomposed LWE assumption introduced in [AMR25]. Specifically, the decomposed LWE assumption (with dimension ℓ and width parameter σ) along with LWE parameters (n, m, q, χ) asserts that

$$\left(\mathbf{B}, \mathbf{s}^{\mathsf{T}}(\mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) + \mathrm{vec}(\mathbf{I}_\ell)^{\mathsf{T}} \otimes \mathbf{G}) + \mathbf{e}^{\mathsf{T}}, \mathbf{W}, \mathbf{R}\right) \overset{c}{\approx} \left(\mathbf{B}, \mathbf{c}^{\mathsf{T}}, \mathbf{W}, \mathbf{R}\right),$$

where $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times \ell m}$, $\mathbf{R} \leftarrow D_{\mathbb{Z},\sigma}^{m \times \ell m}$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^{\ell^2 m}$, and $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell^2 m}$. Wee then shows that an analogous matrix commitment scheme (satisfying the properties in Lemma 3.8) where the public parameters are given by $\mathsf{pp}_{\mathsf{com}} = (\hat{\mathbf{B}}, \mathbf{W}, \mathbf{R})$, where $\mathbf{W} \in \mathbb{Z}_q^{n \times 2m^3}$, $\mathbf{R} \in D_{\mathbb{Z},\sigma}^{m \times 2m^3}$, and

$$\hat{\mathbf{B}} := \mathbf{W}(\mathbf{I}_{2m^2 \otimes \mathbf{R}} + \mathrm{vec}(\mathbf{I}_{2m^2})^{\mathsf{T}} \otimes \mathbf{G}) \in \mathbb{Z}_q^{n \times 4m^5}. \tag{4.1}$$

Then, the decomposed LWE assumption (with dimension $2m^2$ and width parameter σ) asserts that

$$\left(\mathsf{pp}_{\mathsf{com}}, \mathbf{s}^{\mathsf{T}}\hat{\mathbf{B}} + \mathbf{e}^{\mathsf{T}}\right) \overset{c}{\approx} \left(\mathsf{pp}_{\mathsf{com}}, \mathbf{c}^{\mathsf{T}}\right),$$

where $\mathsf{pp}_{\mathsf{com}} = (\hat{\mathbf{B}}, \mathbf{W}, \mathbf{R})$, $\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times 2m^3}$, $\mathbf{R} \leftarrow D_{\mathbb{Z},\sigma}^{m \times 2m^3}$, $\hat{\mathbf{B}} \in \mathbb{Z}_q^{n \times 4m^5}$ as in Eq. (4.1), $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^{4m^5}$, and $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{4m^5}$. As such, we can substitute this alternative instantiation for the matrix commitment in Construction 4.2 to obtain a distributed broadcast encryption scheme from the decomposed LWE assumption with dimension $2m^2$ and width parameter σ. Compared to the basic instantiation from Construction 4.2, this has the following advantages and disadvantages:

- **Transparent setup:** The public parameters $\mathsf{pp}_{\mathsf{com}}$ for the matrix commitment scheme in the above instantiation can be described by $\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times 2m^3}$ and $\mathbf{R} \leftarrow D_{\mathbb{Z},\sigma}^{m \times 2m^3}$. These can be derived from a uniform random string. The remaining components of the public key in Construction 4.2 is a uniform random matrix $\mathbf{A}$ and a uniform random vector $\mathbf{p}$. Thus, this yields a construction with a *transparent* setup process. The construction based on succinct LWE requires including the succinct LWE trapdoor $\mathbf{T}$ as part of the commitment parameters, which results in the need for a structured string.

- **Larger parameters:** A downside of using decomposed LWE instead of succinct LWE is the matrix $\hat{\mathbf{B}} \in \mathbb{Z}_q^{n \times 4m^5}$ now has width $4m^5$ (compared to $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ in Construction 4.2). Thus, the ciphertexts are longer in the modified construction. Note though that this is a polynomial blowup in the parameter size (since $m = \mathsf{poly}(\lambda, \log N)$).

5 Key-Policy Registered Attribute-Based Encryption

In this section, we show how to construct a registered key-policy ABE for general circuits from the succinct LWE assumption in the random oracle model. Our construction combines ideas from the recent registered ABE construction from [CHW25] with our distributed broadcast encryption scheme from Section 4.

Slotted registered ABE. Similar to prior works on registered ABE [HLWW23, FWW23, ZZGQ23, GLWW24, AT24, CHW25], we focus on the simpler "slotted" primitive introduced in [HLWW23]. In a slotted registered ABE scheme, there is a *fixed* number of users N and each user is associated with a specific slot index. Instead of users joining the system at arbitrary times, there is instead an aggregation algorithm that takes all N keys (together with their associated policies) and aggregates them into a single succinct master public key. The work of [HLWW23] show that a slotted registered ABE scheme implies the normal notion that supports dynamic registrations with only polylogarithmic overhead (using a standard powers-of-two transformation implicit in earlier works such as [GHMR18]). In this work, we focus exclusively on the slotted primitive because it is simpler to describe and construct.

Definition 5.1 (Slotted Key-Policy Registered ABE [HLWW23, adapted])**.** Let λ be a security parameter and τ be a policy-family parameter. Let $\mathcal{X} = \{\mathcal{X}_\tau\}_{\tau \in \mathbb{N}}$ be a family of attributes and $\mathcal{F} = \{\mathcal{F}_\tau\}_{\tau \in \mathbb{N}}$ be a set of policies (where each $f \in \mathcal{F}_\tau$ is a function $f \colon \mathcal{F}_\tau \to \{0, 1\}$). Throughout this work, we adopt the convention that an attribute $x \in \mathcal{X}_\tau$ satisfies a policy $f \in \mathcal{F}_\tau$ if $f(x) = 0$ (and does not satisfy the policy if $f(x) = 1$). A slotted key-policy registered ABE scheme with attribute space $\mathcal{X}$ and policy space $\mathcal{F}$ is a tuple of efficient algorithms $\Pi_{\mathsf{sRABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^\tau, N) \to \mathsf{pp}$: On input the security parameter λ, the policy-family parameter τ, and the number of slots N (in *binary*), the setup algorithm outputs the public parameters pp. We assume that pp implicitly defines 1^λ, 1^τ, and N.
- $\mathsf{KeyGen}(\mathsf{pp}, i, f) \to (\mathsf{pk}, \mathsf{sk})$: On input the public parameters pp, a slot index $i \in [N]$ and a function $f \in \mathcal{F}_\tau$, the key-generation algorithm outputs a public key pk and a secret key sk.
- $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) \to b$: On input the public parameters pp, a slot index $i \in [N]$, a function $f \in \mathcal{F}_\tau$, and a public key pk, the validity-checking algorithm outputs a bit $b \in \{0, 1\}$.
- $\mathsf{Aggregate}(\mathsf{pp}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N)) \to (\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N)$: On input the common reference string pp, a collection of N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, and their respective functions $f_1, \ldots, f_N \in \mathcal{F}_\tau$, the aggregation algorithm outputs a master public key mpk and a collection of helper decryption keys $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_N$. This algorithm is deterministic.
- $\mathsf{Encrypt}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$: On input the master public key mpk, an attribute $x \in \mathcal{X}_\tau$, and a message $\mu \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{hsk}, f, x, \mathsf{ct}) \to \mu$: On input the secret key sk, a helper decryption key hsk, a policy $f \in \mathcal{F}_\tau$, an attribute $x \in \mathcal{X}_\tau$, and a ciphertext ct, the decryption algorithm outputs a message $\mu \in \{0, 1\}$.

Moreover, Π_{sRABE} should satisfy the following properties:

- **Completeness:** For all $\lambda, \tau \in \mathbb{N}$, all $N \leq 2^\lambda$, all indices $i \in [N]$, and all policies $f \in \mathcal{F}_\tau$,
$$\Pr\left[\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) = 1 : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\tau, N) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i, f) \end{array}\right] = 1.$$
- **Correctness:** We say Π_{sRABE} is correct if for all λ, τ, all $N \leq 2^\lambda$, all indices $i \in [N]$, all policies $f_i \in \mathcal{F}_\tau$, all attributes $x \in \mathcal{X}_\tau$ where $f_i(x) = 0$, all pp in the support of $\mathsf{Setup}(1^\lambda, 1^\tau, N)$, all $\{(j, f_j, \mathsf{pk}_j)\}_{j \neq i}$ where $\mathsf{IsValid}(\mathsf{pp}, j, f_j, \mathsf{pk}_j) = 1$, and all messages $\mu \in \{0, 1\}$,
$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{hsk}_i, f, x, \mathsf{ct}) = \mu : \enspace \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, x, \mu) \enspace \right] = 1,$$
where $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) = \mathsf{Aggregate}(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$.
- **Compactness:** There exists a polynomial $\mathsf{poly}(\cdot, \cdot)$ such that for all $\lambda, \tau \in \mathbb{N}$, all $N \leq 2^\lambda$, all pp in the support of $\mathsf{Setup}(1^\lambda, 1^\tau, N)$, all $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N)$ where $\mathsf{IsValid}(\mathsf{pp}, i, f_i, \mathsf{pk}_i) = 1$ for all $i \in [N]$, and all $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N)$ in the support of $\mathsf{Aggregate}(\mathsf{pp}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$, it holds that
$$|\mathsf{mpk}| \leq \mathsf{poly}(\lambda, \tau) \quad \text{and} \quad \forall i \in [N] : |\mathsf{hsk}_i| \leq \mathsf{poly}(\lambda, \tau).$$

- **Security:** Let $b \in \{0,1\}$ be a bit. For a policy-family parameter $\tau \in \mathbb{N}$, a slot number N, and an adversary $\mathcal{A}$, we define the following security game between $\mathcal{A}$ and a challenger:
  - **Setup phase:** The challenger begins by sampling $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\tau, N)$ and gives pp to $\mathcal{A}$. The challenger also initializes a counter $\mathsf{ctr} = 0$ and an (empty) dictionary D to keep track of key-generation queries.
  - **Query phase:** Adversary $\mathcal{A}$ can now issue the following queries:
    - **Key-generation query:** In a key-generation query, the adversary specifies a slot index $i \in [N]$ and a function $f \in \mathcal{F}_\tau$. The challenger increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$ and samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i, f)$. The challenger replies to $\mathcal{A}$ with $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}})$ to $\mathcal{A}$ and then adds the mapping $\mathsf{ctr} \mapsto (i, f, \mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}})$ to D.
    - **Corruption query:** In a corruption query, the adversary specifies an index $1 \le c \le \mathsf{ctr}$. In response, the challenger looks up the tuple $(i', f', \mathsf{pk}', \mathsf{sk}') = \mathsf{D}[c]$ and replies to $\mathcal{A}$ with sk'.
  - **Challenge phase:** Algorithm $\mathcal{A}$ now outputs a list of tuples $(c_1, f_1, \mathsf{pk}_1), \ldots, (c_N, f_N, \mathsf{pk}_N)$ where either $c_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $c_i = \perp$ to reference a key outside this set. The adversary also specifies a challenge attribute $x^* \in \mathcal{X}_\tau$. The challenger then checks the following:
    - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', f', \mathsf{pk}', \mathsf{sk}')$ and checks that $i = i'$. If not, the challenger outputs 0. Otherwise, the challenger sets $\mathsf{pk}_i^* = \mathsf{pk}'$ and $f_i^* = f'$. In addition, if the adversary issued a "corruption" query on index c_i, then the challenger additionally checks that $f'(x^*) = 1$ and outputs 0 if not.
    - If $c_i = \perp$, then the challenger checks that $\mathsf{IsValid}(\mathsf{pp}, i, f_i, \mathsf{pk}_i) = 1$ and $f_i(x^*) = 1$. If so, then the challenger sets $\mathsf{pk}_i^* = \mathsf{pk}_i$ and $f_i^* = f_i$. Otherwise, the challenger outputs 0.

    The challenger computes $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) = \mathsf{Aggregate}(\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*) \ldots, (\mathsf{pk}_N^*, f_N^*))$ and replies with the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, x^*, b)$.
  - **Output phase:** At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment. If $\mathcal{A}$ aborts before this point, then the output of the experiment is 0.

  We say that a slotted registered ABE scheme is secure if for all polynomials $\tau = \tau(\lambda)$ and $N = N(\lambda)$, and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

  $$|\Pr[b' = 1 : b = 0] - \Pr[b' = 1 : b = 1]| = \mathsf{negl}(\lambda)$$

  in the above security game.

Relaxed security notions. We now describe two relaxations of the security definition in Definition 5.1.

Definition 5.2 (Attribute-Selective Security)**.** We say a slotted registered ABE scheme Π_{sRABE} satisfies attribute-selective security if the adversary must declare its challenger attribute $x^* \in \mathcal{X}_\tau$ at the beginning of the setup phase (before seeing pp).

Definition 5.3 (Security without Corruptions)**.** We say a slotted registered ABE scheme Π_{sRABE} satisfies security without corruptions if the adversary in the security game of Definition 5.1 is not allowed to make corruption queries.

Remark 5.4 (Relationship between Definitions)**.** While Definitions 5.2 and 5.3 relax the main security notion in Definition 5.1, there are generic ways to transform a scheme satisfying the relaxed notions of security into one that satisfies full security:

- First, if Π_{sRABE} satisfies attribute-selective security, we can obtain an adaptively-secure scheme (where the attribute is chosen in the challenge phase) via standard complexity leveraging [BB04]. Using complexity leveraging will require assuming sub-exponential hardness, and moreover, the size of the scheme parameters will scale with the attribute length.

- If Π_{sRABE} satisfies security without corruptions, we can apply the generic transformation from [FWW23] to obtain a scheme with security against adversaries that can make corruption queries in the random oracle model. Since our base registered ABE scheme is already in the random oracle model, this transformation is essentially free (the transformation itself only incurs constant overhead over the base scheme).

5.1 Additional Building Blocks

Similar to the registered ABE scheme from [CHW25], our construction also relies on simulation-sound extractable NIZK arguments, an explainable discrete Gaussian sampler, as well as a Gaussian preimage smudging lemma. We review these notions below:

Simulation-sound extractable NIZKs. The first primitive we require is a simulation-sound extractable non-interactive zero-knowledge (NIZK) argument for NP [BFM88, FLS90, Sah99, DDO+01]. Similar to [CHW25], we will use the NIZK argument of knowledge to extract secret keys in the security analysis of the registered ABE scheme. We give the definition below (taken mostly verbatim from [CHW25]):

Definition 5.5 (Simulation-Sound Extractable NIZK)**.** A simulation-sound extractable NIZK argument Π_{NIZK} for NP is a tuple of efficient algorithms $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{TrapSetup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim}, \mathsf{Extract})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$: On input the security parameter λ, the setup algorithm outputs a common reference string crs.
- $\mathsf{TrapSetup}(1^\lambda) \to (\mathsf{crs}, \mathsf{td})$: On input the security parameter λ, the trapdoor setup algorithm outputs a common reference string crs and a trapdoor td.
- $\mathsf{Prove}(\mathsf{crs}, C, x, w) \to \pi$: On input the common reference string crs, a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$, the prove algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, C, x, \pi) \to b$: On input the common reference string crs, a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.
- $\mathsf{Sim}(\mathsf{td}, C, x) \to \pi$: On input the trapdoor td, a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, and a statement $x \in \{0,1\}^n$, the simulation algorithm outputs a proof π.
- $\mathsf{Extract}(\mathsf{td}, C, x, \pi) \to w$: On input the trapdoor td, a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, the extraction algorithm outputs a witness $w \in \{0,1\}^h$ (or a special symbol $\perp$).

We require that Π_{NIZK} satisfy the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$, all Boolean circuits $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, all statements $x \in \{0,1\}^n$ and witnesses $w \in \{0,1\}^h$ where $C(x,w) = 1$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, C, x, \pi) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, x, w) \end{array}\right] = 1.$$

- **Zero-knowledge:** For a security parameter λ, an adversary $\mathcal{A}$, and a bit $b \in \{0,1\}$, we define the zero-knowledge security game as follows:
  - If $b = 0$, the challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and if $b = 1$, the challenger samples $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\lambda)$. The challenger gives crs to $\mathcal{A}$.
  - Algorithm $\mathcal{A}$ can now make adaptive queries of the form (C, x, w), where $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ is a Boolean circuit, $x \in \{0,1\}^n$ is a statement, and $w \in \{0,1\}^h$ is a witness.
    - The challenger first checks if $C(x,w) = 1$. If not, the challenger responds with $\perp$.
    - Otherwise, if $b = 0$, the challenger replies with $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, C, x, w)$. If $b = 1$, the challenger replies with $\pi \leftarrow \mathsf{Sim}(\mathsf{td}, C, x)$.
  - After $\mathcal{A}$ is finished making queries, it outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

  We say that Π_{NIZK} satisfies computational zero-knowledge if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda)$ in the zero-knowledge security game.

- **Simulation extractability:** For a security parameter λ, and an adversary $\mathcal{A}$, we define the simulation extractability games as follows:
  - The challenger starts by sampling $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\lambda)$ and gives crs to $\mathcal{A}$. The challenger also initializes an (empty) list Q.
  - Algorithm $\mathcal{A}$ can now make adaptive queries (C, x) where $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ is a Boolean circuit and $x \in \{0,1\}^n$ is a statement. The challenger replies with $\pi \leftarrow \mathsf{Sim}(\mathsf{td}, C, x)$ and adds (C, x, π) to Q.
  - After $\mathcal{A}$ is finished making queries, it outputs a Boolean circuit $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, a statement $x \in \{0,1\}^n$, and a proof π.
  - The challenger computes $w = \mathsf{Extract}(\mathsf{td}, C, x, \pi)$ and outputs $b' = 1$ if $\mathsf{Verify}(\mathsf{crs}, C, x, \pi) = 1$, $(C, x, \pi) \notin Q$ and $C(x, w) = 0$. Otherwise, the challenger outputs $b' = 0$.

  We say that Π_{NIZK} satisfies simulation extractability if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[b' = 1] = \mathsf{negl}(\lambda)$ in the simulation extractability game.

The work of [DDO+01] show how to construct a simulation-sound extractable NIZK for NP from any NIZK for NP together with a public-key encryption scheme (and a one-time signature scheme, which is implied by public-key encryption). Both NIZKs for NP [PS19, Wat24, WWW25, BCD+25] and public-key encryption [Reg05] are known from the plain LWE assumption.

Remark 5.6 (On Semi-Malicious Security)**.** In our registered ABE construction (Construction 5.11), each user's public key includes a NIZK proof of knowledge of the randomness used to sample the key. Essentially, the NIZK serves to provide a proof of well-formedness for public keys. A natural question is whether we can provide a more modular description where we first show that the registered ABE scheme without NIZK proofs of well-formedness is secure against *semi-malicious* adversaries. In this model, when the adversary specifies a public key in the challenge phase, it must also reveal the key-generation randomness used to sample the key. We can then upgrade a scheme with semi-malicious security to one with full security by requiring public keys to include a NIZK proof of knowledge of the key-generation randomness; indeed, such a transformation was recently described in [LWW25] in the context of multi-authority registered ABE. Unfortunately, this modular approach does not apply in our setting (for the same reason it did not apply in [CHW25]). This is because in the security proof, the reduction algorithm needs to know the key-generation randomness associated with adversarially-chosen public keys when it is simulating *random oracle queries* (before the challenge phase). It is not meaningful to restrict the adversary to only query the random oracle on inputs that are well-formed. As a result, our current proof strategy critically relies on the ability to extract the key-generation randomness from *any* well-formed public key (as opposed to only the subset of public keys chosen during the challenge phase). We refer to the proof of Theorem 5.14 for the full details.

Explainable Gaussian sampling. Our construction relies on the re-randomization technique introduced in [CHW25]. Namely, the key-aggregation algorithm will sample and register a "dummy key," and the randomness for sampling the dummy key will be derived from the random oracle. In the security analysis, the reduction algorithm programs the aggregation randomness in order to correctly simulate the challenge ciphertext. To implement this, the work of [CHW25] relies on an *explainable* discrete Gaussian sampler for sampling from the distribution $\mathbf{A}_\sigma^{-1}(\mathbf{z})$. Specifically, there is an Explain algorithm that takes a preimage $\mathbf{y} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{z})$ and outputs the randomness (for the sampler algorithm) that would produce $\mathbf{x}$. The work of [CHW25] observed that the Gentry-Peikert-Vaikuntanathan preimage sampling algorithm [GPV08] is explainable (when instantiated with an explainable discrete Gaussian sampler over the integers [LW22]). We give the formal definition and the required properties from [CHW25] below:

Definition 5.7 (Explainable Discrete Gaussian Preimage Sampler [CHW25, Definition 4.1])**.** Let λ be a security parameter, and n, m, q be lattice parameters. A $(\rho, \sigma_{\mathsf{loss}})$-explainable discrete Gaussian preimage sampler Π_{DGS} with randomness length ρ and width parameter σ_{loss} is a pair of efficient algorithms $\Pi_{\mathsf{DGS}} = (\mathsf{SamplePre}, \mathsf{Explain})$ with the following syntax:

- $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{z}, \sigma; r)$: On input the security parameter λ, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$, a target vector $\mathbf{z} \in \mathbb{Z}_q^n$, a width parameter $\sigma > 0$, and randomness $r \in \{0,1\}^\rho$, the preimage sampling algorithm outputs a vector $\mathbf{y} \in \mathbb{Z}_q^m$.

- $\mathsf{Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{T}, \mathbf{z}, \mathbf{y}, \sigma) \to r$: On input the security parameter λ, the precision parameter κ, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$, a target vector $\mathbf{z} \in \mathbb{Z}_q^n$, a preimage $\mathbf{y} \in \mathbb{Z}_q^m$, and a width parameter $\sigma > 0$, the explain algorithm outputs a string $r \in \{0,1\}^\rho$.

Moreover, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, trapdoors $\mathbf{T} \in \mathbb{Z}_q^{m \times m}$ where $\mathbf{AT} = \mathbf{G}$, all targets $\mathbf{z} \in \mathbb{Z}_q^n$ where $\|\mathbf{z}\| \leq 2^\lambda$, and all width parameters σ where $\|\mathbf{T}\| \cdot \sigma_{\text{loss}} \leq \sigma \leq 2^\lambda$, the following two properties hold:

- **Correctness:** The following distributions are statistically indistinguishable:
$$\{\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)\} \quad \text{and} \quad \{\mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})\}.$$
Moreover, for all $\mathbf{x}$ in the support of $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$, we have $\mathbf{Ax} = \mathbf{y}$.
- **Explainability:** For all $\kappa \in \mathbb{N}$, the statistical distance between the following distributions is bounded by $1/\kappa + \mathsf{negl}(\lambda)$:
  - $\mathcal{D}_{\mathsf{SamplePre}}$: Sample $r \xleftarrow{R} \{0,1\}^\rho$ and $\mathbf{y} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{z}, \sigma; r)$. Output $(\mathbf{y}, r)$.
  - $\mathcal{D}_{\mathsf{Explain}}$: Sample $r' \xleftarrow{R} \{0,1\}^\rho$ and $\mathbf{y} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{z}, \sigma; r')$ and $r \leftarrow \mathsf{Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{T}, \mathbf{z}, \mathbf{y}, \sigma)$. Output $(\mathbf{y}, r)$.

