Registered ABE and Adaptively-Secure Broadcast Encryption from Succinct LWE

Jeffrey Champion
UT Austin
jchampion@utexas.edu

Yao-Ching Hsieh[*]
University of Washington
ychsieh@cs.washington.edu

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

Abstract

Registered attribute-based encryption (ABE) is a generalization of public-key encryption that enables fine-grained access control to encrypted data (like standard ABE), but *without* needing a central trusted authority. In a key-policy registered ABE scheme, users choose their own public and private keys and then register their public keys together with a decryption policy with an (untrusted) key curator. The key curator aggregates all of the individual public keys into a short master public key which serves as the public key for an ABE scheme.

Currently, we can build registered ABE for restricted policies (e.g., Boolean formulas) from pairing-based assumptions and for general policies using witness encryption or indistinguishability obfuscation. In this work, we construct a key-policy registered ABE for general policies (specifically, bounded-depth Boolean circuits) from the ℓ-succinct learning with errors (LWE) assumption in the random oracle model. The ciphertext size in our registered ABE scheme is $\mathsf{poly}(\lambda, d)$, where λ is a security parameter and d is the depth of the circuit that computes the policy circuit C. Notably, this is independent of the length of the attribute $\mathbf{x}$ and is optimal up to the $\mathsf{poly}(d)$ factor.

Previously, the only lattice-based instantiation of registered ABE uses witness encryption, which relies on private-coin evasive LWE, a stronger assumption than ℓ-succinct LWE. Moreover, the ciphertext size in previous registered ABE schemes that support general policies (i.e., from obfuscation or witness encryption) scales with $\mathsf{poly}(\lambda, |\mathbf{x}|, |C|)$. The ciphertext size in our scheme depends only on the depth of the circuit (and not the length of the attribute or the size of the policy). This enables new applications to identity-based distributed broadcast encryption.

Our techniques are also useful for constructing *adaptively-secure* (distributed) broadcast encryption, and we give the first scheme from the ℓ-succinct LWE assumption in the random oracle model. Previously, the only lattice-based broadcast encryption scheme with adaptive security relied on witness encryption in the random oracle model. All other lattice-based broadcast encryption schemes only achieved selective security.

1 Introduction

Attribute-based encryption (ABE) [SW05, GPSW06] is a generalization of public-key encryption that enables fine-grained access control to encrypted data. For example, in a key-policy ABE scheme, decryption keys are associated with an access policy f and ciphertexts are associated with a set of attributes $\mathbf{x}$. Decryption is possible whenever the access policy is satisfied. While ABE augments classic public-key encryption with powerful new capabilities, it comes at the price of changing the trust model. Whereas individual users sample their own secret keys in standard public-key encryption schemes, in ABE, there is a central *trusted* authority who is responsible for issuing keys to different users. To do so, the central authority holds on to a long-term master secret key, and if this secret key is ever leaked or exfiltrated, then the attacker compromises the security of every user in the system. ABE thus introduces a central point of failure that does not exist in the decentralized model of public-key encryption.

Registration-based cryptography. Garg, Hajiabadi, Mahmoody, and Rahimi [GHMR18] introduced the registration-based model to augment public-key encryption with fine-grained decryption *without* a central trusted party. Their work specifically considers identity-based encryption (IBE) where secret keys and ciphertexts are both associated

[*] Part of this work was done while visiting UT Austin.

with an identity, and decryption is successful whenever the identity associated with the secret key and the ciphertext match. In registration-based encryption, users generate their own public/secret keys and then register their public keys with a key curator. The key curator *aggregates* the public keys from the different users (together with their identities) into a single short master public key. The master public key functions as the public key for a standard identity-based encryption scheme. Crucially, the key curator in this model is deterministic and transparent (i.e., holds no secrets). Thus, registration-based encryption provides a way to realize IBE without a trusted key issuer. Since the original work of Garg et al., many works have studied constructions of registration-based encryption [GHM+19, GV20, CES21, GKMR23, DKL+23, FKdP23].

Registered ABE. In this work, we focus on a recent generalization of registration-based encryption to the setting of ABE introduced by Hohenberger, Lu, Waters, and Wu [HLWW23]. In a registered (key-policy) ABE scheme, users also generate their own public/secret key-pairs and register their public key with the key curator along with a decryption policy. The key curator aggregates the public keys into a short master public that serves as a standard ABE public key. The key curator also provides each user a helper decryption key that they use for decryption. Currently, we have constructions of (ciphertext-policy) registered ABE for Boolean formulas and arithmetic branching programs from pairing-based assumptions [HLWW23, ZZGQ23, GLWW24, AT24] as well as constructions that support general policies from advanced tools like witness encryption (in conjunction with function-binding hash functions) [FWW23] or indistinguishability obfuscation (in conjunction with one-way functions) [HLWW23].

1.1 Our Results

In this work, we give the first construction of (key-policy) registered ABE for arbitrary (bounded-depth) circuit policies from falsifiable lattice assumptions in the random oracle model. Security of our scheme relies on the ℓ-succinct learning with errors (LWE) assumption introduced by Wee [Wee24]. Previously, the only lattice-based construction of registered ABE goes through general-purpose witness encryption [FWW23], which itself relies on the *private-coin* evasive LWE assumption [Tsa22, VWW22]. Multiple recent works [VWW22, BÜW24, BDJ+24] have demonstrated the implausibility of some versions of the private-coin evasive LWE assumption. In light of these counter-examples, an important goal in lattice-based cryptography is to move towards assumptions that are simpler to state and analyze. The ℓ-succinct LWE assumption is an example of a simple, but useful, generalization of the LWE assumption. It is falsifiable, instance-independent, and also implied by the *public-coin* evasive LWE assumption (in conjunction with plain LWE). We refer to [Wee24, §1.4] and [CW24, §1] for additional discussion on the advantages of succinct LWE over evasive LWE. The ℓ-succinct LWE assumption has found several applications to succinct ciphertext-policy ABE [Wee24], succinct functional commitments for circuits [WW23a], and distributed broadcast encryption [CW24].

Another appealing feature of our construction is it only relies on standard lattice homomorphic evaluation machinery (similar complexity as vanilla lattice-based ABE [GVW13, BGG+14]) and is fully black-box in the use of cryptographic primitives. The previous approach based on witness encryption (or indistinguishability obfuscation) makes *non-black-box* use of function-binding hash functions (or one-way functions). Obtaining constructions that do not rely on non-black-box use of other cryptographic primitives is an important step towards bringing registration-based cryptography closer to practice, and has been a major motivating factor behind a number of works in both the pairing-based setting [GKMR23, HLWW23, FKdP23] and the lattice-based setting [DKL+23]. We summarize our construction with the following theorem:

Theorem 1.1 (Informal)**.** *Let λ be a security parameter and N be a bound on the number of users. Let $\mathcal{F}$ be a family of decryption policies on attributes $\mathbf{x}$ that can be computed by a Boolean circuit of depth at most d. Then, assuming polynomial hardness of the ℓ-succinct LWE assumption (with $\ell \geq \max(|\mathbf{x}|, N \cdot \mathsf{poly}(\lambda, \log N))$) with a sub-exponential modulus-to-noise ratio, there exists a key-policy registered ABE scheme that supports up to N users and policy family $\mathcal{F}$ in the random oracle model. The scheme satisfies attribute-selective security and has the following efficiency properties (and ignoring polylogarithmic factors):*

- *The scheme has a structured reference string of size $(N^2 + |\mathbf{x}|^2) \cdot \mathsf{poly}(\lambda, d)$.*
- *Each user's public key has size $N \cdot \mathsf{poly}(\lambda, d)$. The user's secret key has size $\mathsf{poly}(\lambda, d)$.*

- *The aggregated master public key and each user's helper decryption key has size* $\mathsf{poly}(\lambda, d)$.
- *A ciphertext has size* $\mathsf{poly}(\lambda, d)$.

If we assume sub-exponential hardness of ℓ-succinct LWE, then the scheme is adaptively secure (and all parameter sizes now additionally scale with $\mathsf{poly}(|\mathbf{x}|)$*).*

Succinct ciphertexts and identity-based distributed broadcast encryption. Much like Wee's centralized ABE scheme [Wee24] from the ℓ-succinct LWE assumption, our registered ABE scheme has succinct ciphertexts where the ciphertext size is *independent* of the attribute length.[1] This is the first registered ABE scheme for general policies from *any* assumption that has *succinct* ciphertexts. The ciphertext size in previous registered ABE schemes for circuits based on witness encryption [FWW23] or indistinguishability obfuscation [HLWW23] all scale with $\mathsf{poly}(\lambda, |\mathbf{x}|, |C|)$, where C is the size of the policy circuit. In these constructions, the ciphertext contains an obfuscated program that computes $C(\mathbf{x})$ or a witness encryption ciphertext with respect to an NP relation that computes $C(\mathbf{x})$.

Registered ABE with succinct ciphertexts immediately gives a (selectively-secure) identity-based distributed broadcast encryption scheme (see Remark 5.38). Normally, in a distributed broadcast encryption scheme [WQZDF10, BZ14], the encrypter needs to look up the public key for each recipient during encryption. With identity-based distributed broadcast encryption, the encrypter only needs to know the recipients' identities (e.g., their usernames) and there is no need for a separate public key lookup. Theorem 1.1 gives the first such scheme with these properties from ℓ-succinct LWE in the random oracle model.

Adaptively-secure broadcast encryption. Beyond giving the first construction of registered ABE for general circuit policies from falsifiable lattice assumptions, the techniques we develop (see Section 2) are also applicable for constructing *adaptively-secure* broadcast encryption. Broadcast encryption [FN93] allows a user to encrypt a message to a set of users S with a ciphertext whose size scales sublinearly with $|S|$. In this work, we show how to adapt the techniques underlying our registered ABE scheme to obtain an adaptively-secure broadcast encryption scheme from the ℓ-succinct LWE assumption in the random oracle model. Like [CW24], our scheme is a distributed broadcast encryption scheme [WQZDF10, BZ14], which is a trustless version of broadcast encryption where users choose their own keys.

Prior to our work, the only instantiation of adaptively-secure broadcast encryption from lattice assumptions relied on witness encryption in the random oracle model [FWW23]. Other constructions of broadcast encryption from lattice assumptions [Wee22, Wee24, CW24], including constructions from evasive LWE, only satisfy selective security, where the adversary has to declare the challenge set at the start of the security game. In the context of broadcast encryption, selective security does not imply adaptive security via complexity leveraging (since complexity leveraging does not preserve succinctness). The work of [FWW23] also describe a generic way to build distributed broadcast encryption from registered ABE, but they only prove *selective* security of the resulting construction. We summarize our results in the following informal theorem:

Theorem 1.2 (Informal). *Let λ be a security parameter and N be a bound on the number of users. Then, assuming polynomial hardness of the ℓ-succinct LWE assumption (with $\ell \geq N \cdot \mathsf{poly}(\lambda, \log N)$) with a sub-exponential modulus-to-noise ratio, there exists an adaptively-secure distributed broadcast encryption scheme that supports up to N users in the random oracle with the following properties:*

- *The common reference string consists of a structured string of size* $N^2 \cdot \mathsf{poly}(\lambda, \log N)$.
- *Each user's public key has size* $N \cdot \mathsf{poly}(\lambda, \log N)$ *and secret key has size* $\mathsf{poly}(\lambda, \log N)$.
- *An encryption to a set of users* $S \subseteq [N]$ *has size* $\mathsf{poly}(\lambda, \log N)$.

[1] The decryption algorithm in an ABE scheme takes the attribute $\mathbf{x}$ as input, so it is possible for the ciphertext size to be independent of $|\mathbf{x}|$.

1.2 Concurrent and Subsequent Work

In a concurrent and independent work, Zhu, Zhang, Chen, Gong, and Qian [ZZC$^+$25] construct registered ABE that supports an unbounded number of users with transparent setup from the private-coin evasive LWE assumption (an assumption that implies witness encryption). The main improvement over the [FWW23] construction based on witness encryption is they can handle corruptions without random oracles. The ciphertext size in their scheme scales polynomially with the attribute length and the depth of the circuit.

Our scheme requires a structured reference string and assumes an a priori bound on the number of users. However, the scheme has succinct ciphertexts (independent of the attribute length). The latter property is important for the implication to identity-based distributed broadcast encryption (see Remark 5.38). More importantly, security of our scheme relies on the much weaker (and falsifiable) succinct LWE assumption. As noted previously, recent attacks on private-coin evasive LWE [VWW22, BÜW24, BDJ$^+$24] raise significant doubts on the plausibility of private-coin evasive LWE. At the very least, security of their scheme necessarily relies on a carefully-crafted evasive LWE assumption (tailored to the specific structure of their construction) in order to avoid known counter-examples.

On selective security notions for registered ABE. In a subsequent and independent work, Abram, Malavolta, and Roy [AMR25] propose a (slotted) registered ABE scheme for general policies from a new assumption they call the ℓ-decomposed LWE assumption in the *plain* model. The ℓ-decomposed LWE assumption is a variant of LWE that is implied by the ℓ-succinct LWE assumption. To support N users, their scheme requires a structured string of size $N \cdot \mathsf{poly}(\lambda, \log N)$; in our scheme, the size of the CRS is quadratic in the number of users. Like our scheme, the [AMR25] construction has succinct ciphertexts (independent of the attribute length). An important distinction of their scheme is it satisfies a significantly weaker notion they call "very-selective" security.[2] In their definition, they require that the adversary in the registered ABE security game to pre-commit to the *key-generation randomness* for all of the adversarially-chosen keys in advance. This is a substantial relaxation of security compared to the usual concept of selective security proposed in the study of notions like digital signatures [GMR88], verifiable random functions [MRV99], or identity-based encryption [CHK03, BB04]. Here, we offer some perspectives on the different notions of selective security commonly encountered in the literature:

- The standard notion of selective security corresponds to the setting where the adversary has to pre-commit to its challenge at the beginning of the security game. For instance, in the case of signatures [GMR88], the adversary commits to the message it forges on ahead of time. For identity-based encryption [CHK03, BB04], the adversary commits to the identity associated with the challenge ciphertext. A common rationale cited in these settings for why selective security is a reasonable relaxation of adaptive security is that with complexity leveraging and sub-exponential hardness assumptions, selective security implies the normal notion of adaptive security [MRV99, BB04].

- A stronger notion, which we call "query-selective" security goes one step further and requires that the adversary also commit to any queries it makes in the security game ahead of time. For instance, in the signature case, this corresponds to the adversary declaring its signing queries ahead of time. In identity-based encryption, this corresponds to the adversary declaring its key-generation queries. Additionally, several works on multi-authority ABE [RW15, DKW21, WWW22] have considered query-selective (or "static") security. Unlike the case with standard selective security, query-selective security cannot be lifted to adaptive security via complexity leveraging (unless we impose an a priori bound on the number of queries the adversary makes). In some sense, query-selective security should be treated as a heuristic substitution for adaptive security since we cannot generically lift to full adaptive security. Alternatively, query-selective security is perhaps sufficient for applications where the adversary does not have the ability to make arbitrary queries.

- The "very-selective" notion of security for registered ABE from [AMR25] is like a query-selective notion since it asks the adversary to commit to all of its registration queries ahead of time, but it goes one step further by additionally restricting the adversary to registering keys that are in the support of the honest key-generation

[2] Separately, their work considers correctness only in the setting where *all* of the keys are *honestly-generated*. Using non-interactive zero-knowledge proofs, it seems plausible that we can lift this to the standard definition where correctness holds for honest users even if some keys are adversarially chosen. Thus, we ignore this distinction here.

algorithm. There is no such restriction in the standard registered ABE security game. This is another example where we cannot lift a scheme satisfying this security notion to one that is adaptively secure in the usual sense. Indeed, it is also easy to build a (contrived) scheme that satisfies very-selective security, but would be completely broken against an adversary that can register arbitrary keys (e.g., a scheme where encryption always outputs the message in the clear if the master public key was derived by a aggregating a special key embedded in the public parameters). Thus, we need to be careful whenever we consider relaxed notions of security along these lines. More generally, we find the concept of query-selective notion of security to be somewhat counter to the original premise of registration-based cryptography. The goal in registration-based cryptography is for users to choose their own public keys and a security definition that removes this capability from the adversary seems to be at odds with the primary objective of registration-based cryptography.

The techniques we develop in this work are designed to prove security even against adversarially-chosen public keys. We view this as one of the primary challenges when designing registered ABE schemes. In fact, it is straightforward to prove the security of our scheme in the *plain* model if we are satisfied with the "very selective" notion of security from [AMR25]. As we discuss more in Remark 5.40, many of the tools we develop in this work are unnecessary if we settle for this weaker notion of security. However, since we do not know of a way to generically lift a scheme under the very-selective security definition into one under the standard definition, we opt to directly construct a scheme that satisfies the standard security requirement for registered ABE. We also note that in our analysis, we do consider some relaxations of the standard registered ABE security definition (e.g., attribute-selective security, security without corruptions). The critical difference though is that there are standard ways to lift the relaxations we consider in our analysis to the standard security definition for registered ABE (see Remark 5.5).

2 Technical Overview

In this section, we provide a high-level overview of our main constructions. To start, we first introduce some notation. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{y} \in \mathbb{Z}_q^n$ in the column-space of $\mathbf{A}$, we write $\mathbf{x} \leftarrow \mathbf{A}^{-1}(\mathbf{y})$ to denote sampling $\mathbf{x}$ from a discrete Gaussian distribution conditioned on $\mathbf{A}\mathbf{x} = \mathbf{y}$. We can efficiently sample from $\mathbf{A}^{-1}(\mathbf{y})$ given a trapdoor for $\mathbf{A}$. We write $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^{\mathsf{T}}$ to denote the gadget vector where $\mathbf{I}_n$ is the identity matrix of dimension n and $\mathbf{g}^{\mathsf{T}} = [1, 2, \ldots, 2^{\lceil \log q \rceil - 1}]$. To simplify the description in the overview, we use *curly underlines* to suppress small (low-norm) error terms. 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 small error vector.

Homomorphic computation using lattices. Our construction relies on the machinery from [GSW13, BGG⁺14] for homomorphic computation on matrix encodings. Specifically, these works describe an efficient algorithm that takes as input a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times \ell m}$, a Boolean circuit $C \colon \{0,1\}^\ell \to \{0,1\}$, and an input $\mathbf{x} \in \{0,1\}^\ell$ and outputs a short matrix $\mathbf{H}_{\mathbf{B},C,\mathbf{x}}$ where

$$(\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{B},C,\mathbf{x}} = \mathbf{B}_C - C(\mathbf{x}) \cdot \mathbf{G}, \tag{2.1}$$

where $\mathbf{B}_C$ is a matrix that only depends on the matrix $\mathbf{B}$ and the circuit C.

The [CW24] distributed broadcast scheme. Our starting point in this work is the recent construction of distributed broadcast encryption from the ℓ-succinct LWE assumption by Champion and Wu [CW24]. In a distributed broadcast encryption scheme [WQZDF10, BZ14], each user generates their own public and secret keys $(\mathsf{pk}_i, \mathsf{sk}_i)$. The encryption algorithm takes as input a collection of public keys $\{\mathsf{pk}_i\}_{i \in S}$ together with a message μ and outputs a short ciphertext which encrypts μ to the set of users S. We start by recalling their construction. In the following, let N be a bound on the number of users in the system.

- **Common reference string:** The common reference string (CRS) consists of matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a target vector $\mathbf{p} \in \mathbb{Z}_q^n$. In addition, for each $i \in [N]$, the common reference string also includes a short vector $\mathbf{r}_i \in \mathbb{Z}_q^m$. Finally, to allow users to sample their own keys, the common reference string includes the matrix

$$\mathbf{V} = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \in \mathbb{Z}_q^{nN \times (mN+k)}, \tag{2.2}$$

where $\mathbf{Z} \in \mathbb{Z}_q^{n \times mk}$ and $k = O(nm \log q)$ along with a trapdoor $\mathsf{td}_\mathbf{V}$ for $\mathbf{V}$.

- **Key generation:** To sample a public/secret key for index $i \in [N]$, user i uses the trapdoor $\mathsf{td}_\mathbf{V}$ to sample $\mathbf{y}_{i,j} \in \mathbb{Z}_q^m$, and $\mathbf{d}_i \in \mathbb{Z}_q^k$ such that

$$\left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \cdot \begin{bmatrix} \mathbf{y}_{i,1} \\ \vdots \\ \mathbf{y}_{i,N} \\ \mathbf{d}_i \end{bmatrix} = \boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}\mathbf{r}_i), \tag{2.3}$$

where $\boldsymbol{\eta}_i \in \mathbb{Z}_q^N$ is the i^{th} canonical basis vector. Let $\mathbf{W}_i \coloneqq \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m) \in \mathbb{Z}_q^{n \times m}$. Then, for all $j \in [N]$,

$$\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)\mathbf{d}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)(1 \otimes \mathbf{r}_j) = \mathbf{W}_i \mathbf{r}_j,$$

Then, from Eqs. (2.2) and (2.3), we have for all $j \in [N]$

$$\mathbf{A}\mathbf{y}_{i,j} = \begin{cases} \mathbf{W}_i \mathbf{r}_j & j \neq i \\ \mathbf{W}_i \mathbf{r}_i + \mathbf{p} + \mathbf{B}\mathbf{r}_i & j = i. \end{cases}$$

The user's public key consists of $\mathbf{W}_i$ and the "cross terms" $\mathbf{y}_{i,j}$ for $j \neq i$. The user's secret key is $\mathbf{y}_{i,i}$. In other words, the public key is a short vector that recodes from $\mathbf{A}$ to $\mathbf{W}_i \mathbf{r}_j$ for $i \neq j$, whereas the secret key recodes from $\mathbf{A}$ to $\mathbf{W}_i \mathbf{r}_i + \mathbf{B}\mathbf{r}_i + \mathbf{p}$. These two properties will be crucial for decryption.

- **Encryption:** To encrypt a bit $\mu \in \{0, 1\}$ to a set of public keys $\{\mathsf{pk}_i\}_{i \in S}$ where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i})$, the encrypter samples an LWE secret key $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and computes $\mathbf{W}_S = \sum_{j \in S} \mathbf{W}_j$. The ciphertext is then

$$\mathsf{ct}_S = \left(\utilde{\mathbf{s}^\mathsf{T}\mathbf{A}}\,,\ \utilde{\mathbf{s}^\mathsf{T}(\mathbf{B} + \mathbf{W}_S)}\,,\ \utilde{\mathbf{s}^\mathsf{T}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor}\right).$$

- **Decryption:** Decryption relies on the fact that when $i \in S$, we have

$$\utilde{\mathbf{s}^\mathsf{T}\mathbf{A}}\left(\mathbf{y}_{i,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) \approx \mathbf{s}^\mathsf{T}\mathbf{W}_i\mathbf{r}_i + \mathbf{s}^\mathsf{T}\mathbf{p} + \mathbf{s}^\mathsf{T}\mathbf{B}\mathbf{r}_i + \sum_{j \in S \setminus \{i\}} \mathbf{s}^\mathsf{T}\mathbf{W}_j\mathbf{r}_i = \mathbf{s}^\mathsf{T}\mathbf{p} + \mathbf{s}^\mathsf{T}\mathbf{B}\mathbf{r}_i + \mathbf{s}^\mathsf{T}\mathbf{W}_S\mathbf{r}_i,$$

where $\mathbf{y}_{i,i}$ is the secret key of user i and $\mathbf{y}_{j,i}$ are the components of the public keys for other users. To decrypt, user i computes

$$\utilde{\mathbf{s}^\mathsf{T}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor} + \utilde{\mathbf{s}^\mathsf{T}(\mathbf{B} + \mathbf{W}_S)}\mathbf{r}_i - \utilde{\mathbf{s}^\mathsf{T}\mathbf{A}}\left(\mathbf{y}_{i,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) \approx \mu \cdot \lfloor q/2 \rfloor\,.$$

The [BGG+14] ABE scheme. To construct our key-policy registered ABE scheme, we combine the structure of the [CW24] distributed broadcast encryption scheme with the key-policy ABE scheme from [BGG+14]. In the [BGG+14] ABE scheme, an encryption of a message μ with respect to an attribute $\mathbf{x} \in \{0, 1\}^\ell$ is a triple

$$\left(\utilde{\mathbf{s}^\mathsf{T}\mathbf{A}}\,,\ \utilde{\mathbf{s}^\mathsf{T}(\mathbf{B} - \mathbf{x}^\mathsf{T} \otimes \mathbf{G})}\,,\ \utilde{\mathbf{s}^\mathsf{T}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor}\right),$$

and the secret key for a policy $C \colon \{0, 1\}^\ell \to \{0, 1\}$ is a short vector $\mathbf{y}_C$ where $[\mathbf{A} \mid \mathbf{B}_C]\mathbf{y}_C = \mathbf{p}$. We say that $\mathbf{x}$ satisfies the policy C if $C(\mathbf{x}) = 0$. When $C(\mathbf{x}) = 0$, by Eq. (2.1),

$$\utilde{\mathbf{s}^\mathsf{T}(\mathbf{B} - \mathbf{x}^\mathsf{T} \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathbf{B},C,\mathbf{x}} = \utilde{\mathbf{s}^\mathsf{T}(\mathbf{B}_C - C(\mathbf{x}) \cdot \mathbf{G})} = \utilde{\mathbf{s}^\mathsf{T}\mathbf{B}_C}.$$

Using the secret key $\mathbf{y}_C$, the user can now compute

$$\left[\utilde{\mathbf{s}^\mathsf{T}\mathbf{A}} \mid \utilde{\mathbf{s}^\mathsf{T}\mathbf{B}_C}\right] \cdot \mathbf{y}_C \approx \mathbf{s}^\mathsf{T}\mathbf{p},$$

which is sufficient to recover μ.

Key-policy (slotted) registered ABE. In registered ABE, users are allowed to dynamically join the system at any time (and the key curator updates the master public key after each registration). To simplify the construction of registered ABE, the work of [HLWW23] shows that it suffices to construct a simpler *slotted* registered ABE scheme. A slotted registered ABE scheme supports an a priori fixed number of users N, and moreover, each user is associated with a specific slot index $i \in [N]$. When sampling their public keys, the users sample it for their particular slot index. In our setting, the user also specifies their decryption policy at key-generation time. Finally, there is an aggregation algorithm that takes as input N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ together with their respective decryption policies $C_1, \ldots, C_N$ and aggregates them together into a master public key (whose size is sublinear in N). Our key-policy registered ABE scheme leverages features of the [CW24] distributed broadcast encryption scheme and the [BGG+14] key-policy ABE scheme. We start with the basic structure of our scheme:

- **Common reference string:** The CRS contains the following components:
  - **Encryption components:** The CRS contains a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and vector $\mathbf{p} \in \mathbb{Z}_q^{n \times m}$. These constitute a public key for a dual Regev encryption scheme (c.f., [GPV08]) and play the same role as in the aforementioned schemes [BGG+14, CW24].
  - **Attribute-embedding component:** Similar to [BGG+14], the matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times \ell m}$ is used to encode attributes in the ciphertext.
  - **Key-generation components:** Similar to [CW24], the CRS contains a short vector $\mathbf{r}_i \in \mathbb{Z}_q^m$ for each slot $i \in [N]$, a matrix $\mathbf{Z} \in \mathbb{Z}_q^{n \times mk}$ and $\mathbf{V} \in \mathbb{Z}_q^{nN \times (mN+k)}$ from Eq. (2.2) together with the trapdoor $\mathsf{td}_{\mathbf{V}}$. Users will use the matrix $\mathbf{V}$ and trapdoor to sample public keys, just as in [CW24].
  - **Smudging components:** For each slot $i \in [N]$, the CRS also contains a random vector $\mathbf{t}_i \xleftarrow{\text{R}} \mathbb{Z}_q^m$ which will be used for a critical noise smudging step in our security analysis (see Section 2.1).
- **Public key structure:** Like [CW24], a public key for slot i contains a matrix $\mathbf{W}_i \in \mathbb{Z}_q^{n \times m}$ and cross-terms $\mathbf{y}_{i,j}$ for all $j \neq i$ where $\mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i \mathbf{r}_j$. These can be sampled using the trapdoor $\mathsf{td}_{\mathbf{V}}$ for $\mathbf{V}$ (see Eq. (2.3)).
- **Aggregated master public key:** Let $\mathbf{W}_1, \ldots, \mathbf{W}_N$ together with $\mathbf{y}_{i,j}$ for all $i \neq j$ be a collection of N public keys. In a registered ABE scheme, ciphertexts are encrypted to *all* users, with the stipulation that only users who satisfy the policy can decrypt. Thus, the aggregated master public key mpk is $\mathsf{mpk} = \widehat{\mathbf{W}} = \sum_{i \in [N]} \mathbf{W}_i$ and the helper decryption key hsk_i for each user $i \in [N]$ is the sum of the associated cross terms $\mathsf{hsk}_i = \widehat{\mathbf{y}}_i = \sum_{j \neq i} \mathbf{y}_{j,i}$. We can view mpk as a public key associated with broadcasting to *all* users in the [CW24] scheme, and hsk_i as a pre-computed helper decryption component.
- **Ciphertext:** To encrypt a bit $\mu \in \{0,1\}$ with respect to an attribute $\mathbf{x} \in \{0,1\}^\ell$ and the master public key $\mathsf{mpk} = \widehat{\mathbf{W}}$, the encrypter samples an LWE secret key $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and outputs

$$\mathsf{ct}_S = \left(\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{A}}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}}}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})}\,,\ \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor}\right). \tag{2.4}$$

  We can view $\mathbf{s}^{\mathsf{T}}(\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$ as the attribute-embedding component from [BGG+14] and $\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}}$ as the broadcast component from [CW24]. The latter serves to ensure that only registered users (i.e., users whose keys have been aggregated as part of $\mathsf{mpk} = \widehat{\mathbf{W}}$) are able to decrypt.
- **Secret key structure:** A secret key for slot i and policy C is a short vector $\mathbf{y}_{i,i} \in \mathbb{Z}_q^m$ where $\mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i \mathbf{r}_i + \mathbf{B}_C \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p}$. First, observe that the user (for slot i) can jointly sample their public key $\mathbf{W}_i$, their secret key $\mathbf{y}_{i,i}$, and the the cross terms $\mathbf{y}_{i,j}$ for $j \neq i$ by using $\mathsf{td}_{\mathbf{V}}$ to sample a solution to the system

$$\left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \begin{bmatrix} \mathbf{y}_{i,1} \\ \vdots \\ \mathbf{y}_{i,N} \\ \mathbf{d}_i \end{bmatrix} = \boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_C \mathbf{G}^{-1}(\mathbf{t}_i)), \tag{2.5}$$

and setting $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$. This is the same procedure as in Eq. (2.3), except the user now targets $\mathbf{p}+\mathbf{B}_C\mathbf{G}^{-1}(\mathbf{t}_i)$ in the i^{th} index, which corresponds to the structure of its secret key. Given the secret key $\mathbf{y}_{i,i}$, together with the helper decryption key $\mathsf{hsk}_i = \widehat{\mathbf{y}}_i = \sum_{j \neq i} \mathbf{y}_{j,i}$, the user can decrypt a ciphertext encrypted to any attribute $\mathbf{x} \in \{0,1\}^\ell$ where $C(\mathbf{x}) = 0$ as follows:

- **Attribute check:** When $C(\mathbf{x}) = 0$, we have by Eq. (2.1) that

$$\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})} \cdot \mathbf{H}_{\mathbf{B},C,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{t}_i) \approx \mathbf{s}^{\mathsf{T}}\mathbf{B}_C\mathbf{G}^{-1}(\mathbf{t}_i). \tag{2.6}$$

- **Slot check:** Since $\mathbf{r}_i$ and $\widehat{\mathbf{y}}_i$ are both short, the user can now compute

$$\utilde{\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}}}\mathbf{r}_i - \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{A}}\widehat{\mathbf{y}}_i \approx \mathbf{s}^{\mathsf{T}} \sum_{j \in [N]} \mathbf{W}_j \mathbf{r}_i - \mathbf{s}^{\mathsf{T}} \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} = \mathbf{s}^{\mathsf{T}}\mathbf{W}_i\mathbf{r}_i, \tag{2.7}$$

  since the cross-terms $\mathbf{y}_{j,i}$ satisfy $\mathbf{A}\mathbf{y}_{j,i} = \mathbf{W}_j\mathbf{r}_i$.

- **Combining the pieces:** Finally, the user can use its secret key $\mathbf{y}_{i,i}$ to compute

$$\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{A}}\mathbf{y}_{i,i} \approx \mathbf{s}^{\mathsf{T}}(\mathbf{p} + \mathbf{W}_i\mathbf{r}_i + \mathbf{B}_C\mathbf{G}^{-1}(\mathbf{t}_i)). \tag{2.8}$$

Subtracting Eqs. (2.6) and (2.7) from Eq. (2.8) now yields $\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p}}$, which can be combined with $\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor}$ in the ciphertext to recover the message μ. We can view Eq. (2.6) as ensuring that the attribute satisfies the decryption policy and Eq. (2.7) as ensuring that the user is registered to some slot i.

The construction described here satisfies correctness. While the structure of the scheme is similar to the distributed broadcast encryption scheme of [CW24], we require a different approach to prove security. We view this as the primary technical challenge of this work, and elaborate further in Section 2.1.

2.1 Proving Security of our Registered ABE Scheme

Like [CW24], security of our construction relies on the ℓ-succinct LWE assumption introduced in [Wee24]. The ℓ-succinct LWE assumption asserts that the LWE assumption holds with respect to a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ (i.e., that $\utilde{\mathbf{s}^{\mathsf{T}}\mathbf{A}}$ is pseudorandom) given a trapdoor for a related matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ where $\mathbf{U} \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^{\ell n \times m}$. The work of [CW24, §4] describes a transformation that takes any trapdoor for the matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ and converts it into a trapdoor for the matrix $\mathbf{V} \in \mathbb{Z}_q^{nN \times (mN+k)}$ from Eq. (2.2) so long as $\ell \geq N \cdot O(n \log q)$ and $k \geq 3nm \log q$. In the following description, we will primarily work with the structured matrix $\mathbf{V}$ and its trapdoor $\mathsf{td}_{\mathbf{V}}$.

The [CW24] partitioning approach. We first describe the key principles underlying the partitioning strategy from [CW24] that are used to argue *selective* security of their distributed broadcast encryption scheme:

- **Programming the challenge set:** In the selective security game for broadcast encryption, the adversary has to declare its challenge set S upfront. Moreover, in broadcast encryption, the public keys corresponding to users in S are honestly generated. Otherwise, the adversary can trivially decrypt. Thus, the reduction algorithm samples the public keys $\mathbf{W}_i$ for the honest users $i \in S$ itself, and then *programs* the challenge set into the public parameters by defining $\mathbf{B} := \mathbf{B}^* - \sum_{i \in S} \mathbf{W}_i$, where $\mathbf{B}^* \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}$. When simulating the challenge ciphertext, the reduction algorithm has to simulate $\utilde{\mathbf{s}^{\mathsf{T}}(\mathbf{B} + \sum_{i \in S} \mathbf{W}_i)} = \utilde{\mathbf{s}^{\mathsf{T}}\mathbf{B}^*}$. By setting $\mathbf{B}^* = \mathbf{A}\mathbf{K}$ for a random short $\mathbf{K}$, the reduction algorithm can simulate this using the terms from the ℓ-succinct LWE challenge.

- **Sampling honest user keys:** In order to program the honest users' public keys $\mathbf{W}_i$ into the public matrix $\mathbf{B}$, the reduction algorithm needs to sample $\mathbf{W}_i$ without knowledge of $\mathbf{B}$. It does so by modifying the honest sampling algorithm (Eq. (2.3)) which samples a preimage of $\boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}\mathbf{r}_i)$ to instead sample a preimage of $\mathbf{0}^{nN}$. By properties of the Gaussian distribution, this does *not* affect the marginal distribution of the components of the public key $\mathbf{y}_{i,j}$ for $j \neq i$ or $\mathbf{d}_i$. In this way, the reduction algorithm can sample the public keys for the honest users without knowledge of $\mathbf{B}$, which is enough to complete the partitioning argument.

Security of [CW24] critically relies on being able to embed the challenge set into the public parameters of the scheme.

The trouble with adversarial registrations. It is unclear how to leverage the [CW24] proof strategy in our setting of registered ABE. Unlike broadcast encryption, there are *two* reasons for why a user cannot decrypt in a registered ABE scheme:

- Their public key is not registered in the system.
- Their public key is registered in the system, but the ciphertext does *not* satisfy their decryption policy.

In the setting of broadcast encryption, the first case corresponds to whether a user is in the broadcast set or not, whereas there is no analog of the second case. In registered ABE, the adversary is allowed to register keys for any policy that does not satisfy the challenge attribute. For our specific registered ABE construction, this means the adversary can choose the public keys $\mathbf{W}_i$ for some subset of the slots $S \subseteq [N]$. The aggregated public key is an aggregation of *all* of the registered public keys $\widehat{\mathbf{W}} = \sum_{i \in [N]} \mathbf{W}_i$. Therefore, when simulating the challenge ciphertext, the reduction algorithm needs to simulate a component of the form $\utilde{\mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}}}$. However, $\widehat{\mathbf{W}}$ necessarily depends on the public keys chosen by the adversary, so the reduction algorithm cannot program it into the public parameters during setup (and $\widehat{\mathbf{W}}$ is also too big to guess). Moreover, because the structure of the public keys depends on the CRS, we cannot ask the adversary to "commit" to them *before* seeing the CRS (e.g., we do not have a meaningful notion of "selective-registration" security in this setting). Thus, the [CW24] proof strategy would only be applicable if we completely preclude the adversary from registering keys altogether, which is an unreasonable notion of security for registered ABE. Thus, we need a new proof strategy that does *not* rely on programming the master public key into the CRS itself.

Our approach: randomizing during aggregation. Our approach for arguing security is to introduce additional randomness at aggregation time. Specifically, during aggregation, the aggregator chooses a matrix $\mathbf{W}_0 \in \mathbb{Z}_q^{n \times m}$ together with short preimages $\mathbf{y}_{0,i}$ where $\mathbf{A}\mathbf{y}_{0,i} = \mathbf{W}_0 \mathbf{r}_i$ for all $i \in [N]$ The aggregated public key is now $\widehat{\mathbf{W}} = \mathbf{W}_0 + \sum_{i \in [N]} \mathbf{W}_i$ and each user's helper decryption key is now $\mathsf{hsk}_i = \widehat{\mathbf{y}}_i = \mathbf{y}_{0,i} + \sum_{j \neq i} \mathbf{y}_{j,i}$. The components $(\mathbf{W}_0, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N})$ can be derived using the trapdoor $\mathsf{td}_{\mathbf{V}}$ by sampling $(\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{d}_0)$ from $\mathbf{V}^{-1}(\mathbf{0})$ and setting $\mathbf{W}_0 = \mathbf{A}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. The slot check in the decryption relation (Eq. (2.7)) for user i now becomes

$$\utilde{\mathbf{s}^{\mathsf{T}}\big(\mathbf{W}_0 + \textstyle\sum_{j \in [N]} \mathbf{W}_j\big)}\mathbf{r}_i - \utilde{\mathbf{s}^{\mathsf{T}}}\mathbf{A}\big(\mathbf{y}_{0,i} + \textstyle\sum_{j \neq i} \mathbf{y}_{j,i}\big) \approx \underbrace{\mathbf{s}^{\mathsf{T}}\mathbf{W}_0\mathbf{r}_i - \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{y}_{0,i}}_{\mathbf{0}} + \underbrace{\textstyle\sum_{j \in [N]} \mathbf{s}^{\mathsf{T}}\mathbf{W}_j\mathbf{r}_i - \sum_{j \neq i} \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{y}_{j,i}}_{\mathbf{s}^{\mathsf{T}}\mathbf{W}_i\mathbf{r}_i} = \mathbf{s}^{\mathsf{T}}\mathbf{W}_i\mathbf{r}_i.$$

The aggregator essentially introduces additional entropy by registering a "virtual party" and including the corresponding cross-terms as part of each user's helper decryption key. The problem with this approach is that in registered ABE, the aggregator is untrusted. For this reason, we require a *deterministic* aggregation algorithm so there is no room for the aggregator to cheat. With a randomized scheme, a malicious aggregator could choose "bad" randomness that jeopardizes security. For instance, while the honest aggregator in this case is supposed to register a key for which it does not know the corresponding secret key, a malicious aggregator may not do this. Thus, we need a way to limit the aggregator's ability to rig the master public key. We solve this by relying on the random oracle heuristic. Namely, the aggregator derives the aggregation randomness (i.e., the matrix $\mathbf{W}_0$ and the cross terms $\mathbf{y}_{0,i}$) by hashing the public keys of each user and then using the hash value as the randomness for sampling $\mathbf{V}^{-1}(\mathbf{0})$. This way, the aggregation algorithm is deterministic. The next question is how to prove security of this scheme.

Attribute-selective security. In this work, we consider attribute-selective security where the adversary declares the attribute associated with the challenge ciphertext at the beginning of the security game. This is a standard relaxation in lattice-based (non-registered) ABE schemes [GVW13, BGG⁺14, HLL23, Wee24]. The previous (ciphertext-policy) registered ABE schemes from witness encryption [FWW23] was also selectively secure. Note that selective security implies adaptive security via complexity leveraging and relying on sub-exponential hardness.

We consider a reduction to ℓ-succinct LWE. Consider an ℓ-succinct LWE challenge $(\mathbf{A}, \mathbf{v}^{\mathsf{T}}, \mathbf{U}, \mathsf{td})$, where td is a trapdoor for $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$. As noted before, the work of [CW24] shows how to use $\mathbf{U}$ and td to sample a matrix $\mathbf{Z}$, $\mathbf{r}_1, \ldots, \mathbf{r}_N$, together with a trapdoor $\mathsf{td}_{\mathbf{V}}$ for the matrix $\mathbf{V}$ in Eq. (2.2). The reduction algorithm proceeds as follows:

- The reduction gets the matrix $\mathbf{A}$, the matrix $\mathbf{Z}$, the vectors $\mathbf{r}_1, \ldots, \mathbf{r}_N$, and the trapdoor $\mathsf{td}_{\mathbf{V}}$ from the ℓ-succinct LWE challenger (and then applies the [CW24] transformation to derive $\mathbf{Z}, \mathbf{r}_1, \ldots, \mathbf{r}_N, \mathsf{td}_{\mathbf{V}}$).
- At the beginning of the security game, the adversary commits to the attribute $\mathbf{x} \in \{0,1\}^\ell$. The reduction algorithm programs $\mathbf{x}$ into the public parameters by sampling a short matrix $\mathbf{K}_{\mathbf{B}}$ and setting $\mathbf{B} = \mathbf{A}\mathbf{K}_{\mathbf{B}} + \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}$.
- The reduction algorithm samples a short vector $\mathbf{k}_{\mathbf{p}}$ and sets $\mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}}$.
- For each $i \in [N]$, the reduction algorithm samples a vector $\mathbf{k}_{\mathbf{t}_i}$ (from a discrete Gaussian distribution) and sets $\mathbf{t}_i = \mathbf{A}\mathbf{k}_{\mathbf{t}_i}$.

To answer a key-generation query for a slot $i \in [N]$ and policy C, the reduction algorithm samples $(\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,N}, \mathbf{d}_i)$ by sampling $\mathbf{V}^{-1}(\boldsymbol{\eta}_i \otimes \mathbf{t}_i)$ and then sets $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$. In particular,

$$\mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i\mathbf{r}_i + \mathbf{t}_i \quad \text{and} \quad \forall i \neq j : \mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i\mathbf{r}_j. \tag{2.9}$$

Note that the reduction algorithm changes the i^{th} target from $\mathbf{p} + \mathbf{B}_C\mathbf{G}^{-1}(\mathbf{t}_i)$ as in the real scheme to the vector $\mathbf{t}_i$. Targeting $\mathbf{t}_i = \mathbf{A}\mathbf{k}_{\mathbf{t}_i}$ will allow the reduction to simulate the challenge ciphertext later on. As shown in [CW24], changing the i^{th} target only changes the marginal distribution of $\mathbf{y}_{i,i}$ and does *not* change the distribution of the other components $\mathbf{y}_{i,j}$ for $j \neq i$ and $\mathbf{d}_i$ by a noticeable amount. Since the public key for user i just consists of $\mathbf{y}_{i,j}$ for $j \neq i$ and $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$, the reduction algorithm correctly simulates the public key for user i.

This reduction strategy does not simulate the secret key $\mathbf{y}_{i,i}$ correctly (in fact, $\mathbf{y}_{i,i}$ cannot decrypt any ciphertext). As such, we can only prove security in a model where the adversary cannot "corrupt" an honestly-generated key and obtain the associated decryption key. The work of [FWW23] shows how to generically upgrade any registered ABE scheme that does not support corruption queries into one that supports corruption queries in the random oracle model. Since our base construction already relies on random oracles, we can leverage the [FWW23] transformation without introducing additional cryptographic or modeling assumptions. Thus, for the rest of this overview (and also in the technical sections), we focus on the setting where the adversary cannot request secret keys for honest users. Using the [FWW23] transformation, we then obtain a scheme that does support adversarial corruptions.

Simulating the challenge ciphertext. The main challenge is simulating the challenge ciphertext. Let $\widehat{\mathbf{W}} = \mathbf{W}_0 + \sum_{i \in [N]} \mathbf{W}_i$ be the aggregated master public key. Recall that $\mathbf{W}_0$ is derived from the random oracle (by hashing the inputs to the aggregation algorithm) and each $\mathbf{W}_i$ is either an honest user's public key sampled by the reduction or a public keys chosen by the adversary. We start by showing security in the simpler setting where we allow the reduction algorithm to completely pick the value of $\mathbf{W}_0$ and the cross terms $\mathbf{y}_{0,i}$, so long as $\mathbf{A}\mathbf{y}_{0,i} = \mathbf{W}_0\mathbf{r}_i$.

Suppose the adversary chooses a public key $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i})$ for slot i along with an associated policy C. We say the public key pk_i is valid if the following holds:

- For all $j \neq i$, we require that $\mathbf{y}_{i,j}$ is short and is a valid cross term: $\mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i\mathbf{r}_j$.
- Since the public key is generated with respect to a circuit C, we also require that the adversary prove knowledge of the associated secret key. Here, it does so by providing a non-interactive zero-knowledge (NIZK) proof of knowledge π_i of a short vector $\mathbf{y}_{i,i}$ where $\mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i\mathbf{r}_i + \mathbf{p} + \mathbf{B}_C\mathbf{G}^{-1}(\mathbf{t}_i)$. The NIZK proof of knowledge guarantees that the adversary indeed sampled a secret key for the policy C. Note that this NIZK proof is proving knowledge of a secret key, which is a significantly simpler relation than proving that the public key was derived by running the honest key-generation algorithm. The latter approach would lead to a NIZK proof that scales with the size of the policy *and* the number of users, whereas proving knowledge of the secret key requires a NIZK proof whose size scales only with the depth of C and polylogarithmically with the number of users. Note that in the security proof, the reduction will need to extract when answering *random oracle* queries. As such, including a NIZK is crucial for our proof strategy.[3]

[3] It may seem tempting to consider a semi-honest or semi-malicious version of the scheme, prove security without the NIZK, and then appeal to a standard compilers (e.g., [GMW86]) to get security against malicious adversaries. This strategy does not apply in our setting because the reduction needs to extract secret keys when simulating random oracle queries. There is no notion of what constitutes a "semi-honest" random oracle query.

In the registered ABE security game, the adversary is required to provide valid public keys. Note that checking whether a public key is valid or not is a *public* operation (and thus, can be performed by the aggregator).