Theorem 5.8 (Explainable Discrete Gaussian Preimage Sampler [CHW25, Theorem 4.2])**.** *Let λ be a security parameter, and n, m, q be lattice parameters. There exists an explicit $(\rho, \sigma_{\text{loss}})$-explainable discrete Gaussian preimage sampler where $\rho = \mathsf{poly}(\lambda, n, m, \log q)$ and $\sigma_{\text{loss}} = 18m^{3/2}\log(m\lambda)\log\log q$.*

Explainable re-randomizer. Let $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ be the public parameters for a matrix commitment scheme. To apply the re-randomization technique from [CHW25] to prove security of our registered ABE scheme, we require an explainable sampler for sampling from the following distribution:
$$\left\{ (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) : \begin{array}{c} \mathbf{C} \xleftarrow{R} \mathbb{Z}_q^{n \times m} \\ \mathbf{y}_i \leftarrow \mathbf{B}_\sigma^{-1}(-\mathbf{C}\mathbf{v}_i) \end{array} \right\}$$
where $\mathbf{V} = [\mathbf{v}_1 \mid \cdots \mathbf{v}_N] = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}, 1^N)$. To simplify the exposition of our main construction, we abstract out the properties we require and then show how to build an explainable sampler for this distribution below:

Theorem 5.9 (Explainable Re-randomizer)**.** *Let λ be a security parameter, n, m, q be lattice parameters, and ρ be a randomness parameter. There exists a pair of efficient algorithms* $(\mathsf{Sample}, \mathsf{Explain})$ *with the following syntax:*

- $\mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r) \to (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$*: On input the public parameters* pp*, the security parameter λ, the dimension N, a width parameter $\sigma > 0$, and randomness $r \in \{0,1\}^\rho$, output $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^m$.*
- $\mathsf{Explain}(\mathsf{pp}, 1^\lambda, 1^\kappa, (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N), \sigma) \to r$*: On input the public parameters* pp*, a security parameter λ, a precision parameter κ, a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$, vectors $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^m$, and a width parameter $\sigma > 0$, output randomness $r \in \{0,1\}^\rho$.*

There exists a polynomial $\rho = \mathsf{poly}(\lambda, N, n, m, \log q)$ such that for all $n \geq \lambda$ and $m \geq 2n\log q$, for all but a negligible fractions of matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, and all $\mathbf{W} \in \mathbb{Z}_q^{2m^2 n \times m}$, $\mathbf{T} \in \mathbb{Z}_q^{(2m^2+1)m \times 2m^3}$ where $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}] \cdot \mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$, all parameters $N \in \mathbb{N}$, and setting $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, the following properties hold:

- **Sampling distribution:** *For all width parameters $\|\mathbf{T}\| \cdot O(m^{12}N^3) < \sigma < 2^\lambda$, the statistical distance between the following distributions is bounded by* $\mathsf{negl}(\lambda)$*:*
  - $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r)$.
  - $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ *where* $\mathbf{C} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{y}_i \leftarrow \mathbf{B}_\sigma^{-1}(-\mathbf{C}\mathbf{v}_i)$*, and* $[\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N] = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}, 1^N)$.

Moreover, for all $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ *in the support of* $\mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma)$*, it holds that* $\mathbf{B}\mathbf{y}_i = -\mathbf{C}\mathbf{v}_i$ *and* $\|\mathbf{y}_i\| \le \sqrt{m}\sigma$*.*

- **Explainability:** *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all precision parameters* κ*, dimensions* $N \in \mathbb{N}$*, and width parameters* $\|\mathbf{T}\| \cdot O(m^{12}N^3) < \sigma < 2^\lambda$*, the statistical distance between the following distributions is bounded by* $1/\kappa + \mathsf{negl}(\lambda)$*:*
  - $\mathcal{D}_{\mathsf{Sample}}$*: Sample* $r \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ *and* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r)$*. Output* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N, r)$*.*
  - $\mathcal{D}_{\mathsf{Explain}}$*: Sample* $r' \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ *and* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r')$*. Then, sample the randomness* $r \leftarrow \mathsf{Explain}(\mathsf{pp}, 1^\lambda, 1^\kappa, (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N), \sigma)$*. Output* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N, r)$*.*

Proof. We define the algorithms as follows:

- $\mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r)$: On input the public parameters $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, the security parameter λ, the dimension $N \in \mathbb{N}$, a width parameter $\sigma > 0$, and randomness $r \in \{0,1\}^\rho$, the sampling algorithm proceeds as follows:
  1. Compute $\mathbf{V}_N = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}, 1^N)$. Parse $\mathbf{V}_N = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N]$ where each $\mathbf{v}_i \in \mathbb{Z}_q^m$.
  2. Let $\mathbf{G}_N = \mathbf{I}_N \otimes \mathbf{G} \in \mathbb{Z}_q^{nN \times mN}$, and let $\mathbf{m}_1, \ldots, \mathbf{m}_{mN} \in \mathbb{Z}_q^{nN}$ be the columns of $\mathbf{G}_N$. Let $\mathbf{M}_i \in \mathbb{Z}_q^{n \times N}$ be the matrix where $\mathrm{vec}(\mathbf{M}_i) = \mathbf{m}_i$. Then, for each $i \in [mN]$, compute the commitment $\mathbf{C}_i = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_i) \in \mathbb{Z}_q^{n \times m}$ and the opening $\mathbf{Z}_i = \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_i) \in \mathbb{Z}_q^{m \times N}$.
  3. Define the vector $\mathbf{r}_i \in \mathbb{Z}_q^{mN+k}$
  $$\mathbf{r}_i = \begin{bmatrix} \mathrm{vec}(\mathbf{Z}_i) \\ \mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) \end{bmatrix} \in \mathbb{Z}_q^{mN+k}.$$
  Let $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_{mN}] \in \mathbb{Z}_q^{(mN+k) \times mN}$.
  4. Let $k = nm \lceil \log q \rceil$ and let $\mathbf{D} = [\mathbf{D}_1 \mid \cdots \mid \mathbf{D}_k] \in \mathbb{Z}_q^{n \times mk}$ be the matrix where $\mathrm{vec}(\mathbf{D}_i)$ is the i^{th} column of $\mathbf{G}_{nm}$. Now define the matrix $\mathbf{A}$ as follows:
  $$\mathbf{A} = \left[\begin{array}{ccc|c} \mathbf{B} & & & \mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_1) \\ & \ddots & & \vdots \\ & & \mathbf{B} & \mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_N) \end{array}\right] \in \mathbb{Z}_q^{nN \times (mN+k)}. \tag{5.1}$$
  5. Sample
  $$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y}_0 \end{bmatrix} \leftarrow \mathsf{DGS.Sample}(1^\lambda, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \sigma; r),$$
  where $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^m$ and $\mathbf{y}_0 \in \mathbb{Z}_q^k$. If $\|\mathbf{y}_i\| > \sqrt{m}\sigma$ for any $i \in [N]$, then output $\mathbf{C} = \mathbf{0}^{n \times m}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_N = \mathbf{0}^m$. Otherwise, output the matrix $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$ and the vectors $\mathbf{y}_1, \ldots, \mathbf{y}_N$.
- $\mathsf{Explain}(\mathsf{pp}, 1^\lambda, 1^\kappa, (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N), \sigma)$: On input the public parameters $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, the precision parameter κ, a target $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$, and a width parameter σ, the explain algorithm proceeds as follows:
  - Compute $\mathbf{y}_0 \leftarrow \mathsf{SamplePre}(\mathbf{G}_{nm}, \mathbf{I}_k, \mathrm{vec}(\mathbf{C}), \sigma)$.
  - Let $\mathbf{y}^{\mathsf{T}} = [\mathbf{y}_1^{\mathsf{T}} \mid \cdots \mid \mathbf{y}_N^{\mathsf{T}} \mid \mathbf{y}_0^{\mathsf{T}}]^{\mathsf{T}}$. Compute $\mathbf{A}, \mathbf{R}$ as in $\mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma)$. By construction, $\mathbf{A}, \mathbf{R}$ is a deterministic function of pp. Output $r \leftarrow \mathsf{DGS.Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \mathbf{y}, \sigma)$.

We now show that $(\mathsf{Sample}, \mathsf{Explain})$ satisfy the requirements in Theorem 5.9.

Sampling distribution. First, we argue that $\mathbf{AR} = \mathbf{I}_N \otimes \mathbf{G}$. By Remark 3.9, for all $i \in [mN]$, $\mathbf{C}_i \mathbf{V}_N = \mathbf{M}_i - \mathbf{B}\mathbf{Z}_i$. By Eq. (3.2), this means

$$\mathbf{m}_i = \mathrm{vec}(\mathbf{M}_i) = (\mathbf{I}_N \otimes \mathbf{B}) \cdot \mathrm{vec}(\mathbf{Z}_i) + (\mathbf{I}_N \otimes \mathbf{C}_i) \cdot \mathrm{vec}(\mathbf{V}_N). \tag{5.2}$$

Next, recall that $\mathbf{D} = [\mathbf{D}_1 \mid \cdots \mid \mathbf{D}_k]$ where $\mathbf{D}_i \in \mathbb{Z}_q^{n \times m}$. For any vector $\mathbf{x} \in \mathbb{Z}_q^k$, we have

$$\mathbf{D}(\mathbf{x} \otimes \mathbf{I}_m) = [\mathbf{D}_1 \mid \cdots \mid \mathbf{D}_k](\mathbf{x} \otimes \mathbf{I}_m) = \sum_{i \in [k]} x_i \mathbf{D}_i.$$

Then,

$$\mathrm{vec}(\mathbf{D}(\mathbf{x} \otimes \mathbf{I}_m)) = \mathrm{vec}\left(\sum_{i \in [k]} x_i \mathbf{D}_i \right) = \sum_{i \in [k]} x_i \cdot \mathrm{vec}(\mathbf{D}_i) = [\mathrm{vec}(\mathbf{D}_1) \mid \cdots \mid \mathrm{vec}(\mathbf{D}_k)] \cdot \mathbf{x} = \mathbf{G}_{nm} \cdot \mathbf{x}. \tag{5.3}$$

This means

$$\mathrm{vec}(\mathbf{C}_i) = \mathbf{G}_{nm} \cdot \mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) = \mathrm{vec}(\mathbf{D}(\mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) \otimes \mathbf{I}_m)).$$

Since $\mathbf{C}_i$ and $\mathbf{D}(\mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) \otimes \mathbf{I}_m)$ are matrices with the same dimension, we conclude that

$$\mathbf{C}_i = \mathbf{D}(\mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) \otimes \mathbf{I}_m). \tag{5.4}$$

Now,

$$\begin{aligned}
\mathbf{A}\mathbf{r}_i &= (\mathbf{I}_N \otimes \mathbf{B}) \cdot \mathrm{vec}(\mathbf{Z}_i) + \begin{bmatrix} \mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_1) \\ \vdots \\ \mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_N) \end{bmatrix} \mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) && \text{by Eq. (5.1)} \\
&= (\mathbf{I}_N \otimes \mathbf{B}) \cdot \mathrm{vec}(\mathbf{Z}_i) + \begin{bmatrix} \mathbf{D}(\mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) \otimes \mathbf{I}_m)\mathbf{v}_1 \\ \vdots \\ \mathbf{D}(\mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{C}_i)) \otimes \mathbf{I}_m)\mathbf{v}_N \end{bmatrix} && \text{by Eq. (3.1)} \\
&= (\mathbf{I}_N \otimes \mathbf{B}) \cdot \mathrm{vec}(\mathbf{Z}_i) + (\mathbf{I}_N \otimes \mathbf{C}_i)\mathrm{vec}(\mathbf{V}_N) && \text{by Eq. (5.4)} \\
&= \mathbf{m}_i && \text{by Eq. (5.2).}
\end{aligned}$$

We conclude that $\mathbf{AR} = [\mathbf{m}_1 \mid \cdots \mid \mathbf{m}_{mN}] = \mathbf{G}_N$, as required. By Remark 3.9, for all $i \in [mN]$,

$$\|\mathbf{Z}_i\| \leq O(\|\mathbf{T}\| \cdot m^7 \log q \log N).$$

Thus, $\|\mathbf{R}\| \leq O(\|\mathbf{T}\| \cdot m^7 \log q \log N)$. Let $\sigma'_{\mathrm{loss}} = 18(mN + k)^{3/2} \log((mN + k)\lambda) \log\log q \leq O(m^4 N^2)$. Then,

$$\|\mathbf{R}\| \cdot \sigma'_{\mathrm{loss}} \leq \|\mathbf{T}\| \cdot O(m^{12} N^3).$$

If $\|\mathbf{R}\| \cdot \sigma'_{\mathrm{loss}} \leq \|\mathbf{T}\| \cdot O(m^{12} N^3) \leq \sigma \leq 2^\lambda$, the distribution of $(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ output by $\mathsf{DGS.Sample}(1^\lambda, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \sigma; r)$ is statistically close to sampling

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y}_0 \end{bmatrix} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{0}^{nN}). \tag{5.5}$$

We now characterize the distribution of $(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N)$. We proceed by a hybrid argument:

- $\mathcal{D}_0$: Sample $(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ according to Eq. (5.5). Set $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$ and output $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ if for all $i \in [N]$, $\|\mathbf{y}_i\| \leq \sqrt{m}\sigma$. Otherwise, if $\|\mathbf{y}_i\| > \sqrt{m}\sigma$ for some $i \in [N]$, output $(\mathbf{0}^{n \times m}, \mathbf{0}^m, \ldots, \mathbf{0}^m)$.

- $\mathcal{D}_1$: Sample $\mathbf{y}_0 \leftarrow D^k_{\mathbb{Z},\sigma}$ and set $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$. Then, for $i \in [N]$, sample $\mathbf{y}_i \leftarrow \mathbf{B}^{-1}_\sigma(-\mathbf{C}\mathbf{v}_i)$. Output $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ if for all $i \in [N]$, $\|\mathbf{y}_i\| \le \sqrt{m}\sigma$. Otherwise, if $\|\mathbf{y}_i\| > \sqrt{m}\sigma$ for some $i \in [N]$, output $(\mathbf{0}^{n\times m}, \mathbf{0}^m, \ldots, \mathbf{0}^m)$.
- $\mathcal{D}_2$: Sample $\mathbf{C} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$ and for each $i \in [N]$, sample $\mathbf{y}_i \leftarrow \mathbf{B}^{-1}_\sigma(-\mathbf{C}\mathbf{v}_i)$. Output $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ if for all $i \in [N]$, $\|\mathbf{y}_i\| \le \sqrt{m}\sigma$. Otherwise, if $\|\mathbf{y}_i\| > \sqrt{m}\sigma$ for some $i \in [N]$, output $(\mathbf{0}^{n\times m}, \mathbf{0}^m, \ldots, \mathbf{0}^m)$.
- $\mathcal{D}_3$: Sample $\mathbf{C} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$ and for each $i \in [N]$, sample $\mathbf{y}_i \leftarrow \mathbf{B}^{-1}_\sigma(-\mathbf{C}\mathbf{v}_i)$. Output $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$. In particular, there is no additional check on the norms of $\mathbf{y}_1, \ldots, \mathbf{y}_N$.

As argued above, when $\|\mathbf{T}\| \cdot \sigma_{\text{loss}} \le \sigma \le 2^\lambda$, the output of $\mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r)$ is statistically close to $\mathcal{D}_0$. We now show that $\mathcal{D}_0$ is statistically close to $\mathcal{D}_3$:

- By Lemma 3.4, for all but a negligible fraction of matrices $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$, the distribution of $(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ is statistically close to sampling $\mathbf{y}_0 \leftarrow D_{\mathbb{Z},\sigma^k}$ and $\mathbf{y}_i \leftarrow \mathbf{B}^{-1}_\sigma(-\mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_i)\mathbf{y}_0)$. By Eq. (3.1), we can write
$$\mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_i)\mathbf{y}_0 = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)\mathbf{v}_i = \mathbf{C}\mathbf{v}_i.$$
This is the distribution in $\mathcal{D}_1$.
- By Eq. (5.3), we can write $\text{vec}(\mathbf{C}) = \text{vec}(\mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)) = \mathbf{G}_{nm} \cdot \mathbf{y}_0$. In $\mathcal{D}_1$, $\mathbf{y}_0 \leftarrow D^k_{\mathbb{Z},\sigma}$ so by Lemma 3.3, $\mathbf{G}_{nm} \cdot \mathbf{y}_0$ is statistically close to uniform over $\mathbb{Z}_q^{nm}$. Correspondingly, this means that the marginal distribution of $\mathbf{C}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$, as required.
- By Lemma 3.2, for all $i \in [N]$, we have $\Pr[\|\mathbf{y}_i\| \ge \sqrt{m}\sigma] \le O(2^{-m})$. Since $m \ge 2\lambda$ and $N \le 2^\lambda$, the probability that there exists $i \in [N]$ where $\|\mathbf{y}_i\| \ge \sqrt{m}\sigma$ is at most $2^\lambda/2^m \le 2^{-\lambda}$. Thus, with overwhelming probability, the vectors $\|\mathbf{y}_i\|$ sampled in $\mathcal{D}_2$ all satisfy $\|\mathbf{y}_i\| \le \sqrt{m}\sigma$, so $\mathcal{D}_2$ and $\mathcal{D}_3$ are statistically indistinguishable.

The claim now follows by a hybrid argument. Finally, the worst-case guarantee that $\mathbf{B}\mathbf{y}_i = -\mathbf{C}\mathbf{v}_i$ follows immediately by the worst-case guarantee of Π_{DGS}. Namely, the $(\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ output by $\mathsf{DGS.Sample}$ always satisfies $\mathbf{y}_i = \mathbf{B}(-\mathbf{D}(\mathbf{I}_k \otimes \mathbf{v}_i)\mathbf{y}_0) = \mathbf{B}(-\mathbf{C}\mathbf{v}_i)$. The norm constraint is ensured by construction.

Explainability. We proceed via a hybrid argument. We start by defining a sequence of distributions:

- $\mathcal{D}_0$: This is distribution $\mathcal{D}_{\mathsf{Sample}}$. Namely, sample $r \xleftarrow{\text{R}} \{0,1\}^\rho$, and $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma; r)$. Concretely, compute $\mathbf{A}, \mathbf{R}$ as in $\mathsf{Sample}(\mathsf{pp}, 1^\lambda, 1^N, \sigma)$. Then sample
$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y}_0 \end{bmatrix} \leftarrow \mathsf{DGS.Sample}(1^\lambda, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \sigma; r),$$
where $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^m$ and $\mathbf{y}_0 \in \mathbb{Z}_q^k$. Let $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$. If there exists $i \in [N]$ where $\|\mathbf{y}_i\| > \sqrt{m}\sigma$, then set $\mathbf{C} = \mathbf{0}^{n\times m}$ and $\mathbf{y}_1, \ldots, \mathbf{y}_N = \mathbf{0}^m$. Output $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N, r)$.
- $\mathcal{D}_1$: Same as $\mathcal{D}_0$ except the distribution no longer checks the norm constraints $\|\mathbf{y}_i\| > \sqrt{m}\sigma$.
- $\mathcal{D}_2$: Same as $\mathcal{D}_1$ except after sampling $\mathbf{y}$ and $\mathbf{C}$, sample $r' \leftarrow \mathsf{DGS.Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \mathbf{y}, \sigma)$ and output $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N, r')$.
- $\mathcal{D}_3$: Same as $\mathcal{D}_2$, except sample $\mathbf{y} \leftarrow \mathbf{A}^{-1}_\sigma(\mathbf{0}^{nN})$.
- $\mathcal{D}_4$: Same as $\mathcal{D}_3$, except instead sample $\mathbf{y}_0 \leftarrow D^k_{\mathbb{Z},\sigma}$. Then set $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$ and sample $\mathbf{y}_i \leftarrow \mathbf{B}^{-1}_\sigma(-\mathbf{C}\mathbf{v}_i)$ for each $i \in [N]$. In this experiment, the vector $\mathbf{y}$ is still defined as
$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y}_0 \end{bmatrix}.$$

- $\mathcal{D}_5$: Same as $\mathcal{D}_4$, except sample $\mathbf{C} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ and $\mathbf{y}_0 \leftarrow (\mathbf{G}_{nm})_\sigma^{-1}(\mathrm{vec}(\mathbf{C}))$.
- $\mathcal{D}_6$: Same as $\mathcal{D}_5$ except sample $\mathbf{y}_0' \leftarrow D_{\mathbb{Z},\sigma}^k$ and set $\mathbf{C} = \mathbf{D}(\mathbf{y}_0' \otimes \mathbf{I}_m)$. Next, sample $\mathbf{y}_0 \leftarrow (\mathbf{G}_{nm})_\sigma^{-1}(\mathrm{vec}(\mathbf{C}))$ and for each $i \in [N]$, sample $\mathbf{y}_i \leftarrow \mathbf{B}_\sigma^{-1}(-\mathbf{C}\mathbf{v}_i)$.
- $\mathcal{D}_7$: Same as $\mathcal{D}_6$ except sample
$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y}_0' \end{bmatrix} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{0}^{nN}).$$
Then set $\mathbf{C} = \mathbf{D}(\mathbf{y}_0' \otimes \mathbf{I}_m)$ and compute $\mathbf{y}_0 \leftarrow (\mathbf{G}_{nm})_\sigma^{-1}(\mathrm{vec}(\mathbf{C}))$. Let $\mathbf{y} = [\mathbf{y}_1^{\mathsf{T}} \mid \cdots \mid \mathbf{y}_N^{\mathsf{T}} \mid \mathbf{y}_0^{\mathsf{T}}]^{\mathsf{T}}$ and $r' \leftarrow \mathsf{DGS.Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \mathbf{y}, \sigma)$ as before.
- $\mathcal{D}_8$: Same as $\mathcal{D}_7$ except sample
$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{y}_0' \end{bmatrix} \leftarrow \mathsf{DGS.Sample}(1^\lambda, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \sigma).$$
Set $\mathbf{C} = \mathbf{D}(\mathbf{y}_0' \otimes \mathbf{I}_m)$ and compute $\mathbf{y}_0 \leftarrow (\mathbf{G}_{nm})_\sigma^{-1}(\mathrm{vec}(\mathbf{C}))$. Let $\mathbf{y} = [\mathbf{y}_1^{\mathsf{T}} \mid \cdots \mid \mathbf{y}_N^{\mathsf{T}} \mid \mathbf{y}_0^{\mathsf{T}}]^{\mathsf{T}}$ and $r' \leftarrow \mathsf{DGS.Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{R}, \mathbf{0}^{nN}, \mathbf{y}, \sigma)$ as before.
- $\mathcal{D}_9$: Same as $\mathcal{D}_8$ except sample $\mathbf{y}_0 \leftarrow \mathsf{SamplePre}(\mathbf{G}_{nm}, \mathbf{I}_k, \mathrm{vec}(\mathbf{C}), \sigma)$. This is the distribution $\mathcal{D}_{\mathsf{Explain}}$.

We now analyze each adjacent pair of distributions. As in the analysis of the sampling distribution, we assume that $\|\mathbf{R}\| \cdot \sigma'_{\mathsf{loss}} \le \|\mathbf{T}\| \cdot O(m^{12}N^3) \le \sigma \le 2^\lambda$, where $\sigma'_{\mathsf{loss}} = 18(mN+k)^{3/2} \log((mN+k)\lambda) \log\log q \le O(m^4 N^2)$.

- Distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are statistically indistinguishable by the above analysis on the sampling distribution.
- Distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ have statistical distance $1/\kappa + \mathsf{negl}(\lambda)$ by the explainability property of Π_{DGS}.
- Distributions $\mathcal{D}_2$ and $\mathcal{D}_3$ are statistically indistinguishable by the correctness property of Π_{DGS}.
- Distributed $\mathcal{D}_3$ and $\mathcal{D}_4$ are statistically indistinguishable by Lemma 3.4. Specifically, for all but a negligible fraction of matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, the distribution of $\mathbf{y} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{0}^{nN})$ is statistically close to sampling $\mathbf{y}_0 \leftarrow D_{\mathbb{Z},\sigma^k}$ and $\mathbf{y}_i \leftarrow \mathbf{B}_\sigma^{-1}(-\mathbf{C}\mathbf{v}_i)$ where $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$. The former sampling procedure corresponds to $\mathcal{D}_2$ while the latter corresponds to $\mathcal{D}_3$.
- Distributions $\mathcal{D}_4$ and $\mathcal{D}_5$ are statistically indistinguishable by Lemma 3.3. Specifically, by Eq. (5.3), we can write $\mathrm{vec}(\mathbf{C}) = \mathrm{vec}(\mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)) = \mathbf{G}_{nm} \cdot \mathbf{y}_0$. By Lemma 3.3, the following distributions are statistically indistinguishable:
  - $(\mathbf{C}, \mathbf{y}_0)$ where $\mathbf{y}_0 \leftarrow D_{\mathbb{Z},\sigma}^k$ and $\mathbf{C} = \mathbf{D}(\mathbf{y}_0 \otimes \mathbf{I}_m)$.
  - $(\mathbf{C}, \mathbf{y}_0)$ where $\mathbf{C} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{y}_0 \leftarrow (\mathbf{G}_{nm})_\sigma^{-1}(\mathrm{vec}(\mathbf{C}))$.

  The left distribution corresponds to $\mathcal{D}_4$ while the right distribution corresponds to $\mathcal{D}_5$.
- Distributions $\mathcal{D}_5$ and $\mathcal{D}_6$ are also statistically indistinguishable by Lemma 3.3 (via the same argument as in the previous case).
- Distributions $\mathcal{D}_6$ and $\mathcal{D}_7$ are statistically indistinguishable by Lemma 3.4 (via the same argument as used to argue indistinguishability of $\mathcal{D}_2$ and $\mathcal{D}_3$).
- Distributions $\mathcal{D}_7$ and $\mathcal{D}_8$ are statistically indistinguishable by the correctness property of Π_{DGS}.

- Distributions $\mathcal{D}_8$ and $\mathcal{D}_9$ are statistically indistinguishable by Lemma 3.5.

By a hybrid argument, the statistical distance between $\mathcal{D}_0 \equiv \mathcal{D}_{\mathsf{Sample}}$ and $\mathcal{D}_9 \equiv \mathcal{D}_{\mathsf{Explain}}$ can be bounded by $1/\kappa + \mathsf{negl}(\lambda)$, as required. □

Gaussian preimage smudging. We will also need the following noise smudging lemma from [CHW25]:

Lemma 5.10 (Gaussian Preimage Smudging [CHW25, Theorem 4.3, adapted])**.** *Let n, m, q be lattice parameters such that $m \geq 2n \log q$ and q is prime. Then, for all but a q^{-n}-fraction of matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, for all vectors $\mathbf{y} \in \mathbb{Z}_q^n$ in the column-span of $\mathbf{A}$, all $\mathbf{z} \in \mathbb{Z}_q^m$, and all width parameters $\sigma > \lambda^{\omega(1)} \sqrt{m} \, \|z\|$, then the following distributions are statistically indistinguishable:*

$$\left\{\mathbf{A}_\sigma^{-1}(\mathbf{y} + \mathbf{A}\mathbf{z})\right\} \quad \textit{and} \quad \left\{\mathbf{A}_\sigma^{-1}(\mathbf{y}) + \mathbf{z}\right\}.$$

5.2 Key-Policy Registered ABE Construction

In this section, we give our construction of a key-policy registered ABE scheme that supports arbitrary (bounded-depth) Boolean circuits. Specifically, the scheme has short parameters and thus, can be used to obtain a registered ABE scheme that supports an arbitrary polynomial number of users. All previous registered ABE schemes [HLWW23, ZZGQ23, GLWW24, AT24, CHW25] that does not rely on indistinguishability obfuscation or witness encryption could only support an *a priori* bounded number of users.

Construction 5.11 (Key-Policy Slotted Registered ABE)**.** Let λ be a security parameter and τ be a policy-family parameter. We define the following quantities:

- Let (n, m, q, χ) be LWE parameters (which may be functions of λ, τ). Let $m' = n \lceil \log q \rceil$. Let $\sigma_{\mathsf{td}}, \sigma_{\mathsf{agg}} > 0$ be width parameters.
- Let $\ell = \ell(\tau)$ be the attribute length. Let $\mathcal{X} = \{\mathcal{X}_\tau\}_{\tau \in \mathbb{N}}$ where $\mathcal{X}_\tau = \{0,1\}^{\ell(\tau)}$ be the attribute space.
- Let $\mathcal{F}_\tau$ be the family of functions f that can be computed by a Boolean circuit of depth at most $d = d(\tau)$.
- Let λ_{DGS} be the security parameter for the explainable re-randomizer $(\mathsf{Sample}, \mathsf{Explain})$ from Theorem 5.9. Let $\rho = \mathsf{poly}(\lambda_{\mathsf{DGS}}, N, n, m, \log q)$ be the randomness length from Theorem 5.9.
- Let $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.TrapSetup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify}, \mathsf{NIZK.Sim}, \mathsf{NIZK.Extract})$ be a simulation-sound extractable NIZK argument for NP. Let C_{valid} be the Boolean circuit that takes a statement $(i, \mathbf{A}, \mathbf{d}, \mathbf{p}, \mathbf{B}, \mathbf{t})$, a witness $\mathbf{r}$, and outputs 1 if $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{A}\mathbf{G}^{-1}(\mathbf{d}) + \mathbf{p}$.[2]
- Let $H_1 \colon \mathbb{N} \to \mathbb{Z}_q^m$ and $H_2 \colon \{0,1\}^* \to \{0,1\}^\rho$ be hash functions (which we model as random oracles in the security analysis).

We construct a key-policy slotted registered ABE scheme $\Pi_{\mathsf{sRABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^\tau)$: On input the security parameter λ and the policy family parameter τ, sample

  $$\begin{aligned} \mathsf{crs}_{\mathsf{NIZK}} &\leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\ (\mathbf{B}, \mathbf{T}_\mathbf{B}) &\leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{2m^2 n \times m} \\ \mathbf{T} &\leftarrow \mathsf{SamplePre}\left([\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}], \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_\mathbf{B} \\ \mathbf{0} \end{smallmatrix}\right], \mathbf{I}_{2m^2} \otimes \mathbf{G}, \sigma_{\mathsf{td}}\right) \\ \mathbf{B}_0 &\xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}, \mathbf{p} \xleftarrow{\text{R}} \mathbb{Z}_q^n. \end{aligned}$$

  If $\|\mathbf{T}\| > \sqrt{m}\sigma_{\mathsf{td}}$, then set $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_\mathbf{B} \\ \mathbf{0} \end{smallmatrix}\right]$. Let $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and output the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$.

[2] Note that the circuit ignores the index i, but including it as part of the statement allows us to bind a proof to the specific index i (when appealing to simulation-sound extractability).

- $\mathsf{KeyGen}(\mathsf{pp}, i, f)$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$, where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, compute

$$\begin{aligned} \mathbf{d}_i &= H_1(i) \in \mathbb{Z}_q^n \\ \mathbf{V}_{\ell m} &= \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m}) \in \mathbb{Z}_q^{m \times \ell m} \\ \mathbf{A} &= -\mathbf{B}_0 \mathbf{V}_{\ell m} \in \mathbb{Z}_q^{n \times \ell m} \\ \mathbf{A}_f &= \mathsf{EvalF}(\mathbf{A}, f) \in \mathbb{Z}_q^{n \times m}. \end{aligned} \tag{5.6}$$

Sample $\mathbf{r} \xleftarrow{\mathrm{R}} \{0,1\}^m$ and compute

$$\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p} \in \mathbb{Z}_q^n.$$

Compute a NIZK proof $\pi \leftarrow \mathsf{NIZK.Prove}(C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r})$. Output the public key $\mathsf{pk} = (\mathbf{t}, \pi)$ and the secret key $\mathsf{sk} = (\mathbf{t}, \mathbf{r})$.

- $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk})$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and a public key $\mathsf{pk} = (\mathbf{t}, \pi)$, compute $\mathbf{d}_i, \mathbf{A}_f$ according to Eq. (5.6) and output 1 if

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \pi) = 1.$$

Otherwise, output 0.

- $\mathsf{Aggregate}(\mathsf{pp}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, a list of public keys $\mathsf{pk}_1 = (\mathbf{t}_1, \pi_1), \ldots, \mathsf{pk}_N = (\mathbf{t}_N, \pi_N)$ and their associated policies $f_1, \ldots, f_N$, the aggregation algorithm computes for each $i \in [N]$,

$$\begin{aligned} \mathbf{C}_i &= \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i) \in \mathbb{Z}_q^{n \times m} \\ \mathbf{Z}_i &= \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i) \in \mathbb{Z}_q^{n \times N}, \end{aligned}$$

where $\mathbf{u}_i \in \mathbb{Z}_q^N$ is the i^{th} unit vector. The aggregation algorithm parses $\mathbf{Z}_i = [\mathbf{z}_{i,1} \mid \cdots \mid \mathbf{z}_{i,N}]$. Next, it computes the re-randomization terms

$$\begin{aligned} \xi &= (\mathsf{pp}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N)) \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathrm{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; H_2(\xi)). \end{aligned}$$

Output $\mathsf{mpk} = \mathbf{C}_0 + \sum_{i \in [N]} \mathbf{C}_i$ and $\mathsf{hsk}_i = (N, f_i, \mathbf{z}_{0,i} + \sum_{j \in [N]} \mathbf{z}_{j,i})$ for all $i \in [N]$.

- $\mathsf{Encrypt}(\mathsf{mpk}, \mathbf{x}, \mu)$: On input the master public key $\mathsf{mpk} = \widehat{\mathbf{C}}$, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a message $\mu \in \{0,1\}$, sample the following:

$$\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \ , \ \mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m \ , \ \mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m} \ , \ \mathbf{r}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m.$$

If $\|\mathbf{e}\| > \sqrt{m}\chi$, then set $\mathbf{e} = \mathbf{0}^m$. Next, compute $\mathbf{C}_{\mathbf{x}} = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \in \mathbb{Z}_q^{n \times m}$ and output the ciphertext

$$\mathsf{ct} = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \ , \ \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}} \ , \ \mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}) + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0} \ , \ \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot \mu \right).$$

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{hsk}, f, \mathbf{x}, \mathsf{ct})$: On input the secret key $\mathsf{sk} = (\mathbf{t}, \mathbf{r})$, the helper decryption key $\mathsf{hsk} = (N, f, \hat{\mathbf{z}})$, the function f, the attribute $\mathbf{x} \in \{0,1\}^\ell$, and the ciphertext $\mathsf{ct} = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4)$, the decryption algorithm first computes

$$\begin{aligned} \mathbf{V}_{\ell m} &= \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m}) \in \mathbb{Z}_q^{m \times \ell m} \\ \mathbf{Z}_{\mathbf{x}} &= \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \in \mathbb{Z}_q^{m \times \ell m}. \end{aligned}$$

Next, it computes the homomorphic evaluation matrices

$$\begin{aligned}\mathbf{A} &= -\mathbf{B}_0\mathbf{V}_{\ell m} \in \mathbb{Z}_q^{n\times \ell m}\\ \mathbf{A}_f &= \mathsf{EvalF}(\mathbf{A}, f) \in \mathbb{Z}_q^{n\times m}\\ \mathbf{H}_{\mathbf{A},f,\mathbf{x}} &= \mathsf{EvalFX}(\mathbf{A}, f, \mathbf{x}) \in \mathbb{Z}_q^{\ell m\times m}\end{aligned}$$

Finally, it computes $\mathbf{d} = H_1(i) \in \mathbb{Z}_q^n$ and

$$\tilde{c}_3^{\mathsf{T}} = [\mathbf{c}_1^{\mathsf{T}} \mid \mathbf{c}_3^{\mathsf{T}}] \cdot \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}).$$

Next, it computes $\mathbf{v} = \mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, Nm, i)$ and the (noisy) encoding of the message

$$\tilde{\mu} = c_4 + \mathbf{c}_1^{\mathsf{T}}(\mathbf{r} - \hat{\mathbf{z}}) - \mathbf{c}_2^{\mathsf{T}}\mathbf{v} + \tilde{c}_3.$$

It outputs 0 if $-q/4 < \tilde{\mu} < q/4$ and 1 otherwise.

Theorem 5.12 (Completeness). *If Π_{NIZK} is complete, then Construction 5.11 is complete.*

Proof. Take any $\lambda, \tau \in \mathbb{N}$, any $N \leq 2^\lambda$, any $i \in [N]$, and policy f. Take any $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ in the support of $\mathsf{Setup}(1^\lambda, 1^\tau, N)$ and any $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, i, f)$. By construction, we can parse $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and $\mathsf{pk} = (\mathbf{t}, \pi)$ where

$$\begin{aligned}\mathbf{d}_i &= H_1(i) \in \mathbb{Z}_q^n\\ \mathbf{V}_{\ell m} &= \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m}) \in \mathbb{Z}_q^{m\times \ell m}\\ \mathbf{A} &= -\mathbf{B}_0\mathbf{V}_{\ell m} \in \mathbb{Z}_q^{n\times \ell m}\\ \mathbf{A}_f &= \mathsf{EvalF}(\mathbf{A}, f) \in \mathbb{Z}_q^{n\times m}\\ \mathbf{t} &= \mathbf{B}\mathbf{r} + \mathbf{A}_f\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p} \in \mathbb{Z}_q^n\\ \pi &\leftarrow \mathsf{NIZK.Prove}(C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r}),\end{aligned}$$

and $\mathbf{r} \in \{0,1\}^m$. By design, $C_{\mathsf{valid}}((i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r}) = 1$ so by completeness of Π_{NIZK},

$$\mathsf{NIZK.Verify}(C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \pi) = 1.$$

Correspondingly, $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk})$ outputs 1 and the claim holds. □

Theorem 5.13 (Correctness). *Suppose $m \geq n \log q$ and suppose $q > m^{O(d)} \cdot \chi \cdot (\sigma_{\mathsf{td}} \cdot \ell \log \ell + \sigma_{\mathsf{td}} \cdot N \log N + \sigma_{\mathsf{agg}})$.*

Proof. Take $\lambda, \tau \in \mathbb{N}$, $N \leq 2^\lambda$, any index $i \in [N]$, any policy $f_i \in \mathcal{F}_\tau$, any attribute $\mathbf{x} \in \{0,1\}^\ell$ where $f_i(\mathbf{x}) = 0$, any pp in the support of $\mathsf{Setup}(1^\lambda, 1^\tau, N)$, any $(\mathsf{pk}_i, \mathsf{sk}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, i, f_i)$, any set $\{(j, f_j, \mathsf{pk}_j)\}_{j\neq i}$ where $\mathsf{IsValid}(\mathsf{pp}, j, f_j, \mathsf{pk}_j) = 1$ for all $j \neq i$, and any message $\mu \in \{0,1\}$. Let

$$\begin{aligned}(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) &= \mathsf{Aggregate}(\mathsf{pp}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)\\ \mathsf{ct} &\leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathbf{x}, \mu).\end{aligned}$$

Then the following holds:

- First, $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. By construction, $[\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]\mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G}$ and $\|\mathbf{T}\| \leq \sqrt{m}\sigma_{\mathsf{td}}$.
- Let $\mathbf{V}_{\ell m} = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m})$, $\mathbf{A} = -\mathbf{B}_0\mathbf{V}_{\ell m}$. For each $j \in [N]$, let $\mathbf{A}_{f_j} = \mathsf{EvalF}(\mathbf{A}, f_j)$ and $\mathbf{d}_j = H_1(j)$.
- Since $(\mathsf{pk}_i, \mathsf{sk}_i)$ is in the support of $\mathsf{KeyGen}(\mathsf{pp}, i, f_i)$, we can write $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ where

  $$\mathbf{t}_i = \mathbf{B}\mathbf{r}_i + \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$$

  and $\mathsf{sk}_i = (\mathbf{t}_i, \mathbf{r}_i)$ for some $\mathbf{r}_i \in \{0,1\}^m$.

- Consider the quantities constructed by the aggregation algorithm. For each $j \in [N]$, the aggregation algorithm computes

$$\begin{aligned}\mathbf{C}_j &= \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j) \\ \mathbf{Z}_j &= \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j).\end{aligned}$$

  The aggregation algorithm parses $\mathbf{Z}_j = [\mathbf{z}_{j,1} \mid \cdots \mid \mathbf{z}_{j,N}]$. Then it computes

$$\begin{aligned}\xi &= (\mathsf{pp}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N)) \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; H_2(\xi)).\end{aligned}$$

  The master public key mpk and helper decryption key hsk_i is then defined to be

$$\mathsf{mpk} = \widehat{\mathbf{C}} = \mathbf{C}_0 + \sum_{j \in [N]} \mathbf{C}_j \quad \text{and} \quad \mathsf{hsk}_i = \hat{\mathbf{z}}_i = \mathbf{z}_{0,i} + \sum_{j \in [N]} \mathbf{z}_{j,i}.$$

- Next, consider the ciphertext ct. By definition,

$$\mathsf{ct} = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4) = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \;,\; \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}} \;,\; \mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}) + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0} \;,\; \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot \mu. \right),$$

  where $\mathbf{C}_{\mathbf{x}} = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$, $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m$, $\mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} \in \{0,1\}^{m \times m}$ and $\mathbf{r}_{\mathbf{p}} \in \{0,1\}^m$.

Consider now $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{hsk}_i, f_i, \mathbf{x}, \mathsf{ct})$:

- Let $\mathbf{V}_{\ell m} = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m})$ and $\mathbf{Z}_{\mathbf{x}} = \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$. By Remark 3.9, we have

$$\mathbf{C}_{\mathbf{x}} \cdot \mathbf{V}_{\ell m} = \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G} - \mathbf{B} \cdot \mathbf{Z}_{\mathbf{x}}.$$

  Recalling that $\mathbf{A} = -\mathbf{B}_0\mathbf{V}_{\ell m}$, this means

$$[\mathbf{B} \mid \mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}] \cdot \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} = -\mathbf{B} \cdot \mathbf{Z}_{\mathbf{x}} - \mathbf{B}_0 \cdot \mathbf{V}_{\ell m} - \mathbf{C}_{\mathbf{x}} \cdot \mathbf{V}_{\ell m} = \mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}.$$

  Let $\mathbf{H}_{\mathbf{A},f_i,\mathbf{x}} = \mathsf{EvalFX}(\mathbf{A}, f_i, \mathbf{x})$. Since $f_i(\mathbf{x}) = 0$, we appeal to Theorem 3.6 to conclude that

$$(\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}} = \mathbf{A}_{f_i} - f_i(\mathbf{x}) \cdot \mathbf{G} = \mathbf{A}_{f_i}.$$

  Thus,

$$\begin{aligned}\tilde{c}_3^{\mathsf{T}} &= [\mathbf{c}_1^{\mathsf{T}} \mid \mathbf{c}_3^{\mathsf{T}}] \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) \\ &= \mathbf{s}^{\mathsf{T}}[\mathbf{B} \mid \mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}] \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \underbrace{\mathbf{e}^{\mathsf{T}}[\mathbf{I}_m \mid \mathbf{R}_{\mathbf{B}_0}] \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i)}_{\tilde{e}_1} \\ &= \mathbf{s}^{\mathsf{T}}\mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i) + \tilde{e}_1.\end{aligned}$$

- Let $\mathbf{V}_N = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N] = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^N)$. For all $j \in [N]$, $\mathbf{C}_j$ is a commitment to $\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j$ and $\mathbf{Z}_j$ is an opening to $\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j$. By Remark 3.9, $\mathbf{C}_j\mathbf{V}_N = (\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j) - \mathbf{B}\mathbf{Z}_j$, and in particular,

$$\begin{aligned}\forall j \neq i : \mathbf{C}_j\mathbf{v}_i &= -\mathbf{B}\mathbf{z}_{j,i} \\ \mathbf{C}_i\mathbf{v}_i &= \mathbf{t}_i - \mathbf{B}\mathbf{z}_{i,i} = \mathbf{B}\mathbf{R}_i + \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p} - \mathbf{B}\mathbf{z}_{i,i}.\end{aligned}$$

By Theorem 5.9, we have that $\mathbf{B}\mathbf{z}_{0,j} = -\mathbf{C}_0\mathbf{v}_j$ for all $j \in [N]$. Combining these two relations, we conclude

$$\widehat{\mathbf{C}}\mathbf{v}_i + \mathbf{B}\hat{\mathbf{z}}_i = \mathbf{C}_0\mathbf{v}_i + \sum_{j\in[N]} \mathbf{C}_j\mathbf{v}_i + \mathbf{B}\mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{B}\mathbf{z}_{j,i} = \mathbf{C}_i\mathbf{v}_i = \mathbf{B}\mathbf{r}_i + \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}.$$

Thus,

$$\begin{aligned} \mathbf{c}_1^{\mathsf{T}}(\mathbf{r}_i - \hat{\mathbf{z}}_i) - \mathbf{c}_2^{\mathsf{T}}\mathbf{v}_i + \tilde{c}_3 &= \mathbf{s}^{\mathsf{T}}(\mathbf{B}\mathbf{r}_i - \mathbf{B}\hat{\mathbf{z}}_i - \widehat{\mathbf{C}}\mathbf{v}_i + \mathbf{A}_{f_i}\mathbf{G}^{-1}(\mathbf{d}_i)) + \tilde{e}_2 \\ &= -\mathbf{s}^{\mathsf{T}}\mathbf{p} + \tilde{e}_2 \end{aligned}$$

where

$$\tilde{e}_2 = \mathbf{e}^{\mathsf{T}}\mathbf{r}_i - \mathbf{e}^{\mathsf{T}}\hat{\mathbf{z}}_i - \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \tilde{e}_1.$$

Thus, we have

$$\begin{aligned} \tilde{\mu} &= c_4 + \mathbf{c}_1^{\mathsf{T}}(\mathbf{r}_i - \hat{\mathbf{z}}_i) - \mathbf{c}_2^{\mathsf{T}}\mathbf{v}_i + \tilde{c}_3 \\ &= \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot \mu - \mathbf{s}^{\mathsf{T}}\mathbf{p} + \tilde{e}_2 \\ &= \lfloor q/2 \rfloor \cdot \mu + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \tilde{e}_2. \end{aligned}$$

As long as $|\mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \tilde{e}_2| < q/4$, correctness holds.

To complete, the proof, it suffices to bound the error $|\mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} - \tilde{e}_2|$.