Now, we describe how to simulate the challenge ciphertext. Simulating the ciphertext components (see Eq. (2.4)) that are independent of $\widehat{\mathbf{W}}$ is straightforward:

$$(\mathbf{v}^\top, \mathbf{v}^\top \mathbf{K}_\mathbf{B}, \mathbf{v}^\top \mathbf{k}_\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor).$$

When $\mathbf{v}^\top = \utilde{\mathbf{s}^\top \mathbf{A}}$, and using the fact that $\mathbf{A}\mathbf{K}_\mathbf{B} = \mathbf{B} - \mathbf{x}^\top \otimes \mathbf{G}$, $\mathbf{A}\mathbf{k}_\mathbf{p} = \mathbf{p}$, this corresponds to

$$\begin{aligned}(\mathbf{v}^\top, \mathbf{v}^\top \mathbf{K}_\mathbf{B}, \mathbf{v}^\top \mathbf{k}_\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor) &= \big(\utilde{\mathbf{s}^\top \mathbf{A}}\ ,\ \utilde{\mathbf{s}^\top \mathbf{A}}\mathbf{K}_\mathbf{B}\ ,\ \utilde{\mathbf{s}^\top \mathbf{A}}\mathbf{k}_\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor\ \big) \\ &= \big(\utilde{\mathbf{s}^\top \mathbf{A}}\ ,\ \utilde{\mathbf{s}^\top(\mathbf{B} - \mathbf{x}^\top \otimes \mathbf{G})}\ ,\ \utilde{\mathbf{s}^\top \mathbf{p} + \mu \cdot \lfloor q/2 \rfloor}\big).\end{aligned}$$

This is precisely the distribution of the corresponding components in Eq. (2.4). When $\mathbf{v}$ is random, then by the leftover hash lemma, these components are statistically close to uniform. The remaining question is simulating $\utilde{\mathbf{s}^\top(\mathbf{W}_0 + \sum_{j \in [N]} \mathbf{W}_j)}$. The idea is for the reduction algorithm to sample a short matrix $\mathbf{K}_\mathbf{W}$ and "program"

$$\mathbf{W}_0 \coloneqq \mathbf{A}\mathbf{K}_\mathbf{W} - \textstyle\sum_{j \in [N]} \mathbf{W}_j.$$

With this choice of variables, the reduction algorithm can simulate the challenge ciphertext component with $\mathbf{v}^\top \mathbf{K}_\mathbf{W}$. When $\mathbf{v}^\top = \utilde{\mathbf{s}^\top \mathbf{A}}$, we have

$$\mathbf{v}^\top \mathbf{K}_\mathbf{W} = \utilde{\mathbf{s}^\top \mathbf{A}}\mathbf{K}_\mathbf{W} = \utilde{\mathbf{s}^\top(\mathbf{W}_0 + \textstyle\sum_{j \in [N]} \mathbf{W}_j)},$$

which is distributed as in the real scheme. Similarly, when $\mathbf{v}^\top$ is random, then $\mathbf{v}^\top \mathbf{K}_\mathbf{W}$ is statistically close to uniform. The catch, however, is that the reduction cannot simply choose $\mathbf{W}_0$ arbitrarily. It also needs to simulate the cross-terms $\mathbf{y}_{0,i}$ where $\mathbf{A}\mathbf{y}_{0,i} = \mathbf{W}_0 \mathbf{r}_i$ for all $i \in [N]$. Moreover, the *joint distribution* of $(\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{W}_0)$ must be distributed as in the real scheme (e.g., derived from a fresh sample $\mathbf{V}^{-1}(\mathbf{0})$). We first show that the reduction algorithm can obtain *some* short $\mathbf{y}_{0,i} \in \mathbb{Z}_q^m$ where $\mathbf{A}\mathbf{y}_{0,i} = \mathbf{W}_0 \mathbf{r}_i$. Afterwards, we revisit the distribution question. We consider two cases:

- Suppose the public key $\mathbf{W}_i$ associated with slot $i \in [N]$ is honestly generated (e.g., chosen by the reduction algorithm). In this case, the reduction knows a short $\mathbf{y}_{i,i}$ where $\mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i \mathbf{r}_i + \mathbf{t}_i$ (see Eq. (2.9)). Moreover, for $j \neq i$, the reduction also knows a short $\mathbf{y}_{j,i}$ where $\mathbf{A}\mathbf{y}_{j,i} = \mathbf{W}_j \mathbf{r}_i$. Namely, either the reduction algorithm sampled $\mathbf{y}_{j,i}$ itself in response to an honest key-generation query (in which case Eq. (2.9) holds) or the adversary chose the public key pk_j, which necessarily includes a short $\mathbf{y}_{j,i}$ where $\mathbf{A}\mathbf{y}_{j,i} = \mathbf{W}_j \mathbf{r}_i$ (otherwise, pk_j is an invalid public key). In either case,

$$\mathbf{W}_0 \mathbf{r}_i = \left(\mathbf{A}\mathbf{K}_\mathbf{W} - \sum_{j \in [N]} \mathbf{W}_j\right)\mathbf{r}_i = \mathbf{A}\mathbf{K}_\mathbf{W}\mathbf{r}_i - \sum_{j \in [N]} \mathbf{W}_j \mathbf{r}_i = \mathbf{A}\mathbf{K}_\mathbf{W}\mathbf{r}_i - \sum_{j \in [N]} \mathbf{A}\mathbf{y}_{j,i} + \mathbf{t}_i = \mathbf{t}_i + \mathbf{A}\underbrace{\left(\mathbf{K}_\mathbf{W}\mathbf{r}_i - \sum_{j \in [N]} \mathbf{y}_{j,i}\right)}_{\text{short}}.$$

- Suppose the public key $\mathbf{W}_i$ associated with slot $i \in [N]$ is chosen by the adversary. We require in this case that the associated policy C_i does *not* satisfy the challenge attribute $\mathbf{x}$ (i.e., $C_i(\mathbf{x}) = 1$). By Eq. (2.1), this means

$$(\mathbf{B} - \mathbf{x}^\top \otimes \mathbf{G})\mathbf{H}_{\mathbf{B},C_i,\mathbf{x}} = \mathbf{B}_{C_i} - C_i(\mathbf{x}) \cdot \mathbf{G} = \mathbf{B}_{C_i} - \mathbf{G}.$$

Since $\mathbf{A}\mathbf{K}_\mathbf{B} = \mathbf{B} - \mathbf{x}^\top \otimes \mathbf{G}$, this means

$$\mathbf{A}\mathbf{K}_\mathbf{B}\mathbf{H}_{\mathbf{B},C_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{t}_i) = (\mathbf{B} - \mathbf{x}^\top \otimes \mathbf{G})\mathbf{H}_{\mathbf{B},C_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{t}_i) = \mathbf{B}_{C_i}\mathbf{G}^{-1}(\mathbf{t}_i) - \mathbf{t}_i \tag{2.10}$$

Recall that each public key contains a NIZK proof of knowledge of the associated decryption key. In this case, the reduction algorithm uses the knowledge extractor to extract a short vector $\mathbf{y}_{i,i}$ where $\mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i \mathbf{r}_i + \mathbf{p} + \mathbf{B}_{C_i}\mathbf{G}^{-1}(\mathbf{t}_i)$. Moreover, as in the previous case, the reduction algorithm also knows a short $\mathbf{y}_{j,i}$ where $\mathbf{A}\mathbf{y}_{j,i} = \mathbf{W}_j \mathbf{r}_i$ for all

$j \neq i$. This allows us to write

$$\begin{aligned}
\mathbf{W}_0\mathbf{r}_i = \left(\mathbf{A}\mathbf{K}_{\mathbf{W}} - \sum_{j\in[N]} \mathbf{W}_j\right)\mathbf{r}_i &= \mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in[N]} \mathbf{W}_j\mathbf{r}_i \\
&= \mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in[N]} \mathbf{A}\mathbf{y}_{j,i} + \mathbf{p} + \mathbf{B}_{C_i}\mathbf{G}^{-1}(\mathbf{t}_i) \\
&= \mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in[N]} \mathbf{A}\mathbf{y}_{j,i} + \mathbf{A}\mathbf{k}_{\mathbf{p}} + \mathbf{B}_{C_i}\mathbf{G}^{-1}(\mathbf{t}_i) \\
&= \mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in[N]} \mathbf{A}\mathbf{y}_{j,i} + \mathbf{A}\mathbf{k}_{\mathbf{p}} + \mathbf{A}\mathbf{K}_{\mathbf{B}}\mathbf{H}_{\mathbf{B},C_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{t}_i \\
&= \mathbf{t}_i + \mathbf{A}\underbrace{\left(\mathbf{K}_{\mathbf{W}}\mathbf{r}_i + \mathbf{k}_{\mathbf{p}} + \mathbf{K}_{\mathbf{B}}\mathbf{H}_{\mathbf{B},C_i,\mathbf{x}}\mathbf{G}^{-1}(\mathbf{t}_i) - \sum_{j\in[N]} \mathbf{y}_{j,i}\right)}_{\text{short}}.
\end{aligned}$$

To summarize, the above analysis shows that when $\mathbf{W}_0 = \mathbf{A}\mathbf{K}_{\mathbf{W}} - \sum_{j\in[N]} \mathbf{W}_j$, the reduction algorithm can construct a preimage $\tilde{\mathbf{y}}_{0,i} \in \mathbb{Z}_q^m$ where

$$\mathbf{W}_0\mathbf{r}_i = \mathbf{A}\tilde{\mathbf{y}}_{0,i} + \mathbf{t}_i = \mathbf{A}(\tilde{\mathbf{y}}_{0,i} + \mathbf{k}_{\mathbf{t}_i})$$

for all $i \in [N]$. In the real aggregation algorithm, if the cross terms $\mathbf{y}_{0,i}$ and $\mathbf{W}_0$ are obtained by sampling from $\mathbf{V}^{-1}(\mathbf{0})$ (with randomness derived from the random oracle), the distribution of $(\mathbf{W}_0, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N})$ is statistically close to sampling

$$\mathbf{W}_0 \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m} \quad \text{and} \quad \forall i \in [N] : \mathbf{y}_{0,i} \leftarrow \mathbf{A}^{-1}(\mathbf{W}_0\mathbf{r}_i).$$

We now need to argue that the matrix $\mathbf{W}_0$ and the cross terms $\mathbf{y}_{0,i}$ that the reduction samples also has this distribution.

The claim reduces to showing that the distribution of $\tilde{\mathbf{y}}_{0,i} + \mathbf{k}_{\mathbf{t}_i}$ is statistically close to sampling $\mathbf{A}^{-1}(\mathbf{W}_0\mathbf{r}_i)$. When the vector $\mathbf{k}_{\mathbf{t}_i}$ is sampled from a sufficiently-wide discrete Gaussian (i.e., one whose width is super-polynomially larger than the norm of $\tilde{\mathbf{y}}_{0,i}$), then the distributions of $\mathbf{k}_{\mathbf{t}_i} + \tilde{\mathbf{y}}_{0,i}$ and $\mathbf{A}^{-1}(\mathbf{W}_0\mathbf{r}_i)$ are statistically close. We formalize this using a Gaussian preimage smudging lemma that says that for any target vector $\mathbf{t} \in \mathbb{Z}_q^n$, any vector $\mathbf{z} \in \mathbb{Z}_q^m$, if we consider a discrete Gaussian whose width is at least $\lambda^{\omega(1)} \cdot \|\mathbf{z}\|$, then the distributions $\mathbf{A}^{-1}(\mathbf{t} + \mathbf{A}\mathbf{z})$ and $\mathbf{A}^{-1}(\mathbf{t}) + \mathbf{z}$ are statistically indistinguishable (see Section 4.2 and Theorem 4.3). Using our Gaussian preimage smudging lemma, we can then show that the cross terms $\mathbf{y}_{0,i} \coloneqq \mathbf{k}_{\mathbf{t}_i} + \tilde{\mathbf{y}}_{0,i}$ sampled by the reduction are statistically close to honestly-generated cross terms (e.g., those derived by computing $\mathbf{V}^{-1}(\mathbf{0})$). Thus, the distribution of $(\mathbf{W}_0, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N})$ sampled by the reduction algorithm is statistically close to that sampled by the real aggregation algorithm.

Explainable sampling. The remaining issue is that in the real scheme, the public matrix $\mathbf{W}_0$ and the associated cross terms $\mathbf{y}_{0,i}$ are obtained by sampling from $\mathbf{V}^{-1}(\mathbf{0})$ using the randomness γ derived from the random oracle (by hashing the inputs to the aggregation algorithm). This is necessary to ensure a deterministic aggregation process. In the proof, the reduction needs a way to reverse engineer this process: given a (properly-distributed) tuple $(\mathbf{W}_0, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N})$ find a random string γ that "explains" it. If we have such an algorithm, then the reduction algorithm can simply program the random oracle to output the target string γ when it is queried on the inputs to the aggregation algorithm.

For this to be possible, we require that the discrete Gaussian sampling algorithm used to sample from $\mathbf{V}^{-1}(\mathbf{0})$ to be "explainable" [LW22]: namely, given $\mathbf{x} \leftarrow \mathbf{V}^{-1}(\mathbf{y})$, there is an explain algorithm that outputs a set of (uniformly-random) coins that would cause the sampling algorithm to output the preimage $\mathbf{x}$. For our application, we observe that the classic Gentry-Peikert-Vaikuntanathan [GPV08] preimage sampler is explainable (Sections 4.1 and 7). Thus, in our security proof, we can implement our reduction strategy described above for simulating the challenge ciphertext, and then program the random oracle to output the randomness needed to explain the programmed tuple $(\mathbf{W}_0, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N})$. One technicality here is that our reduction algorithm above programs the matrix $\mathbf{W}_0$, whereas the preimage sampler outputs a short vector $\mathbf{d}_0$ where $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. Thus, the reduction algorithm needs an efficient way to sample a short $\mathbf{d}_0$ such that $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. It turns out that the [CW24] transformation from the ℓ-succinct LWE trapdoor to the trapdoor for $\mathbf{V}$ produces an associated trapdoor for a matrix related to $\mathbf{Z}$ that allows one to efficiently sample such a $\mathbf{d}_0$. We defer the details to Section 4.3 (see Lemma 4.7).

Ciphertext compression using ℓ-succinct LWE. In the description above, we embed the attributes in the ciphertext in the same way as the centralized ABE scheme from [BGG+14] (i.e., $\mathbf{s}^{\mathsf{T}}(\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G})$). This means the size of the ciphertext scales linearly with the length of the attribute. Recently, Wee [Wee24] show how to use the ℓ-succinct LWE assumption to compress the attribute (a similar approach was also used in the succinct functional commitment from [WW23a]). Specifically, Wee showed how to compress the attribute encoding $\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}$ using the ℓ-succinct LWE trapdoor. To illustrate, we first write the ℓ-succinct LWE trapdoor for the matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ as follows:

$$\left[\begin{array}{ccc|c} \mathbf{A} & & & \mathbf{U}_1 \\ & \ddots & & \vdots \\ & & \mathbf{A} & \mathbf{U}_\ell \end{array}\right] \cdot \begin{bmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_\ell \\ \underline{\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \mathbf{G} & & \\ & \ddots & \\ & & \mathbf{G} \end{bmatrix},$$

where $\mathbf{A}, \mathbf{U}_1, \ldots, \mathbf{U}_\ell \in \mathbb{Z}_q^{n \times m}$. The observation in [Wee24] is that

$$\left[\mathbf{A} \mid \textstyle\sum_{i \in [\ell]} x_i \mathbf{U}_i\right] \begin{bmatrix} \sum_{i \in [\ell]} \mathbf{T}_i \\ \underline{\mathbf{T}} \end{bmatrix} = \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}.$$

Then, for a matrix $\mathbf{A}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$,

$$\left[\mathbf{A} \mid \mathbf{A}_0 + \textstyle\sum_{i \in [\ell]} x_i \mathbf{U}_i\right] \begin{bmatrix} -\sum_{i \in [\ell]} \mathbf{T}_i \\ -\underline{\mathbf{T}} \end{bmatrix} = -\mathbf{A}_0 \underline{\mathbf{T}} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}.$$

Let $\mathbf{B} = -\mathbf{A}_0 \underline{\mathbf{T}}$. Then, $\left[\mathbf{A} \mid \mathbf{A}_0 + \sum_{i \in [\ell]} x_i \mathbf{U}_i\right] \in \mathbb{Z}_q^{n \times 2m}$ is a *compressed* representation of $\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times \ell m}$. The work of [Wee24] uses this technique to obtain an ABE scheme where the ciphertext size is independent of the attribute length; we can use the same compression technique in our scheme to obtain a registered ABE scheme where the size of the ciphertext size is also independent of the attribute length. To take advantage of this compression technique, the master public key of the ABE scheme would need to include the trapdoor for the ℓ-succinct LWE matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$. Since we already need this trapdoor to derive the components for key-generation, we can take advantage of this compression with no overhead. Our complete scheme is described in Section 5.2 (Construction 5.6). As noted in Section 1.1, this is the first registered ABE scheme from *any* assumption with succinct ciphertexts (independent of attribute length). This in turn also implies an identity-based distributed broadcast encryption scheme (see Remark 5.38).

2.2 Adaptively-Secure Distributed Broadcast Encryption

The re-randomization approach described in Section 2.1 for proving security of our registered ABE scheme can also be applied to the distributed broadcast encryption scheme from [CW24] to obtain a distributed broadcast encryption scheme with *semi-static* security. In a semi-statically-secure broadcast encryption scheme [GW09], the adversary is required to declare a *superset* S^* of its challenge set at the beginning of the security game and is not allowed to request decryption keys for any index $i \in S^*$. In the challenge phase, the adversary is allowed to choose any set $S \subseteq S^*$ that is a subset of S^*. This is a stronger security property than selective security which requires the adversary to declare its actual challenge set at the beginning of the security game.

As discussed in Section 2.1, the work of [CW24] only considers *selective* security, and moreover, their proof strategy critically relied on the ability to program the exact challenge set into the scheme parameters. Using our randomized aggregation technique, we can show a variant of the [CW24] distributed broadcast encryption scheme satisfies semi-static security in the random oracle model. Specifically, instead of programming the keys for the challenge set into the public parameters (as in [CW24]), the reduction instead re-randomizes the aggregated key (for the challenge set) at encryption time. The reduction uses the same re-randomization technique as our registered ABE scheme. Our broadcast encryption scheme is *not* adaptively secure because the reduction cannot answer *adaptive* key-generation queries (for the same reason that our base registered ABE scheme does not support corruptions). The only adaptivity we can support is in the adversary's choice of the challenge set; this coincides with the notion of semi-static security. We give our construction of semi-statically-secure distributed broadcast encryption from ℓ-succinct LWE in the random oracle model in Section 6.

The work of [GW09] shows how to transform a semi-statically-secure broadcast encryption into an *adaptively* secure broadcast encryption scheme with only constant overhead in the random oracle model. A similar transformation is also possible with a semi-statically-secure distributed broadcast encryption schemes [KMW23]. In combination with our semi-statically secure distributed broadcast encryption in the random oracle model, we obtain an adaptively-secure distributed broadcast encryption from the ℓ-succinct LWE assumption in the random oracle model. Previously, the only lattice-based construction of adaptively-secure (distributed) broadcast encryption relied on witness encryption in the random oracle model [FWW23]. All other lattice-based constructions of (centralized or distributed) broadcast encryption [BV22, Wee22, Wee24, CW24] only achieved selective security.

3 Preliminaries

We write λ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $[n] := \{1, \ldots, n\}$. For a finite set S, we write $x \xleftarrow{\text{R}} S$ to denote that x is a uniform random draw from S. We write $x \leftarrow \mathcal{D}$ to denote that x is sampled from the distribution $\mathcal{D}$. We write $\mathsf{poly}(\lambda)$ to denote a fixed polynomial in λ and $\mathsf{negl}(\lambda)$ to denote a function that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. 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 $\lambda \in \mathbb{N}$,

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

We say that they are statistically indistinguishable if their statistical distance $\Delta(\mathcal{D}_0, \mathcal{D}_1)$ is bounded by $\mathsf{negl}(\lambda)$. We say an event occurs with overwhelming probability if the probability of its complement occurring is negligible.

Simulation-sound extractable NIZKs. Next, we recall the notion of a simulation-sound extractable non-interactive zero-knowledge (NIZK) argument for NP [BFM88, FLS90, Sah99, DDO+01]. We give the definition below.

Definition 3.1 (Simulation-Sound Extractable NIZK)**.** A simulation-sound extractable NIZK 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. This yields the following instantiation:

Fact 3.2 (Simulation-Sound Extractable NIZK from LWE)**.** Under the plain LWE assumption (with a polynomial modulus-to-noise ratio), there exists a simulation-sound extractable NIZK for NP.

3.1 Lattice Preliminaries

We now recall some basic facts about lattices. Throughout this work, we use bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices and bold lowercase letters (e.g., $\mathbf{u}, \mathbf{v}$) to denote vectors. We use non-boldface letters to denote their components (e.g., $\mathbf{v} = [v_1, \ldots, v_n]$). For a vector $\mathbf{v} \in \mathbb{R}^n$, we write $\|\mathbf{v}\| = \max_i |v_i|$ to denote the ℓ_∞-norm of $\mathbf{v}$, and for a matrix $\mathbf{V}$ we write $\|\mathbf{V}\| = \max_{i,j} \left|V_{i,j}\right|$. We write $\|\mathbf{v}\|_2$ to denote the ℓ_2-norm of $\mathbf{v}$ (i.e., $\|\mathbf{v}\|_2^2 \coloneqq \sum_{i \in [n]} v_i^2$). When $\mathbf{v} \in \mathbb{Z}_q^n$, we write $\|\mathbf{v}\|$ (resp., $\|\mathbf{v}\|_2$) to denote the ℓ_∞-norm (resp., ℓ_2-norm) of the vector obtained by associating each component v_i with its unique representative in the interval $(-q/2, q/2]$.

Tensor products and vectorization. For matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{B} \in \mathbb{Z}_q^{k \times \ell}$, we write $\mathbf{A} \otimes \mathbf{B} \in \mathbb{Z}_q^{nk \times m\ell}$ to denote their tensor (Kronecker) product. For matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ where the products $\mathbf{AC}$ and $\mathbf{BD}$ are well-defined, then

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

For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we write $\mathsf{vec}(\mathbf{A})$ to denote the vectorization of $\mathbf{A}$ (i.e., the vector $\mathbf{a} \in \mathbb{Z}_q^{nm}$ obtained by concatenating together the columns of $\mathbf{A}$ in left-to-right order).

Leftover hash lemma. Next, we recall a generalization of the leftover hash lemma [HILL99, DORS08, ABB10]:

Lemma 3.3 (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 $\mathsf{negl}(n)$:*

$$\left\{ (\mathbf{A}, \mathbf{AK}, \mathbf{e}^{\mathsf{T}}\mathbf{K}) : \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{K} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times k} \right\} \quad \textit{and} \quad \left\{ (\mathbf{A}, \mathbf{U}, \mathbf{e}^{\mathsf{T}}\mathbf{K}) : \begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times k} \\ \mathbf{K} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times k} \end{array} \right\}.$$

Corollary 3.4 (Column Space of Random Matrix [GPV08, Lemma 5.1])**.** *Let n, m, q be lattice parameters where q is prime and $m \geq 2n \log q$. Then, for all but a $\mathsf{negl}(n)$ fraction of matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the columns of $\mathbf{A}$ generate $\mathbb{Z}_q^n$.*

Discrete Gaussians and gadget matrices. We write $D_{\mathbb{Z},\sigma}$ to denote the discrete Gaussian distribution over $\mathbb{Z}$ with width parameter $\sigma > 0$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a target vector $\mathbf{y} \in \mathbb{Z}_q^n$ in the column-space of $\mathbf{A}$, we write $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ to denote a random variable $\mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}^m$ conditioned on $\mathbf{Ax} = \mathbf{y} \bmod q$. We extend $\mathbf{A}_\sigma^{-1}(\cdot)$ to matrices by applying $\mathbf{A}_\sigma^{-1}(\cdot)$ to each column of the input. For positive integers $n, q \in \mathbb{N}$, let $\mathbf{G}_n = \mathbf{I}_n \otimes \mathbf{g}^{\mathsf{T}} \in \mathbb{Z}_q^{n \times m'}$ be the gadget matrix [MP12] where $\mathbf{I}_n$ is the identity matrix of dimension n, $\mathbf{g}^{\mathsf{T}} = [1, 2, \ldots, 2^{\lceil \log q \rceil - 1}]$, and $m' = n \lceil \log q \rceil$. We write $\mathbf{G}_n^{-1} \colon \mathbb{Z}_q^n \to \{0,1\}^{m'}$ to denote the operator that expands each component of the input into its binary decomposition (i.e., $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{x}) = \mathbf{x}$ for all $\mathbf{x} \in \mathbb{Z}_q^n$). We extend $\mathbf{G}_n^{-1}(\cdot)$ to operate on matrices in a column-wise manner. For $m > m'$, we overload $\mathbf{G}_n$ to denote the padded gadget matrix $\mathbf{G}_n = [\mathbf{I}_n \otimes \mathbf{g}^{\mathsf{T}} \mid \mathbf{0}^{n \times (m - m')}]$. We define $\mathbf{G}_n^{-1}$ analogously (i.e., padding the output with zeroes). We now recall some basic properties of the discrete Gaussian distribution.

Lemma 3.5 (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 $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, for all $\sigma > \log m$, and all vectors $\mathbf{y} \in \mathbb{Z}_q^n$ in the span of $\mathbf{A}$,*

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

For the particular case of the discrete Gaussian distribution over the integers and any $\lambda \in \mathbb{N}$,

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

Lemma 3.6 (Gaussian Samples [GPV08, adapted])**.** *Let n, m, q, σ be lattice parameters such that $\sigma \geq \log m$, $m \geq 2n \log q$, and q is prime. There exist a negligible function $\mathsf{negl}(\cdot)$ such that for all but a q^{-n}-fraction of matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the statistical distance between the following distributions is at most $\mathsf{negl}(n)$:*

$$\left\{ (\mathbf{x}, \mathbf{Ax}) : \mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}^m \right\} \quad \textit{and} \quad \left\{ (\mathbf{x}, \mathbf{y}) : \mathbf{y} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n, \mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}) \right\}.$$

Lemma 3.7 (Marginal of Gaussian Preimages [WW23b, 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 $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, all matrices $\mathbf{B} \in \mathbb{Z}_q^{n\ell \times k}$ and matrices $\mathbf{C} = [\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{B}]$, 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 $\mathsf{negl}(n)$:*

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

Lattice trapdoors. We recall the notion of a gadget trapdoor [MP12]:

Lemma 3.8 (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:*

- $\mathsf{TrapGen}(1^n, q, m) \to (\mathbf{A}, \mathbf{T})$: *On input the lattice dimension n, the modulus q, and the number of samples m, the trapdoor-generation algorithm outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m \times m'}$ where $m' = n \lceil \log q \rceil$.*
- $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma) \to \mathbf{x}$: *On input 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 $\mathbf{y} \in \mathbb{Z}_q^n$, and a Gaussian width parameter σ, the preimage-sampling algorithm outputs a vector $\mathbf{x} \in \mathbb{Z}_q^m$.*

Moreover, the above algorithms satisfy the following properties:

- **Trapdoor distribution:** *If $(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ and $\mathbf{A}' \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, then $\Delta(\mathbf{A}, \mathbf{A}') = \mathsf{negl}(n)$. Moreover, $\mathbf{AT} = \mathbf{G}_n \in \mathbb{Z}_q^{n \times m'}$ and $\|\mathbf{T}\| = 1$.*
- **Preimage sampling:** *For all matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T} \in \mathbb{Z}_q^{m \times m'}$, width parameter $\sigma > 0$, and all target vectors $\mathbf{y} \in \mathbb{Z}_q^n$ in the column span of $\mathbf{A}$, the output $\mathbf{x} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$ satisfies $\mathbf{Ax} = \mathbf{y}$.*
- **Preimage distribution:** *There exist a negligible function $\mathsf{negl}(\cdot)$ such that for all $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{T} \in \mathbb{Z}_q^{m \times m'})$ where $\mathbf{T}$ is a gadget trapdoor for $\mathbf{A}$ (i.e., $\mathbf{AT} = \mathbf{G}_n$), all $\sigma \geq m \|\mathbf{T}\| \log n$ and all target vectors $\mathbf{y} \in \mathbb{Z}_q^n$, the statistical distance between the following distributions is at most $\mathsf{negl}(n)$:*

$$\{\mathbf{x} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)\} \quad \textit{and} \quad \{\mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})\}.$$

Homomorphic evaluation. Our construction of registered ABE for circuits will rely on the lattice homomorphic evaluation procedure developed in [GSW13, BGG+14]. Our presentation is adapted from that in [BV15, BCTW16, BTVW17].

Theorem 3.9 (Homomorphic Encodings [GSW13, BGG+14]). *Let λ be a security parameter and $n = n(\lambda)$, $q = q(\lambda)$ be lattice parameters. Take any $m \geq n \lceil \log q \rceil$, and let $\ell = \ell(\lambda)$ be an input length. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of functions $f \colon \{0,1\}^\ell \to \{0,1\}$ that can be computed by a Boolean circuit of depth at most $d = d(\lambda)$. Then, there exist a pair of efficient algorithms* $(\mathsf{EvalF}, \mathsf{EvalFX})$ *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 $f \in \mathcal{F}$, the input-independent evaluation algorithm outputs 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 $f \in \mathcal{F}$, and an input $\mathbf{x} \in \{0,1\}^\ell$, the input-dependent evaluation algorithm outputs a matrix $\mathbf{H}_{\mathbf{A},f,\mathbf{x}} \in \mathbb{Z}_q^{\ell m \times m}$.*

Moreover for all security parameters $\lambda \in \mathbb{N}$, matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times \ell m}$, all functions $f \in \mathcal{F}$, and all 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:

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

Learning with errors and ℓ-succinct LWE. The learning with errors (LWE) assumption [Reg05] with parameters (n, m, q, σ) states that the distributions of $(\mathbf{A}, \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T})$ is computationally indistinguishable from $(\mathbf{A}, \mathbf{v}^\mathsf{T})$ when $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\sigma}^m$, and $\mathbf{v} \xleftarrow{\text{R}} \mathbb{Z}_q^m$. We will also use the ℓ-succinct LWE assumption introduced by Wee [Wee24], which asserts that LWE is hard with respect to $\mathbf{A}$ even given a trapdoor for the matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ where $\mathbf{U} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\ell \times m}$. We now give the formal statement of the assumption:

Assumption 3.10 (ℓ-Succinct LWE [Wee24])**.** Let λ be a security parameter and let $n = n(\lambda), m = m(\lambda), q = q(\lambda), \sigma = \sigma(\lambda)$ be lattice parameters. Let $s = s(\lambda)$ be a Gaussian width parameter and $\ell = \ell(\lambda)$ be a dimension. We say that the ℓ-succinct LWE assumption with parameters (n, m, q, σ, s) holds if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$:

$$\left|\Pr[\mathcal{A}(1^\lambda, \mathbf{A}, \mathbf{s}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}, \mathbf{U}, \mathbf{T}) = 1] - \Pr[\mathcal{A}(1^\lambda, \mathbf{A}, \mathbf{v}^\mathsf{T}, \mathbf{U}, \mathbf{T}) = 1]\right| = \mathsf{negl}(\lambda),$$

where $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\sigma}^m$, $\mathbf{v} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$, $\mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\ell\times m}$, and $\mathbf{T} \leftarrow [\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]_s^{-1}(\mathbf{G}_{n\ell})$.[4]

4 Lattice Building Blocks

In this section, we introduce several new building blocks that we use in our main constructions. We believe that abstracting out these components provide a simpler and more modular view of our constructions (Sections 5 and 6). The building blocks we describe here are general and may also be useful in other settings. These include our explainable discrete Gaussian preimage sampler (Section 4.1), our Gaussian preimage smudging lemma (Section 4.2), and some simple transformations for using the ℓ-succinct LWE trapdoor (Section 4.3).

4.1 Explainable Discrete Gaussian Preimage Sampler

As described in Section 2.1, a key ingredient in our ciphertext re-randomization technique is an "explainable algorithm" for sampling from the distribution $\mathbf{A}_\sigma^{-1}(\mathbf{y})$. Namely, there is an Explain algorithm that takes any preimage $\mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})$ and outputs a sequence of random coins that would cause the sampling algorithm to output $\mathbf{x}$. Previously, the work of Lu and Waters [LW22] showed how to construct an explainable discrete Gaussian sampler for sampling from a discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ over the integers. For our application, we require a scheme for sampling over an arbitrary lattice coset. We give the precise definition here, and in Section 7, we show that combining an explainable discrete Gaussian sampler over the integers with the Gentry-Peikert-Vaikuntanathan preimage sampling algorithm [GPV08] yields an explainable discrete Gaussian sampler for sampling from an arbitrary lattice.

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

- $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r) \to \mathbf{x}$: On input a security parameter λ, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a gadget trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m\times m'}$, a target vector $\mathbf{y} \in \mathbb{Z}_q^n$, a width parameter σ, and randomness $r \in \{0,1\}^{\rho(\lambda,n,m,q)}$, the preimage sampling algorithm outputs a vector $\mathbf{x} \in \mathbb{Z}_q^m$.
- $\mathsf{Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{T}, \mathbf{y}, \mathbf{x}, \sigma) \to r$: On input a security parameter λ, a precision parameter κ, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, a gadget trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m\times m'}$, a target vector $\mathbf{y} \in \mathbb{Z}_q^n$, a preimage $\mathbf{x} \in \mathbb{Z}_q^m$, and a width parameter σ, the explain algorithm outputs a string $r \in \{0,1\}^{\rho(\lambda,n,m,q)}$.

Moreover, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}, \mathbf{T} \in \mathbb{Z}_q^{m\times m'}$ such that $\mathbf{AT} = \mathbf{G}$, all targets $\mathbf{y} \in \mathbb{Z}_q^n$ where $\|\mathbf{y}\| \le 2^\lambda$, and all width parameters σ where $\|\mathbf{T}\| \cdot \sigma_{\mathrm{loss}}(\lambda, n, m, q) \le \sigma \le 2^\lambda$, the following two properties holds.

- **Correctness:** The statistical distance between the following distributions is bounded by $\mathsf{negl}(\lambda)$:

$$\left\{\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)\right\} \quad \text{and} \quad \left\{\mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})\right\}.$$

Moreover, for all $\mathbf{x}$ in the support of $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$, we have $\mathbf{Ax} = \mathbf{y}$.

[4] If $\mathbf{G}_{n\ell}$ is not in the image of $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$, we set $\mathbf{T} = \perp$. When $m \ge 2n\log q$, the matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ is full rank with overwhelming probability.

- **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 $r \xleftarrow{\mathrm{R}} \{0,1\}^{\rho}$ and $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^{\lambda}, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r)$. Output $(\mathbf{x}, r)$.
  - $\mathcal{D}_{\mathsf{Explain}}$: Sample $r' \xleftarrow{\mathrm{R}} \{0,1\}^{\rho}$, $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^{\lambda}, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r')$, and $r \leftarrow \mathsf{Explain}(1^{\lambda}, 1^{\kappa}, \mathbf{A}, \mathbf{T}, \mathbf{y}, \mathbf{x}, \sigma)$. Output $(\mathbf{x}, r)$.

Theorem 4.2 (Explainable Discrete Gaussian Preimage Sampler)**.** *There exist a $(\rho, \sigma_{\mathrm{loss}})$-explainable discrete Gaussian preimage sampler Π_{DGS} for $\rho \in \mathsf{poly}(\lambda, n, m, \log q)$*[5] *and $\sigma_{\mathrm{loss}}(\lambda, n, m, q) = 18m^{3/2}\log(m\lambda)\log\log q$*

We give the full construction and analysis for the explainable discrete Gaussian preimage sampler in Section 7.

4.2 Noise Smudging for Gaussian Preimages

Our analysis will rely on the following smudging lemma that roughly states that the distribution of $\mathbf{A}_{\sigma}^{-1}(\mathbf{u} + \mathbf{A}\mathbf{z})$ and $\mathbf{A}_{\sigma}^{-1}(\mathbf{u}) + \mathbf{z}$ is statistically close whenever the width σ of the Gaussian distribution is much larger than $\|\mathbf{z}\|_2$. Roughly speaking, this boils down to the statement that a small *translation* of a sufficiently-wide Gaussian does not affect the distribution. We give the formal statement here and defer the proof to Appendix B.1.

Theorem 4.3 (Gaussian Preimage Smudging)**.** *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 $\sigma > \max(\log m, \|\mathbf{z}\|_2)$ the statistical distance between the following distributions is $O(\sqrt{\|\mathbf{z}\|_2 / \sigma})$:*

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

Remark 4.4 (Gaussian Preimage Smudging)**.** Several prior works [GMPW20, GP21] have considered similar, though incomparable variants of the Gaussian preimage smudging lemma from Theorem 4.3. For instance, the Gaussian convolution lemmas from [GMPW20, §4] show that the distributions of $\mathbf{A}_{\sigma_1}^{-1}(\mathbf{u}+\mathbf{v})$ and $\mathbf{A}_{\sigma_1}^{-1}(\mathbf{u})+\mathbf{A}_{\sigma_2}^{-1}(\mathbf{v})$ are statistically close when $\sigma_1 \gg \sigma_2$. This implies a special case of Theorem 4.3 for the case where $\mathbf{z}$ is distributed according to a discrete Gaussian. However, our application requires this to hold for arbitrary (non-Gaussian) vectors $\mathbf{z}$. The work of [GP21, §3.2.3] design an alternative preimage sampling procedure with the property that the output distributions are statistically close under small translations of the input. However, our applications require that the output distribution are distributed according to a discrete Gaussian distribution, which is not the case for their construction.

4.3 Sampling and Using ℓ-Succinct Trapdoors

Our construction relies on the ℓ-succinct LWE assumption [Wee24]. The ℓ-succinct LWE assumption assets that LWE is hard with respect to matrix $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ given a trapdoor for the matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ where $\mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\ell \times m}$. Our work builds on the work of [CW24] who show how to transform a trapdoor for $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ into a trapdoor for the matrix

$$\mathbf{V} = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \in \mathbb{Z}_q^{nN \times (mN+k)}, \tag{4.1}$$

where $\mathbf{Z} \in \mathbb{Z}_q^{n \times mk}$ and $\mathbf{r}_i \in \mathbb{Z}_q^m$ Our construction relies on both types of trapdoors. To simplify our description, we start by defining some simple transformations on these trapdoors which we use in our construction.

The ℓ-succinct trapdoor sampler. We start by defining an analog of $\mathsf{TrapGen}$ that samples a matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ together with a trapdoor $\mathbf{T}$ of bounded norm:

[5] Without loss of generality, we take ρ to be a monotone function (since an algorithm can always choose to ignore extra random bits).

Algorithm 1: The ℓ-succinct trapdoor sampler algorithm $\mathsf{SuccinctTrapGen}$.

```
$\mathsf{SuccinctTrapGen}(1^n, 1^\ell, q, m, \sigma)$:
    • Sample $(\mathbf{A}, \mathbf{R}) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ and $\mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell n \times m}$.
    • Sample $\mathbf{T} \leftarrow \mathsf{SamplePre}\big([\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}], \big[\begin{smallmatrix} \mathbf{I}_\ell \otimes \mathbf{R} \\ \mathbf{0} \end{smallmatrix}\big], \mathbf{G}_{n\ell}, \sigma\big)$. If $\|\mathbf{T}\| > \sqrt{m}\sigma$, then output $\big[\begin{smallmatrix} \mathbf{I}_\ell \otimes \mathbf{R} \\ \mathbf{0} \end{smallmatrix}\big]$.
    • Output $(\mathbf{A}, \mathbf{U}, \mathbf{T})$.
```

Lemma 4.5 (ℓ-Succinct Trapdoor Sampler). *Let λ be a security parameter and let n, m, q, σ be parameters where $n \ge \lambda$ and $m \ge 3n \log q$. Then, for all polynomials $\ell = \ell(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all $\sigma \ge (m\ell + m) \cdot \log(n\ell)$, the statistical distance between the following distributions is $\mathsf{negl}(\lambda)$:*

$$\big\{(\mathbf{A}, \mathbf{U}, \mathbf{T}) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^\ell, q, m, \sigma)\big\} \quad \text{and} \quad \left\{(\mathbf{A}, \mathbf{U}, \mathbf{T}) : \begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{U} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\ell \times m} \\ \mathbf{T} \leftarrow [\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]_\sigma^{-1}(\mathbf{G}_{n\ell}) \end{array}\right\}.$$

In addition, if $(\mathbf{A}, \mathbf{U}, \mathbf{T}) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^\ell, q, m, \sigma)$, then $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T} = \mathbf{G}_{n\ell}$ and $\|\mathbf{T}\| \le \sqrt{m}\sigma$.

Proof. Take $(\mathbf{A}, \mathbf{U}, \mathbf{T}) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^\ell, q, m, \sigma)$. We first show that the two properties hold:

- By Lemma 3.8, we have that $\mathbf{AR} = \mathbf{G}_n$. This guarantees that $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T} = \mathbf{G}_{n\ell}$.
- By Lemma 3.8, $\|\mathbf{R}\| = 1$, so either $\|\mathbf{T}\| \le \sqrt{m}\sigma$ or $\|\mathbf{T}\| = 1$.

We now analyze the distribution of $(\mathbf{A}, \mathbf{U}, \mathbf{T})$. This follows by a simple hybrid argument:

- $\mathcal{D}_0$: Sample and output $(\mathbf{A}, \mathbf{U}, \mathbf{T}) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^\ell, q, m, \sigma)$.
- $\mathcal{D}_1$: Same as $\mathcal{D}_0$ except sample $\mathbf{T} \leftarrow [\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]_\sigma^{-1}(\mathbf{G}_{n\ell})$.
- $\mathcal{D}_2$: Same as $\mathcal{D}_1$ except remove the check that $\|\mathbf{T}\| > \sqrt{m}\sigma$.
- $\mathcal{D}_3$: Same as $\mathcal{D}_2$ except sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$.

First $\mathcal{D}_0$ and $\mathcal{D}_1$ are statistically indistinguishable by Lemma 3.8. Similarly, $\mathcal{D}_2$ and $\mathcal{D}_3$ are also statistically indistinguishable by Lemma 3.8. To complete the proof, it suffices to argue that $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable. The only difference between these two experiments is if $\|\mathbf{T}\| > \sqrt{m}\sigma$. Let E be the event that this occurs. We bound the probability of E in $\mathcal{D}_1$:

- Consider the probability that $\|\mathbf{T}\| > \sqrt{m}\sigma$ when $\mathbf{T} \leftarrow [\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]_\sigma^{-1}(\mathbf{G}_{n\ell})$ and $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$. By Lemmas 3.5 and 3.7, this happens with negligible probability.
- By Lemma 3.8, the distributions $\mathbf{A} \leftarrow \mathsf{TrapGen}(1^n, q, m)$ and $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ are statistically indistinguishable. If event E occurs with negligible probability when $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, then it also happens with negligible probability when $(\mathbf{A}, \mathbf{R}) \leftarrow \mathsf{TrapGen}(1^n, q, m)$.

Thus, event E happens with negligible probability in $\mathcal{D}_1$, and so $\mathcal{D}_1$ and $\mathcal{D}_2$ are also statistically indistinguishable. The lemma now follows by a hybrid argument. □

Dimension reduction. In our constructions, we start with a trapdoor for the structured matrix $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}]$ for arbitrary $\mathbf{U} \in \mathbb{Z}_q^{n\ell \times t}$ and need to derive from it a trapdoor of $[\mathbf{I}_k \otimes \mathbf{A} \mid \mathbf{U}_S]$, where $S \subseteq [\ell]$ is a set of size k and $\mathbf{U}_S$ is the ordered vertical concatenation of blocks $\mathbf{U}_i \in \mathbb{Z}_q^{n \times t}$ from $\mathbf{U}$ such that $i \in S$.

Algorithm 2: A structured trapdoor dimension reduction algorithm DimRed.

$\mathsf{DimRed}(\mathbf{A}, \mathbf{U}, \mathbf{T}, S)$:

- Parse $\mathbf{T} \in \mathbb{Z}_q^{(m\ell+t)\times \ell m'}$ and $\mathbf{U} \in \mathbb{Z}_q^{n\ell \times t}$ into blocks as follows:

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_\ell \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{B}_{1,1} & \cdots & \mathbf{B}_{1,\ell} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{\ell,1} & \cdots & \mathbf{B}_{\ell,\ell} \\ \mathbf{D}_1 & \cdots & \mathbf{D}_\ell \end{bmatrix} \quad \text{where} \quad \mathbf{U}_i \in \mathbb{Z}_q^{n\times t}, \mathbf{D}_i \in \mathbb{Z}_q^{t\times m'}, \mathbf{B}_{i,j} \in \mathbb{Z}_q^{m\times m'} \text{ for } i,j \in [\ell].$$

- Parse $S = \{i_1, \ldots, i_k\} \subseteq [\ell]$ and output $(\mathbf{U}_S, \mathbf{T}_S)$ where

$$\mathbf{U}_S = \begin{bmatrix} \mathbf{U}_{i_1} \\ \vdots \\ \mathbf{U}_{i_k} \end{bmatrix} \in \mathbb{Z}_q^{nk\times t} \quad \text{and} \quad \mathbf{T}_S = \begin{bmatrix} \mathbf{B}_{i_1,i_1} & \cdots & \mathbf{B}_{i_1,i_k} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{i_k,i_1} & \cdots & \mathbf{B}_{i_k,i_k} \\ \mathbf{D}_{i_1} & \cdots & \mathbf{D}_{i_k} \end{bmatrix} \in \mathbb{Z}_q^{(mk+t)\times km'}.$$

Lemma 4.6 (Dimension Reduction for Structured Trapdoors). *Let n, m, q be lattice parameters, ℓ be a dimension, and set $m' = n\lceil \log q \rceil$. Then, the algorithm* $\mathsf{DimRed}(\mathbf{A}, \mathbf{U}, \mathbf{T}, S)$ *(Algorithm 2) on input $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n\ell\times t}$, $\mathbf{T} \in \mathbb{Z}_q^{(m\ell+t)\times \ell m'}$, $S \subseteq [\ell]$ of size k, and $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T} = \mathbf{G}_{n\ell}$, outputs $(\mathbf{U}_S \in \mathbb{Z}_q^{nk\times t}, \mathbf{T}_S \in \mathbb{Z}_q^{(mk+t)\times km'})$ such that*

$$[\mathbf{I}_k \otimes \mathbf{A} \mid \mathbf{U}_S] \cdot \mathbf{T}_S = \mathbf{G}_{nk} \quad \textit{and} \quad \|\mathbf{T}_S\| \le \|\mathbf{T}\|.$$

Proof. Since the output of Algorithm 2 are submatrices $\mathbf{U}_S \in \mathbb{Z}_q^{nk\times t}$ and $\mathbf{T}_S \in \mathbb{Z}_q^{(mk+t)\times km'}$ of the inputs $\mathbf{U}$ and $\mathbf{T}$, we immediately have $\|\mathbf{T}_S\| \le \|\mathbf{T}\|$. Since $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T} = \mathbf{G}_{n\ell}$ holds, we have

$$\left[\begin{array}{ccc|c} \mathbf{A} & & & \mathbf{U}_1 \\ & \ddots & & \vdots \\ & & \mathbf{A} & \mathbf{U}_\ell \end{array}\right] \cdot \begin{bmatrix} \mathbf{B}_{1,1} & \cdots & \mathbf{B}_{1,\ell} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{\ell,1} & \cdots & \mathbf{B}_{\ell,\ell} \\ \mathbf{D}_1 & \cdots & \mathbf{D}_\ell \end{bmatrix} = \begin{bmatrix} \mathbf{G}_n & & \\ & \ddots & \\ & & \mathbf{G}_n \end{bmatrix}.$$

This implies that for $i \in [\ell]$, we have $\mathbf{A}\mathbf{B}_{i,i} + \mathbf{U}_i\mathbf{D}_i = \mathbf{G}_n$, and furthermore, $\mathbf{A}\mathbf{B}_{j,i} + \mathbf{U}_j\mathbf{D}_i = \mathbf{0}$ for $j \in [\ell]$ such that $j \ne i$. Thus, $\mathbf{T}_S$ satisfies the relation $[\mathbf{I}_k \otimes \mathbf{A} \mid \mathbf{U}_S] \cdot \mathbf{T}_S = \mathbf{G}_{nk}$. □

Transformation to structured trapdoors. Next, we show how to transform a ℓ-succinct trapdoor into a trapdoor for the structured matrix from Eq. (4.1). The transformation is implicit in [CW24, Theorem 5.1], but we abstract out the main requirements we use for our applications. For completeness, we include a proof in Appendix B.2.

Lemma 4.7 (ℓ-succinct LWE Trapdoor Transformation [CW24, adapted]). *Let n, m, q be lattice parameters and let $m' = n\lceil \log q \rceil$. Suppose $m \ge m'$. There exists an efficient randomized algorithm* $\mathsf{Transform}(\mathbf{A}, \mathbf{U}, \mathbf{T}, N)$ *that takes as input a tuple $(\mathbf{A}, \mathbf{U}, \mathbf{T}, N)$ where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{U} \in \mathbb{Z}_q^{\ell n\times m}$, $\mathbf{T} \in \mathbb{Z}_q^{(\ell+1)m\times \ell m'}$, and $N \in \mathbb{N}$, such that $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T} = \mathbf{G}_{n\ell}$ and $\ell \ge Nm'$, and outputs a tuple $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$ with the following properties:*

- *The distribution of $\mathbf{Z}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times mk}$ where $k = 3nm\lceil \log q \rceil$.*
- *Write $\mathbf{Z} = [\mathbf{Z}_1 \mid \cdots \mid \mathbf{Z}_k]$ where $\mathbf{Z}_k \in \mathbb{Z}_q^{n\times m}$. Let $\tilde{\mathbf{Z}} = [\tilde{\mathbf{z}}_1 \mid \cdots \mid \tilde{\mathbf{z}}_k] \in \mathbb{Z}_q^{nm\times k}$ be the matrix where $\tilde{\mathbf{z}}_i = \mathrm{vec}(\mathbf{Z}_i)$ for all $i \in [k]$. Then $\tilde{\mathbf{Z}} \cdot \mathbf{T}_{\tilde{\mathbf{Z}}} = \mathbf{G}_{nm}$ and $\|\mathbf{T}_{\tilde{\mathbf{Z}}}\| = 1$.*

- *Write* $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N] \in \mathbb{Z}_q^{m\times N}$. *The matrix* $\mathbf{V}$ *satisfies*

$$\mathbf{V} = \left[\begin{array}{ccc|ccc} \mathbf{A} & & & -\mathbf{Z}_1\mathbf{r}_1 & \cdots & -\mathbf{Z}_k\mathbf{r}_1 \\ & \ddots & & \vdots & \ddots & \vdots \\ & & \mathbf{A} & -\mathbf{Z}_1\mathbf{r}_N & \cdots & -\mathbf{Z}_k\mathbf{r}_N \end{array}\right] = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right],$$

  and the trapdoor $\mathbf{T}_\mathbf{V}$ *satisfies* $\mathbf{V} \cdot \mathbf{T}_\mathbf{V} = \mathbf{G}_{nN}$.

- *Finally,* $\|\mathbf{T}_\mathbf{V}\|, \|\mathbf{R}\| \leq \|\mathbf{T}\| \cdot \ell m^2$.