- By definition, we have $\|\mathbf{T}\| \le \sqrt{m}\sigma_{\mathrm{td}}$ and $\|\mathbf{e}\| \le \sqrt{m}\chi$. In addition, $\mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} \in \{0,1\}^{m\times m}$ and $\mathbf{r}_i, \mathbf{r}_{\mathbf{p}} \in \{0,1\}^m$.
- By Remark 3.9, for all $j \in [N]$, we have

$$\begin{aligned} \|\mathbf{V}_{\ell m}\|, \|\mathbf{V}_N\| &\le O(\|\mathbf{T}\| \cdot m^4 \log q) = O(\sigma_{\mathrm{td}} m^{9/2} \log q) \\ \|\mathbf{Z}_{\mathbf{x}}\| &\le O(\|\mathbf{T}\| \cdot m^7 \log q \log(\ell m)) = O(\sigma_{\mathrm{td}} m^{15/2} \log q \log(\ell m)) \\ \|\mathbf{Z}_j\| &\le O(\|\mathbf{T}\| \cdot m^7 \log q \log N) = O(\sigma_{\mathrm{td}} m^{15/2} \log q \log N). \end{aligned}$$

  From Theorem 5.9, we have that $\|\mathbf{z}_{0,i}\| \le \sqrt{m}\sigma_{\mathrm{agg}}$. Since $\hat{\mathbf{z}}_i = \mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{z}_{j,i}$, this means

$$\|\hat{\mathbf{z}}_i\| \le \|\mathbf{z}_{0,i}\| + \sum_{j\in[N]} \|\mathbf{z}_{j,i}\| \le \sqrt{m}\sigma_{\mathrm{agg}} + N \cdot O(\sigma_{\mathrm{td}} m^{15/2} \log q \log N).$$

- By Lemma 3.5, we have $\|\mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\| \le m^{O(d)}$.
- By definition, $\tilde{e}_1 = \mathbf{e}^{\mathsf{T}}[\mathbf{I}_m \mid \mathbf{R}_{\mathbf{B}_0}]\left[\begin{smallmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{smallmatrix}\right] \cdot \mathbf{H}_{\mathbf{A},f_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i)$. This means

$$\begin{aligned} |\tilde{e}_1| &\le (\sqrt{m}\chi) \cdot m \cdot O(\sigma_{\mathrm{td}} m^{15/2} \log q \log(\ell m)) \cdot (2m) \cdot m^{O(d)} \cdot \ell m \cdot m \\ &\le m^{O(d)} \cdot \sigma_{\mathrm{td}} \ell \chi \log q \log \ell. \end{aligned}$$

- Next $\tilde{e}_2 = \mathbf{e}^{\mathsf{T}}\mathbf{r}_i - \mathbf{e}^{\mathsf{T}}\hat{\mathbf{z}}_i - \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \tilde{e}_1$. Then,

$$|\tilde{e}_2| \le m^{O(d)} \cdot \sigma_{\mathrm{td}} \ell \chi \log q \log \ell + m^2 \chi \sigma_{\mathrm{agg}} + N \cdot O(\sigma_{\mathrm{td}} \chi m^9 \log q \log N).$$

  Finally, $|\mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}}| \le m\sqrt{m}\chi$.

If $m \ge n \log q$, setting

$$q > m^{O(d)} \cdot \chi \cdot (\sigma_{\mathrm{td}} \cdot \ell \log \ell + \sigma_{\mathrm{td}} \cdot N \log N + \sigma_{\mathrm{agg}})$$

suffices for correctness. □

Theorem 5.14 (Attribute-Selective Security without Corruptions)**.** *Suppose Π_{NIZK} is complete, zero-knowledge, and simulation-sound extractable, and that the $(2m^2, \sigma_{\mathsf{td}})$-succinct LWE assumption holds with parameters (n, m, q, χ). Suppose also $n \geq \lambda$, $m \geq O(n \log q)$, $q > 2$ is prime, $\chi \geq \log m$, $\sigma_{\mathsf{td}} > O(m^3 \log m)$, $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot \sigma_{\mathsf{td}} \cdot m^{O(d)} \cdot N^3 \log \ell$, and $\lambda_{\mathsf{DGS}} > \log \sigma_{\mathsf{agg}}$. Then, Construction 5.11 satisfies attribute-selective security without corruptions.*

Proof. Our proof follows a similar structure as the corresponding proof from [CHW25]:

- First, we replace the NIZK proofs in the keys sampled by the challenger with simulated proofs. For the adversarially-generated keys, the reduction extracts an associated secret key.
- Next, we change the distribution of the honest keys. Instead of sampling them as $\mathbf{t} = \mathbf{Br} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$, they are replaced with $\mathbf{t} = \mathbf{Br} + \mathbf{d}_i$. In particular, the reduction algorithm no longer has a secret key for the honestly-generated public keys.
- Next, we program the challenge attribute $\mathbf{x}$ into the public parameters (i.e., we set $\mathbf{B}_0 - \mathbf{BR}_{\mathbf{B}_0} - \mathbf{C}_{\mathbf{x}}$, where $\mathbf{C}_{\mathbf{x}}$ is a commitment to $\mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}$ and $\mathbf{x}$ is the challenge attribute).
- Then we use the explainable re-randomizer to program the challenge public keys into $\mathbf{C}_0$. Specifically, we choose $\mathbf{C}_0$ such that the aggregated key $\widehat{\mathbf{C}}$ can be written as $\widehat{\mathbf{C}} = \mathbf{C}_0 + \sum_{i \in [N]} \mathbf{C}_i^* = \mathbf{BR}_{\mathbf{C}_0}$, where $\mathbf{C}_i^*$ is the commitment to the (adversarially-chosen) public key for slot i. Just as in [CHW25], the re-randomization term $\mathbf{C}_0$ is chosen to "cancel" out the adversarially-chosen public keys. This is the critical step that enables the reduction algorithm (to succinct LWE) to simulate the challenge ciphertext.
- Finally, we rely on succinct LWE to argue that the challenge ciphertext is pseudorandom.

We now give the formal argument. Take polynomials $\tau = \tau(\lambda)$ and $N = N(\lambda)$ and let $\mathcal{A}$ be an efficient adversary for the attribute-selective security game (without corruptions). In the security analysis, we model the hash functions H_1 and H_2 as random oracles. For ease of exposition, we assume that $\mathcal{A}$ has the following properties:

- Algorithm $\mathcal{A}$ does not query H_1 or H_2 on the same input more than once.
- Algorithm $\mathcal{A}$ always queries H_2 on the tuple $\xi^* = (\mathsf{pp}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ associated with the challenge query before entering the challenge phase.

Observe that both properties hold without loss of generality. Namely, any efficient algorithm $\mathcal{A}$ that does not satisfy these properties can be generically compiled into an algorithm that does. Finally, let Q_{ro} be a bound on the number of random oracle queries algorithm $\mathcal{A}$ makes (to either H_1 or H_2).

Hybrid experiments. We now define the sequence of hybrid experiments for our security analysis. Each experiment is parameterized by a bit $b \in \{0, 1\}$ and a precision parameter $\kappa = \kappa(\lambda)$ (i.e., the input to the explainable sampling procedure). For simplicity of notation, we omit the index κ when the behavior of the experiment is independent of the choice of κ.

- $\mathsf{Hyb}_0^{(b)}$: This is the semi-malicious attribute-selective security experiment with challenge bit b:
  - **Setup phase:** On input the security parameter 1^λ, the policy-family parameter 1^τ, and the number of slots N, algorithm $\mathcal{A}$ outputs the challenge attribute $\mathbf{x} \in \{0, 1\}^{\ell(\tau)}$. The challenger samples

$$\begin{aligned} \mathsf{crs}_{\mathsf{NIZK}} &\leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\ (\mathbf{B}, \mathbf{T}_{\mathbf{B}}) &\leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{2m^2 n \times m} \\ \mathbf{T} &\leftarrow \mathsf{SamplePre}([\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}], \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}} \\ \mathbf{0} \end{smallmatrix}\right], \mathbf{I}_{2m^2} \otimes \mathbf{G}, \sigma_{\mathsf{td}}) \\ \mathbf{B}_0 &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n. \end{aligned}$$

  If $\|\mathbf{T}\| > \sqrt{m}\sigma_{\mathsf{td}}$, then the challenger sets $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}} \\ \mathbf{0} \end{smallmatrix}\right]$. Let $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. The challenger gives the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ to $\mathcal{A}$. The challenger also initializes a counter $\mathsf{ctr} = 0$ and a dictionary D. In addition, whenever $\mathcal{A}$ queries H_1 on an index $i \in [N]$, the challenger responds with $\mathbf{d}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$. Whenever algorithm $\mathcal{A}$ queries H_2 on a string $\xi \in \{0,1\}^*$, the challenger responds with a string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$.

- **Query phase:** When $\mathcal{A}$ makes a key-generation query on an index $i \in [N]$ and a function f, the challenger increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$. Then it computes

$$\begin{aligned} \mathbf{d}_i &= H_1(i) \in \mathbb{Z}_q^n \\ \mathbf{V}_{\ell m} &= \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m}) \in \mathbb{Z}_q^{m \times \ell m} \\ \mathbf{A} &= -\mathbf{B}_0 \mathbf{V}_{\ell m} \in \mathbb{Z}_q^{n \times \ell m} \\ \mathbf{A}_f &= \mathsf{EvalF}(\mathbf{A}, f) \in \mathbb{Z}_q^{n \times m}. \end{aligned} \tag{5.7}$$

Then, it samples $\mathbf{r} \xleftarrow{\mathrm{R}} \{0,1\}^m$ and computes

$$\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p} \in \mathbb{Z}_q^n.$$

The challenger then computes $\pi \leftarrow \mathsf{NIZK.Prove}(C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r})$ and responds with the public key $\mathsf{pk} = (\mathbf{t}, \pi)$. The challenger adds the mapping $\mathsf{ctr} \mapsto (i, f, \mathbf{t})$ to D.

- **Challenge phase:** Let $((c_1, f_1, \mathsf{pk}_1), \ldots, (c_N, f_N, \mathsf{pk}_N))$ be algorithm $\mathcal{A}$'s challenge query. For each $i \in [N]$, the challenger proceeds as follows:
  - $*$ If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up $(i', f', \mathsf{pk}') = \mathsf{D}[c_i]$ and checks that $i = i'$. If not, the challenger outputs 0. Otherwise, the challenger sets $\mathsf{pk}_i^* = \mathsf{pk}' = (\mathbf{t}_i, \pi_i)$ and $f_i^* = f'$.
  - $*$ If $c_i = \bot$, then the challenger checks that $f_i(\mathbf{x}) = 1$. If so, it parses $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$. Then, it checks that $\mathsf{IsValid}(\mathsf{pp}, i, f_i, \mathsf{pk}_i) = 1$. If so, the challenger sets $\mathsf{pk}_i^* = \mathsf{pk}_i$ and $f_i^* = f_i$. Otherwise, the challenger outputs 0.

  Next, for each $i \in [N]$, the challenger computes $\mathbf{C}_i^* = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i)$ and the re-randomization matrix $\mathbf{C}_0$ as

$$\begin{aligned} \xi^* &= (\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)) \\ \gamma^* &= H_2(\xi^*) \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; \gamma^*). \end{aligned} \tag{5.8}$$

  Finally, the challenger sets

$$\widehat{\mathbf{C}} = \mathbf{C}_0 + \sum_{i \in [N]} \mathbf{C}_i^*.$$

  Next, to construct the challenge ciphertext, the challenger samples the following:

$$\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \ , \ \mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m \ , \ \mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m} \ , \ \mathbf{r}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m.$$

  If $\|\mathbf{e}\| > \sqrt{m}\chi$, it sets $\mathbf{e} = \mathbf{0}^m$. Next, compute $\mathbf{C}_{\mathbf{x}} = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \in \mathbb{Z}_q^{n \times m}$ and the ciphertext

$$\mathsf{ct} = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \ , \ \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}} \ , \ \mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}) + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0} \ , \ \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot b \right).$$

  The challenger gives ct to $\mathcal{A}$.

- **Output phase:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

- $\mathsf{Hyb}_1^{(b)}$: Same as $\mathsf{Hyb}_0^{(b)}$ except at the beginning of the experiment, the challenger samples an index $\mathsf{ind} \xleftarrow{\mathrm{R}} [Q_{\mathsf{ro}}]$. In the challenge phase, after the challenger computes ξ^* according to Eq. (5.8), the challenger additionally checks that algorithm $\mathcal{A}$ has made at least ind queries to H_2, and if so, that the ind^{th} query $\xi_{\mathsf{ind}} \in \{0,1\}^*$ algorithm $\mathcal{A}$ made to H_2 satisfies $\xi_{\mathsf{ind}} = \xi^*$. If this is not the case, then the challenger halts with output 0.

- $\mathsf{Hyb}_2^{(b)}$: Same as $\mathsf{Hyb}_1^{(b)}$ except when responding to key-generation queries (on an index i and function f), the challenger outputs 0 if $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) = 0$.

- $\mathsf{Hyb}_3^{(b)}$: Same as $\mathsf{Hyb}_2^{(b)}$ except the challenger replaces the NIZK proofs in the key-generation queries with simulated NIZK proofs:

  - In the setup phase, the challenger samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$.
  - On each key-generation query, the challenger computes $\pi \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}))$.

- $\mathsf{Hyb}_4^{(b)}$: Same as $\mathsf{Hyb}_3^{(b)}$ except the challenger extracts secret keys for the public keys associated with the indth query ξ_{ind} to H_2. Specifically, the experiment proceeds as follows:

  - During the setup phase, the challenger initializes an empty dictionary D_{sk}.
  - Whenever $\mathcal{A}$ makes a key-generation query on an index i and function f, after the challenger samples $\mathbf{r} \xleftarrow{\mathrm{R}} \{0,1\}^m$ and computes $\mathbf{t} = \mathbf{B}\mathbf{r} - \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{p}$, the challenger adds the mapping $(i, \mathbf{A}_f, \mathbf{t}) \mapsto (0, \mathbf{r})$ to D_{sk} if $(i, \mathbf{A}_f, \mathbf{t})$ is not already contained in D_{sk}.

  When responding to the indth query ξ_{ind} to H_2, the challenger proceeds as follows:

  - Parse $\xi_{\mathsf{ind}} = (\mathsf{pp}^*, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$, where $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$. If ξ_{ind} does not have this form or $\mathsf{pp}^* \neq \mathsf{pp}$, then the challenger outputs 0. If $\mathsf{IsValid}(\mathsf{pp}, i, f_i^*, \mathsf{pk}_i^*) = 0$ for any $i \in [N]$, the challenger outputs 0.
  - For each $i \in [N]$, the challenger computes $\mathbf{A}_{f_i^*} = \mathsf{EvalF}(\mathbf{A}, f_i^*)$. If $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ is not contained in D_{sk}, the challenger first checks that $f_i^*(\mathbf{x}) = 1$. If not, the challenger outputs 0. Otherwise, the challenger computes
  $$\mathbf{r}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \pi_i^*).$$
  If $\mathbf{r}_i^* \notin \{0,1\}^m$ or $\mathbf{B}\mathbf{r}_i^* \neq \mathbf{t}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$, then the challenger outputs 0. Otherwise, the challenger adds the mapping $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ to D_{sk}.

  If all of the checks pass, the challenger samples $\gamma^* \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ and responds with $H_2(\xi_{\mathsf{ind}}) \coloneqq \gamma^*$. In this experiment, for every $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$ and accompanying policy f_i^* in ξ_{ind}, there is a mapping $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (b_i^*, \mathbf{r}_i^*)$ in D_{sk}. The bit b_i^* indicates whether $\mathbf{r}_i^*$ was sampled by the challenger (in response to a key-generation query) or if $\mathbf{r}_i^*$ was chosen by the adversary.

- $\mathsf{Hyb}_5^{(b)}$: Same as $\mathsf{Hyb}_4^{(b)}$, except when sampling the public parameters, the challenger samples
  $$\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m} \quad \text{and} \quad \mathbf{T} \leftarrow [\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]_{\sigma_{\mathsf{td}}}^{-1}(\mathbf{I}_{2m^2} \otimes \mathbf{G}).$$

- $\mathsf{Hyb}_6^{(b)}$: Same as $\mathsf{Hyb}_5^{(b)}$, except the challenger samples the matrices $\mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}$ and $\mathbf{r}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m$ during the setup phase instead. Then, the challenger computes $\mathbf{C}_{\mathbf{x}} = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$. It sets $\mathbf{B}_0 = \mathbf{B}\mathbf{R}_{\mathbf{B}_0} - \mathbf{C}_{\mathbf{x}}$ and $\mathbf{p} = \mathbf{B}\mathbf{r}_{\mathbf{p}}$. When responding to a key-generation query (on an index i and function f), the challenger instead sets $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{d}_i$. The challenger still adds the mapping $(i, \mathbf{A}_f, \mathbf{t}) \mapsto (0, \mathbf{r})$ to D_{sk} as before.

- $\mathsf{Hyb}_{7,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_6^{(b)}$, except when the adversary makes the indth query ξ_{ind} to H_2, the challenger now samples
  $$\begin{aligned} \gamma &\xleftarrow{\mathrm{R}} \{0,1\}^\rho \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; \gamma) \\ \gamma^* &\leftarrow \mathsf{Explain}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^\kappa, (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}), \sigma_{\mathsf{agg}}). \end{aligned}$$

  The challenger replies to $\mathcal{A}$ with γ^* (i.e., implicitly setting $H_2(\xi_{\mathsf{ind}}) \coloneqq \gamma^*$).

- $\mathsf{Hyb}_{8,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{7,\kappa}^{(b)}$, except when responding to the indth query ξ_{ind} to H_2, the challenger now samples

  $$\mathbf{C}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m} \quad \text{and} \quad \forall i \in [N] : \mathbf{z}_{0,i} \leftarrow \mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}(-\mathbf{C}_0 \mathbf{v}_i)$$

  where $\mathbf{V}_N = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N] = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^N)$.

- $\mathsf{Hyb}_{9,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{8,\kappa}^{(b)}$ except the challenger changes how it samples $\mathbf{C}_0$ and $\mathbf{z}_{0,i}$ when responding to the indth query ξ_{ind} to H_2. Specifically, let $\xi_{\mathsf{ind}} = (\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$ where $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$. If ξ_{ind} does not have this form, then the challenger outputs 0 (as in $\mathsf{Hyb}_4^{(b)}$ and all subsequent hybrids). The challenger populates the dictionary D_{sk} usually the same procedure described in $\mathsf{Hyb}_4^{(b)}$. Then, for each $i \in [N]$, the challenger computes

  $$\begin{aligned} \mathbf{C}_i^* &= \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i^*) \\ \mathbf{Z}_i^* &= \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i^*) \end{aligned}$$

  The challenger parses $\mathbf{Z}_i^* = [\mathbf{z}_{i,1}^* \mid \cdots \mid \mathbf{z}_{i,N}^*]$. Next, the challenger sets $\mathbf{C}_0 = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} - \sum_{j \in [N]} \mathbf{C}_j^*$. By construction, if the challenger has not halted, we have for every $i \in [N]$, there exists a mapping $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (b_i^*, \mathbf{r}_i^*)$ in D_{sk}. Then, for each $i \in [N]$, it defines $\mathbf{z}_{0,i}$ as follows:

  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (0, \mathbf{r}_i^*)$ in D_{sk}. Then, the challenger sets

    $$\mathbf{z}_{0,i} \leftarrow \mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}\left(\mathbf{d}_i - \mathbf{B}\big(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + \textstyle\sum_{j \in [N]} \mathbf{z}_{j,i}^*\big)\right).$$

  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ in D_{sk}. Then, the challenger sets

    $$\mathbf{z}_{0,i} \leftarrow \mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}\left(\mathbf{d}_i - \mathbf{B}\big(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A}, f_i^*, \mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{r}_{\mathbf{p}} + \textstyle\sum_{j \in [N]} \mathbf{z}_{j,i}^*\big)\right).$$

  The rest of the experiment proceeds as in $\mathsf{Hyb}_{8,\kappa}^{(b)}$.

- $\mathsf{Hyb}_{10,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{9,\kappa}^{(b)}$ except the challenger changes how it samples $\mathbf{z}_{0,i}$ when responding to the indth query ξ_{ind} to H_2. As usual, let $\xi_{\mathsf{ind}} = (\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$ where $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$.

  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (0, \mathbf{r}_i^*)$ in D_{sk}. Then, the challenger samples $\tilde{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{d}_i)$ and sets

    $$\mathbf{z}_{0,i} = \tilde{\mathbf{z}}_{0,i} - \mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - \sum_{j \in [N]} \mathbf{z}_{j,i}^*.$$

  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ in D_{sk}. Then, the challenger samples $\tilde{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{d}_i)$ and sets

    $$\mathbf{z}_{0,i} = \tilde{\mathbf{z}}_{0,i} - \mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A}, f_i^*, \mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{r}_{\mathbf{p}} - \sum_{j \in [N]} \mathbf{z}_{j,i}^*.$$

- $\mathsf{Hyb}_{11,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{10,\kappa}^{(b)}$, except when simulating $\mathbf{d}_i = H_1(i)$, instead of sampling $\mathbf{d}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, the challenger instead samples $\tilde{\mathbf{z}}_{0,i} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{agg}}}^m$ and sets $\mathbf{d}_i = \mathbf{B}\tilde{\mathbf{z}}_{0,i}$. When answering the indth query to H_2, the challenger uses these values of $\tilde{\mathbf{z}}_{0,i}$ for all $i \in [N]$ instead of sampling them.

- $\mathsf{Hyb}_{12,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{11,\kappa}^{(b)}$, except when sampling the public parameters, the challenger no longer checks the condition $\|\mathbf{T}\| > \sqrt{m}\sigma_{\mathsf{td}}$. Similarly, when constructing the challenge ciphertext ct^*, the challenger no longer checks if $\|\mathbf{e}\| > \sqrt{m}\chi$.

- $\mathsf{Hyb}_{13,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{12,\kappa}^{(b)}$ except when simulating the challenge ciphertext, the challenger now samples $\tilde{\mathbf{c}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$ and defines

  $$\mathsf{ct} = \left(\tilde{\mathbf{c}}^{\mathsf{T}},\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}},\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0},\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot b\right).$$

- $\mathsf{Hyb}^{(b)}_{14,\kappa}$: Same as $\mathsf{Hyb}^{(b)}_{13,\kappa}$ except when responding to the ind^{th} query ξ_{ind} to H_2, the challenger samples $\mathbf{z}_{0,i}$ as follows:
  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (0, \mathbf{r}_i^*)$ in D_{sk}. Then, the challenger sets
$$\mathbf{z}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}\left(\mathbf{d}_i - \mathbf{B}\big(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + \textstyle\sum_{j\in[N]} \mathbf{z}^*_{j,i}\big)\right).$$
  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ in D_{sk}. Then, the challenger sets
$$\mathbf{z}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}\left(\mathbf{d}_i - \mathbf{B}\big(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{r}_{\mathbf{p}} + \textstyle\sum_{j\in[N]} \mathbf{z}^*_{j,i}\big)\right).$$

    Note that this is equivalent to sampling
$$\mathbf{z}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}\left(\mathbf{d}_i - \mathbf{B}\big(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \textstyle\sum_{j\in[N]} \mathbf{z}^*_{j,i}\big) + \mathbf{p}\right).$$

    This latter expression only depends on $\mathbf{p}$ and *not* $\mathbf{r}_{\mathbf{p}}$.
- $\mathsf{Hyb}^{(b)}_{15,\kappa}$: Same as $\mathsf{Hyb}^{(b)}_{14,\kappa}$ except when simulating the challenger ciphertext, the challenger samples $c_4 \xleftarrow{\text{R}} \mathbb{Z}_q$ and outputs
$$\mathsf{ct} = \left(\tilde{\mathbf{c}}^{\mathsf{T}}\,,\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}}\,,\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0}\,,\ c_4\right).$$

  Notably, the challenger's behavior in this experiment is *independent* of the bit $b \in \{0,1\}$.