Remark 4.8 (On the Width of $\mathbf{U}$)**.** The work of [Wee24] also consider a more general version of ℓ-succinct LWE with an additional parameter $\hat{m}$ which corresponds to the width of the matrix $\mathbf{U} \in \mathbb{Z}_q^{\ell n\times \hat{m}}$ in Algorithm 1. The transformation in Lemma 4.7 also works for succinct trapdoors with a matrix $\mathbf{U}$ of any width $\hat{m} \geq m$. In this the case, the Transform algorithm outputs a matrix $\mathbf{Z} \in \mathbb{Z}_q^{n\times \hat{m}k}$ where $k = 3n\hat{m}\log q$ and $\mathbf{R} \in \mathbb{Z}_q^{\hat{m}\times N}$. While we focus on the particular case where $\hat{m} = m$ (the standard setting for ℓ-succinct LWE), structured trapdoors with other widths may also be useful.

Gaussian preimage sampling using structured trapdoors. A core component of the correctness and security analysis of our constructions is characterizing the distribution of Gaussian preimages sampled according to $\mathbf{V}_\sigma^{-1}(\cdot)$.

Lemma 4.9 (Marginals of Structured Gaussian Preimages)**.** *Let* n, m, q, k *be lattice parameters where* $n \geq \lambda$, $m \geq 3n\log q$, q *is prime, and* $k \geq 2nm\log q$. *Then for all polynomials* $N = N(\lambda)$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all but a* q^{-n}*-fraction of matrices* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and all but a* q^{-nm}*-fraction of matrices* $\mathbf{Z} \in \mathbb{Z}_q^{n\times km}$, *all matrices*

$$\mathbf{V} = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \in \mathbb{Z}_q^{Nn\times(Nm+k)},$$

where $\mathbf{r}_1, \ldots, \mathbf{r}_N \in \mathbb{Z}_q^m$, *all matrices* $\mathbf{T}$ *where* $\mathbf{VT} = \mathbf{G}_{Nn}$, *all* $\sigma \geq (Nm + k)\|\mathbf{T}\|\log(Nn)$, *all target vectors* $\mathbf{y} \in \mathbb{Z}_q^{Nn}$, *and all* $\lambda \in \mathbb{N}$, *the statistical distance between the following distributions is* $\mathsf{negl}(\lambda)$*:*

$$\{\mathbf{u} : \mathbf{u} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}, \mathbf{y}, \sigma)\} \;\; \textit{and} \;\; \left\{ \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{v} \end{bmatrix} : \begin{array}{c} \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}, \mathbf{v} \leftarrow \tilde{\mathbf{Z}}_\sigma^{-1}(\mathsf{vec}(\mathbf{W})) \\ \mathbf{x}_i \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}_i + \mathbf{W}\mathbf{r}_i) \end{array} \right\},$$

where the vector $\mathbf{y}$ *is the vertical concatenation of* $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^n$, $\mathbf{Z} = [\mathbf{Z}_1 \mid \ldots \mid \mathbf{Z}_k]$ *where* $\mathbf{Z}_i \in \mathbb{Z}_q^{n\times m}$, *and* $\tilde{\mathbf{Z}} = [\mathsf{vec}(\mathbf{Z}_1) \mid \cdots \mid \mathsf{vec}(\mathbf{Z}_k)] \in \mathbb{Z}_q^{nm\times k}$.

Proof. Take any polynomial $N = N(\lambda)$ and $k = k(\lambda)$. We define the following distributions:

- $\mathcal{D}_0$: Sample and output $\mathbf{u} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}, \mathbf{y}, \sigma)$.
- $\mathcal{D}_1$: Sample and output $\mathbf{u} \leftarrow (\mathbf{V})_\sigma^{-1}(\mathbf{y})$.
- $\mathcal{D}_2$: Sample $\mathbf{v} \leftarrow D_{\mathbb{Z},\sigma}^k$ and for each $i \in [N]$, sample $\mathbf{x}_i \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}_i + \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{v})$. Output $[\mathbf{x}_1^\mathsf{T} \mid \cdots \mid \mathbf{x}_N^\mathsf{T} \mid \mathbf{v}^\mathsf{T}]^\mathsf{T}$.
- $\mathcal{D}_3$: Sample $\mathbf{v} \leftarrow D_{\mathbb{Z},\sigma}^k$ and let $\mathbf{W} = \mathbf{Z}(\mathbf{v} \otimes \mathbf{I}_m) \in \mathbb{Z}_q^{n\times m}$. Then for each $i \in [N]$, sample $\mathbf{x}_i \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}_i + \mathbf{W}\mathbf{r}_i)$. Output $[\mathbf{x}_1^\mathsf{T} \mid \cdots \mid \mathbf{x}_N^\mathsf{T} \mid \mathbf{v}^\mathsf{T}]^\mathsf{T}$.
- $\mathcal{D}_4$: Sample $\mathbf{v} \leftarrow D_{\mathbb{Z},\sigma}^k$ and define $\mathbf{W} \in \mathbb{Z}_q^{n\times m}$ to be the matrix where $\mathsf{vec}(\mathbf{W}) = \tilde{\mathbf{Z}}\mathbf{v}$. Then for each $i \in [N]$, sample $\mathbf{x}_i \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}_i + \mathbf{W}\mathbf{r}_i)$. Output $[\mathbf{x}_1^\mathsf{T} \mid \cdots \mid \mathbf{x}_N^\mathsf{T} \mid \mathbf{v}^\mathsf{T}]^\mathsf{T}$.

- $\mathcal{D}_5$: Sample $\mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}$ and $\mathbf{v} \leftarrow \tilde{\mathbf{Z}}_\sigma^{-1}(\text{vec}(\mathbf{W}))$. For each $i \in [N]$, sample $\mathbf{x}_i \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}_i + \mathbf{W}\mathbf{r}_i)$. Output $[\mathbf{x}_1^\intercal \mid \cdots \mid \mathbf{x}_N^\intercal \mid \mathbf{v}^\intercal]^\intercal$.

We argue that each consecutive pair of distributions is statistically indistinguishable.

- Distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are statistically indistinguishable by Lemma 3.8.
- Distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable by Lemma 3.7.
- Distributions $\mathcal{D}_2$ and $\mathcal{D}_3$ are identical since $\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{v} = \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)(\mathbf{v} \otimes 1) = \mathbf{Z}(\mathbf{v} \otimes \mathbf{I}_m)\mathbf{r}_i = \mathbf{W}\mathbf{r}_i$.
- Distributions $\mathcal{D}_3$ and $\mathcal{D}_4$ are identical since $\text{vec}(\mathbf{W}) = \text{vec}(\mathbf{Z}(\mathbf{v} \otimes \mathbf{I}_m)) = \tilde{\mathbf{Z}}\mathbf{v}$.
- Distributions $\mathcal{D}_4$ and $\mathcal{D}_5$ are statistically indistinguishable by Lemma 3.6. Specifically, by Lemma 3.6, when $k \ge 2nm\log q$, then for all but a q^{-nm}-fraction of matrices $\tilde{\mathbf{Z}}$ and all $\sigma \ge \log k$, the statistical distance between the following two distributions is $\text{negl}(nm)$:

  $$\left\{(\mathbf{v}, \tilde{\mathbf{Z}}\mathbf{v}) : \mathbf{v} \leftarrow D_{\mathbb{Z},\sigma}^k\right\} \quad \text{and} \quad \left\{(\mathbf{v}, \tilde{\mathbf{w}}) : \tilde{\mathbf{w}} \xleftarrow{\text{R}} \mathbb{Z}_q^{nm}, \mathbf{v} \leftarrow \tilde{\mathbf{Z}}_\sigma^{-1}(\mathbf{v})\right\}.$$

  The left distribution correspond to $\mathcal{D}_4$ while the right one corresponds to $\mathcal{D}_5$, where $\mathbf{W} \in \mathbb{Z}_q^{n\times m}$ is the matrix satisfying $\text{vec}(\mathbf{W}) = \tilde{\mathbf{w}}$.

The claim now follows by a hybrid argument. □

Corollary 4.10 (Marginals of Structured Gaussian Preimages). *Let λ be a security parameter and let $N, \ell, n, m, q, \sigma_0, \sigma$ be parameters where $n \ge \lambda$, $m \ge 3n\log q$, q is prime, $\ell > Nn\lceil\log q\rceil$, $\sigma_0 \ge (m\ell + m)\log(n\ell)$, and $\sigma \ge 3\ell^3 m^{9/2} \cdot \sigma_0$. Suppose we sample $(\mathbf{A}, \mathbf{V}_S, \mathbf{Z}, \mathbf{T}_S)$ using the following process:*

- $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^\ell, q, m, \sigma_0)$.
- $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_\mathbf{V}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N)$ *where* $\mathbf{V} = [\mathbf{I}_N \otimes \mathbf{A} \mid \mathbf{M}_{\mathbf{Z},\mathbf{R}}]$ *and* $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$.
- $(\mathbf{M}_S, \mathbf{T}_S) \leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{M}_{\mathbf{Z},\mathbf{R}}, \mathbf{T}_\mathbf{V}, S)$ *and set* $\mathbf{V}_S = [\mathbf{I}_{|S|} \otimes \mathbf{A} \mid \mathbf{M}_S]$.

There exist a negligible function $\text{negl}(\cdot)$ such that for all non-empty sets $S \subseteq [N]$, all target vectors $\mathbf{y} \in \mathbb{Z}_q^{|S|n}$, and all $\lambda \in \mathbb{N}$, with overwhelming probability over the choice of $(\mathbf{A}, \mathbf{V}_S, \mathbf{Z}, \mathbf{T}_S)$, the statistical distance between the following distributions is $\text{negl}(\lambda)$:

$$\{\mathbf{u} : \mathbf{u} \leftarrow \mathsf{SamplePre}(\mathbf{V}_S, \mathbf{T}_S, \mathbf{y}, \sigma)\} \quad \textit{and} \quad \left\{ \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{|S|} \\ \mathbf{v} \end{bmatrix} : \begin{array}{c} \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n\times m}, \mathbf{v} \leftarrow \tilde{\mathbf{Z}}_\sigma^{-1}(\text{vec}(\mathbf{W})) \\ \mathbf{x}_i \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y}_i + \mathbf{W}\mathbf{r}_i) \end{array} \right\},$$

where the vector $\mathbf{y}$ is the vertical concatenation of $\mathbf{y}_1, \ldots, \mathbf{y}_{|S|} \in \mathbb{Z}_q^n$, $\mathbf{Z} = [\mathbf{Z}_1 \mid \ldots \mid \mathbf{Z}_k]$ where $\mathbf{Z}_i \in \mathbb{Z}_q^{n\times m}$, and $\tilde{\mathbf{Z}} = [\text{vec}(\mathbf{Z}_1) \mid \cdots \mid \text{vec}(\mathbf{Z}_k)] \in \mathbb{Z}_q^{nm\times k}$. Note that the statement also holds for $(\mathbf{A}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_\mathbf{V})$ without the DimRed *step, since* DimRed *is the identity function when $S = [N]$.*

Proof. The corollary follows directly from Lemma 4.5, Lemma 4.7, Lemma 4.6, and Lemma 4.9:

- By Lemma 4.5, given $n \ge \lambda$, $m \ge 3n\log q$, and $\sigma_0 \ge (m\ell + m)\log(n\ell)$, we know that the marginal distribution of $\mathbf{A}$ is statistically close to uniform, $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}_0] \cdot \mathbf{T}_0 = \mathbf{G}_{n\ell}$, and $\|\mathbf{T}\| \le \sqrt{m}\sigma_0$.
- Next, by Lemma 4.7, since $\ell > Nn\lceil\log q\rceil$ and $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}_0] \cdot \mathbf{T}_0 = \mathbf{G}_{n\ell}$, the marginal distribution of $\mathbf{Z}$ is statistically close to uniform. In addition, the matrix $\mathbf{V}$ satisfies

  $$\mathbf{V} = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right],$$

  where $k = 3nm\log q \le 3m^2$, $\mathbf{V} \cdot \mathbf{T}_\mathbf{V} = \mathbf{G}_{nN}$, and $\|\mathbf{T}_\mathbf{V}\| \le \|\mathbf{T}\| \cdot \ell m^2 \le \sqrt{m}\sigma_0 \cdot \ell m^2 \le \ell m^{5/2}\sigma_0$.

- Let $S = \{i_1, \ldots, i_{|S|}\}$. By Lemma 4.6, the matrix $\mathbf{V}_S = [\mathbf{I}_{|S|} \otimes \mathbf{A} \mid \mathbf{M}_S]$ satisfies

$$\mathbf{V}_S = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_{i_1}) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_{i_{|S|}}) \end{array}\right].$$

In addition, the trapdoor $\mathbf{T}_S$ satisfies $\mathbf{V}_S \cdot \mathbf{T}_S = \mathbf{G}_{n|S|}$ and $\|\mathbf{T}_S\| \le \|\mathbf{T}_\mathbf{V}\| \le \ell m^{5/2} \sigma_0$.

- Since $(\mathbf{A}, \mathbf{Z})$ are statistically close to uniform and

$$(m|S| + k) \, \|\mathbf{T}_S\| \log(n|S|) \le (m\ell + 3m^2) \cdot \ell m^{5/2} \sigma_0 \cdot \log(n\ell) \le 3\ell^3 m^{9/2} \cdot \sigma_0 < \sigma,$$

the corollary immediately follows from Lemma 4.9. □

5 Registered Attribute-Based Encryption for General Policies

In this section, we show how to construct a registered key-policy ABE scheme for general circuits from the ℓ-succinct LWE assumption in the random oracle model. We start by constructing a "slotted" registered ABE scheme [HLWW23], which is a simpler primitive that can be generically transformed into a registered ABE scheme. Then, in Section 5.2, we give our construction of the slotted registered ABE scheme. We refer to Section 2 for an overview of our construction.

5.1 Slotted Registered Attribute-Based Encryption

We first recall the notion of a slotted registered attribute-based encryption (ABE) scheme [HLWW23]. In slotted registered ABE, there is an *a priori* bound on the number of users N and each user is associated with a slot $i \in [N]$. Users generate their public keys with respect to a particular slot index $i \in [N]$. Then, there is an aggregation algorithm that takes as input a collection of N public keys and aggregates them into a *short* master public key for the scheme. In particular, there is no notion of users joining the system dynamically in a slotted registered ABE scheme. The work of [HLWW23] describes a generic compiler that transforms any slotted registered ABE scheme into one that supports dynamic registrations with polylogarithmic overhead. The transformation relies on a powers-of-two approach similar to those from earlier works on registration-based encryption [GHMR18, GHM+19]. Throughout this paper, we focus exclusively on the simpler notion of slotted registered ABE. We give the definition for the slotted primitive here and defer the full definition of registered ABE to Appendix A. Our definitions closely follow that from [HLWW23], and we highlight the only differences in Remark 5.2. Note that for generality, we decouple the policy-family parameter τ from the security parameter λ in our definition (i.e., allow these to be chosen independently).

Definition 5.1 (Slotted Registered ABE). 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 set of attributes and $\mathcal{P} = \{\mathcal{P}_\tau\}_{\tau \in \mathbb{N}}$ be a set of policies (where each $P \in \mathcal{P}_\tau$ is a mapping $P \colon \mathcal{X}_\tau \to \{0,1\}$). A slotted registered *key-policy* ABE scheme with attribute space $\mathcal{X}$ and policy space $\mathcal{P}$ is a tuple of efficient algorithm $\Pi_{\mathsf{sRABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^N, 1^\tau) \to \mathsf{crs}$: On input the security parameter λ, the number of slots N, and the policy-family parameter τ, the setup algorithm outputs a common reference string crs. We assume crs contains an implicit description of 1^λ and 1^τ.

- $\mathsf{KeyGen}(\mathsf{crs}, i, P) \to (\mathsf{pk}_i, \mathsf{sk}_i)$: On input the common reference string crs, a slot index $i \in [N]$, and a policy $P \in \mathcal{P}_\tau$, the key-generation algorithm outputs a public key pk_i and a secret key sk_i for slot i.

- $\mathsf{IsValid}(\mathsf{crs}, i, P, \mathsf{pk}_i) \to b$: On input the common reference string crs, a slot index $i \in [N]$, a policy $P \in \mathcal{P}_\tau$, and a public key pk_i, the key-validation algorithm outputs a bit $b \in \{0,1\}$. This algorithm is *deterministic*.

- $\mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, P_1), \ldots, (\mathsf{pk}_N, P_N)) \to (\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N)$: On input the common reference string crs and a list of public keys and the associated policies $(\mathsf{pk}_1, P_1), \ldots, (\mathsf{pk}_N, P_N)$, the aggregate algorithm outputs the master public key mpk and a collection of helper decryption keys $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_N$. This algorithm is *deterministic*. We assume that mpk includes an implicit description of 1^λ and the policy-family parameter 1^τ.

- $\mathsf{Encrypt}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$: 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.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{hsk}, x, \mathsf{ct}) \to \mu$: On input a decryption key sk, the helper decryption key hsk, the attribute $x \in \mathcal{X}_\tau$, and a ciphertext ct, the decryption algorithm outputs a message $\mu \in \{0, 1\}$. This algorithm is *deterministic*.

Moreover, the above algorithms should satisfy the following properties:

- **Completeness:** For all $\lambda, N, \tau \in \mathbb{N}$, and all indices $i \in [N]$, all policies $P \in \mathcal{P}_\tau$,

$$\Pr\left[\mathsf{IsValid}(\mathsf{crs}, i, P, \mathsf{pk}_i) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^\tau) \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i, P) \end{array}\right] = 1.$$

- **Correctness:** We say Π_{sRABE} is correct if for all $\lambda, N, \tau \in \mathbb{N}$, all indices $i \in [N]$, all policies $P \in \mathcal{P}_\tau$, all attributes $x \in \mathcal{X}_\tau$ where $P(x) = 1$, all crs in the support of $\mathsf{Setup}(1^\lambda, 1^N, 1^\tau)$, all $(\mathsf{pk}_i, \mathsf{sk}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{crs}, i, P)$, all collections of tuples $\{(j, P_j, \mathsf{pk}_j)\}_{j \neq i}$ where $\mathsf{IsValid}(\mathsf{crs}, j, P, \mathsf{pk}_j) = 1$, and all messages $\mu \in \{0, 1\}$, we have

$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{hsk}_i, x, \mathsf{ct}) = \mu : \ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, x, \mu)\ \right] = 1,$$

  where $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) = \mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, P_1), \ldots, (\mathsf{pk}_N, P_N))$.

- **Compactness:** There exists a universal polynomial poly such that for all $\lambda, N, \tau \in \mathbb{N}$, all crs in the support of $\mathsf{Setup}(1^\lambda, 1^N, 1^\tau)$, all triples (i, pk_i, P_i) where $\mathsf{IsValid}(\mathsf{crs}, i, P, \mathsf{pk}_i) = 1$, and all $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N)$ in the support of $\mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, P_1), \ldots, (\mathsf{pk}_N, P_N))$, it holds that $|\mathsf{mpk}| \leq \mathrm{poly}(\lambda, \tau, \log N)$ and for all $i \in [N]$, $|\mathsf{hsk}_i| \leq \mathrm{poly}(\lambda, \tau, \log N)$.

- **Security:** Let $b \in \{0, 1\}$ be a bit. For an adversary $\mathcal{A}$, define the following security game between $\mathcal{A}$ and a challenger:
  - **Setup phase:** On input the security parameter 1^λ, algorithm $\mathcal{A}$ sends a slot count 1^N and the policy-family parameter 1^τ to the challenger. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^\tau)$ and gives crs to $\mathcal{A}$. The challenger also initializes a counter $\mathsf{ctr} = 0$, an (initially-empty) dictionary D, and a set of slot indices $C = \varnothing$.
  - **Pre-challenge 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 policy $P \in \mathcal{P}_\tau$. The challenger responds by incrementing the counter $\mathsf{ctr} = \mathsf{ctr} + 1$, sampling $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i, P)$ and replies with $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}})$ to $\mathcal{A}$. The challenger adds the mapping $\mathsf{ctr} \mapsto (i, P, \mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}})$ to the dictionary D.
    - **Corruption query:** In a corruption query, the adversary specifies an index $1 \leq c \leq \mathsf{ctr}$. In response, the challenger looks up the tuple $(i', P', \mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and replies to $\mathcal{A}$ with sk'.
  - **Challenge phase:** For each slot $i \in [N]$, adversary $\mathcal{A}$ must specify a tuple $(c_i, P_i^*, \mathsf{pk}_i^*)$ where either $c_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $c_i = \bot$ to reference a key outside this set. The adversary also specifies a challenge attribute $x^* \in \mathcal{X}_\lambda$. The challenger then checks the following:
    - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', P', \mathsf{pk}', \mathsf{sk}')$. If $i \neq i'$ or $P_i^* \neq P'$ or $\mathsf{pk}_i^* \neq \mathsf{pk}'$, then the challenger halts with output 0. If the adversary issued a "corruption" query on index c_i, then the challenger adds the slot index i to C.
    - If $c_i = \bot$, then the challenger checks that $\mathsf{IsValid}(\mathsf{crs}, i, P_i^*, \mathsf{pk}_i^*) = 1$. If not, the experiment outputs 0. Otherwise, the challenger adds i to C.

  The challenger computes $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) = \mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1^*, P_1^*), \ldots, (\mathsf{pk}_N^*, P_N^*))$ 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 an adversary $\mathcal{A}$ is admissible if for all corrupted slot indices $i \in C$, $P_i^*(x^*) = 0$. We say that a slotted registered ABE scheme is secure if for all efficient and admissible 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 experiment.

Remark 5.2 (Policy-Dependent Key Generation)**.** In the original notion of (ciphertext-policy) registered ABE from [HLWW23], the key-generation algorithm is independent of the attribute, whereas in our notion of (key-policy) registered ABE, we allow the key-generation algorithm to take the policy as input. In other words, the policy needs to be specified at key-generation time rather than registration time. We view this as a minor restriction since the scheme still retains the primary requirement in registered ABE of users being able to choose their own keys (without relying on any trusted key-issuer).

Definition 5.3 (Attribute-Selective Security)**.** We say that a slotted registered key-policy ABE scheme Π_{sRABE} satisfies *attribute-selective* security if the adversary in Definition 5.1 is required to choose its challenge attribute x^* at the beginning of the setup phase *before* it sees the common reference string.

Definition 5.4 (Security without Corruptions)**.** We say that a slotted registered key-policy ABE scheme Π_{sRABE} satisfies security *without corruptions* if the adversary in Definition 5.1 is not allowed to make any corruption queries.

Remark 5.5 (Achieving Adaptive Security)**.** The work of [FWW23] shows how to generically transform an attribute-selective slotted registered ABE scheme that does not support corruptions (i.e., Definitions 5.3 and 5.4) into an attribute-selective construction that does support corruptions in the random oracle model. Moreover, by relying on sub-exponential hardness, we can use standard complexity leveraging (c.f., [BB04]) to transform a registered ABE scheme with policy-selective security into one that is adaptively secure (i.e., a scheme that satisfies the security requirement in Definition 5.1). Note that the latter transformation (to obtain security against an adaptively-chosen attribute) would increase the size of the scheme parameters by a polynomial in the attribute length. In the context of our construction (Construction 5.6), the ciphertext size in the final adaptively-secure scheme would no longer be sublinear in the attribute length.

5.2 Slotted Key-Policy Registered ABE for Circuits

In this section, we give our construction of a slotted key-policy registered ABE for general circuit policies from the ℓ-succinct LWE assumption in the random oracle model.

Construction 5.6 (Slotted Registered Key-Policy Attribute-Based Encryption)**.** Let λ be a security parameter, N be the number of users, and τ be a policy parameter. We define the following parameters:

- Let $\ell = \ell(\tau)$ is the attribute length and define the attribute space $\mathcal{X} = \{\mathcal{X}_\tau\}_{\tau \in \mathbb{N}}$ where $\mathcal{X}_\tau = \{0,1\}^{\ell(\tau)}$.
- Let $\mathcal{P}_\tau$ be the family of policies that can be computed by a Boolean circuit $C \colon \{0,1\}^{\ell(\tau)} \to \{0,1\}$ of depth at most $d = d(\tau)$. In the following, we adopt the convention that an attribute $\mathbf{x} \in \{0,1\}^{\ell(\tau)}$ satisfies a policy with circuit C if $C(\mathbf{x}) = 0$.
- Let n, m, q be lattice parameters (which can be functions of λ, N, τ). Let $m' = n \lceil \log q \rceil$, $\ell_0 = \max(\ell, Nm')$, and $k = 3nm \log q$ be fixed dimensions.
- Let $\sigma_{\mathsf{LWE}}, \sigma_{\mathsf{crs}}, \sigma_{\mathsf{key}}, \sigma_{\mathsf{agg}}$ be Gaussian width parameters and $\beta_{\mathsf{key}}, \beta_{\mathsf{agg}}$ be norm bounds (which are functions of λ, N, τ).