We write $\mathsf{Hyb}^{(b)}_i(\mathcal{A})$ to denote the distribution of the output of $\mathsf{Hyb}^{(b)}_i$ with adversary $\mathcal{A}$. We now analyze each pair of adjacent distributions.

Lemma 5.15. *For all $b \in \{0,1\}$, $\Pr[\mathsf{Hyb}^{(b)}_1(\mathcal{A}) = 1] = \frac{1}{Q_{\mathsf{ro}}}\Pr[\mathsf{Hyb}^{(b)}_0(\mathcal{A}) = 1]$.*

Proof. By construction, the view of adversary $\mathcal{A}$ is identical in $\mathsf{Hyb}^{(b)}_0$ and $\mathsf{Hyb}^{(b)}_1$. By assumption, algorithm $\mathcal{A}$ is guaranteed to query H_2 on ξ^* prior to the challenge phase. Algorithm $\mathcal{A}$ makes at most Q_{ro} queries to H_2 so let $\mathsf{ind}^* \in [Q_{\mathsf{ro}}]$ be the index of query ξ^*. Since all the queries $\mathcal{A}$ makes to H_2 are distinct, with probability $1/Q_{\mathsf{ro}}$ over the choice of $\mathsf{ind} \xleftarrow{\text{R}} [Q_{\mathsf{ro}}]$, it will be the case that $\mathsf{ind} = \mathsf{ind}^*$. The output in experiment $\mathsf{Hyb}^{(b)}_1$ is 1 if and only if $\mathsf{ind} = \mathsf{ind}^*$ and the output in experiment $\mathsf{Hyb}^{(b)}_0$ is 1. We conclude
$$\Pr[\mathsf{Hyb}^{(b)}_1(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}^{(b)}_0(\mathcal{A}) = 1 \wedge \mathsf{ind} = \mathsf{ind}^*] = \frac{1}{Q_{\mathsf{ro}}}\Pr[\mathsf{Hyb}^{(b)}_0(\mathcal{A}) = 1]. \qquad \square$$

Lemma 5.16. *Suppose Π_{NIZK} is complete. Then, for all $b \in \{0,1\}$, $\mathsf{Hyb}^{(b)}_1(\mathcal{A})$ and $\mathsf{Hyb}^{(b)}_2(\mathcal{A})$ are identically distributed.*

Proof. By Theorem 5.12, Construction 5.11 is complete so $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) = 1$ for all pk in the support of $\mathsf{KeyGen}(\mathsf{pp}, i, f)$. Thus, the additional abort condition never triggers and the two experiments are identical. □

Lemma 5.17. *Suppose Π_{NIZK} satisfies zero-knowledge. Then, for all $b \in \{0,1\}$, $\mathsf{Hyb}^{(b)}_2(\mathcal{A})$ and $\mathsf{Hyb}^{(b)}_3(\mathcal{A})$ are computationally indistinguishable.*

Proof. Suppose $|\Pr[\mathsf{Hyb}^{(b)}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_3(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks zero-knowledge of Π_{NIZK}:

- **Setup phase:** At the beginning of the game, algorithm $\mathcal{B}$ receives a common reference string $\mathsf{crs}_{\mathsf{NIZK}}$ from the zero-knowledge challenger. Algorithm $\mathcal{B}$ samples an index $\mathsf{ind} \xleftarrow{\text{R}} [Q_{\mathsf{ro}}]$ and starts running $\mathcal{A}$ on input the security parameter 1^λ, the policy-family parameter 1^τ, and the number of slots N. Algorithm $\mathcal{A}$ outputs the challenge attribute $\mathbf{x} \in \{0,1\}^{\ell(\tau)}$. Algorithm $\mathcal{B}$ now samples

$$\begin{aligned}(\mathbf{B}, \mathbf{T_B}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} &\xleftarrow{\text{R}} \mathbb{Z}_q^{2m^2 n \times m} \\ \mathbf{T} \leftarrow \mathsf{SamplePre}([\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}], \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T_B} \\ \mathbf{0} \end{smallmatrix}\right], \mathbf{I}_{2m^2} &\otimes \mathbf{G}, \sigma_{\mathsf{td}}) \\ \mathbf{B}_0 \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}, \mathbf{p} &\xleftarrow{\text{R}} \mathbb{Z}_q^n.\end{aligned}$$

If $\|\mathbf{T}\| > \sqrt{m}\sigma_{\mathsf{td}}$, then algorithm $\mathcal{B}$ sets $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T_B} \\ \mathbf{0} \end{smallmatrix}\right]$. Let $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. Algorithm $\mathcal{B}$ gives the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ to $\mathcal{A}$. Algorithm $\mathcal{B}$ also initializes a counter $\mathsf{ctr} = 0$ and a dictionary D. In addition, whenever $\mathcal{A}$ queries H_1 on an index $i \in [N]$, algorithm $\mathcal{B}$ responds with $\mathbf{d}_i \xleftarrow{\text{R}} \mathbb{Z}_q^n$. Whenever algorithm $\mathcal{A}$ queries H_2 on a string $\xi \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds with a string $\gamma \xleftarrow{\text{R}} \{0,1\}^\rho$.

- **Query phase:** When $\mathcal{A}$ makes a key-generation query on an index $i \in [N]$ and a function f, algorithm $\mathcal{B}$ increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$. Then it computes $\mathbf{d}_i, \mathbf{V}_\ell, \mathbf{A}, \mathbf{A}_f$ according to Eq. (5.7) and samples $\mathbf{r} \xleftarrow{\text{R}} \{0,1\}^m$. Algorithm $\mathcal{B}$ computes $\mathbf{t} = \mathbf{Br} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$ and submits $(C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r})$ to the zero-knowledge challenger. The zero-knowledge challenger replies with a proof π. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the public key $\mathsf{pk} = (\mathbf{t}, \pi)$ and also adds the mapping $\mathsf{ctr} \mapsto (i, f, \mathbf{t})$ to D. Finally, algorithm $\mathcal{B}$ checks if $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) = 0$ and outputs 0 if so.
- **Challenge phase:** Let $((c_1, f_1, \mathsf{pk}_i), \ldots, (c_N, f_N, \mathsf{pk}_N))$ be algorithm $\mathcal{A}$'s challenge query. For each $i \in [N]$, algorithm $\mathcal{B}$ proceeds as follows:
  - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then algorithm $\mathcal{B}$ looks up $(i', f', \mathsf{pk}') = \mathsf{D}[c_i]$ and checks that $i = i'$. If not, algorithm $\mathcal{B}$ outputs 0. Otherwise, algorithm $\mathcal{B}$ sets $\mathsf{pk}_i^* = \mathsf{pk}' = (\mathbf{t}_i, \pi_i)$ and $f_i^* = f_i$.
  - If $c_i = \perp$, then algorithm $\mathcal{B}$ checks that $f_i(\mathbf{x}) = 1$. If so, it parses $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$. Then, it checks that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, (i, \mathbf{A}_{f_i}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i), \pi_i) = 1$ where $\mathbf{A}_{f_i} = \mathsf{EvalF}(\mathbf{A}, f_i)$, and $\mathbf{d}_i = H_1(i)$). If the checks pass, algorithm $\mathcal{B}$ sets $\mathsf{pk}_i^* = \mathsf{pk}_i$ and $f_i^* = f_i$. Otherwise, algorithm $\mathcal{B}$ outputs 0.

  Next, for each $i \in [N]$, algorithm $\mathcal{B}$ computes $\mathbf{C}_i^* = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i)$ and the re-randomization matrix $\mathbf{C}_0$ as

  $$\begin{aligned}\xi^* &= (\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)) \\ \gamma^* &= H_2(\xi^*) \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; \gamma^*).\end{aligned}$$

  If adversary $\mathcal{A}$ has not yet made at least ind queries to H_2, or if its ind^{th} query ξ_{ind} to H_2 satisfies $\xi_{\mathsf{ind}} \neq \xi^*$, then algorithm $\mathcal{B}$ outputs 0. Otherwise, algorithm $\mathcal{B}$ computes

  $$\widehat{\mathbf{C}} = \mathbf{C}_0 + \sum_{i \in [N]} \mathbf{C}_i^* \ , \ \mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n \ , \ \mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m \ , \ \mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} \xleftarrow{\text{R}} \{0,1\}^{m \times m} \ , \ \mathbf{r_p} \xleftarrow{\text{R}} \{0,1\}^m.$$

  If $\|\mathbf{e}\| > \sqrt{m}\chi$, algorithm $\mathcal{B}$ sets $\mathbf{e} = \mathbf{0}^m$. Next, algorithm $\mathcal{B}$ computes $\mathbf{C_x} = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \in \mathbb{Z}_q^{n \times m}$ and the ciphertext

  $$\mathsf{ct} = \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \ , \ \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}} \ , \ \mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C_x}) + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0} \ , \ \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{r_p} + \lfloor q/2 \rfloor \cdot b\right).$$

  Algorithm $\mathcal{B}$ gives ct to $\mathcal{A}$.
- **Output phase:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which $\mathcal{B}$ also outputs.

By construction, on every key-generation query, we have that

$$\mathbf{t} = \mathbf{Br} + \mathbf{A}_f \mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p},$$

so $C_{\mathsf{valid}}((i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r}) = 1$. We now consider two possibilities:

- Suppose the zero-knowledge challenger sampled $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and constructed the proofs π by setting $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}), \mathbf{r})$. Then, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_2^{(b)}$ and outputs 1 with probability $\Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1]$.
- Suppose the zero-knowledge challenger sampled $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$ and constructed the proof π by setting $\pi \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}))$. Then, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_3^{(b)}$ and outputs 1 with probability $\Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1]$.

We conclude that algorithm $\mathcal{B}$ breaks zero-knowledge with the same advantage ε. □

Lemma 5.18. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, and q is prime. If Π_{NIZK} satisfies simulation-sound extractability, then for all $b \in \{0,1\}$, $\mathsf{Hyb}_3^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_4^{(b)}(\mathcal{A})$ are computationally indistinguishable.*

Proof. The only difference between $\mathsf{Hyb}_3^{(b)}$ and $\mathsf{Hyb}_4^{(b)}$ is the extra check that the challenger performs when responding to the ind^{th} query ξ_{ind} to H_2. Specifically, the two experiments can only differ when the following occurs:

- Let $((c_1, f_1, \mathsf{pk}_i), \ldots, (c_N, f_N, \mathsf{pk}_N))$ be adversary $\mathcal{A}$'s challenge query and ξ^* be the challenger's input to H_2 from Eq. (5.8). Then,
$$\xi_{\mathsf{ind}} = \xi^* = (\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)),$$
where $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$. Otherwise, the output in both experiments output 0. Moreover, it is also the case that $\mathsf{IsValid}(\mathsf{pp}, i, f_i^*, \mathsf{pk}_i^*) = 1$ for all $i \in [N]$. To see this, we consider two cases:
  - Suppose $c_i \in \{1, \ldots, \mathsf{ctr}\}$. In this case, pk_i^* was obtained as the result of an honest key-generation query (on index i and function f_i^*). In both experiments, the challenger outputs 0 if $\mathsf{IsValid}(\mathsf{pp}, i, f_i^*, \mathsf{pk}_i^*) = 0$.
  - Suppose $c_i = \perp$. Then, the challenger in both experiments affirms that $\mathsf{IsValid}(\mathsf{pp}, i, f_i, \mathsf{pk}_i^*) = 1$. In this case, the challenger in both experiments sets $f_i^* = f_i$. We conclude that $\mathsf{IsValid}(\mathsf{pp}, i, f_i^*, \mathsf{pk}_i^*) = 1$.
- For each $i \in [N]$, let $\mathbf{A}_{f_i^*} = \mathsf{EvalF}(\mathbf{A}, f_i^*)$. It must be the case that there exists some index $i \in [N]$ where $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ is not contained in D_{sk} at the time algorithm $\mathcal{A}$ queried H_2 on ξ_{ind}. If not, then the challenger's behavior in the two experiments is identical. Moreover, one of the following conditions must occur for one such index $i \in [N]$ where $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ is not contained in D_{sk}:
  - $f_i^*(\mathbf{x}) \neq 1$; or
  - The extracted vector $\mathbf{r}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \pi_i^*)$ satisfies either $\mathbf{r}_i^* \notin \{0,1\}^m$ or $\mathbf{B}\mathbf{r}_i^* \neq \mathbf{t}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$.

Suppose $|\Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4^{(b)}(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. Then, it must be the case that in an execution of $\mathsf{Hyb}_3^{(b)}$, the above conditions hold with probability ε. We consider the two possibilities:

- Suppose there exists some index $i \in [N]$ where $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ is not contained in D_{sk} and $f_i^*(\mathbf{x}) \neq 1$. In this case, the challenger in $\mathsf{Hyb}_4^{(b)}$ always outputs 0. We claim that this is the case with overwhelming probability in $\mathsf{Hyb}_3^{(b)}$. Recall also that $\xi_{\mathsf{ind}} = \xi^*$. By construction of ξ^*, we now have the following:
  - Suppose $c_i \in \{1, \ldots, \mathsf{ctr}\}$. In this case, the challenger in $\mathsf{Hyb}_3^{(b)}$ sets $\mathsf{pk}_i^* = \mathsf{pk}'$ where $(i', f', \mathsf{pk}') = \mathsf{D}[c_i]$. By construction, $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$ is the public key the challenger generated when responding to a key-generation query on index i and function $f' = f_i^*$. Also, $i = i'$ as otherwise, the output in $\mathsf{Hyb}_3^{(b)}$ is also 0. In $\mathsf{Hyb}_3^{(b)}$, the challenger would then add the mapping $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ to D_{sk}. However, since $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ is not contained in D_{sk} at the time $\mathcal{A}$ made its ind^{th} query to H_2, this case can only happen if the challenger inserted the entry $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ into D_{sk} *after* algorithm $\mathcal{A}$ queried H_2 on ξ_{ind}. This can only happen if the challenger sampled $\mathbf{t}_i^*$ as the public key in one of the subsequent key-generation queries. However, in a key-generation query, the challenger samples $\mathbf{r} \xleftarrow{\text{R}} \{0,1\}^m$ and then sets $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{A}_f\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$. Since $n \geq \lambda$, $m \geq 2n \log q$, and $q > 2$ is prime, the marginal distribution of $\mathbf{B}\mathbf{r}$ is statistically close to

uniform over $\mathbb{Z}_q^n$. Thus, with overwhelming probability, it will be the case that $\mathbf{t} \neq \mathbf{t}_i^*$. Since the adversary can make at most a polynomial number of key-generation queries, we conclude by a union bound that the probability that the challenger samples $\mathbf{t}^*$ in a key-generation query (after $\mathcal{A}$ queries H_2 on ξ_{ind}) is negligible so this case occurs with negligible probability.

  - Suppose $c_i = \perp$. In this case, the challenger in $\mathsf{Hyb}_3^{(b)}$ outputs 0 if $f_i^*(\mathbf{x}) = 1$.

  We conclude that in this case, the challenger in $\mathsf{Hyb}_3^{(b)}$ also outputs 0 with overwhelming probability.

- Since the challenger's behavior in $\mathsf{Hyb}_3^{(b)}$ and $\mathsf{Hyb}_4^{(b)}$ is identical with overwhelming probability when $f_i^*(\mathbf{x}) = 0$, it must be the case that with probability $\varepsilon - \mathsf{negl}(\lambda)$, $f_i^*(\mathbf{x}) = 1$ and the extracted vector $\mathbf{r}_i^*$ satisfies either $\mathbf{r}_i^* \notin \{0,1\}^m$ or $\mathbf{t}_i^* \neq \mathbf{B}\mathbf{r}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$. In particular, this means that $C_{\mathsf{valid}}((i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \mathbf{r}_i^*) = 0$. We show below that this implies an adversary $\mathcal{B}$ that can break simulation-extractability of Π_{NIZK} with the same advantage $\varepsilon - \mathsf{negl}(\lambda)$.

We now use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the simulation-sound extractability game:

- At the beginning of the game, algorithm $\mathcal{B}$ receives a common reference string $\mathsf{crs}_{\mathsf{NIZK}}$ from the simulation-sound extractability challenger. Algorithm $\mathcal{B}$ samples an index $\mathsf{ind} \xleftarrow{\mathrm{R}} [Q_{\mathsf{ro}}]$ and starts running $\mathcal{A}$ on input the security parameter 1^λ, the policy-family parameter 1^τ, and the number of slots N.

- Algorithm $\mathcal{A}$ outputs the challenge attribute $\mathbf{x} \in \{0,1\}^{\ell(\tau)}$. Algorithm $\mathcal{B}$ now samples

$$
\begin{aligned}
(\mathbf{B}, \mathbf{T}_{\mathbf{B}}) &\leftarrow \mathsf{TrapGen}(1^n, 1^m, q), \mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{2m^2 n \times m} \\
\mathbf{T} &\leftarrow \mathsf{SamplePre}([\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}], \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}} \\ \mathbf{0} \end{smallmatrix}\right], \mathbf{I}_{2m^2} \otimes \mathbf{G}, \sigma_{\mathsf{td}}) \\
\mathbf{B}_0 &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n.
\end{aligned}
$$

  If $\|\mathbf{T}\| > \sqrt{m}\sigma_{\mathsf{td}}$, then algorithm $\mathcal{B}$ sets $\mathbf{T} = \left[\begin{smallmatrix} \mathbf{I}_{2m^2} \otimes \mathbf{T}_{\mathbf{B}} \\ \mathbf{0} \end{smallmatrix}\right]$. Let $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$. Algorithm $\mathcal{B}$ gives the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ to $\mathcal{A}$. In addition, whenever $\mathcal{A}$ queries H_1 on an index $i \in [N]$, algorithm $\mathcal{B}$ responds with $\mathbf{d}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$. Whenever algorithm $\mathcal{A}$ queries H_2 on a string $\xi \in \{0,1\}^*$, algorithm $\mathcal{B}$ responds with a string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$. Finally, algorithm $\mathcal{B}$ also initializes an empty dictionary D_{sk}.

- Whenever $\mathcal{A}$ makes a key-generation query on an index $i \in [N]$ and a function f, algorithm $\mathcal{B}$ increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$. Then it computes $\mathbf{d}_i, \mathbf{V}_\ell, \mathbf{A}, \mathbf{A}_f$ according to Eq. (5.7) and samples $\mathbf{r} \xleftarrow{\mathrm{R}} \{0,1\}^m$. Algorithm $\mathcal{B}$ computes $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{A}_f\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$ and submits $(C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}))$ to the simulation-sound extractability challenger and receives a proof π. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the public key $\mathsf{pk} = (\mathbf{t}, \pi)$ and adds the mapping $(i, \mathbf{A}_f, \mathbf{t}) \mapsto (0, \mathbf{r})$ to D_{sk} if such a mapping does not already exist. Finally, algorithm $\mathcal{B}$ also checks if $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) = 0$ and outputs 0 if so.

- When $\mathcal{A}$ makes its ind^{th} query to H_2, algorithm $\mathcal{B}$ parses $\xi_{\mathsf{ind}} = (\mathsf{pp}^*, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$, where $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$. If ξ_{ind} does not have this form or $\mathsf{pp}^* \neq \mathsf{pp}$, then algorithm $\mathcal{B}$ outputs $\perp$. Otherwise, algorithm $\mathcal{B}$ samples a random index $i^* \xleftarrow{\mathrm{R}} [N]$ and and outputs $(C_{\mathsf{valid}}, (i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_{i^*}^*), \pi_{i^*}^*)$.

- If $\mathcal{A}$ does not make ind^{th} queries to H_2 prior to the challenge phase, algorithm $\mathcal{B}$ outputs $\perp$.

Since the challenger samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$ and constructs the (simulated) proofs π as $\pi \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}))$, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_3^{(b)}$ and $\mathsf{Hyb}_4^{(b)}$ for $\mathcal{A}$. By assumption, with probability $\varepsilon - \mathsf{negl}(\lambda)$, the query $\xi_{\mathsf{ind}} = (\mathsf{pp}^*, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$ has the property that there exists an index $i \in [N]$ where

- $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \notin \mathsf{D}_{\mathsf{sk}}$. This means algorithm $\mathcal{B}$ did not query for a proof on $(i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*)$.

- $\mathsf{IsValid}(\mathsf{pp}^*, i, f_i^*, \mathsf{pk}_i^*) = 1$ which means $\mathsf{NIZK.Verify}\big(\mathsf{crs}_{\mathsf{NIZK}}, (i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \pi^*\big) = 1$.

- The extracted vector $\mathbf{r}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \pi_i^*)$ satisfies either $\mathbf{r}_i^* \notin \{0,1\}^m$ or $\mathbf{t}_i^* \neq \mathbf{B}\mathbf{r}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$. In particular, this means $C_{\mathsf{valid}}((i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \mathbf{r}^*) = 0$.

Observe that these precisely coincide with the winning conditions in the simulation-extractability game. Thus, whenever $i^* = i$, then algorithm $\mathcal{B}$ successfully breaks simulation-extractability. Since algorithm $\mathcal{B}$ samples $i^* \xleftarrow{\mathsf{R}} [N]$, algorithm $\mathcal{B}$ breaks simulation-extractability with advantage at least $(\varepsilon - \mathsf{negl}(\lambda))/N$, which is non-negligible since $N = \mathsf{poly}(\lambda)$. □

Lemma 5.19. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, and $\sigma_{\mathsf{td}} \geq O(m^3 \log m)$. Then, for all $b \in \{0,1\}$, $\mathsf{Hyb}_4^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_5^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. The lemma follows from Lemma 3.5. Specifically, the claim follows from Lemma 3.5 as long as

$$\sigma_{\mathsf{td}} \geq (2m^3 + m)\log(2m^2 n) = O(m^3 \log m).$$

□

Lemma 5.20. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, and $q > 2$ is prime. Then, for all $b \in \{0,1\}$, $\mathsf{Hyb}_5^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_6^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. This follows by the leftover hash lemma (Lemma 3.1). First, consider the distribution of $\mathbf{B}_0$ in the two experiments. In $\mathsf{Hyb}_5^{(b)}$, the challenger samples $\mathbf{B}_0 \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}$ whereas in $\mathsf{Hyb}_6^{(b)}$, the challenger samples $\mathbf{R}_0 \xleftarrow{\mathsf{R}} \{0,1\}^{m \times m}$ and sets $\mathbf{B}_0 = \mathbf{B}\mathbf{R}_{\mathbf{B}_0} - \mathbf{C}_{\mathbf{x}}$. We claim that these two distributions are statistically close:

- First, suppose we sample $\mathbf{B}_0 = \mathbf{B}_0^* - \mathbf{C}_{\mathbf{x}}$ where $\mathbf{B}_0^* \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}$. Since $\mathbf{B}_0^*$ is sampled independently of $\mathbf{C}_{\mathbf{x}}$, the distribution of $\mathbf{B}_0$ remains uniform over $\mathbb{Z}_q^{n \times m}$.
- By Lemma 3.1, the distributions

$$(\mathbf{B}, \mathbf{B}\mathbf{R}_{\mathbf{B}_0}, \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0}) \quad \text{and} \quad (\mathbf{B}, \mathbf{B}_0^*, \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0})$$

  are statistically indistinguishable when $\mathbf{B} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{R}_{\mathbf{B}_0} \xleftarrow{\mathsf{R}} \{0,1\}^{m \times m}$, and $\mathbf{B}_0^* \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}$.