Our construction relies on the following additional primitives:

- We define the NP relation $\mathcal{R}_{\mathsf{sk}}$ for the following relation that checks knowledge of a secret key associated with a public key:

Statement: matrices $\mathbf{A}, \mathbf{B}, \mathbf{W} \in \mathbb{Z}_q^{n \times m}$, vectors $\mathbf{p}, \mathbf{t} \in \mathbb{Z}_q^n$, $\mathbf{r} \in \mathbb{Z}_q^m$, and a norm-bound $\beta \in \mathbb{Z}$
Witness: vector $\mathbf{v} \in \mathbb{Z}_q^m$

Output 1 if $\mathbf{A}\mathbf{v} = \mathbf{W}\mathbf{r} + \mathbf{B}\mathbf{G}^{-1}(\mathbf{t}) + \mathbf{p}$ and $\|\mathbf{v}\| \leq \beta$. Otherwise, output 0.

Figure 1: NP relation $\mathcal{R}_{\mathsf{sk}}$ for proving knowledge of a secret key.

- 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 for NP.
- Let $\Pi_{\mathsf{DGS}} = (\mathsf{DGS.SamplePre}, \mathsf{DGS.Explain})$ be a $(\rho', \sigma_{\mathsf{loss}})$-explainable discrete Gaussian preimage sampler. Let $\lambda_{\mathsf{DGS}} = \lambda_{\mathsf{DGS}}(\lambda, N, \tau)$ be the security parameter for the sampler. Additionally, let $\rho = \rho(\lambda_{\mathsf{DGS}}, \lambda, N, \tau)$ upper-bound ρ' for all sampler instances in the construction.[6]
- Let $\left\{H_\lambda \colon \{0,1\}^* \to \{0,1\}^\lambda\right\}_{\lambda \in \mathbb{N}}$ be a family of hash functions with λ-bit outputs, which we model as a random oracle in the security analysis.

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

- $\mathsf{Setup}(1^\lambda, 1^N, 1^\tau)$: On input the security parameter λ, the bound on the number of slots N, and the policy parameter τ, the setup algorithm proceeds as follows:
  1. Sample $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{crs}})$.
  2. Compute trapdoors
  $$(\mathbf{U}, \mathbf{T}_{\mathsf{ct}}) \leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, [\ell]) \qquad \text{using Algorithm 2}$$
  $$(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N) \qquad \text{using Lemma 4.7.}$$
  Parse
  $$\mathbf{V} = \left[\begin{array}{ccc|ccc} \mathbf{A} & & & -\mathbf{Z}_1\mathbf{r}_1 & \cdots & -\mathbf{Z}_k\mathbf{r}_1 \\ & \ddots & & \vdots & \ddots & \vdots \\ & & \mathbf{A} & -\mathbf{Z}_1\mathbf{r}_N & \cdots & -\mathbf{Z}_k\mathbf{r}_N \end{array}\right] = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \in \mathbb{Z}_q^{nN \times (mN+k)}, \tag{5.1}$$
  and $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$.
  3. Sample $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$.
  4. Sample vectors $\mathbf{p}, \mathbf{t}_1, \ldots, \mathbf{t}_N \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and a matrix $\mathbf{U}_{\mathsf{ct}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$.

  Output
  $$\mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \underbrace{\mathbf{U}, \mathbf{U}_{\mathsf{ct}}, \mathbf{T}_{\mathsf{ct}}, \{\mathbf{t}_i\}_{i \in [N]}}_{\text{for homomorphic evaluation}}, \underbrace{\{\mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}},}_{\text{for key generation}}) \tag{5.2}$$
- $\mathsf{KeyGen}(\mathsf{crs}, i, f)$: On input the common reference string crs (with components parsed according to Eq. (5.2)), an index $i \in [N]$, and a function $f \in \mathcal{P}_\tau$, the key-generation algorithm does the following:
  1. Parse $\mathbf{T}_{\mathsf{ct}} = \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix}$ where $\mathbf{T}_{\mathsf{in}} \in \mathbb{Z}_q^{\ell m \times \ell m'}$ and $\mathbf{T}_{\mathsf{fun}} \in \mathbb{Z}_q^{m \times \ell m'}$. Set $\mathbf{B} = \mathbf{U}_{\mathsf{ct}}\mathbf{T}_{\mathsf{fun}} \in \mathbb{Z}_q^{n \times \ell m'}$ and compute $\mathbf{B}_f = \mathsf{EvalF}(\mathbf{B}, f)$.

[6] Here we slightly abuse notation to allow algorithms DGS.SamplePre and DGS.Explain to take/output a random string that is longer than the original specification. This does not affect explainability since any unused bit can always be explained by a uniformly random bit.

2. Sample the preimage

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{d} \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{t}_i)), \sigma_{\mathsf{key}}), \tag{5.3}$$

where $\boldsymbol{\eta}_i \in \{0,1\}^N$ is the the i^{th} standard basis vector, $\mathbf{y}_i \in \mathbb{Z}^m$ for each $i \in [N]$, and $\mathbf{d} \in \mathbb{Z}^k$. If $\|\mathbf{y}_i\| > \beta_{\mathsf{key}}$ for any $i \in [N]$, then set[7]

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{d} \end{bmatrix} = \mathbf{T}_{\mathbf{V}} \cdot \mathbf{G}_{nN}^{-1}(\boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{t}_i))).$$

3. Set $\mathbf{W} = \mathbf{Z}(\mathbf{d} \otimes \mathbf{I}_m) \in \mathbb{Z}_q^{n \times m}$ and construct the NIZK proof

$$\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_f, \mathbf{W}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_i),$$

Output $\mathsf{pk}_i = (\mathbf{W}, \{\mathbf{y}_j\}_{j \neq i}, \pi)$ and $\mathsf{sk}_i = (i, f, \mathbf{y}_i)$.

- $\mathsf{IsValid}(\mathsf{crs}, i, f, \mathsf{pk}_i)$: On input the common reference string crs (with components parsed according to Eq. (5.2)), an index $i \in [N]$, a policy $f \in \mathcal{P}_\tau$, and a public key $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$, the validity-checking algorithm sets $\mathbf{B} = \mathbf{U}_{\mathsf{ct}} \mathbf{T}_{\mathsf{fun}} \in \mathbb{Z}_q^{n \times \ell m}$ as in KeyGen, computes $\mathbf{B}_f = \mathsf{EvalF}(\mathbf{B}, f)$, and outputs 1 if

  $$\forall j \neq i : \mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i \mathbf{r}_j \quad \text{and} \quad \|\mathbf{y}_{i,j}\| \leq \beta_{\mathsf{key}} \quad \text{and} \quad \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_f, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \pi_i) = 1.$$

  Otherwise, the algorithm outputs 0.

- $\mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$: On input the common reference string crs and a list of public keys and polices $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N)$, the aggregate algorithm parses

  $$\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i) \quad \text{and} \quad \mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \mathbf{U}, \mathbf{U}_{\mathsf{ct}}, \mathbf{T}_{\mathsf{ct}}, \{\mathbf{t}_i, \mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}).$$

  Then, it proceeds as follows:

  - Compute $\gamma = H_\rho((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ and sample

    $$\begin{bmatrix} \mathbf{y}_{0,1} \\ \vdots \\ \mathbf{y}_{0,N} \\ \mathbf{d}_0 \end{bmatrix} \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma), \tag{5.4}$$

    where $\mathbf{y}_{0,i} \in \mathbb{Z}^m$ for each $i \in [N]$ and $\mathbf{d}_0 \in \mathbb{Z}^k$. If $\|\mathbf{y}_{0,i}\| > \beta_{\mathsf{agg}}$ for any $i \in [N]$, then it sets $\mathbf{W}_0 = \mathbf{0}^{n \times m}$ and $\mathbf{y}_{0,i} = \mathbf{0}^m$ for all $i \in [N]$. Otherwise, it sets $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$ (and leaves $\mathbf{y}_{0,i}$ unchanged).

  The aggregation algorithm outputs $\mathsf{mpk} = \mathbf{W}_0 + \sum_{i \in [N]} \mathbf{W}_i$ and $\mathsf{hsk}_i = \mathbf{y}_{0,i} + \sum_{j \neq i} \mathbf{y}_{j,i}$ for all $i \in [N]$.

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

  $$\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n \,,\; \mathbf{e} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{LWE}}}^m \,,\; \mathbf{K}_{\mathbf{U}} \xleftarrow{\text{R}} \{0,1\}^{m \times m} \,,\; \mathbf{K}_{\mathbf{W}} \xleftarrow{\text{R}} \{0,1\}^{m \times m} \,,\; \mathbf{k}_{\mathbf{p}} \xleftarrow{\text{R}} \{0,1\}^m.$$

  If $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$, it sets $\mathbf{e} = \mathbf{0}^m$ instead. Finally, it outputs

  $$\mathsf{ct} = \left(\mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}} \,,\; \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}} \,,\; \mathbf{s}^{\mathsf{T}}(\mathbf{U}_{\mathsf{ct}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}) + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{U}} \,,\; \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + \mu \cdot \lfloor q/2 \rfloor \right).$$

[7] This resampling guarantees a bound on the norm and is helpful for ensuring *perfect* completeness and correctness.

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{hsk}, \mathbf{x}, \mathsf{ct})$: On input a decryption key $\mathsf{sk} = (i, f, \mathbf{y}_{\mathsf{sk}})$, a helper key $\mathsf{hsk} = \mathbf{y}_{\mathsf{hsk}}$, an attribute $\mathbf{x} \in \{0,1\}^\ell$, and a ciphertext $\mathsf{ct} = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4)$, the decryption algorithm parses $\mathbf{T}_{\mathsf{ct}} = \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix}$ as in KeyGen, sets $\mathbf{B} = \mathbf{U}_{\mathsf{ct}} \mathbf{T}_{\mathsf{fun}} \in \mathbb{Z}_q^{n \times \ell m'}$, and computes $\mathbf{H}_{\mathbf{B},f,\mathbf{x}} = \mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x})$. Then, it computes
$$z = c_4 + [\mathbf{c}_1^{\mathsf{T}} \mid \mathbf{c}_3^{\mathsf{T}}] \cdot \begin{bmatrix} -(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{c}_2^{\mathsf{T}} \mathbf{r}_i - \mathbf{c}_1^{\mathsf{T}}(\mathbf{y}_{\mathsf{sk}} + \mathbf{y}_{\mathsf{hsk}}) \in \mathbb{Z}_q.$$
Finally, it outputs $\lfloor z \rceil$ where $\lfloor z \rceil = 0$ if $-q/4 \le z < q/4$ and $\lfloor z \rceil = 1$ otherwise.

Theorem 5.7 (Completeness). *Suppose q is prime, $n \ge \lambda$, $m \ge 3n \log q$, $\sigma_{\mathsf{crs}} \ge O(\ell_0{}^2 m^2)$, $\beta_{\mathsf{key}} > \sigma_{\mathsf{crs}} \cdot O(\ell_0{}^2 m^3)$, and Π_{NIZK} is complete. Then Construction 5.6 is complete.*

Proof. Let $\lambda, N, \tau \in \mathbb{N}$. Take any index $i \in [N]$ and any policy $f \in \mathcal{P}_\tau$. Let
$$\mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \mathbf{U}, \mathbf{U}_{\mathsf{ct}}, \mathbf{T}_{\mathsf{ct}}, \{\mathbf{t}_i, \mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^\tau),$$
and sample $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i, f)$. Then, we can write
$$\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \ne i}, \pi_i) \quad \text{and} \quad \mathsf{sk}_i = (i, f, \mathbf{y}_{i,i}).$$
We show that $\mathsf{IsValid}(\mathsf{crs}, i, f, \mathsf{pk}_i) = 1$ with probability 1:

- Since $\sigma_{\mathsf{crs}} \ge O(\ell_0{}^2 m^2) \ge (m\ell_0 + m) \cdot \log(n\ell_0)$, Lemma 4.5 implies that $\|\mathbf{T}_0\| \le \sqrt{m}\sigma_{\mathsf{crs}}$. By Lemma 4.7, this means $\|\mathbf{T}_{\mathbf{V}}\| \le \sqrt{m}\sigma_{\mathsf{crs}} \cdot \ell_0 m^2 \le \sigma_{\mathsf{crs}} \cdot O(\ell_0 m^3)$.
- Next, $\|\mathbf{T}_{\mathbf{V}} \cdot \mathbf{G}_{nN}^{-1}(\boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{t}_i)))\| \le m' N \|\mathbf{T}_{\mathbf{V}}\| \le \sigma_{\mathsf{crs}} \cdot O(\ell_0{}^2 m^3) < \beta_{\mathsf{key}}$ by definition of ℓ_0. Thus, $\|\mathbf{y}_{i,j}\| \le \beta_{\mathsf{key}}$ for all $j \in [N]$.
- By construction of $\mathbf{V}$ and the fact that $\mathbf{V} \cdot \mathbf{T}_{\mathbf{V}} = \mathbf{G}_{nN}$, Lemma 3.8 ensures that
$$\mathbf{A}\mathbf{y}_{i,j} - \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)\mathbf{d}_i = \begin{cases} \mathbf{0}^n & j \ne i \\ \mathbf{p} + \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{t}_i) & j = i. \end{cases}$$
By construction, KeyGen sets $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$. By Eq. (3.1), we also have
$$\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)\mathbf{d}_i = \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)(\mathbf{d}_i \otimes 1) = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)\mathbf{r}_j = \mathbf{W}_i \mathbf{r}_j.$$
Correspondingly, this means that
$$\mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i \mathbf{r}_j \quad \text{and} \quad \mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i \mathbf{r}_i + \mathbf{p} + \mathbf{B}_f \mathbf{G}^{-1}(\mathbf{t}_i).$$
In particular, this means that $C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_f, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i,i}) = 1$.
- Since KeyGen samples $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_f, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i,i})$, completeness of Π_{NIZK} now implies that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_f, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \pi_i) = 1$. Correspondingly, $\mathsf{IsValid}(\mathsf{crs}, i, f, \mathsf{pk}_i)$ outputs 1. □

Theorem 5.8 (Correctness). *Suppose q is prime, $n \ge \lambda$, $m \ge 2n \log q$, $\sigma_{\mathsf{crs}} \ge O(\ell_0^2 m^2)$, $\beta_{\mathsf{key}} > \sigma_{\mathsf{crs}} \cdot O(\ell_0{}^2 m^3)$, $q \ge m^{O(d)} \cdot O(\ell_0{}^2) \cdot \sigma_{\mathsf{LWE}} \sigma_{\mathsf{crs}} + 4m^{3/2} \cdot \sigma_{\mathsf{LWE}}(N\beta_{\mathsf{key}} + \beta_{\mathsf{agg}})$, and Π_{DGS} is correct. Then, Construction 5.6 is correct.*

Proof. Take any $\lambda, N, \tau \in \mathbb{N}$, index $i \in [N]$, policy $f \in \mathcal{P}_\tau$, and attribute $\mathbf{x} \in \mathcal{X}_\tau$ where $f(\mathbf{x}) = 0$. (Recall our convention is that $\mathbf{x}$ satisfies the policy f if $f(\mathbf{x}) = 0$). Let
$$\mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \mathbf{U}, \mathbf{U}_{\mathsf{ct}}, \mathbf{T}_{\mathsf{ct}}, \{\mathbf{t}_i, \mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^\tau)$$

and $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i, f)$. Parse $\mathbf{T}_{\mathsf{ct}} = \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix}$, $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$, and $\mathsf{sk}_i = (i, f, \mathbf{y}_{i,i})$. Set $\mathbf{B} = \mathbf{U}_{\mathsf{ct}}\mathbf{T}_{\mathsf{fun}} \in \mathbb{Z}_q^{n \times \ell m'}$ and compute $\mathbf{B}_f = \mathsf{EvalF}(\mathbf{B}, f)$, $\mathbf{H}_{\mathbf{B},f,\mathbf{x}} = \mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x})$. From the analysis in Theorem 5.7, we always have

$$\|\mathbf{y}_{i,i}\| \leq \beta_{\mathsf{key}} \quad \text{and} \quad \mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i\mathbf{r}_i + \mathbf{p} + \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{t}_i). \tag{5.5}$$

Take any set of tuples $\{(j, f_j, \mathsf{pk}_j)\}_{j \neq i}$ where $\mathsf{IsValid}(\mathsf{crs}, j, f, \mathsf{pk}_j) = 1$ for all $j \neq i$. This implies that for each $j \in [N] \setminus \{i\}$, we have

$$\|\mathbf{y}_{j,i}\| \leq \beta_{\mathsf{key}} \quad \text{and} \quad \mathbf{A}\mathbf{y}_{j,i} = \mathbf{W}_j\mathbf{r}_i. \tag{5.6}$$

Now let $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) = \mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$. By the structure of $\mathbf{V}$ and correctness of Π_{DGS}, the vectors $\mathbf{y}_{0,i}$ and $\mathbf{d}_0$ from Eq. (5.4) satisfy $\mathbf{A}\mathbf{y}_{0,i} - \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{d}_0 = \mathbf{0}^n$. This means

$$\mathbf{A}\mathbf{y}_{0,i} = \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{d}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)\mathbf{r}_i = \mathbf{W}_0\mathbf{r}_i$$

We conclude that Aggregate always computes $(\mathbf{W}_0, \mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N})$ such that

$$\|\mathbf{y}_{0,i}\| \leq \beta_{\mathsf{agg}} \quad \text{and} \quad \mathbf{A}\mathbf{y}_{0,i} = \mathbf{W}_0\mathbf{r}_i \tag{5.7}$$

This means $\mathsf{mpk} = \widehat{\mathbf{W}} = \mathbf{W}_0 + \sum_{i \in [N]} \mathbf{W}_i$. Take any message $\mu \in \{0,1\}$ and let $\mathsf{ct} = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4) \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathbf{x}, \mu)$. Let $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{e} \in \mathbb{Z}_q^m$, $\mathbf{K}_{\mathsf{U}} \in \{0,1\}^{m \times m}$, $\mathbf{K}_{\mathsf{W}} \in \{0,1\}^{m \times m}$, $\mathbf{k}_{\mathsf{p}} \in \{0,1\}^m$ be the components sampled by Encrypt. Consider the output of $\mathsf{Decrypt}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathbf{x}, \mathsf{ct})$. In this case, $\mathbf{y}_{\mathsf{sk}} = \mathbf{y}_{i,i}$ and $\mathbf{y}_{\mathsf{hsk}} = \mathsf{hsk}_i = \mathbf{y}_{0,i} + \sum_{j \neq i} \mathbf{y}_{j,i}$ First,

$$\mathbf{c}_1^{\mathsf{T}}(\mathbf{y}_{\mathsf{sk}} + \mathbf{y}_{\mathsf{hsk}}) = \mathbf{s}^{\mathsf{T}}\mathbf{A}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \neq i} \mathbf{y}_{j,i}\right) + \underbrace{\mathbf{e}^{\mathsf{T}}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \neq i} \mathbf{y}_{j,i}\right)}_{\tilde{e}_1}.$$

Combined with Eqs. (5.5) to (5.7), this becomes

$$\begin{aligned} \mathbf{c}_1^{\mathsf{T}}(\mathbf{y}_{\mathsf{sk}} + \mathbf{y}_{\mathsf{hsk}}) &= \mathbf{s}^{\mathsf{T}}(\mathbf{W}_i\mathbf{r}_i + \mathbf{p} + \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{W}_0\mathbf{r}_i) + \sum_{j \neq i} \mathbf{s}^{\mathsf{T}}\mathbf{W}_j\mathbf{r}_i + \tilde{e}_1 \\ &= \mathbf{s}^{\mathsf{T}}\left(\widehat{\mathbf{W}}\mathbf{r}_i + \mathbf{p} + \mathbf{B}_f\mathbf{G}^{-1}(\mathbf{t}_i)\right) + \tilde{e}_1. \end{aligned}$$

Next,

$$c_4 + \mathbf{c}_2^{\mathsf{T}}\mathbf{r}_i = \mu \cdot \lfloor q/2 \rfloor + \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}}\mathbf{r}_i + \underbrace{\mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathsf{p}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathsf{W}}\mathbf{r}_i}_{\tilde{e}_2}.$$

We now break down the term $[\mathbf{c}_1^{\mathsf{T}} \mid \mathbf{c}_3^{\mathsf{T}}] \cdot \begin{bmatrix} -(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{t}_i)$. We start by observing

$$\mathbf{x}^{\mathsf{T}} \otimes \mathbf{G} = (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)(\mathbf{I}_\ell \otimes \mathbf{G}) = (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T}_{\mathsf{ct}} = [\mathbf{A}(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m) \mid (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}] \cdot \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix}.$$

Then, recalling that $\mathbf{B} = \mathbf{U}_{\mathsf{ct}}\mathbf{T}_{\mathsf{fun}} \in \mathbb{Z}_q^{n \times \ell m'}$, we have

$$[\mathbf{A} \mid \mathbf{U}_{\mathsf{ct}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}] \cdot \begin{bmatrix} -(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix} = [-\mathbf{A}(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m) \mid \mathbf{U}_{\mathsf{ct}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}] \cdot \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix} = \mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}. \tag{5.8}$$

By Theorem 3.9 and the fact that $f(\mathbf{x}) = 0$, this yields

$$\begin{aligned} [\mathbf{c}_1^{\mathsf{T}} \mid \mathbf{c}_3^{\mathsf{T}}] \cdot \begin{bmatrix} -(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{t}_i) &= \mathbf{s}^{\mathsf{T}}(\mathbf{B} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{B},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{t}_i) \\ &\quad + \underbrace{(-\mathbf{e}^{\mathsf{T}}(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathsf{U}}\mathbf{T}_{\mathsf{fun}}) \cdot \mathbf{H}_{\mathbf{B},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{t}_i)}_{\tilde{e}_3} \\ &= \mathbf{s}^{\mathsf{T}}\mathbf{B}_f\mathbf{G}^{-1}(\mathbf{t}_i) + \tilde{e}_3. \end{aligned}$$

Putting everything together, we have

$$c_4 + [\mathbf{c}_1^{\mathsf{T}} \mid \mathbf{c}_3^{\mathsf{T}}] \cdot \begin{bmatrix} -(\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B},f,\mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{c}_2^{\mathsf{T}}\mathbf{r}_i - \mathbf{c}_1^{\mathsf{T}}(\mathbf{y}_{\mathsf{sk}} + \mathbf{y}_{\mathsf{hsk}}) = \mu \cdot \lfloor q/2 \rfloor - \tilde{e}_1 + \tilde{e}_2 + \tilde{e}_3.$$

It suffices to show that $|\tilde{e}_1| + |\tilde{e}_2| + |\tilde{e}_3| < q/4$ always holds:

- By construction, $\|\mathbf{e}\| \le \sqrt{m}\sigma_{\mathsf{LWE}}$.
- Since $\|\mathbf{y}_{j,i}\| \le \beta_{\mathsf{key}}$ for $j \in [N]$ and $\|\mathbf{y}_{0,i}\| \le \beta_{\mathsf{agg}}$ by Eqs. (5.5) to (5.7), it follows that

$$|\tilde{e}_1| \le m^{3/2}\sigma_{\mathsf{LWE}}(N\beta_{\mathsf{key}} + \beta_{\mathsf{agg}}).$$

- Next, $\mathbf{k}_{\mathsf{p}} \in \{0,1\}^m$, $\mathbf{K}_{\mathsf{W}} \in \{0,1\}^{m \times m}$, and $\|\mathbf{r}_i\| \le \|\mathbf{T}_0\| \cdot \ell_0 m^2$ by Lemma 4.7. Since $\|\mathbf{T}_0\| \le \sqrt{m}\sigma_{\mathsf{crs}}$ by Lemma 4.5, we have

$$|\tilde{e}_2| \le \left|\mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathsf{p}}\right| + \left|\mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathsf{W}}\mathbf{r}_i\right| \le m^{3/2}\sigma_{\mathsf{LWE}} + m^{3/2}\sigma_{\mathsf{LWE}} \cdot \|\mathbf{r}_i\| \le O(\ell_0 m^4) \cdot \sigma_{\mathsf{LWE}}\sigma_{\mathsf{crs}}.$$

- Finally, $\|\mathbf{G}^{-1}(\mathbf{t}_i)\| = 1$. Next, $\|\mathbf{T}_{\mathsf{ct}}\| \le \|\mathbf{T}_0\|$ by Lemma 4.6 and $\|\mathbf{H}_{\mathbf{B},f,\mathbf{x}}\| \le m^{O(d)}$ by Theorem 3.9. Thus,

$$\|\mathbf{e}^{\mathsf{T}}(\mathbf{K}_{\mathsf{U}}\mathbf{T}_{\mathsf{fun}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}})\| \le 2\ell m^2 \, \|\mathbf{T}_0\| \, \|\mathbf{e}\| \le 2\ell m^3 \sigma_{\mathsf{crs}}\sigma_{\mathsf{LWE}}$$

  and

$$|\tilde{e}_3| \le m^{O(d)} \cdot \ell m^2 \cdot (2\ell m^3 \sigma_{\mathsf{crs}}\sigma_{\mathsf{LWE}}) \le m^{O(d)} \cdot O(\ell^2) \cdot \sigma_{\mathsf{LWE}}\sigma_{\mathsf{crs}}.$$

Correctness holds when $q \ge 4(|\tilde{e}_1| + |\tilde{e}_2| + |\tilde{e}_3|)$, so it suffices to take

$$q = m^{O(d)} \cdot O({\ell_0}^2) \cdot \sigma_{\mathsf{LWE}}\sigma_{\mathsf{crs}} + 4m^{3/2} \cdot \sigma_{\mathsf{LWE}}(N\beta_{\mathsf{key}} + \beta_{\mathsf{agg}}). \qquad \square$$

Theorem 5.9 (Attribute-Selective Security). *Suppose the following constraints hold:*

- *Lattice parameters:* $n \ge \lambda$, $m \ge 3n \log q$, *and* $q > 2$ *is prime.*
- *Width parameters:* $\sigma_{\mathsf{crs}} \ge O({\ell_0}^2 m^2)$, $\sigma_{\mathsf{key}} \ge O({\ell_0}^3 m^5) \cdot \sigma_{\mathsf{crs}}$, $\beta_{\mathsf{key}} \ge \sqrt{m}\sigma_{\mathsf{key}}$, $\beta_{\mathsf{agg}} \ge \sqrt{m}\sigma_{\mathsf{agg}}$, *and*

$$2^{\lambda_{\mathsf{DGS}}} > \sigma_{\mathsf{agg}} \ge \max\{2^{\lambda}(\beta_{\mathsf{key}} + m^{O(d)}\sigma_{\mathsf{crs}}), O(\ell_0 m^{5/2}) \cdot \sigma_{\mathsf{crs}}\sigma_{\mathsf{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q)\}.$$

Suppose also that Π_{NIZK} *is complete, simulation-sound extractable, and zero-knowledge, and that* Π_{DGS} *is correct and explainable. Then, under the* ℓ_0*-succinct LWE assumption (Assumption 3.10) with parameters* $(n, m, q, \sigma_{\mathsf{LWE}}, \sigma_{\mathsf{crs}})$*, Construction 5.6 is attribute-selective secure without corruptions in the random oracle model.*

Proof. Take any polynomials $N = N(\lambda)$, $\tau = \tau(\lambda)$ and any efficient adversary $\mathcal{A}$ for the attribute-selective security game. Suppose $\mathcal{A}$ wins the game with non-negligible advantage ε. In addition, let Q_{ro} be a bound on the number of random oracle queries algorithm $\mathcal{A}$ makes. For ease of exposition, we assume that $\mathcal{A}$ has the following properties:

- It never queries the random oracle on the same input more than once.
- It always makes a random oracle query on the tuple $((\mathsf{pk}_1^*, f_1), \ldots, (\mathsf{pk}_N^*, f_N))$ associated with its challenge query (i.e., the input to the Aggregate algorithm during the challenge phase).