Combining the above statements, we conclude that the distribution of $\mathbf{B}_0$ in the two experiments are statistically indistinguishable. Next, consider the distribution of $\mathbf{p}$. By Lemma 3.1, the distributions

$$(\mathbf{B}, \mathbf{B}\mathbf{r}_{\mathbf{p}}, \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}}) \quad \text{and} \quad (\mathbf{B}, \mathbf{p}, \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}})$$

are statistically indistinguishable when $\mathbf{B} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{r}_{\mathbf{p}} \xleftarrow{\mathsf{R}} \{0,1\}^m$, and $\mathbf{p} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^n$. The left distribution maps to $\mathsf{Hyb}_6^{(b)}$ while the right distribution maps to $\mathsf{Hyb}_5^{(b)}$. Finally, consider the key-generation queries:

- In $\mathsf{Hyb}_5^{(b)}$, the challenger sets $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{A}_f\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$ where $\mathbf{r} \xleftarrow{\mathsf{R}} \{0,1\}^m$.
- In $\mathsf{Hyb}_6^{(b)}$, the challenger sets $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{d}_i$ where $\mathbf{r} \xleftarrow{\mathsf{R}} \{0,1\}^m$.

By the Lemma 3.1, the distribution of $(\mathbf{B}, \mathbf{B}\mathbf{r})$ is statistically indistinguishable from $(\mathbf{B}, \mathbf{t}^*)$ where $\mathbf{t}^* \xleftarrow{\mathsf{R}} \mathbb{Z}_q^n$. The claim now follows via a similar argument as used to analyze $\mathbf{B}_0$. □

Lemma 5.21. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, and $\sigma_{\mathsf{td}} \cdot O(m^{25/2} N^3) < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the statistical distance between $\mathsf{Hyb}_6^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{7,\kappa}^{(b)}(\mathcal{A})$ is at most $1/\kappa + \mathsf{negl}(\lambda)$.*

Proof. By construction, in $\mathsf{Hyb}_{7,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{8,\kappa}^{(b)}(\mathcal{A})$, the challenger samples $\mathbf{B} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}$ and sets $\mathbf{T}$ such that $\|\mathbf{T}\| \leq \sqrt{m}\sigma_{\mathsf{td}}$. The claim now follows by Theorem 5.9 (specifically, the explainability property). □

Lemma 5.22. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, and $\sigma_{\mathsf{td}} \cdot O(m^{25/2} N^3) < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{7,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{8,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. By construction, in $\mathsf{Hyb}_{7,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{8,\kappa}^{(b)}(\mathcal{A})$, the challenger samples $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}$ and sets $\mathbf{T}$ such that $\|\mathbf{T}\| \leq \sqrt{m}\sigma_{\mathsf{td}}$. The claim now follows by Theorem 5.9 (specifically, the sampling distribution property). □

Lemma 5.23. *Suppose $n \geq \lambda$, $m \geq 2n\log q$, and q is prime. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{8,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{9,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. By construction, the only difference between these two distributions is the distribution of $\mathbf{C}_0$ and $\mathbf{z}_{0,i}$ for $i \in [N]$. We show that these components are statistically indistinguishable:

- Since $\mathbf{B} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}$ and $\mathbf{R}_{\widehat{\mathbf{C}}} \xleftarrow{\mathrm{R}} \{0,1\}^{m\times m}$, the distribution of $\mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$ by the leftover hash lemma (Lemma 3.1). Thus, the distribution of $\mathbf{C}_0 = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} - \sum_{j\in[N]} \mathbf{C}_j^*$ in $\mathsf{Hyb}_{9,\kappa}^{(b)}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$, which coincides with the distribution of $\mathbf{C}_0$ in $\mathsf{Hyb}_{8,\kappa}^{(b)}$
- Consider $\mathbf{z}_{0,i}$ for an index $i \in [N]$ where $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (0, \mathbf{r}_i^*)$ in D_{sk}. By construction, this means $\mathbf{t}_i^* = \mathbf{d}_i + \mathbf{B}\mathbf{r}_i^*$ where $\mathbf{r}_i^* \in \{0,1\}^m$. Now,
$$\mathbf{C}_0\mathbf{v}_i = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \sum_{j\in[N]} \mathbf{C}_j^*\mathbf{v}_i.$$
Since $\mathbf{C}_j^*$ is a matrix commitment to $\mathbf{u}_j^{\mathsf{T}} \otimes \mathbf{t}_j^*$ and $\mathbf{Z}_j^*$ is the associated opening, we appeal to Lemma 3.8 to conclude that
$$\begin{aligned} \forall j \neq i : \mathbf{C}_j^*\mathbf{v}_i &= -\mathbf{B}\mathbf{z}_{j,i}^* \\ \mathbf{C}_i^*\mathbf{v}_i &= \mathbf{t}_i^* - \mathbf{B}\mathbf{z}_{i,i}^* \\ &= \mathbf{d}_i + \mathbf{B}\mathbf{r}_i^* - \mathbf{B}\mathbf{z}_{i,i}^* \end{aligned}$$
Thus, in $\mathsf{Hyb}_{9,\kappa}^{(b)}$, we have
$$\begin{aligned} \mathbf{C}_0\mathbf{v}_i &= \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{d}_i - \mathbf{B}\mathbf{r}_i^* + \sum_{j\in[N]} \mathbf{B}\mathbf{z}_{j,i}^* \\ &= -\mathbf{d}_i + \mathbf{B}\left(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + \sum_{j\in[N]} \mathbf{z}_{j,i}^*\right). \end{aligned}$$
In this case, the distribution of $\mathbf{z}_{0,i}$ in $\mathsf{Hyb}_{9,\kappa}^{(b)}$ is precisely $\mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}(-\mathbf{C}_0\mathbf{v}_i)$, which matches the distribution in $\mathsf{Hyb}_{8,\kappa}^{(b)}$.
- Consider $\mathbf{z}_{0,i}$ for an index $i \in [N]$ where $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ in D_{sk}. By construction, this means $\mathbf{B}\mathbf{r}_i^* = \mathbf{t}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$ and moreover $f_i^*(\mathbf{x}) = 1$. Let $\mathbf{Z}_{\mathbf{x}} = \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$. Using the fact that $\mathbf{A} = -\mathbf{B}_0\mathbf{V}_{\ell m}$, we have
$$[\mathbf{B} \mid \mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}] \cdot \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} = -\mathbf{B}\mathbf{Z}_{\mathbf{x}} - \mathbf{B}_0\mathbf{V}_{\ell m} - \mathbf{C}_{\mathbf{x}}\mathbf{V}_{\ell m} = \mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}.$$
Let $\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}} = \mathsf{EvalFX}(\mathbf{A}, f_i^*, \mathbf{x})$. Using the fact that $\mathbf{B}_0 = \mathbf{B}\mathbf{R}_{\mathbf{B}_0} - \mathbf{C}_{\mathbf{x}}$, we now have
$$\mathbf{B}[\mathbf{I}_m \mid \mathbf{R}_{\mathbf{B}_0}] \cdot \begin{bmatrix} -\mathbf{Z}_{\mathbf{x}} \\ -\mathbf{V}_{\ell m} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}} = (\mathbf{A} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}} = \mathbf{A}_{f_i^*} - f_i^*(\mathbf{x}) \cdot \mathbf{G} = \mathbf{A}_{f_i^*} - \mathbf{G}.$$
This means we can write
$$\mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) = \mathbf{d}_i + \mathbf{B}(-\mathbf{Z}_{\mathbf{x}} - \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i). \tag{5.9}$$

Using the fact that $\mathbf{p} = \mathbf{B}\mathbf{r}_{\mathbf{p}}$, we now have

$$\begin{aligned}
\forall j \neq i : \mathbf{C}_j^* \mathbf{v}_i &= -\mathbf{B}\mathbf{z}_{j,i}^* \\
\mathbf{C}_i^* \mathbf{v}_i &= \mathbf{t}_i^* - \mathbf{B}\mathbf{z}_{i,i}^* \\
&= \mathbf{B}\mathbf{r}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p} - \mathbf{B}\mathbf{z}_{i,i}^* \\
&= \mathbf{B}\mathbf{r}_i^* + \mathbf{d}_i + \mathbf{B}(-\mathbf{Z}_{\mathbf{x}} - \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{B}\mathbf{r}_{\mathbf{p}} - \mathbf{B}\mathbf{z}_{i,i}^* \\
&= \mathbf{d}_i + \mathbf{B}(\mathbf{r}_i^* - (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{r}_{\mathbf{p}} - \mathbf{z}_{i,i}^*).
\end{aligned}$$

In this case,

$$\mathbf{C}_0\mathbf{v}_i = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \sum_{j\in[N]} \mathbf{C}_j^*\mathbf{v}_i = -\mathbf{d}_i + \mathbf{B}\left(\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i - \mathbf{r}_i^* + (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) - \mathbf{r}_{\mathbf{p}} + \sum_{j\in[N]} \mathbf{z}_{j,i}^*\right).$$

Thus the distribution of $\mathbf{z}_{0,i}$ in this case in $\mathsf{Hyb}_{9,\kappa}^{(b)}$ is precisely $\mathbf{B}_{\sigma_{\mathrm{agg}}}^{-1}(-\mathbf{C}_0\mathbf{v}_i)$, which is the distribution in $\mathsf{Hyb}_{8,\kappa}^{(b)}$.

Since the distribution of $\mathbf{C}_0$ is statistically indistinguishable between $\mathsf{Hyb}_{8,\kappa}^{(b)}$ and $\mathsf{Hyb}_{9,\kappa}^{(b)}$ and $\mathbf{z}_{0,i}$ is constructed using identical procedures, the two distributions are statistically indistinguishable. □

Lemma 5.24. *Suppose $\sigma_{\mathrm{agg}} > \lambda^{\omega(1)} \cdot \sigma_{\mathrm{td}} \cdot m^{O(d)} \log q \cdot (N \log N + \log(\ell m))$. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{9,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{10,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. This follows by the Gaussian preimage smudging lemma (Lemma 5.10), as long as σ_{agg} is sufficiently large. In the following analysis, it suffices to consider the setting where $\|\mathbf{T}\| \leq \sqrt{m}\sigma_{\mathrm{td}}$. Otherwise, the output in both experiments is 0. In $\mathsf{Hyb}_{10,\kappa}^{(b)}$, each $\mathbf{z}_{0,i}$ can be written as $\mathbf{z}_{0,i} = \tilde{\mathbf{z}}_{0,i} + \mathbf{z}'_{0,i}$. By Lemma 5.10, if $\sigma_{\mathrm{agg}} > \lambda^{\omega(1)} \cdot \sqrt{m}\|\mathbf{z}'_{0,i}\|$, then the distributions of $\mathbf{z}_{0,i}$ in $\mathsf{Hyb}_{9,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{10,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable. The claim then holds by a hybrid argument over all $N = \mathrm{poly}(\lambda)$ indices i. It suffices to analyze $\|\mathbf{z}'_{0,i}\|$ for each $i \in [N]$:

- Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (0, \mathbf{r}_i^*)$ in D_{sk}. In this case, $\mathbf{z}'_{0,i} = -\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - \sum_{j\in[N]} \mathbf{z}_{j,i}^*$. By Lemma 3.8,

$$\begin{aligned}
\|\mathbf{v}_i\| &\leq O(\|\mathbf{T}\| \cdot m^4 \log q) = O(\sigma_{\mathrm{td}} \cdot m^{9/2} \log q) \\
\left\|\mathbf{z}_{j,i}^*\right\| &\leq O(\|\mathbf{T}\| \cdot m^7 \log q \log N) \leq O(\sigma_{\mathrm{td}} \cdot m^{15/2} \log q \log N).
\end{aligned} \tag{5.10}$$

  Since $\mathbf{R}_{\widehat{\mathbf{C}}} \in \{0,1\}^{m\times m}, \mathbf{r}_i^* \in \{0,1\}^m$, we conclude that

$$\|\mathbf{z}'_{0,i}\| = \left\|-\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - \textstyle\sum_{j\in[N]} \mathbf{z}_{j,i}^*\right\| \leq O(N\sigma_{\mathrm{td}} \cdot m^{15/2} \log q \log N).$$

- Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ in D_{sk}. In this case,

$$\mathbf{z}'_{0,i} = -\mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{r}_{\mathbf{p}} - \sum_{j\in[N]} \mathbf{z}_{j,i}^*.$$

  By Theorem 3.6, $\|\mathbf{H}_{\mathbf{A},f_i^*,\mathbf{x}}\| \leq m^{O(d)}$. By Lemma 3.8,

$$\begin{aligned}
\|\mathbf{V}_{\ell m}\| &\leq O(\|\mathbf{T}\| \cdot m^4 \log q) = O(\sigma_{\mathrm{td}} \cdot m^{9/2} \log q) \\
\|\mathbf{Z}_{\mathbf{x}}\| &\leq O(\|\mathbf{T}\| \cdot m^7 \log q \log(\ell m)) = O(\sigma_{\mathrm{td}} \cdot m^{15/2} \log q \log(\ell m)).
\end{aligned}$$

  Next, $\mathbf{R}_{\mathbf{B}_0} \in \{0,1\}^{m\times m}$. Combined with Eq. (5.10), we can bound

$$\left\|\mathbf{z}'_{0,i}\right\| \leq \sigma_{\mathrm{td}} \cdot m^{O(d)} \log q \cdot (N \log N + \log(\ell m)).$$

If $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot \sigma_{\mathsf{td}} \cdot m^{O(d)} \log q \cdot (N \log N + \log(\ell m))$, then $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot \sqrt{m} \|\mathbf{z}_{0,i'}\|$ for all $i \in [N]$ and the claim holds. □

Lemma 5.25. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} > \log m$. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{10,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{11,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. The only difference between these two experiments is the distribution of $(\tilde{\mathbf{z}}_{0,i}, \mathbf{d}_i)$ for each $i \in [N]$. In $\mathsf{Hyb}_{10,\kappa}^{(b)}$, the challenger samples $\mathbf{d}_i \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and $\tilde{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{d}_i)$ whereas in $\mathsf{Hyb}_{11,\kappa}^{(b)}$, the challenger samples $\tilde{\mathbf{z}}_{0,i} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}^m$ and sets $\mathbf{d}_i = \mathbf{B}\tilde{\mathbf{z}}_{0,i}$. Under the given conditions, these two distributions are statistically indistinguishable by Lemma 3.3. □

Lemma 5.26. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, q is prime, $\chi \geq \log m$, and $\sigma_{\mathsf{td}} \geq O(\log m)$. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{11,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{12,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. Follows immediately from Lemmas 3.2 and 3.4. □

Lemma 5.27. *Suppose the $(2m^2, \sigma_{\mathsf{td}})$-succinct LWE assumption with lattice parameters (n, m, q, χ) holds. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{12,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{13,\kappa}^{(b)}(\mathcal{A})$ is computationally indistinguishable.*

Proof. Suppose $|\Pr[\mathsf{Hyb}_{12,\kappa}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{13,\kappa}^{(b)}(\mathcal{A}) = 1]| = \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ for the $(2m^2, \sigma_{\mathsf{td}})$-succinct LWE assumption:

- **Setup phase:** At the beginning of the game, algorithm $\mathcal{B}$ receives the challenge $(\mathbf{B}, \tilde{\mathbf{c}}, \mathbf{W}, \mathbf{T})$. Algorithm $\mathcal{B}$ starts running algorithm $\mathcal{A}$ on input 1^λ, 1^τ, and N. Algorithm $\mathcal{A}$ begins by outputting the attribute $\mathbf{x} \in \{0,1\}^\ell$. Algorithm $\mathcal{B}$ sets $\mathsf{pp}_{\mathsf{com}} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and then samples the following:

$$\begin{aligned}
(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) &\leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda) \\
\mathbf{C}_{\mathbf{x}} &= \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \\
\mathbf{R}_{\widehat{\mathbf{C}}}, \mathbf{R}_{\mathbf{B}_0} &\xleftarrow{\text{R}} \{0,1\}^{m \times m} \;,\; \mathbf{r}_{\mathbf{p}} \xleftarrow{\text{R}} \{0,1\}^m \\
\mathbf{B}_0 &= \mathbf{B}\mathbf{R}_{\mathbf{B}_0} - \mathbf{C}_{\mathbf{x}} \;,\; \mathbf{p} = \mathbf{B}\mathbf{r}_{\mathbf{p}}.
\end{aligned}$$

  Algorithm $\mathcal{B}$ sets $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{B}_0, \mathbf{p})$ and gives pp to $\mathcal{A}$. Algorithm $\mathcal{B}$ also initializes a counter $\mathsf{ctr} = 0$, dictionaries D, D_{sk}, and samples an index $\mathsf{ind} \xleftarrow{\text{R}} [Q_{\mathsf{ro}}]$.

- **Queries to H_1:** Whenever $\mathcal{A}$ queries H_1 on an index $i \in \mathbb{N}$, algorithm $\mathcal{B}$ samples $\tilde{\mathbf{z}}_{0,i} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}^m$ and responds with $H_1(i) = \mathbf{d}_i = \mathbf{B}\tilde{\mathbf{z}}_{0,i}$.

- **Queries to H_2:** Whenever $\mathcal{A}$ queries H_2 on an input $\xi \in \{0,1\}^*$, if this is not the ind^{th} query to H_2, algorithm $\mathcal{B}$ responds with $\gamma \xleftarrow{\text{R}} \{0,1\}^\rho$. If it is the ind^{th} query, then algorithm $\mathcal{B}$ proceeds as follows:

  - Algorithm $\mathcal{B}$ parses $\xi_{\mathsf{ind}} = (\mathsf{pp}^*, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$, where $\mathsf{pk}_i^* = (\mathbf{t}_i^*, \pi_i^*)$. If ξ_{ind} does not have this form or $\mathsf{pp}^* \neq \mathsf{pp}$, then $\mathcal{B}$ outputs 0. If $\mathsf{IsValid}(\mathsf{pp}, i, f_i^*, \mathsf{pk}_i^*) = 0$ for any $i \in [N]$, algorithm $\mathcal{B}$ outputs 0.
  - Algorithm $\mathcal{B}$ computes $\mathbf{V}_{\ell m} = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^{\ell m})$ and $\mathbf{A} = -\mathbf{B}_0 \mathbf{V}_{\ell m}$.
  - For each $i \in [N]$, algorithm $\mathcal{B}$ computes $\mathbf{A}_{f_i^*} = \mathsf{EvalF}(\mathbf{A}, f_i^*)$. If $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*)$ is not contained in D_{sk}, algorithm $\mathcal{B}$ checks that $f_i^*(\mathbf{x}) = 1$. If not, it outputs 0. Otherwise, algorithm $\mathcal{B}$ computes

$$\mathbf{r}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_{f_i^*}, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}_i^*), \pi_i^*).$$

    If $\mathbf{r}_i^* \notin \{0,1\}^m$ or $\mathbf{B}\mathbf{r}_i^* \neq \mathbf{t}_i^* + \mathbf{A}_{f_i^*}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{p}$, then algorihtm $\mathcal{B}$ outputs 0. Otherwise, the challenger adds the mapping $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ to D_{sk}.

- Algorithm $\mathcal{B}$ computes $\mathbf{V}_N = \mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, 1^N)$ and parses $\mathbf{V}_N = [\mathbf{v}_1 \mid \cdots \mid \mathbf{v}_N]$. Now, for each $i \in [N]$, algorithm $\mathcal{B}$ computes

$$\begin{aligned} \mathbf{C}_i^* &= \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i^*) \\ \mathbf{Z}_i^* &= \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\mathsf{T}} \otimes \mathbf{t}_i^*) \end{aligned}$$

  It parses $\mathbf{Z}_i^* = [\mathbf{z}_{i,1}^* \mid \cdots \mid \mathbf{z}_{i,N}^*]$ and sets $\mathbf{C}_0 = \mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} - \sum_{j \in [N]} \mathbf{C}_j^*$. For each $i \in [N]$, algorithm $\mathcal{B}$ defines $\mathbf{z}_{0,i}$ as follows:

  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (0, \mathbf{r}_i^*)$ in D_{sk}. Then, it sets

  $$\mathbf{z}_{0,i} = \tilde{\mathbf{z}}_{0,i} - \mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - \sum_{j \in [N]} \mathbf{z}_{j,i}^*.$$

  - Suppose $(i, \mathbf{A}_{f_i^*}, \mathbf{t}_i^*) \mapsto (1, \mathbf{r}_i^*)$ in D_{sk}. Then, it computes $\mathbf{Z}_{\mathbf{x}} = \mathsf{Open}^{\mathsf{mat}}(\mathsf{pp}_{\mathsf{com}}, \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$ and $\mathbf{H}_{\mathbf{A}, f_i^*, \mathbf{x}} = \mathsf{EvalFX}(\mathbf{A}, f_i^*, \mathbf{x})$ and sets

  $$\mathbf{z}_{0,i} = \tilde{\mathbf{z}}_{0,i} - \mathbf{R}_{\widehat{\mathbf{C}}}\mathbf{v}_i + \mathbf{r}_i^* - (\mathbf{Z}_{\mathbf{x}} + \mathbf{R}_{\mathbf{B}_0}\mathbf{V}_{\ell m})\mathbf{H}_{\mathbf{A}, f_i^*, \mathbf{x}}\mathbf{G}^{-1}(\mathbf{d}_i) + \mathbf{r}_{\mathbf{p}} - \sum_{j \in [N]} \mathbf{z}_{j,i}^*.$$

  Here, $\tilde{\mathbf{z}}_{0,i} \in \mathbb{Z}_q^m$ is the value algorithm $\mathcal{B}$ sampled when responding to a query on $H_1(i)$. If algorithm $\mathcal{A}$ has not yet made a query to H_1 on some index $i \in [N]$, algorithm $\mathcal{B}$ samples $\tilde{\mathbf{z}}_{0,i} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{agg}}}^m$ and programs $H_1(i) = \mathbf{d}_i = \mathbf{B}\tilde{\mathbf{z}}_{0,i}$.

- Finally, algorithm $\mathcal{B}$ computes

  $$\gamma^* \leftarrow \mathsf{Explain}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^{\kappa}, (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}), \sigma_{\mathsf{agg}}).$$

  It responds to $\mathcal{A}$ with $H_2(\xi_{\mathsf{ind}}) = \gamma^*$.

- **Key-generation queries:** Whenever $\mathcal{A}$ makes a key-generation query on an index $i \in [N]$ and a function f, algorithm $\mathcal{B}$ samples $\mathbf{r} \xleftarrow{\mathrm{R}} \{0,1\}^m$ and sets $\mathbf{t} = \mathbf{B}\mathbf{r} + \mathbf{d}_i$. It also computes $\mathbf{A}_f = \mathsf{EvalF}(\mathbf{A}, f)$ and $\mathbf{d}_i = H_1(i)$. Then, it generates a (simulated) proof $\pi \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{valid}}, (i, \mathbf{A}_f, \mathbf{d}_i, \mathbf{p}, \mathbf{B}, \mathbf{t}))$ and responds to $\mathcal{A}$ with the public key $\mathsf{pk} = (\mathbf{t}, \pi)$. Algorithm $\mathcal{B}$ adds the mapping $\mathsf{ctr} \mapsto (i, f, \mathbf{t})$ to D and $(i, \mathbf{A}_f, \mathbf{t}) \mapsto (0, \mathbf{r})$ to D_{sk} if such a mapping does not already exist. In addition, algorithm $\mathcal{B}$ checks that $\mathsf{IsValid}(\mathsf{pp}, i, f, \mathsf{pk}) = 1$ and outputs 0 if not.

- **Challenge query:** Let $((c_1, f_1, \mathsf{pk}_i), \ldots, (c_N, f_N, \mathsf{pk}_N))$ be algorithm $\mathcal{A}$'s challenge query. For each $i \in [N]$, algorithm $\mathcal{B}$ proceeds as follows:

  - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then algorithm $\mathcal{B}$ looks up $(i', f', \mathsf{pk}') = \mathsf{D}[c_i]$ and checks that $i = i'$. If not, algorithm $\mathcal{B}$ outputs 0. Otherwise, it sets $\mathsf{pk}_i^* = \mathsf{pk}' = (\mathbf{t}_i, \pi_i)$ and $f_i^* = f'$.
  - If $c_i = \perp$, then algorithm $\mathcal{B}$ checks that $f_i(\mathbf{x}) = 1$. If so, it parses $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ and checks that $\mathsf{IsValid}(\mathsf{pp}, i, f_i, \mathsf{pk}_i) = 1$. If so, algorithm $\mathcal{B}$ sets $\mathsf{pk}_i^* = \mathsf{pk}_i$ and $f_i^* = f_i$. Otherwise, it outputs 0.

  Let $\xi^* = (\mathsf{pp}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$. If algorithm $\mathcal{A}$ has not made at least ind queries to H_2, then algorithm $\mathcal{B}$ outputs 0. Otherwise, let ξ_{ind} be the $\mathsf{ind}^{\mathrm{th}}$ query algorithm $\mathcal{A}$ made to H_2. If $\xi^* \neq \xi_{\mathsf{ind}}$, then algorithm $\mathcal{B}$ outputs 0. Otherwise, algorithm $\mathcal{B}$ constructs the challenge ciphertext as follows:

  $$\mathsf{ct} = \left(\tilde{\mathbf{c}}^{\mathsf{T}}\,,\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}}\,,\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0}\,,\ \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot b\right).$$

  The challenger gives ct to $\mathcal{A}$.

- **Output:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which algorithm $\mathcal{B}$ also outputs.

If $\mathcal{A}$ is efficient, then algorithm $\mathcal{B}$ is also efficient by construction. By definition, the succinct LWE challenger samples $\mathbf{B} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{2m^2 n \times m}$, and $\mathbf{T} \leftarrow [\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}]_\sigma^{-1}(\mathbf{I}_{2m^2} \otimes \mathbf{G})$. Thus, algorithm $\mathcal{B}$ perfectly simulates the public parameters pp according to the specification of $\mathsf{Hyb}_{12,\kappa}^{(b)}$ and $\mathsf{Hyb}_{13,\kappa}^{(b)}$. The random oracle queries and key-generation queries are also simulated exactly according to the specification of $\mathsf{Hyb}_{12,\kappa}^{(b)}$ and $\mathsf{Hyb}_{13,\kappa}^{(b)}$. Consider now the distribution of the challenge ciphertext:

- Suppose $\tilde{\mathbf{c}}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}$ where $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},\chi}^m$. Observe that when $\xi_{\mathsf{ind}} = \xi^*$, we have that
$$\mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} = \mathbf{C}_0 + \sum_{i \in [N]} \mathbf{C}_i^* = \widehat{\mathbf{C}}.$$
Similarly, by definition of $\mathbf{B}_0$ and $\mathbf{p}$, we further have
$$\begin{aligned} \mathbf{B}\mathbf{R}_{\mathbf{B}_0} &= \mathbf{B}_0 + \mathbf{C}_{\mathbf{x}} \\ \mathbf{B}\mathbf{r}_{\mathbf{p}} &= \mathbf{p}. \end{aligned}$$
Then, we can write
$$\begin{aligned} \mathbf{c} &= \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\, \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{R}_{\widehat{\mathbf{C}}} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}} \,,\, \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{R}_{\mathbf{B}_0} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0} \,,\, \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{r}_{\mathbf{p}} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot b\right) \\ &= \left(\mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}} \,,\, \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{C}} + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\widehat{\mathbf{C}}} \,,\, \mathbf{s}^{\mathsf{T}}(\mathbf{B}_0 + \mathbf{C}_{\mathbf{x}}) + \mathbf{e}^{\mathsf{T}}\mathbf{R}_{\mathbf{B}_0} \,,\, \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{r}_{\mathbf{p}} + \lfloor q/2 \rfloor \cdot b\right). \end{aligned}$$
This is the ciphertext distribution in $\mathsf{Hyb}_{12,\kappa}^{(b)}$.

- Suppose $\tilde{\mathbf{c}} \xleftarrow{R} \mathbb{Z}_p^n$. Then, the challenge ciphertext is distributed exactly as in $\mathsf{Hyb}_{13,\kappa}^{(b)}$.

We conclude that algorithm $\mathcal{B}$ breaks the succinct LWE assumption with the same advantage ε. □

Lemma 5.28. *Suppose $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot \sigma_{\mathsf{td}} \cdot m^{O(d)} \log q \cdot (N \log N + \log(\ell m))$. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{13,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{14,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. Follows by the same argument as the proof of Lemma 5.25 (via the Gaussian preimage smudging lemma). □

Lemma 5.29. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, and $q > 2$ is prime. Then, for all polynomials $\kappa = \kappa(\lambda)$ and all $b \in \{0,1\}$, $\mathsf{Hyb}_{14,\kappa}^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_{15,\kappa}^{(b)}(\mathcal{A})$ are statistically indistinguishable.*

Proof. By the leftover hash lemma (Lemma 3.1), the distributions $(\mathbf{B}, \tilde{\mathbf{c}}, \mathbf{A}\mathbf{r}_{\mathbf{p}}, \tilde{\mathbf{c}}^{\mathsf{T}}\mathbf{r}_p + b \cdot \lfloor q/2 \rfloor)$ and $(\mathbf{B}, \tilde{\mathbf{c}}, \mathbf{p}, c_4 + b \cdot \lfloor q/2 \rfloor)$ where $\mathbf{B} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $\tilde{\mathbf{c}} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{r}_p \xleftarrow{R} \{0,1\}^m$, $\mathbf{p} \xleftarrow{R} \mathbb{Z}_q^n$, and $c_4 \xleftarrow{R} \mathbb{Z}_q$ are statistically indistinguishable. The first distribution corresponds to $\mathsf{Hyb}_{14,\kappa}^{(b)}$ while the second corresponds to $\mathsf{Hyb}_{15,\kappa}^{(b)}$. □

Completing the proof. To complete the proof of Theorem 5.14, suppose
$$|\Pr[\mathsf{Hyb}_0^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0^{(1)}(\mathcal{A}) = 1]| = \varepsilon(\lambda)$$
for some non-negligible function. By Lemma 5.15, this means
$$|\Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1]| = \frac{\varepsilon(\lambda)}{Q_{\mathsf{ro}}}. \tag{5.11}$$
Since $Q_{\mathsf{ro}} = \mathsf{poly}(\lambda)$, the quantity $\varepsilon(\lambda)/Q_{\mathsf{ro}}$ is also non-negligible. Thus, there exists a polynomial $\kappa'(\lambda)$ such that for infinitely-many $\lambda \in \mathbb{N}$, $\varepsilon(\lambda)/Q_{\mathsf{ro}} > 1/\kappa'(\lambda)$. Let $\kappa(\lambda) = 3\kappa'(\lambda)$. For $i \leq 6$, we write $\mathsf{Hyb}_{i,\kappa}^{(b)}(\mathcal{A})$ to denote $\mathsf{Hyb}_i^{(b)}(\mathcal{A})$. Moreover, since the challenger's behavior in $\mathsf{Hyb}_{15}^{(b)}$ is independent of the bit $b \in \{0,1\}$,
$$\Pr[\mathsf{Hyb}_{15}^{(0)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_{15}^{(1)}(\mathcal{A}) = 1].$$

By Lemmas 5.16 to 5.29, we now have for all $\lambda \in \mathbb{N}$,

$$\begin{aligned}
|\Pr[\mathsf{Hyb}_0^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1]| \le \sum_{i=1}^{15} &|\Pr[\mathsf{Hyb}_{i,\kappa}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i+1,\kappa}^{(0)}(\mathcal{A}) = 1]| \\
&+ |\Pr[\mathsf{Hyb}_{15}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{15}^{(1)}(\mathcal{A}) = 1]| \\
&+ \sum_{i=1}^{15} |\Pr[\mathsf{Hyb}_{i+1,\kappa}^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i,\kappa}^{(1)}(\mathcal{A}) = 1]| \\
\le 2/\kappa(\lambda) + \mathsf{negl}(\lambda).&
\end{aligned}$$

However, this contradicts Eq. (5.11) which asserts that there are infinitely-many $\lambda \in \mathbb{N}$ where

$$|\Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1]| = \frac{\varepsilon(\lambda)}{Q_{\mathrm{ro}}} > \frac{1}{\kappa'(\lambda)} = \frac{3}{\kappa(\lambda)}.$$

Hence, we conclude that it must be the case that $\varepsilon(\lambda)$ is negligible, and attribute-selective security holds. □

Parameter instantiation. We now provide one example instantiation of the lattice parameters for Construction 5.11 so as to satisfy Theorems 5.13 and 5.14. Let λ be a security parameter and τ be the policy parameter. Let $d = d(\tau)$ and $\ell = \ell(\tau)$ be the bound on the policy depth and the attribute length, respectively. Let $N \le 2^\lambda$ be the number of slots. Let $\varepsilon \in (0, 1)$ be a constant. Then, we set

$$\begin{aligned}
n &= (\lambda d)^{1/\varepsilon} \cdot \mathsf{poly}(\log \lambda, \log d, \log \ell, \log N) \\
m &= nd \cdot \mathsf{poly}(\log \lambda, \log d, \log \ell, \log N) \\
\chi &= \mathsf{poly}(n, \log d, \log \ell) \\
\sigma_{\mathrm{td}} &= O(m^3 \log m) \\
\sigma_{\mathrm{agg}} &= \lambda^{\omega(1)} \cdot m^{O(d)} \cdot N^3 \log \ell \\
q &= \lambda^{\omega(1)} \cdot m^{O(d)} \cdot \mathsf{poly}(n, N, \ell, \log d) \\
\lambda_{\mathrm{DGS}} &= (\lambda + d) \cdot \mathsf{poly}(\log \lambda, \log d, \log \ell)
\end{aligned}$$

For this choice of parameters, $q < 2^{n^\varepsilon}$. In this case, security relies on the $(2m^2, \sigma_{\mathrm{td}})$-succinct LWE assumption with LWE parameters (n, m, q, χ); this corresponds to a sub-exponential modulus-to-noise ratio. We summarize our instantiation in the following corollary:

Corollary 5.30 (Slotted Key-Policy Registered ABE)**.** *Let λ be a security parameter and $N \le 2^\lambda$ be the number of slots. Let $d \le 2^\lambda$ be a depth bound and $\ell \le 2^\lambda$ be an attribute length. Then, under the $\mathsf{poly}(\lambda, d)$-succinct LWE assumption with a sub-exponential modulus-to-noise ratio, there exists a slotted key-policy registered ABE scheme with N slots where the size of the public parameters, the size of the individual public/secret keys, and the size of the ciphertext is $\mathsf{poly}(\lambda, d, \log N, \log \ell)$.*

Remark 5.31 (Transparent Setup via Decomposed LWE)**.** Similar to Remark 4.13, we can replace the matrix commitment scheme in Construction 5.11 with the alternative instantiation from [Wee25, Appendix C] based on decomposed LWE. This yields a key-policy registered ABE scheme that supports an unbounded number of users with a *transparent* setup.

Acknowledgments

This work was initiated at Shonan Meeting No.195 “Workshop on Encrypted Computation / Enhancing Functionality in Cryptography.” David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Microsoft Research Faculty Fellowship, and a Google Research Scholar award.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, 1996.

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: fully succinct randomised encodings and more. In *CRYPTO*, 2025.

[AT24] Nuttapong Attrapadung and Junichi Tomida. A modular approach to registered abe for unbounded predicates. In *CRYPTO*, 2024.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, 2004.

[BCD$^+$25] Pedro Branco, Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta, and Akshayaram Srinivasan. Black-box non-interactive zero knowledge from vector trapdoor hash. In *EUROCRYPT*, 2025.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In *ASIACRYPT*, 2018.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

[BGG$^+$14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *EUROCRYPT*, 2014.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, 2005.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *CRYPTO*, 2014.

[CHW25] Jeffrey Champion, Yao-Ching Hsieh, and David J. Wu. Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In *CRYPTO*, 2025.

[CW24] Jeffrey Champion and David J. Wu. Distributed broadcast encryption from lattices. In *TCC*, 2024.

[DDO$^+$01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, 2001.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1), 2008.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, 1990.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, 1993.

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In *CRYPTO*, 2023.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In *TCC*, 2018.

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In *CRYPTO*, 2024.

[GLWW24] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Reducing the CRS size in registered ABE systems. In *CRYPTO*, 2024.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, 2013.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, 2009.

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In *EUROCRYPT*, 2023.

[HW15] Pavel Hubácek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In *ITCS*, 2015.

[HWW25] Yao-Ching Hsieh, Brent Waters, and David J. Wu. A generic approach to adaptively-secure broadcast encryption in the plain model. In *EUROCRYPT*, 2025.

[KMW23] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. In *ASIACRYPT*, 2023.

[LW22] George Lu and Brent Waters. How to sample a discrete gaussian (and more) from a random oracle. In *TCC*, 2022.

[LWW25] George Lu, Brent Waters, and David J. Wu. Multi-authority registered attribute-based encryption. In *EUROCRYPT*, 2025.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, 2019.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[RY07] Thomas Ristenpart and Scott Yilek. The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In *EUROCRYPT*, 2007.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.

[Wat24] Brent Waters. A new approach for non-interactive zero-knowledge from learning with errors. In *STOC*, 2024.

[Wee24] Hoeteck Wee. Circuit ABE with $\mathsf{poly}(\mathsf{depth}, \lambda)$-sized ciphertexts and keys from lattices. In *CRYPTO*, 2024.

[Wee25] Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. In *EUROCRYPT*, 2025.

[WQZD10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption. In *ACM CCS*, 2010.

[WW23] Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. In *EUROCRYPT*, 2023.

[WWW25] Brent Waters, Hoeteck Wee, and David J. Wu. New techniques for preimage sampling: Improved NIZKs and more from LWE. In *EUROCRYPT*, 2025.

[ZZC+25] Ziqi Zhu, Kai Zhang, Zhili Chen, Junqing Gong, and Haifeng Qian. Black-box registered ABE from lattices. *IACR Cryptol. ePrint Arch.*, 2025. Available at https://eprint.iacr.org/2025/051.pdf.

[ZZGQ23] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered ABE via predicate encodings. In *ASIACRYPT*, 2023.

A The [Wee25] Matrix Commitment Scheme

In this section, we recall the matrix commitment scheme from [Wee25, §3.2]. Let n, m, q be lattice parameters where $m \geq 2n \log q$. In the following, we write

$$\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T}) \quad \text{where} \quad [\mathbf{I}_{2m^2} \otimes \mathbf{B} \mid \mathbf{W}] \cdot \mathbf{T} = \mathbf{I}_{2m^2} \otimes \mathbf{G},$$

where $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{W} \in \mathbb{Z}_q^{2m^2 n \times m}$, and $\mathbf{T} \in \mathbb{Z}_q^{(2m^2+1)m \times 2m^3}$. Next, we define the Split function that takes as input (pp, L) where $L \leq 2m$ and outputs submatrices $\mathbf{W}_L, \mathbf{T}_L$ of $\mathbf{W}, \mathbf{T}$, respectively, such that

$$[\mathbf{I}_{Lm} \otimes \mathbf{B} \mid \mathbf{W}_L] \cdot \mathbf{T}_L = \mathbf{I}_{Lm} \otimes \mathbf{G}.$$

Specifically, the Split function operates as follows:

- $\mathsf{Split}(\mathsf{pp}, L)$: On input $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and $L \leq 2m$, parse

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}^{(1)} \\ \vdots \\ \mathbf{W}^{(2m^2)} \end{bmatrix} \quad \text{and} \quad \mathbf{T} = \left[\begin{array}{c||c} \mathbf{T}_{\text{LT}}^{(1)} & \\ \vdots & \\ \mathbf{T}_{\text{LT}}^{(2m^2)} & \mathbf{T}_{\text{RT}} \\ \underline{\mathbf{T}}_{\text{LT}} & \end{array}\right],$$

where $\mathbf{W}^{(i)} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T}_{\text{LT}}^{(i)}, \underline{\mathbf{T}}_{\text{LT}} \in \mathbb{Z}_q^{m \times Lm^2}$, and $\mathbf{T}_{\text{RT}} \in \mathbb{Z}_q^{(2m^2+1)m \times (2m^2 - Lm)m}$. The split function outputs

$$\mathbf{W}_L := \begin{bmatrix} \mathbf{W}^{(1)} \\ \vdots \\ \mathbf{W}^{(Lm)} \end{bmatrix} \in \mathbb{Z}_q^{Lmn \times m} \quad \text{and} \quad \mathbf{T}_L := \begin{bmatrix} \mathbf{T}_{\text{LT}}^{(1)} \\ \vdots \\ \mathbf{T}_{\text{LT}}^{(Lm)} \\ \underline{\mathbf{T}}_{\text{LT}} \end{bmatrix} \in \mathbb{Z}_q^{(Lm+1)m \times Lm^2}.$$

In the following, we parse $\mathbf{T}_L \in \mathbb{Z}_q^{(Lm+1)m \times Lm^2}$ as $\mathbf{T}_L := \begin{bmatrix} \overline{\mathbf{T}}_L \\ \underline{\mathbf{T}}_L \end{bmatrix}$ where $\overline{\mathbf{T}}_L \in \mathbb{Z}_q^{Lm^2 \times Lm^2}$ and $\underline{\mathbf{T}}_L \in \mathbb{Z}_q^{m \times Lm^2}$. In addition, define

$$\mathsf{bits}(\mathbf{M}) := \mathsf{vec}(\mathbf{G}^{-1}(\mathbf{M})) \in \mathbb{Z}_q^{Lm}.$$

Let $\mathbf{J}_L \in \{0,1\}^{Lm^2 \times L\lceil \log q \rceil}$ be the fixed matrix from [Wee25, Lemma 4] where for all $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$,

$$(\mathsf{bits}(\mathbf{M}) \otimes \mathbf{G}) \cdot \mathbf{J}_L = \mathbf{M} \cdot \mathbf{G}_L.$$

The matrix commitment from [Wee25] is recursive, where the base case corresponds to committing to a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ where $L \leq 2m$. Without loss of generality, when $L > 2m$, we always pad the width of $\mathbf{M}$ (with zeroes) to a value of L where $L = 2^k \cdot \ell$, for some $k \in \mathbb{N}$ and $\ell \in [2m]$. The associated verification matrix and openings can then be derived by truncating the corresponding matrices for the padded matrix. Specifically, if $\mathbf{C} \cdot [\mathbf{V}_{\text{LT}} \mid \mathbf{V}_{\text{RT}}] = [\mathbf{M}_{\text{LT}} \mid \mathbf{M}_{\text{RT}}] - \mathbf{B} \cdot [\mathbf{Z}_{\text{LT}} \mid \mathbf{Z}_{\text{RT}}]$, then $\mathbf{C} \cdot \mathbf{V}_{\text{LT}} = \mathbf{M}_{\text{LT}} - \mathbf{B} \cdot \mathbf{Z}_{\text{LT}}$.

- $\mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$: On input $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$:
  - If $L \leq 2m$, output $\mathbf{C} = (\mathsf{bits}(\mathbf{M})^{\intercal} \otimes \mathbf{I}_n)\mathbf{W}_L \in \mathbb{Z}_q^{n \times m}$ where $(\mathbf{W}_L, \mathbf{T}_L) \leftarrow \mathsf{Split}(\mathsf{pp}, L)$.
  - If $L = 2^k \cdot \ell$ for $k \in \mathbb{N}$ and $\ell \in [2m]$, parse $\mathbf{M} = [\mathbf{M}_0 \mid \mathbf{M}_1]$ where $\mathbf{M}_\beta \in \mathbb{Z}_q^{n \times L/2}$. For $\beta \in \{0, 1\}$, compute $\mathbf{C}_\beta = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_\beta) \in \mathbb{Z}_q^{n \times m}$. Output $\mathbf{C} = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, [\mathbf{C}_0, \mid \mathbf{C}_1]) \in \mathbb{Z}_q^{n \times m}$.
- $\mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^L)$: On input $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and $L \in \mathbb{N}$:
  - If $L \leq 2m$, output $\mathbf{V}_L = \underline{\mathbf{T}}_L \cdot \mathbf{J}_L \in \mathbb{Z}_q^{m \times L\lceil \log q \rceil}$ where $(\mathbf{W}_L, \mathbf{T}_L) \leftarrow \mathsf{Split}(\mathsf{pp}, L)$.
  - If $L = 2^k \cdot \ell$ for $k \in \mathbb{N}$ and $\ell \in [2m]$, compute $\mathbf{V}_{L/2} = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^{L/2})$ and $\mathbf{V}_{2m} = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^{2m})$. Output $\mathbf{V}_L = \mathbf{V}_{2m}(\mathbf{I}_2 \otimes \mathbf{G}_m^{-1}(\mathbf{V}_{L/2})) \in \mathbb{Z}_q^{m \times L\lceil \log q \rceil}$.
- $\mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$: On input $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$:
  - If $L \leq 2m$, output $\mathbf{Z} = (\mathsf{bits}(\mathbf{M})^{\intercal} \otimes \mathbf{I}_m)\overline{\mathbf{T}}_L \cdot \mathbf{J}_L \in \mathbb{Z}_q^{m \times L\lceil \log q \rceil}$ where $(\mathbf{W}_L, \mathbf{T}_L) \leftarrow \mathsf{Split}(\mathsf{pp}, L)$.
  - If $L = 2^k \cdot \ell$ for $k \in \mathbb{N}$ and $\ell \in [2m]$, parse $\mathbf{M} = [\mathbf{M}_0 \mid \mathbf{M}_1]$ where $\mathbf{M}_\beta \in \mathbb{Z}_q^{n \times L/2}$. For $\beta \in \{0, 1\}$, compute $\mathbf{Z}_\beta = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_\beta)$ and $\mathbf{C}_\beta = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_\beta)$. In addition, compute $\mathbf{V}_{L/2} = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^{L/2})$. Finally, compute $\mathbf{Z}' = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, [\mathbf{C}_0 \mid \mathbf{C}_1])$ and output $\mathbf{Z} = \mathbf{Z}' \cdot (\mathbf{I}_2 \otimes \mathbf{G}_m^{-1}(\mathbf{V}_{L/2})) + [\mathbf{Z}_0 \mid \mathbf{Z}_1] \in \mathbb{Z}_q^{m \times L\lceil \log q \rceil}$.