These assumptions are without loss of generality since we can generically transform any adversary that does not satisfy this property into one that satisfies these requirements. Namely, we can consider a "wrapper" adversary around $\mathcal{A}$ that maintains a table of random oracle input/outputs corresponding to the queries $\mathcal{A}$ made and answering any repeated queries using its internal table. Moreover, if $\mathcal{A}$ has not queried the random oracle on input $((\mathsf{pk}_1^*, f_1), \ldots, (\mathsf{pk}_N^*, f_N))$ at the time it submits its challenge query, the wrapper adversary can do so itself before submitting the (same) challenge query to the challenger.

Hybrid sequence. We now define a sequence of hybrid experiments. Our hybrids are parameterized by a bit $b \in \{0,1\}$ and a polynomial $p \in \mathsf{poly}(\lambda)$. We omit the index p when the hybrid definition is independent of the choice of p. We also note that some of the hybrid experiments have inefficient challengers. For clarity of exposition, we will highlight the hybrids where the challenger's behavior can be implemented by an efficient algorithm in purple. When considering reductions to computational assumptions, it will often be important that the challenger in the relevant hybrid distributions be efficiently-implementable.

- $\mathsf{Hyb}_0^{(b)}$: This is the attribute-selective security experiment with challenge bit b:
  - **Setup phase:** On input the security parameter 1^λ, algorithm $\mathcal{A}$ sends a slot count 1^N, the policy-family parameter 1^τ, and an attribute $\mathbf{x} \in \{0,1\}^\ell$ to the challenger. The challenger samples
  $$\mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \mathbf{U}, \mathbf{U}_{\mathsf{ct}}, \mathbf{T}_{\mathsf{ct}}, \{\mathbf{t}_i, \mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^\tau).$$
  In particular, the challenger samples
  $$\begin{aligned} (\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) &\leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{crs}}) \\ (\mathbf{U}, \mathbf{T}_{\mathsf{ct}}) &\leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, [\ell]) \\ (\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) &\leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N) \\ \mathsf{crs}_{\mathsf{NIZK}} &\leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\ \mathbf{p}, \mathbf{t}_1, \ldots, \mathbf{t}_N &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \\ \mathbf{U}_{\mathsf{ct}} &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m} \end{aligned}$$
  It parses $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$. The challenger also initializes a counter $\mathsf{ctr} = 0$, and an (initially-empty) dictionary D.
  - **Key-generation phase:** Adversary $\mathcal{A}$ can make key-generation queries. In a key-generation query, the adversary specifies a slot index $i \in [N]$ and a function $f_i \in \mathcal{P}_\tau$. The challenger responds by incrementing the counter $\mathsf{ctr} = \mathsf{ctr} + 1$ and sampling $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i, f_i)$, where $\mathsf{pk}_{\mathsf{ctr}} = \big(\mathbf{W}_{i,\mathsf{ctr}}, \{\mathbf{y}_{i,j,\mathsf{ctr}}\}_{j \neq i}, \pi_{i,\mathsf{ctr}}\big)$ and $\mathsf{sk}_{\mathsf{ctr}} = \mathbf{y}_{i,i,\mathsf{ctr}}$. In particular, the challenger constructs the components as follows:
    1. First, it parses $\mathbf{T}_{\mathsf{ct}} = \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix}$ and sets $\mathbf{B} = \mathbf{U}_{\mathsf{ct}} \mathbf{T}_{\mathsf{fun}}$. It then computes $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$.
    2. Next, the challenger samples
    $$\begin{bmatrix} \mathbf{y}_{i,1,\mathsf{ctr}} \\ \vdots \\ \mathbf{y}_{i,N,\mathsf{ctr}} \\ \mathbf{d}_{i,\mathsf{ctr}} \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_{f_i} \mathbf{G}^{-1}(\mathbf{t}_i)), \sigma_{\mathsf{key}}),$$
    where $\mathbf{y}_{i,j,\mathsf{ctr}} \in \mathbb{Z}^m$ and $\mathbf{d}_{i,\mathsf{ctr}} \in \mathbb{Z}^k$. If $\|\mathbf{y}_{i,j,\mathsf{ctr}}\| > \beta_{\mathsf{key}}$ for any $j \in [N]$, then the challenger sets
    $$\begin{bmatrix} \mathbf{y}_{i,1,\mathsf{ctr}} \\ \vdots \\ \mathbf{y}_{i,N,\mathsf{ctr}} \\ \mathbf{d}_{i,\mathsf{ctr}} \end{bmatrix} = \mathbf{T}_{\mathbf{V}} \cdot \mathbf{G}_{nN}^{-1}(\boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_{f_i} \mathbf{G}^{-1}(\mathbf{t}_i))). \tag{5.9}$$
    The challenger then sets $\mathbf{W}_{i,\mathsf{ctr}} = \mathbf{Z}(\mathbf{d}_{i,\mathsf{ctr}} \otimes \mathbf{I}_m)$.
    3. Finally, the challenger computes $\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i,i,\mathsf{ctr}})$, where $C_{\mathcal{R}}$ is the circuit computing Fig. 1.

  The challenger sets $\mathsf{pk}_{\mathsf{ctr}} = (\mathbf{W}_{i,\mathsf{ctr}}, \big\{\mathbf{y}_{i,j,\mathsf{ctr}}\big\}_{j \neq i}, \pi_{i,\mathsf{ctr}})$ and replies with $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}})$ to $\mathcal{A}$. It also adds the mapping $\mathsf{ctr} \mapsto (i, f_i, \mathsf{pk}_{\mathsf{ctr}})$ to D.[8]

[8] In the no-corruption setting (see Definition 5.4 and Remark 5.5), the challenger does not need to store the honestly-generated secret keys.

- **Challenge phase:** For each slot $i \in [N]$, adversary $\mathcal{A}$ must specify a tuple $(\mathsf{idx}_i, f_i^*, \mathsf{pk}_i^*)$ where either $\mathsf{idx}_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $\mathsf{idx}_i = \bot$ to reference a key outside this set. The challenger parses $\mathsf{pk}_i^* = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$ and checks the following:
  - If $\mathsf{idx}_i \in [\mathsf{ctr}]$, then the challenger looks up the entry $\mathsf{D}[\mathsf{idx}_i] = (i', f', \mathsf{pk}')$. If $i \neq i'$ or $f_i^* \neq f'$ or $\mathsf{pk}_i^* \neq \mathsf{pk}'$, then the challenger halts with output 0.
  - If $\mathsf{idx}_i = \bot$, the challenger checks that $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) = 1$. In particular, the challenger verifies that
  $$\forall j \neq i : \mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i \mathbf{r}_j \quad \text{and} \quad \|\mathbf{y}_{i,j}\| \leq \beta_{\mathsf{key}}$$
  and that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \pi_i) = 1$, where $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$. If the check fails, the experiment outputs 0.

  The challenger then computes $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_N) = \mathsf{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*))$. Specifically, the challenger first constructs the tuple
  $$\xi^* = \left((\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)\right). \tag{5.10}$$
  Then it computes $\gamma^* = H_\rho(\xi^*)$ and
  $$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,1} \\ \vdots \\ \mathbf{y}_{0,N} \\ \mathbf{d}_0 \end{bmatrix} \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma^*), \tag{5.11}$$
  If $\|\mathbf{y}_{0,i}\| > \beta_{\mathsf{agg}}$ for any $i \in [N]$, then it sets $\mathbf{W}_0 = \mathbf{0}^{n \times m}$ and $\mathbf{y}_{0,i} = \mathbf{0}^m$ for all $i \in [N]$. Otherwise, it sets $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. Finally, the challenger computes
  $$\mathsf{mpk} = \widehat{\mathbf{W}} = \mathbf{W}_0 + \sum_{i \in [N]} \mathbf{W}_i \quad \text{and} \quad \forall i \in [N] : \mathsf{hsk}_i = \mathbf{y}_{0,i} + \sum_{j \neq i} \mathbf{y}_{j,i}.$$
  Next, the challenger constructs the challenge ciphertext $\mathsf{ct}^* = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4) \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathbf{x}, b)$ as follows:
  1. The challenger starts by sampling $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{LWE}}}^m$, $\mathbf{K}_{\mathbf{U}} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}$, $\mathbf{K}_{\mathbf{W}} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}$, and $\mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m$. If $\|\mathbf{e}\| > \sqrt{m}\sigma_{\mathsf{LWE}}$, it sets $\mathbf{e} = \mathbf{0}^m$.
  2. The challenger now computes the components as follows:
  $$\begin{aligned} \mathbf{c}_1^{\mathsf{T}} &= \mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}} \\ \mathbf{c}_2^{\mathsf{T}} &= \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}} \\ \mathbf{c}_3^{\mathsf{T}} &= \mathbf{s}^{\mathsf{T}}(\mathbf{U}_{\mathsf{ct}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}) + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{U}} \\ c_4 &= \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor). \end{aligned}$$

  It gives the challenge ciphertext ct^* to $\mathcal{A}$.
- **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. Note that if $\mathcal{A}$ aborts early, then the output of the experiment is 0.

Throughout this experiment, whenever $\mathcal{A}$ queries the random oracle H_ρ on an input, the challenger always replies with a uniform random string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$.

- $\mathsf{Hyb}_1^{(b)}$: Same as $\mathsf{Hyb}_0^{(b)}$, except at the beginning of the experiment, the challenger samples an index $\mathsf{ind} \xleftarrow{\mathrm{R}} [Q_{\mathsf{ro}}]$. Let $\xi_{\mathsf{ind}} \in \{0,1\}^*$ be the ind^{th} query $\mathcal{A}$ makes to the random oracle. During the challenge phase, after computing ξ^* according to Eq. (5.10), the challenger checks if $\xi_{\mathsf{ind}} = \xi^*$ and halts with output 0 if not. If $\mathcal{A}$ has not made ind queries to the random oracle prior to the challenge phase, the challenger also halts with output 0. In this experiment, if ξ_{ind} cannot be parsed into $((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$ and $f_i \in \mathcal{P}_\tau$, then the output of the experiment is guaranteed to be 0.

- $\mathsf{Hyb}_2^{(b)}$: Same as $\mathsf{Hyb}_1^{(b)}$, except in the key-generation phase, after generating the key $\mathsf{pk}_{\mathsf{ctr}}$ on slot index i with policy f_i, the challenger halts with output 0 if $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_{\mathsf{ctr}}) = 0$.

- $\mathsf{Hyb}_3^{(b)}$: Same as $\mathsf{Hyb}_2^{(b)}$, except the challenger replaces the NIZK proofs in the key-generation queries with simulated proofs.
  - In the setup phase, the experiment samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$.
  - For each key-generation query with slot index $i \in [N]$ and policy f_i, the challenger computes the simulated proof as $\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}))$.

  In this experiment, the adversary's view no longer depends on the secret keys $\mathsf{sk}_{\mathsf{ctr}} = \mathbf{y}_{i,i,\mathsf{ctr}}$ generated in key-generation queries.

- $\mathsf{Hyb}_4^{(b)}$: Same as $\mathsf{Hyb}_3^{(b)}$, except the challenger changes how it samples the public key when responding to key-generation queries. In particular, on each key-generation query (i, f_i), the challenger first defines $\tilde{\mathbf{Z}} = [\mathsf{vec}(\mathbf{Z}_1) \cdots \mid \mathsf{vec}(\mathbf{Z}_k)] \in \mathbb{Z}_q^{nm \times k}$ where $\mathbf{Z} = [\mathbf{Z}_1 \mid \cdots \mid \mathbf{Z}_k]$. Then, it samples

$$\begin{aligned} \mathbf{W}_{i,\mathsf{ctr}} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}\ ,\ \mathbf{d}_{i,\mathsf{ctr}} &\leftarrow \tilde{\mathbf{Z}}_{\sigma_{\mathsf{key}}}^{-1}(\mathsf{vec}(\mathbf{W}_{i,\mathsf{ctr}})) \\ \forall j \neq i : \mathbf{y}_{i,j,\mathsf{ctr}} &\leftarrow \mathbf{A}_{\sigma_{\mathsf{key}}}^{-1}(\mathbf{W}_{i,\mathsf{ctr}} \mathbf{r}_j) \\ \mathbf{y}_{i,i,\mathsf{ctr}} &\leftarrow \mathbf{A}_{\sigma_{\mathsf{key}}}^{-1}(\mathbf{W}_{i,\mathsf{ctr}} \mathbf{r}_i + \mathbf{p} + \mathbf{B}_{f_i} \mathbf{G}^{-1}(\mathbf{t}_i)). \end{aligned}$$

  Next, the challenger checks that $\|\mathbf{y}_{i,j,\mathsf{ctr}}\| \leq \beta_{\mathsf{key}}$ for all $j \in [N]$. If not, it sets $(\mathbf{y}_{i,1,\mathsf{ctr}}, \ldots, \mathbf{y}_{i,N,\mathsf{ctr}}, \mathbf{d}_{i,\mathsf{ctr}})$ according to Eq. (5.9) (exactly as in $\mathsf{Hyb}_3^{(b)}$). Finally, the challenger constructs the proof as in $\mathsf{Hyb}_3^{(b)}$:

$$\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}})).$$

  The challenger replies with the public key $\mathsf{pk}_i = (\mathbf{W}_{i,\mathsf{ctr}}, \left\{\mathbf{y}_{i,j,\mathsf{ctr}}\right\}_{j \neq i}, \pi_{i,\mathsf{ctr}})$. Note that $\mathbf{y}_{i,i,\mathsf{ctr}}$ is *not* given out.

- $\mathsf{Hyb}_5^{(b)}$: Same as $\mathsf{Hyb}_4^{(b)}$, except when responding to key-generation queries, the challenger no longer checks the condition that $\|\mathbf{y}_{i,j,\mathsf{ctr}}\| \leq \beta_{\mathsf{key}}$ for all $j \in [N]$ when generating the key.

- $\mathsf{Hyb}_6^{(b)}$: Same as $\mathsf{Hyb}_5^{(b)}$, except when responding to key-generation queries, the challenger now samples $\mathbf{y}_{i,i,\mathsf{ctr}} \leftarrow \mathbf{A}_{\sigma_{\mathsf{key}}}^{-1}(\mathbf{W}_{i,\mathsf{ctr}} \mathbf{r}_i + \mathbf{t}_i)$.

- $\mathsf{Hyb}_7^{(b)}$: Same as $\mathsf{Hyb}_6^{(b)}$, except we introduce the following abort events to the game:
  - During the setup phase, after sampling $\mathbf{t}_1, \ldots, \mathbf{t}_n \xleftarrow{\text{R}} \mathbb{Z}_q^n$, the challenger aborts if there exists $i \neq j$ where $\mathbf{t}_i = \mathbf{t}_j$. Namely, the challenger checks that the $\mathbf{t}_i$ are all distinct.
  - When responding to key-generation queries, the challenger halts with output 0 if $\|\mathbf{y}_{i,i,\mathsf{ctr}}\| > \beta_{\mathsf{key}}$.
  - Let ξ_{ind} be the ind^{th} random oracle query that algorithm $\mathcal{A}$ makes. Suppose $\xi_{\mathsf{ind}} = ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$. When $\mathcal{A}$ makes a key-generation query (i, f_i) after it has made ind random oracle queries, after the challenger samples $\mathbf{W}_{i,\mathsf{ctr}}$, the challenger checks if $\mathbf{W}_{i,\mathsf{ctr}} = \mathbf{W}_i$. If so, the challenger halts with output 0.

- $\mathsf{Hyb}_8^{(b)}$: Same as $\mathsf{Hyb}_7^{(b)}$, except when responding to key-generation queries, the challenger reverts back to using the trapdoor. In particular, on each key-generation query (i, f_i), the challenger samples the public key by first computing

$$\begin{bmatrix} \mathbf{y}_{i,1,\mathsf{ctr}} \\ \vdots \\ \mathbf{y}_{i,N,\mathsf{ctr}} \\ \mathbf{d}_{i,\mathsf{ctr}} \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes \mathbf{t}_i, \sigma_{\mathsf{key}}) \tag{5.12}$$

  Then, it sets $\mathbf{W}_{i,\mathsf{ctr}} = \mathbf{Z}(\mathbf{d}_{i,\mathsf{ctr}} \otimes \mathbf{I}_m)$ and computes $\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}))$. The challenger in this experiment still checks the same set of abort conditions as in $\mathsf{Hyb}_7^{(b)}$.

- $\mathsf{Hyb}_9^{(b)}$: Same as $\mathsf{Hyb}_8^{(b)}$, except in the challenge phase, after sampling $(\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{d}_0)$, the challenger skips the check that $\|\mathbf{y}_{0,i}\| \le \beta_{\mathsf{agg}}$ for all $i \in [N]$.

- $\mathsf{Hyb}_{10}^{(b)}$: Same as $\mathsf{Hyb}_9^{(b)}$ except the challenger performs some additional checks when responding to the ind^{th} random oracle query $\xi_{\mathsf{ind}} \in \{0,1\}^*$. Specifically, we introduce the following modifications:
  - **Setup phase:** In the setup phase, the challenger initializes an additional (empty) dictionary D_{sk} that maps public keys to decryption keys $\mathbf{y}$.
  - **Key-generation phase:** On each key-generation query (i, f_i), after the challenger computes $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$ and samples $\mathbf{W}_{i,\mathsf{ctr}}$ and the associated secret key $\mathbf{y}_{i,i,\mathsf{ctr}}$ as in $\mathsf{Hyb}_9^{(b)}$, the challenger adds the mapping $(i, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}) \mapsto (0, \mathbf{y}_{i,i,\mathsf{ctr}})$ to D_{sk} if $(i, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}})$ is not already in D_{sk}. As in $\mathsf{Hyb}_9^{(b)}$, if the experiment does not abort, then $\mathbf{A}\mathbf{y}_{i,i,\mathsf{ctr}} = \mathbf{W}_{i,\mathsf{ctr}}\mathbf{r}_i + \mathbf{t}_i$ and $\|\mathbf{y}_{i,i,\mathsf{ctr}}\| \le \beta_{\mathsf{key}}$.

  Next, when responding to the ind^{th} query ξ_{ind} to the random oracle, the challenger proceeds as follows:
  - Parse $\xi_{\mathsf{ind}} = ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$, where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \ne i}, \pi_i)$ and $f_i \in \mathcal{P}_\tau$. If ξ_{ind} does not have this form, then halt with output 0.
  - Check that for all $i \in [N]$, $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_i) = 1$. If not, then halt with output 0.
  - For each $i \in [N]$, compute $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$. If $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$ is not contained in D_{sk}, then the challenger computes
  $$\mathbf{y}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \pi_i).$$
  The experiment aborts and outputs 0 if
  $$C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_i^*) = 0 \quad \text{or} \quad f_i(\mathbf{x}) = 0.$$
  In other words, the experiment only proceeds if
  $$\|\mathbf{y}_i^*\| \le \beta_{\mathsf{key}} \quad \text{and} \quad \mathbf{A}\mathbf{y}_i^* = \mathbf{W}_i\mathbf{r}_i + \mathbf{B}_{f_i}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p} \quad \text{and} \quad f_i(\mathbf{x}) = 1,$$
  If all conditions are satisfied, then the challenger adds the mapping $(i, \mathbf{B}_{f_i}, \mathbf{W}_i) \mapsto (1, \mathbf{y}_i^*)$ to D_{sk}.

  If all of the checks pass, then the challenger samples $\gamma^* \xleftarrow{\text{R}} \{0,1\}^\rho$ and responds with γ^* (as the value of $H_\rho(\xi_{\mathsf{ind}})$). The rest of the experiment proceeds exactly as in $\mathsf{Hyb}_9^{(b)}$. Notably, if the challenger does not halt early in this experiment, then every tuple $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$ associated with ξ_{ind} is contained in D_{sk}, and moreover
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}_i^*)$, then $\|\mathbf{y}_i^*\| \le \beta_{\mathsf{key}}$ and $\mathbf{A}\mathbf{y}_i^* = \mathbf{W}_i\mathbf{r}_i + \mathbf{t}_i$.
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}_i^*)$, then $\|\mathbf{y}_i^*\| \le \beta_{\mathsf{key}}$, $\mathbf{A}\mathbf{y}_i^* = \mathbf{W}_i\mathbf{r}_i + \mathbf{B}_{f_i}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p}$, and $f_i(\mathbf{x}) = 1$.

  The "indicator" bit associated with each entry denotes whether $\mathbf{W}_i$ was sampled by the challenger (as part of an honest key-generation query) or chosen by the adversary.

- $\mathsf{Hyb}_{11}^{(b)}$: Same as $\mathsf{Hyb}_{10}^{(b)}$, except the challenger changes the distribution of $\mathbf{A}$. Specifically, in the setup phase, instead of running $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{crs}})$, the challenger samples
$$\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}, \; \mathbf{U}_0 \xleftarrow{\text{R}} \mathbb{Z}_q^{\ell_0 n \times m}, \; \mathbf{T}_0 \leftarrow [\mathbf{I}_{\ell_0} \otimes \mathbf{A} \mid \mathbf{U}_0]^{-1}_{\sigma_{\mathsf{crs}}}(\mathbf{G}_{n\ell_0}). \tag{5.13}$$
It then computes $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N)$. The challenger aborts and outputs 0 if
$$\|\mathbf{T}_0\| > \sqrt{m}\sigma_{\mathsf{crs}} \quad \text{or} \quad \|\mathbf{T}_{\mathbf{V}}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\| \quad \text{or} \quad \|\mathbf{R}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|.$$

- $\mathsf{Hyb}^{(b)}_{12,p}$: Same as $\mathsf{Hyb}^{(b)}_{11}$ except the challenger uses $\mathsf{DGS.Explain}$ to derive $\gamma^* = H_\rho(\xi_{\mathsf{ind}})$. Specifically, when responding to the ind^{th} query to the random oracle, the challenger samples $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ and computes

$$\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma).$$

  Then, it computes

$$\gamma^* \leftarrow \mathsf{DGS.Explain}(1^{\lambda_{\mathsf{DGS}}}, 1^{p(\lambda)}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \boldsymbol{\kappa}_0, \sigma_{\mathsf{agg}}).$$

  The challenger replies to $\mathcal{A}$ with γ^*.