A.1 Committing to Sparse Matrices and Supporting Local Openings

In this section, we give a proof of Lemma 3.10 and show that the [Wee25] matrix commitment scheme supports efficient commitment to sparse matrices, and moreover, there is an efficient algorithm to locally compute the columns of the verification matrix and the opening matrix.

Committing to sparse matrices. To support commitment to sparse matrices, we first observe that for all $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and all $L \in \mathbb{N}$, $\mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{0}^{n \times L}) = \mathbf{0}^{n \times m}$. Thus, we can define the $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$ algorithm as follows:

- $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$: On input $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$ and a sparse matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$:
  - If $\mathbf{M} = \mathbf{0}^{n \times L}$, return $\mathbf{0}^{n \times m}$.
  - If $L \leq 2m$, output $\mathbf{C} = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$.
  - If $L = 2^k \cdot \ell$ for $k \in \mathbb{N}$ and $\ell \in [2m]$, parse $\mathbf{M} = [\mathbf{M}_0 \mid \mathbf{M}_1]$ where $\mathbf{M}_\beta \in \mathbb{Z}_q^{n \times L/2}$. For $\beta \in \{0, 1\}$, compute $\mathbf{C}_\beta = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_\beta) \in \mathbb{Z}_q^{n \times m}$. Output $\mathbf{C} = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, [\mathbf{C}_0, \mid \mathbf{C}_1]) \in \mathbb{Z}_q^{n \times m}$.

Since $\mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{0}^{n \times L}) = \mathbf{0}^{n \times m}$, we have that $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}) = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}) = \mathsf{Com}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$. It suffices to analyze the running time of $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$. Take any matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ where $\mathbf{M}$ contains at most K non-zero columns. Throughout, we assume that the matrix $\mathbf{M}$ (and its submatrices) are encoded in a sparse format (e.g., the description length of $\mathbf{M}$ is $\mathsf{poly}(K, m, \log q, \log L)$). We show that $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$ runs in time $\mathsf{poly}(K, m, \log q, \log L)$. It suffices to consider the case where $L > 2m$. Write $L = 2^k \cdot \ell$ where $k \in \mathbb{N}$ and $\ell \in [2m]$. Without loss of generality, suppose that $2\ell > 2m$ (if not, we can alternatively write $L = 2^{k-1} \cdot (2\ell)$). We now define a complete binary tree $\mathcal{T}$ with height k as follows:

- First, parse the matrix $\mathbf{M} = [\mathbf{M}_0 \mid \cdots \mid \mathbf{M}_{2^k - 1}]$, where $\mathbf{M}_i \in \mathbb{Z}_q^{n \times \ell}$. We associate the i^{th} leaf node (from left to right) of $\mathcal{T}$ with the sub-matrix $\mathbf{M}_i$.
- With each intermediate node σ in $\mathcal{T}$, let $\mathbf{M}_i, \mathbf{M}_{i+1}, \ldots, \mathbf{M}_j$ be the matrices associated with the leaf nodes in the subtree rooted at σ (from left to right). We associate the matrix $\mathbf{M}_\sigma = [\mathbf{M}_i \mid \cdots \mid \mathbf{M}_j]$ with σ.

We now compute the running time of $\mathsf{ComSparse}^{\mathsf{mat}}$. To do so, we associate a value t_σ with each node $\sigma \in \mathcal{T}$, where t_σ is the running time of $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_\sigma)$ and $\mathbf{M}_\sigma$ is the (sparse) matrix associated with σ. First, let $T = \mathsf{poly}(K, m, \log q, \log L)$ be the running time needed to (1) check if $\mathbf{M} = \mathbf{0}$; (2) partition $\mathbf{M} = [\mathbf{M}_0 \mid \mathbf{M}_1]$; and (3) invoke $\mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \cdot)$ on a matrix with width $2m$. This corresponds to the non-recursive components of the recursive step. Note that T also bounds the cost of a base case of the recursion. Then, we can define the running times t_σ as follows:

- If σ is associated with the zero matrix or σ is a leaf node, then $t_\sigma \leq T$.
- If σ is an internal node associated with a non-zero matrix, let σ_{LT} and σ_{RT} be its left and right children. Then, $t_\sigma \leq t_{\sigma_{\text{LT}}} + t_{\sigma_{\text{RT}}} + T$.

We now show the following inductive invariant. For a node $\sigma \in \mathcal{T}$, let K_σ denote the number of leaf nodes in the sub-tree rooted at σ that are associated with non-zero matrices, and let h_σ denote the height of the sub-tree rooted at σ (i.e., the length of the longest path in the sub-tree). Then, for all $\sigma \in \mathcal{T}$, we show that $t_\sigma \leq (1 + 2K_\sigma h_\sigma)T$. We show this inductively:

- Consider a leaf node $\sigma \in \mathcal{T}$. In this case, $h_\sigma = 0$ and $t_\sigma \leq T = (1 + 2K_\sigma h_\sigma)T$.
- Consider an internal node $\sigma \in \mathcal{T}$ that is associated with the zero matrix. Then $K_\sigma = 0$ and $t_\sigma \leq T = (1+2K_\sigma h_\sigma)T$.
- Consider an internal node $\sigma \in \mathcal{T}$ that is associated with a nonzero matrix. This means $K_\sigma \geq 1$ Then, $t_\sigma \leq t_{\sigma_{\text{LT}}} + t_{\sigma_{\text{RT}}} + T$. By definition, $h_{\sigma_{\text{LT}}} = h_{\sigma_{\text{RT}}} = h_\sigma - 1$ and $K_{\sigma_{\text{LT}}} + K_{\sigma_{\text{RT}}} = K_\sigma$. Now, by the inductive hypothesis,

$$\begin{aligned} t_\sigma &\leq t_{\sigma_{\text{LT}}} + t_{\sigma_{\text{RT}}} + T \\ &\leq (1 + 2K_{\sigma_{\text{LT}}}(h_\sigma - 1))T + (1 + 2K_{\sigma_{\text{RT}}}(h_\sigma - 1))T + T \\ &= (1 + 2K_\sigma h_\sigma)T + 2(1 - K_\sigma)T \\ &\leq (1 + 2K_\sigma h_\sigma)T, \end{aligned}$$

  since $1 - K_\sigma \leq 0$.

By induction on $\mathcal{T}$, we conclude for all $\sigma \in \mathcal{T}$, $t_\sigma \leq (1 + 2K_\sigma h_\sigma)T$. The running time of $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M})$ is then bounded by $(1 + 2Kk)T = \mathsf{poly}(K, m, \log q, \log L)$, as required.

Local access to the verification matrix. Next, we show that given pp and L, we can compute any single column of the verification matrix $\mathbf{V}_L$ in time $\mathsf{poly}(m, \log q, \log L)$. We first define a local version of $\mathsf{Ver}^{\mathsf{mx}}$:

- $\mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L, i)$: On input the public parameters pp, the length $L \in \mathbb{N}$ in *binary*, and an index $i \leq L \lceil \log q \rceil$:
  - If $L \leq 2m$, compute $\mathbf{V}_L = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^L)$ and output the i^{th} column $\mathbf{v}_{L,i}$ of $\mathbf{V}_L$.
  - If $L = 2^k \cdot \ell$ for $k \in \mathbb{N}$ and $\ell \in [2m]$, first compute $\mathbf{V}_{2m} = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^{2m}) \in \mathbb{Z}_q^{m \times 2m \lceil \log q \rceil}$. Parse $\mathbf{V}_{2m} = [\mathbf{V}_{2m,\text{LT}} \mid \mathbf{V}_{2m,\text{RT}}]$ where $\mathbf{V}_{2m,\text{LT}}, \mathbf{V}_{2m,\text{RT}} \in \mathbb{Z}_q^{m \times m \lceil \log q \rceil}$. Then output

$$\mathbf{v}_{L,i} = \begin{cases} \mathbf{V}_{2m,\text{LT}} \cdot \mathbf{G}_m^{-1}(\mathbf{v}_{L/2,i}) & i \leq L \lceil \log q \rceil / 2 \\ \mathbf{V}_{2m,\text{RT}} \cdot \mathbf{G}_m^{-1}(\mathbf{v}_{L/2, i - L \lceil \log q \rceil / 2}) & i > L \lceil \log q \rceil / 2, \end{cases}$$

    where $\mathbf{v}_{L/2,i} = \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i)$ and $\mathbf{v}_{L/2, i - L\lceil \log q \rceil/2} = \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i - L \lceil \log q \rceil / 2)$.

First, we analyze the running time of $\mathsf{VerLocal}^{\mathsf{mx}}$. Let $T(n, m, q, L)$ be the running time of $\mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L, i)$ on any set of public parameters pp with lattice parameters (n, m, q), length L, and any index $i \leq L \lceil \log q \rceil$.

- If $L \leq 2m$, then $T(n, m, q, L) = p_1(m, \log q)$ for a fixed polynomial p_1.[3]

[3] Recall that we are only considering public parameters pp where $m \geq 2n \log q$. Thus, any polynomial dependence in the running time on n can be absorbed by a $\mathsf{poly}(m)$ factor.

- If $L > 2m$, then the running time is $p_2(m, \log q, \log L) + T(n, m, q, L/2)$ for some fixed polynomial p_2.

Thus,

$$T(n, m, q, L) \leq \log L \cdot p_2(m, \log q, \log L) + p_1(m, \log q) = \mathsf{poly}(m, \log q, \log L),$$

as required. It suffices to show that $\mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L, i)$ is correct. We proceed by (strong) induction on L.

- If $L \leq 2m$, correctness holds by construction.
- Suppose $L > 2m$. By definition of $\mathsf{Ver}^{\mathsf{mx}}$, this means

  $$\mathbf{V}_L = \mathbf{V}_{2m}(\mathbf{I}_2 \otimes \mathbf{G}_m^{-1}(\mathbf{V}_{L/2})) = [\mathbf{V}_{2m,\textsc{lt}} \mid \mathbf{V}_{2m,\textsc{rt}}] \cdot \begin{bmatrix} \mathbf{G}_m^{-1}(\mathbf{V}_{L/2}) & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_m^{-1}(\mathbf{V}_{L/2}) \end{bmatrix}.$$

  In particular, the i^{th} column $\mathbf{v}_{L,i}$ of $\mathbf{V}_L$ is then

  $$\mathbf{v}_{L,i} = \begin{cases} \mathbf{V}_{2m,\textsc{lt}} \cdot \mathbf{G}_m^{-1}(\mathbf{v}_{L/2,i}) & i \leq L \lceil \log q \rceil /2 \\ \mathbf{V}_{2m,\textsc{rt}} \cdot \mathbf{G}_m^{-1}(\mathbf{v}_{L/2,i-L\lceil \log q \rceil/2}) & i > L \lceil \log q \rceil /2, \end{cases}$$

  where $\mathbf{v}_{L/2,i}$ is the i^{th} row of $\mathbf{V}_{L/2} = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, L/2)$ and $\mathbf{v}_{L/2,i-L\lceil \log q \rceil/2}$ is the $(i - L \lceil \log q \rceil /2)^{\text{th}}$ row of $\mathbf{V}_{L/2}$. By the inductive hypothesis,

  $$\begin{aligned} \mathbf{v}_{L/2,i} &= \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i) \\ \mathbf{v}_{L/2,i-L\lceil \log q \rceil/2} &= \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i - L \lceil \log q \rceil /2) \end{aligned}$$

  The claim now follows by (strong) induction on L.

Given $\mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L, i)$, it is straightforward to construct the local version of $\mathsf{Ver}^{\mathsf{mat}}$:

- $\mathsf{VerLocal}^{\mathsf{mat}}(\mathsf{pp}, L, i)$: Output $\mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L, (i-1) \lceil \log q \rceil + 1)$.

Correctness follows by construction. Namely, $\mathsf{Ver}^{\mathsf{mat}}(\mathsf{pp}, L)$ outputs $\mathbf{V}_L \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L)$, where $\mathbf{V}_L = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, L)$. By construction, the i^{th} column of $\mathbf{G}_L^{-1}(\mathbf{I}_L)$ is the unit vector $\mathbf{u}_{(i-1)\cdot\lceil \log q \rceil+1} \in \mathbb{Z}_q^{L\lceil \log q \rceil \times L}$. Correspondingly, the i^{th} column of $\mathbf{V}_L \cdot \mathbf{G}_L^{-1}(\mathbf{I}_L)$ is then the $((i-1) \cdot \lceil \log q \rceil + 1)^{\text{th}}$-column of $\mathbf{V}_L$.

Computing local openings. Finally, combining the above two procedures, we obtain an analogous algorithm for computing any single column of the opening $\mathbf{Z} = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$ in time $\mathsf{poly}(K, m, \log q, \log L)$, where K is the number of non-zero columns of $\mathbf{M}$. We start by defining the algorithm $\mathsf{OpenLocal}^{\mathsf{mx}}$:

- $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}, i)$: On input $\mathsf{pp} = (\mathbf{B}, \mathbf{W}, \mathbf{T})$, a sparse matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$, and an index $i \leq L \lceil \log q \rceil$:
  - If $L \leq 2m$, compute $\mathbf{Z} = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$ and output the i^{th} column $\mathbf{z}_i$ of $\mathbf{Z}$.
  - If $L = 2^k \cdot \ell$ for $k \in \mathbb{N}$ and $\ell \in [2m]$, parse $\mathbf{M} = [\mathbf{M}_0 \mid \mathbf{M}_1]$ where $\mathbf{M}_\beta \in \mathbb{Z}_q^{n \times L/2}$. Then compute the following:
    * For $\beta \in \{0, 1\}$, let $\mathbf{C}_\beta = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_\beta)$. Then compute $\mathbf{Z}' = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, [\mathbf{C}_0 \mid \mathbf{C}_1])$. Parse $\mathbf{Z}' = [\mathbf{Z}'_{\textsc{lt}} \mid \mathbf{Z}'_{\textsc{rt}}]$ where $\mathbf{Z}'_{\textsc{lt}}, \mathbf{Z}'_{\textsc{rt}} \in \mathbb{Z}_q^{m \times m\lceil \log q \rceil}$.
    * Compute the vector $\hat{\mathbf{v}} \in \mathbb{Z}_q^m$ as

      $$\hat{\mathbf{v}} = \begin{cases} \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i) & i \leq L \lceil \log q \rceil /2 \\ \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i - L \lceil \log q \rceil /2) & i > L \lceil \log q \rceil /2. \end{cases}$$

    * Compute the vector $\hat{\mathbf{z}} \in \mathbb{Z}_q^m$ as

      $$\hat{\mathbf{z}} = \begin{cases} \mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_0, i) & i \leq L \lceil \log q \rceil /2 \\ \mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_1, i - L \lceil \log q \rceil /2) & i > L \lceil \log q \rceil /2. \end{cases}$$

Output the vector

$$\mathbf{z}_i = \begin{cases} \mathbf{Z}'_{\mathrm{LT}} \cdot \mathbf{G}_m^{-1}(\hat{\mathbf{v}}) + \hat{\mathbf{z}} & i \le L \lceil \log q \rceil /2 \\ \mathbf{Z}'_{\mathrm{RT}} \cdot \mathbf{G}_m^{-1}(\hat{\mathbf{v}}) + \hat{\mathbf{z}} & i > L \lceil \log q \rceil /2. \end{cases}$$

First, we analyze the running time of $\mathsf{OpenLocal}^{\mathsf{mx}}$. Let $T(n, m, q, L, K)$ be the running time of $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}, i)$ on any set of public parameters pp with lattice parameters (n, m, q), a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ with at most K non-zero columns, and any index $i \le L \lceil \log q \rceil$.

- If $L \le 2m$, then $T(n, m, q, L) = p_1(m, \log q)$ for a fixed polynomial p_1.
- If $L > 2m$, then $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}, i)$ computes the following quantities:
    - Computing $\mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_\beta)$ for $\beta \in \{0, 1\}$, requires time $\mathsf{poly}(K, m, \log q, \log L)$. Computing $\mathbf{Z}'$ from $\mathbf{C}_0$ and $\mathbf{C}_1$ requires time $\mathsf{poly}(m, \log q)$.
    - Computing the vector $\hat{\mathbf{v}}$ requires time $\mathsf{poly}(m, \log q, \log L)$.
    - Computing the vector $\hat{\mathbf{z}}$ requires time $T(n, m, q, L/2, K)$.
    - Computing the final output from $\mathbf{Z}'$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{z}}$ requires time $\mathsf{poly}(m, \log q)$.

  The total running time in this case is then $T(n, m, q, L/2, K) = p_2(K, m, \log q, \log L) + T(n, m, q, L/2, K)$, where p_2 is a fixed polynomial.

This means

$$T(n, m, q, L, K) = \log L \cdot p_2(K, m, \log q, \log L) + p_1(m, \log q) = \mathsf{poly}(K, m, \log q, \log L).$$

To complete the proof, we show that $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}, i)$ is correct. Similar to the previous case, we proceed by (strong) induction on L:

- If $L \le 2m$, correctness holds by construction.
- Suppose $L > 2m$. Then, the opening algorithm $\mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M})$ computes $\tilde{\mathbf{C}}_\beta = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_\beta)$ for $\beta \in \{0, 1\}$. and $\tilde{\mathbf{Z}}' = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, [\tilde{\mathbf{C}}_0 \mid \tilde{\mathbf{C}}_1])$. It also computes $\tilde{\mathbf{Z}}_\beta = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_\beta)$ for $\beta \in \{0, 1\}$, $\tilde{\mathbf{V}}_{L/2} = \mathsf{Ver}^{\mathsf{mx}}(\mathsf{pp}, 1^{L/2})$, and finally, sets $\tilde{\mathbf{Z}} = \tilde{\mathbf{Z}}' \cdot (\mathbf{I}_2 \otimes \tilde{\mathbf{G}^{-1}}_m(\mathbf{V}_{L/2})) + [\tilde{\mathbf{Z}}_0 \mid \tilde{\mathbf{Z}}_1]$. Consider the i^{th} column $\tilde{\mathbf{z}}_i$ of $\tilde{\mathbf{Z}}$. We can write it as

  $$\tilde{\mathbf{z}}_i = \begin{cases} \tilde{\mathbf{Z}}'_{\mathrm{LT}} \cdot \mathbf{G}_m^{-1}(\tilde{\mathbf{v}}_{L/2,i}) + \tilde{\mathbf{z}}_{0,i} & i \le L \lceil \log q \rceil /2 \\ \tilde{\mathbf{Z}}'_{\mathrm{RT}} \cdot \mathbf{G}_m^{-1}(\tilde{\mathbf{v}}_{L/2,i-L\lceil \log q \rceil/2}) + \tilde{\mathbf{z}}_{1,i-L\lceil \log q \rceil/2} & i > L \lceil \log q \rceil /2, \end{cases}$$

  where $\tilde{\mathbf{Z}} = [\mathbf{Z}'_{\mathrm{LT}} \mid \tilde{\mathbf{Z}}'_{\mathrm{RT}}]$, $\tilde{\mathbf{v}}_{L/2,i}$ denotes the i^{th} column of $\tilde{\mathbf{V}}_{L/2}$ and $\tilde{\mathbf{z}}_{0,i}, \tilde{\mathbf{z}}_{1,i}$ denote the i^{th} column of $\tilde{\mathbf{Z}}_0$ and $\tilde{\mathbf{Z}}_1$, respectively. Consider now $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}, i)$. By correctness of $\mathsf{ComSparse}^{\mathsf{mat}}$, we have that

  $$\mathbf{C}_\beta = \mathsf{ComSparse}^{\mathsf{mat}}(\mathsf{pp}, \mathbf{M}_\beta) = \mathsf{Com}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_\beta) = \tilde{\mathbf{C}}_\beta$$

  for $\beta \in \{0, 1\}$. This means

  $$\mathbf{Z}' = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, [\mathbf{C}_0 \mid \mathbf{C}_1]) = \mathsf{Open}^{\mathsf{mx}}(\mathsf{pp}, [\tilde{\mathbf{C}}_0 \mid \tilde{\mathbf{C}}_1]) = \tilde{\mathbf{Z}}'.$$

  By correctness of $\mathsf{VerLocal}^{\mathsf{mx}}$, we have

  $$\begin{aligned} \hat{\mathbf{v}} &= \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i) = \tilde{\mathbf{v}}_{L/2,i} && \text{if } i \le L \lceil \log q \rceil /2 \\ \hat{\mathbf{v}} &= \mathsf{VerLocal}^{\mathsf{mx}}(\mathsf{pp}, L/2, i - L \lceil \log q \rceil /2) = \tilde{\mathbf{v}}_{L/2,i-L\lceil \log q \rceil/2} && \text{if } i > L \lceil \log q \rceil /2 \end{aligned}$$

  By the inductive hypothesis, we have

  $$\begin{aligned} \hat{\mathbf{z}} &= \mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_0, i) = \tilde{\mathbf{z}}_{0,i} && \text{if } i \le L \lceil \log q \rceil /2 \\ \hat{\mathbf{z}} &= \mathsf{OpenLocal}^{\mathsf{mx}}(\mathsf{pp}, \mathbf{M}_1, i - L \lceil \log q \rceil /2) = \tilde{\mathbf{z}}_{1,i-L\lceil \log q \rceil/2} && \text{if } i > L \lceil \log q \rceil /2. \end{aligned}$$

Putting the pieces together, we now have

$$\begin{aligned} \mathbf{z}_i &= \mathbf{Z}'_{\text{LT}} \cdot \mathbf{G}_m^{-1}(\hat{\mathbf{v}}) + \hat{\mathbf{z}} = \tilde{\mathbf{Z}}'_{\text{LT}} \cdot \mathbf{G}_m^{-1}(\tilde{\mathbf{v}}_{L/2,i}) + \tilde{\mathbf{z}}_{0,i} = \tilde{\mathbf{z}}_i && \text{if } i \le L \lceil \log q \rceil /2 \\ \mathbf{z}_i &= \mathbf{Z}'_{\text{RT}} \cdot \mathbf{G}_m^{-1}(\hat{\mathbf{v}}) + \hat{\mathbf{z}} = \tilde{\mathbf{Z}}'_{\text{RT}} \cdot \mathbf{G}_m^{-1}(\tilde{\mathbf{v}}_{L/2,i-L\lceil \log q \rceil/2}) + \tilde{\mathbf{z}}_{1,i-L\lceil \log q \rceil/2} = \tilde{\mathbf{z}}_i && \text{if } i > L \lceil \log q \rceil /2. \end{aligned}$$

Thus for all $i \le L \lceil \log q \rceil$, we have that $\mathbf{z}_i = \tilde{\mathbf{z}}_i$, and correctness holds.

Given $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathrm{pp}, L, i)$, it is straightforward to construct the local version of $\mathsf{Open}^{\mathsf{mat}}$:

- $\mathsf{OpenLocal}^{\mathsf{mat}}(\mathrm{pp}, \mathbf{M}, i)$: Output $\mathsf{OpenLocal}^{\mathsf{mx}}(\mathrm{pp}, \mathbf{M}, (i-1) \lceil \log q \rceil + 1)$.

Correctness via the same analysis as for $\mathsf{VerLocal}^{\mathsf{mat}}$ above. □