- $\mathsf{Hyb}^{(b)}_{13,p}$: Same as $\mathsf{Hyb}^{(b)}_{12,p}$ except when sampling $\boldsymbol{\kappa}_0$ (when responding to the ind^{th} random oracle query), the challenger samples

$$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,1} \\ \vdots \\ \mathbf{y}_{0,N} \\ \mathbf{d}_0 \end{bmatrix} \leftarrow \mathbf{V}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{0}^{nN}).$$

- $\mathsf{Hyb}^{(b)}_{14,p}$: Same as $\mathsf{Hyb}^{(b)}_{13,p}$ except when responding to the ind^{th} random oracle query, the challenger now samples

$$\mathbf{d}_0 \leftarrow D^k_{\mathbb{Z},\sigma_{\mathsf{agg}}} \quad \text{and} \quad \mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m) \quad \text{and} \quad \forall i \in [N] : \mathbf{y}_{0,i} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{W}_0 \mathbf{r}_i).$$

- $\mathsf{Hyb}^{(b)}_{15,p}$: Same as $\mathsf{Hyb}^{(b)}_{14,p}$, except when responding to the ind^{th} random oracle query, the challenger now samples $\mathbf{W}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ and $\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathsf{vec}(\mathbf{W}_0), \sigma_{\mathsf{agg}})$. Here, $\tilde{\mathbf{Z}} = [\mathsf{vec}(\mathbf{Z}_1) \cdots \mid \mathsf{vec}(\mathbf{Z}_k)] \in \mathbb{Z}_q^{nm \times k}$ where $\mathbf{Z} = [\mathbf{Z}_1 \mid \cdots \mid \mathbf{Z}_k]$.

- $\mathsf{Hyb}^{(b)}_{16,p}$: Same as $\mathsf{Hyb}^{(b)}_{15,p}$, except the challenger changes how it samples $\mathbf{U}_{\mathsf{ct}}$ and $\mathbf{W}_0$. In the setup phase, the challenger samples $\mathbf{U}^*_{\mathsf{ct}}, \mathbf{W}^*_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$. Then it sets $\mathbf{U}_{\mathsf{ct}} = \mathbf{U}^*_{\mathsf{ct}} + (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}$. When responding to the ind^{th} random oracle query, the challenger sets $\mathbf{W}_0 = \mathbf{W}^*_0 - \sum_{i \in [N]} \mathbf{W}_i$.

- $\mathsf{Hyb}^{(b)}_{17,p}$: Same as $\mathsf{Hyb}^{(b)}_{16,p}$, except the challenger now samples $\mathbf{K}_{\mathbf{U}} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}$, $\mathbf{K}_{\mathbf{W}} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}$, and $\mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m$ during the setup phase (instead of the challenge phase). Then, during the setup phase, it sets $\mathbf{U}^*_{\mathsf{ct}} = \mathbf{A}\mathbf{K}_{\mathbf{U}}$ and $\mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}}$. When respond to the ind^{th} random oracle query, the challenger sets $\mathbf{W}^*_0 = \mathbf{A}\mathbf{K}_{\mathbf{W}}$.

- $\mathsf{Hyb}^{(b)}_{18,p}$: Same as $\mathsf{Hyb}^{(b)}_{17,p}$, except when constructing the response to the ind^{th} random oracle query ξ_{ind}, the challenger first parses $\xi_{\mathsf{ind}} = ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$ and $f_i \in \mathcal{F}_\tau$. As in the previous experiments, if ξ_{ind} does not have this form, then the challenger halts with output 0. The challenger then computes $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$ for each $i \in [N]$ and then populates D_{sk} using the same procedure as described in $\mathsf{Hyb}^{(b)}_{10}$. Similar to the previous experiments, if the challenger does not abort, then for every tuple $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$, there exists an entry $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$ in D_{sk}. The challenger now constructs the vectors $\mathbf{y}_{0,i}$ for each $i \in [N]$ as follows:
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}^*_i)$, then the challenger samples $\mathbf{y}_{0,i} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}^*_i + \mathbf{t}_i)$.
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}^*_i)$, then the challenger first defines $\mathbf{K}^{(i)}_{\mathbf{B}} = \mathbf{K}_{\mathbf{U}}\mathbf{T}_{\mathsf{fun}}\mathbf{H}_{\mathbf{B},f_i,\mathbf{x}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}}\mathbf{H}_{\mathbf{B},f_i,\mathbf{x}}$. Here $\mathbf{H}_{\mathbf{B},f_i,\mathbf{x}} = \mathsf{EvalFX}(\mathbf{B}, f_i, \mathbf{x})$. Then it samples

$$\mathbf{y}_{0,i} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{agg}}}\left(\mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}^*_i + \mathbf{A}\mathbf{K}^{(i)}_{\mathbf{B}}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{A}\mathbf{k}_{\mathbf{p}} + \mathbf{t}_i\right).$$

  Note that this is a purely syntactic change from $\mathsf{Hyb}^{(b)}_{17,p}$.

- $\mathsf{Hyb}_{19,p}^{(b)}$: Same as $\mathsf{Hyb}_{18,p}^{(b)}$ except the challenger changes how it computes each $\mathbf{y}_{0,i}$ when responding to the ind^{th} random oracle query. For each $i \in [N]$, the challenger first samples $\mathbf{k}_{\mathbf{t}_i} \leftarrow \mathbf{A}_{\sigma_{\text{agg}}}^{-1}(\mathbf{t}_i)$ and sets $\mathbf{y}_{0,i}$ as follows:
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}_i^*)$, the challenger sets $\mathbf{y}_{0,i} = \mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}_i^* + \mathbf{k}_{\mathbf{t}_i}$.
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}_i^*)$, the challenger sets $\mathbf{y}_{0,i} = \mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}_i^* + \mathbf{K}_{\mathbf{B}}^{(i)}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{k}_{\mathbf{p}} + \mathbf{k}_{\mathbf{t}_i}$.

- $\mathsf{Hyb}_{20,p}^{(b)}$ Same as $\mathsf{Hyb}_{19,p}^{(b)}$ except the challenger changes how it samples $\mathbf{t}_i$ and $\mathbf{k}_{\mathbf{t}_i}$. The challenger samples $\mathbf{k}_{\mathbf{t}_1}, \ldots, \mathbf{k}_{\mathbf{t}_N} \leftarrow D_{\mathbb{Z}, \sigma_{\text{agg}}}^m$ and sets $\mathbf{t}_i = \mathbf{A}\mathbf{k}_{\mathbf{t}_i}$ for all $i \in [N]$.

- $\mathsf{Hyb}_{21,p}^{(b)}$ Same as $\mathsf{Hyb}_{20,p}^{(b)}$ except when constructing the challenger ciphertext, the challenger no longer checks if $\|\mathbf{e}\| \leq \sqrt{m}\sigma_{\mathsf{LWE}}$. Note that beyond the sampling of the initial trapdoor $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0)$ according to Eq. (5.13), the challenger in this experiment can be implemented efficiently.

- $\mathsf{Hyb}_{22,p}^{(b)}$: Same as $\mathsf{Hyb}_{21,p}^{(b)}$ except in the challenge phase, the challenger samples $\mathbf{c}_1 \xleftarrow{\text{R}} \mathbb{Z}_q^m$. It then computes $\mathbf{c}_2^{\mathsf{T}} = \mathbf{c}_1^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}$, $\mathbf{c}_3^{\mathsf{T}} = \mathbf{c}_1^{\mathsf{T}}\mathbf{K}_{\mathbf{U}}$, and $c_4 = \mathbf{c}_1^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor$. The challenge ciphertext ct^* is then $\mathsf{ct}^* = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4)$.

- $\mathsf{Hyb}_{23,p}^{(b)}$: Same as $\mathsf{Hyb}_{22,p}^{(b)}$, except the challenger undoes the change to the sampling of $\mathbf{t}_i$ and $\mathbf{k}_{\mathbf{t}_i}$. Specifically, in this experiment, the challenger samples $\mathbf{t}_i \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and $\mathbf{k}_{\mathbf{t}_i} \leftarrow \mathbf{A}_{\sigma_{\text{agg}}}^{-1}(\mathbf{t}_i)$ for each $i \in [N]$.

- $\mathsf{Hyb}_{24,p}^{(b)}$: Same as $\mathsf{Hyb}_{23,p}^{(b)}$, except the challenger constructs $\mathbf{y}_{0,i}$ using the procedure from $\mathsf{Hyb}_{18,p}^{(b)}$. Specifically, when responding to the ind^{th} random oracle query, the challenger now constructs $\mathbf{y}_{0,i}$ for $i \in [N]$ as follows:
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}_i^*)$, then the challenger samples $\mathbf{y}_{0,i} \leftarrow \mathbf{A}_{\sigma_{\text{agg}}}^{-1}(\mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}_i^* + \mathbf{t}_i)$.
  - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}_i^*)$, then the challenger first defines $\mathbf{K}_{\mathbf{B}}^{(i)} = \mathbf{K}_{\mathbf{U}}\mathbf{T}_{\mathsf{fun}}\mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}}\mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}}$. Here $\mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} = \mathsf{EvalFX}(\mathbf{B}, f_i, \mathbf{x})$. Then it samples

  $$\mathbf{y}_{0,i} \leftarrow \mathbf{A}_{\sigma_{\text{agg}}}^{-1}\left(\mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}_i^* + \mathbf{A}\mathbf{K}_{\mathbf{B}}^{(i)}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p} + \mathbf{t}_i\right).$$

  Recall that $\mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}}$ in this experiment. Importantly, the quantities in this experiment that depend on $\mathbf{k}_{\mathbf{p}}$ can be constructed given only $\mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}}$ and $\mathbf{c}_1^{\mathsf{T}}\mathbf{k}_{\mathbf{p}}$.

- $\mathsf{Hyb}_{25,p}^{(b)}$: Same as $\mathsf{Hyb}_{24,p}^{(b)}$ except in the challenge phase, the challenger samples $\mathbf{p} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and $c_4 \xleftarrow{\text{R}} \mathbb{Z}_q$. Note that in this experiment, the challenger's behavior is *independent* of the bit $b \in \{0, 1\}$.

For any efficient and admissible adversary $\mathcal{A}$, we write $\mathsf{Hyb}^{(b)}(\mathcal{A})$ to denote the random variable corresponding to the output of an execution of hybrid $\mathsf{Hyb}^{(b)}$ with adversary $\mathcal{A}$ (and an implicit security parameter λ). We now bound the difference between the output distributions of each adjacent pair of hybrid experiments.

Lemma 5.10. *For all $b \in \{0, 1\}$, it holds that* $\Pr[\mathsf{Hyb}_0^{(b)}(\mathcal{A}) = 1] = Q_{\text{ro}} \cdot \Pr[\mathsf{Hyb}_1^{(b)}(\mathcal{A}) = 1]$.

Proof. The adversary's view in $\mathsf{Hyb}_0^{(b)}$ and $\mathsf{Hyb}_1^{(b)}$ is identically distributed. By assumption, algorithm $\mathcal{A}$ always queries the random oracle on ξ^* at some point in the security experiment. Let $\mathsf{ind}^* \in [Q_{\text{ro}}]$ be the index of this query (recall that Q_{ro} is an upper bound on the number of random oracle queries algorithm $\mathcal{A}$ makes). By definition,

$$\Pr[\mathsf{Hyb}_1^{(b)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_0^{(b)}(\mathcal{A}) = 1 \wedge \mathsf{ind} = \mathsf{ind}^*].$$

Since the challenger samples $\mathsf{ind} \xleftarrow{\text{R}} [Q_{\text{ro}}]$ and moreover, ind is independent of all other quantities,

$$\Pr[\mathsf{Hyb}_1^{(b)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_0^{(b)}(\mathcal{A}) = 1 \wedge \mathsf{ind} = \mathsf{ind}^*] = \frac{1}{Q_{\text{ro}}} \Pr[\mathsf{Hyb}_0^{(b)}(\mathcal{A}) = 1]. \qquad \square$$

Lemma 5.11. *Suppose the conditions of Theorem 5.7 hold. Then, for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_1^{(b)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1].$$

Proof. Immediate since Construction 5.6 satisfies (perfect) completeness (Theorem 5.7). □

Lemma 5.12. *Suppose the conditions of Theorem 5.7 hold and Π_{NIZK} satisfies computational zero-knowledge. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The indistinguishability of the two hybrids follows from the zero-knowledge property of NIZK (Definition 3.1). Suppose $|\Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1]| = \varepsilon(\lambda)$. We construct an efficient algorithm $\mathcal{B}$ that breaks zero-knowledge as follows:

1. On input the security parameter 1^λ and the common reference string $\mathsf{crs}_{\mathsf{NIZK}}$, algorithm $\mathcal{B}$ starts running $\mathcal{A}(1^\lambda)$. Whenever $\mathcal{A}$ makes a random oracle query, algorithm $\mathcal{B}$ responds with a random string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$. Recall that we assume $\mathcal{A}$ does not query the random oracle on the same input more than once. It is straightforward to handle repeated queries by having $\mathcal{B}$ maintain a table of queries and responses. We omit this detail for ease of exposition.

2. Algorithm $\mathcal{B}$ constructs crs using same procedure as described in $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$, except it uses $\mathsf{crs}_{\mathsf{NIZK}}$ from the challenger (instead of sampling it itself). It gives crs to $\mathcal{A}$.

3. Whenever $\mathcal{A}$ makes a key-generation query (i, f_i), algorithm $\mathcal{B}$ samples the components $\mathbf{W}_{i,\mathsf{ctr}}$ and $\mathbf{y}_{i,j,\mathsf{ctr}}$ exactly as in $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$. To generate the NIZK proof, algorithm $\mathcal{B}$ forwards the circuit $C_{\mathcal{R}}$, the statement $(\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}})$, and the witness $\mathbf{y}_{i,i,\mathsf{ctr}}$ to the zero-knowledge challenger. The challenger replies with a proof $\pi_{i,\mathsf{ctr}}$. Algorithm $\mathcal{B}$ now responds to $\mathcal{A}$ with the public key $\mathsf{pk}_{\mathsf{ctr}} = \big(\mathbf{W}_{i,\mathsf{ctr}}, \{\mathbf{y}_{i,j,\mathsf{ctr}}\}_{j \neq i}, \pi_{i,\mathsf{ctr}}\big)$.

4. Algorithm $\mathcal{B}$ executes the challenge phase and the output phase exactly as described in $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$. In particular, if the challenger would have halted with output 0 in an execution of $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$ (e.g., if $\xi_{\mathsf{ind}} \neq \xi^*$ in the challenge phase or if $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_{\mathsf{ctr}}) = 0$ in a key-generation query), then algorithm $\mathcal{B}$ also halts with output 0. If algorithm $\mathcal{A}$ halts before the end of the experiment, algorithm $\mathcal{B}$ also outputs 0. If algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$ at the end of the experiment, then algorithm $\mathcal{B}$ also outputs b'.

By Theorem 5.7, for every key-generation query, it is the case that $C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i,i,\mathsf{ctr}}) = 1$. Now, if the challenger samples $\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and

$$\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i,i,\mathsf{ctr}}),$$

then algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_2^{(b)}$. Conversely, if the challenger samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$ and $\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}))$, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_3^{(b)}$. Thus, algorithm $\mathcal{B}$ breaks zero-knowledge with the same advantage ε. □

Lemma 5.13. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{crs}} \geq (m\ell_0 + m)\log(n\ell_0)$, and $\sigma_{\mathsf{key}} \geq 3\ell_0{}^3 m^{9/2} \cdot \sigma_{\mathsf{crs}}$. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The statistical indistinguishability of the two hybrids follows directly from Corollary 4.10. Specifically, for the given choice of parameters, by Corollary 4.10, with overwhelming probability over the choice of $(\mathbf{A}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}})$, the following two distributions have negligible statistical distance:

- Sample
$$\begin{bmatrix} \mathbf{y}_{i,1,\mathsf{ctr}} \\ \vdots \\ \mathbf{y}_{i,N,\mathsf{ctr}} \\ \mathbf{d}_{i,\mathsf{ctr}} \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes (\mathbf{p} + \mathbf{B}_{f_i}\mathbf{G}^{-1}(\mathbf{t}_i)), \sigma_{\mathsf{key}})$$
and output $(\mathbf{y}_{i,1,\mathsf{ctr}}, \ldots, \mathbf{y}_{i,N,\mathsf{ctr}}, \mathbf{d}_{i,\mathsf{ctr}})$.

- Sample $\mathbf{W}_{i,\mathsf{ctr}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{d}_{i,\mathsf{ctr}} \leftarrow \tilde{\mathbf{Z}}^{-1}_{\sigma_{\mathsf{key}}}(\mathsf{vec}(\mathbf{W}_{i,\mathsf{ctr}}))$, $\mathbf{y}_{i,j,\mathsf{ctr}} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{key}}}(\mathbf{W}_{i,\mathsf{ctr}}\mathbf{r}_j)$ for all $j \neq i$, and $\mathbf{y}_{i,i,\mathsf{ctr}} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{key}}}(\mathbf{p} + \mathbf{B}_{f_i}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{W}_{i,\mathsf{ctr}}\mathbf{r}_i)$. Output $(\mathbf{y}_{i,1,\mathsf{ctr}}, \ldots, \mathbf{y}_{i,N,\mathsf{ctr}}, \mathbf{d}_{i,\mathsf{ctr}})$.

The first distribution corresponds to $\mathsf{Hyb}_3^{(b)}$ while the second corresponds to $\mathsf{Hyb}_4^{(b)}$. Finally, algorithm $\mathcal{A}$ makes a polynomial number of key-generation queries, so the two experiments are statistically indistinguishable by a hybrid argument. □

Lemma 5.14. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{key}} > \log m$, and $\beta_{\mathsf{key}} > \sqrt{m}\sigma_{\mathsf{key}}$. There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,* $|\Pr[\mathsf{Hyb}_4^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_5^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The only difference between $\mathsf{Hyb}_4^{(b)}$ and $\mathsf{Hyb}_5^{(b)}$ is the challenger in $\mathsf{Hyb}_4^{(b)}$ additionally checks that $\|\mathbf{y}_{i,j,\mathsf{ctr}}\| \leq \beta_{\mathsf{key}}$ when answering key-generation queries. In $\mathsf{Hyb}_4^{(b)}$, the challenger samples $\mathbf{y}_{i,j,\mathsf{ctr}} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{key}}}(\cdot)$. We show that $\|\mathbf{y}_{i,j,\mathsf{ctr}}\| \leq \beta_{\mathsf{key}}$ with overwhelming probability:

- Since $n \geq \lambda$, $m \geq 3n \log q$, and q is prime, we appeal to Lemma 4.5 to conclude that the distribution of $\mathbf{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$ (and correspondingly, has full column rank).

- By Lemma 3.5, if $\sigma_{\mathsf{key}} > \log m$, then with overwhelming probability, $\|\mathbf{y}_{i,j,\mathsf{ctr}}\| \leq \sqrt{m}\sigma_{\mathsf{key}} \leq \beta_{\mathsf{key}}$.

The number of such vectors $\mathbf{y}_{i,j,\mathsf{ctr}}$ that the challenger samples in $\mathsf{Hyb}_4^{(b)}$ is $N \cdot Q_{\mathsf{keygen}}$, where Q_{keygen} is the number of key-generation queries algorithm $\mathcal{A}$ makes. Since this is polynomially-bounded, the two experiments are statistically indistinguishable by a union bound. □

Lemma 5.15. *For all $b \in \{0,1\}$,* $\Pr[\mathsf{Hyb}_5^{(b)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_6^{(b)}(\mathcal{A}) = 1]$.

Proof. In $\mathsf{Hyb}_5^{(b)}$ and $\mathsf{Hyb}_6^{(b)}$, the adversary's view is *independent* of $\mathbf{y}_{i,i,\mathsf{ctr}}$ for all ctr. In particular, the challenger never gives out $\mathbf{y}_{i,i,\mathsf{ctr}}$ in a key-generation query (i.e., it would correspond to a user's *secret* key). Thus, the adversary's view in these two experiments is identical. □

Lemma 5.16. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{key}} > \log m$, and $\beta_{\mathsf{key}} > \sqrt{m}\sigma_{\mathsf{key}}$. There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$,* $|\Pr[\mathsf{Hyb}_6^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_7^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The two experiments are identical except for the addition of the additional abort events in $\mathsf{Hyb}_7^{(b)}$. We argue that each of these events occurs with negligible probability in $\mathsf{Hyb}_6^{(b)}$:

- In $\mathsf{Hyb}_6^{(b)}$, the challenger samples $\mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ for all $i \in [N]$. By a union bound, the probability that there exists $i \neq j$ where $\mathbf{t}_i = \mathbf{t}_j$ is at most N^2/q^n, which is negligible since $N = \mathsf{poly}(\lambda)$.

- In $\mathsf{Hyb}_6^{(b)}$, on each key-generation query (i, f_i), the challenger samples $\mathbf{y}_{i,i,\mathsf{ctr}} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{key}}}(\mathbf{W}_{i,\mathsf{ctr}}\mathbf{r}_i + \mathbf{t}_i)$. By the same argument as in the proof of Lemma 5.14, $\|\mathbf{y}_{i,i,\mathsf{ctr}}\| \leq \beta_{\mathsf{key}}$ holds with overwhelming probability. The adversary makes a polynomial number of key-generation queries, so by a union bound, the probability that there exists ctr such that $\|\mathbf{y}_{i,i,\mathsf{ctr}}\| > \beta_{\mathsf{key}}$ is negligible.

- For the third event, we use the fact that in $\mathsf{Hyb}_6^{(b)}$, for all values of ctr, the challenger samples $\mathbf{W}_{i,\mathsf{ctr}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ and independently of the matrix $\mathbf{W}_i$ that appears in ξ_{ind}. Thus, the probability that $\mathbf{W}_{i,\mathsf{ctr}} = \mathbf{W}_i$ is exactly q^{-nm}, which is negligible. Again taking a union bound over the total number of key-generation queries algorithm $\mathcal{A}$ makes, the probability that this condition occurs is negligible. □

Lemma 5.17. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{crs}} \geq (m\ell_0 + m)\log(n\ell_0)$, and $\sigma_{\mathsf{key}} \geq 3{\ell_0}^3 m^{9/2} \cdot \sigma_{\mathsf{crs}}$. There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_7^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_8^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. This lemma follows by the same argument as in the proof of Lemma 5.13 □

Lemma 5.18. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, Π_{DGS} satisfies correctness, $\sigma_{\mathsf{crs}} \geq O({\ell_0}^2 m^2)$, $\beta_{\mathsf{agg}} \geq \sqrt{m}\sigma_{\mathsf{agg}}$, and $2^{\lambda_{\mathsf{DGS}}} > \sigma_{\mathsf{agg}} \geq \sigma_{\mathsf{crs}} \cdot O(\ell_0 m^{5/2}) \cdot \sigma_{\mathsf{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q)$. There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_8^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_9^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. Since $\sigma_{\mathsf{crs}} \geq O({\ell_0}^2 m^2) \geq (m\ell_0 + m) \cdot \log(n\ell_0)$, Lemma 4.5 implies that $\|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathsf{crs}}$. By Lemma 4.7, this means $\|\mathbf{T}_{\mathbf{V}}\| \leq \sqrt{m}\sigma_{\mathsf{crs}} \cdot \ell_0 m^2 \leq \sigma_{\mathsf{crs}} \cdot O(\ell_0 m^{5/2})$. Moreover, the following also hold:

- First, $2^{\lambda_{\mathsf{DGS}}} > \sigma_{\mathsf{agg}} \geq \|\mathbf{T}_{\mathbf{V}}\| \cdot \sigma_{\mathsf{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q)$, which follows from the constraint on σ_{agg}.

- Next, $\|\mathbf{0}^{nN}\| \leq 2^{\lambda_{\mathsf{DGS}}}$.

Since $\mathbf{V} \cdot \mathbf{T}_{\mathbf{V}} = \mathbf{G}_{nN}$, by correctness of Π_{DGS}, the distribution of $\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{d}_0$ output by Eq. (5.11) is statistically close to sampling from $\mathbf{V}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{0}^{nN})$. By the structure of $\mathbf{V}$, and the fact that the distribution of $\mathbf{A}$ is statistically close to uniform (Lemma 4.5), we can appeal to Lemma 3.7 to conclude that the marginal distribution of each $\mathbf{y}_{0,i}$ is statistically close to $\mathbf{A}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{W}_0 \mathbf{r}_i)$ where $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. By Lemma 3.5, with overwhelming probability over the choice of $\mathbf{y}_{0,i}$, we have $\|\mathbf{y}_{0,i}\| \leq \sqrt{m}\sigma_{\mathsf{agg}} < \beta_{\mathsf{agg}}$. Since $N = \mathsf{poly}(\lambda)$, by a union bound over all $i \in [N]$, we conclude that with overwhelming probability, $\|\mathbf{y}_{0,i}\| \leq \beta_{\mathsf{agg}}$ for all $i \in [N]$. In this case, the challenger's behavior in $\mathsf{Hyb}_8^{(b)}$ is identical to its behavior in $\mathsf{Hyb}_9^{(b)}$. □

Lemma 5.19. *Suppose Π_{NIZK} is simulation-extractable and $\mathcal{A}$ is admissible. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_9^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{10}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. The only difference between $\mathsf{Hyb}_9^{(b)}$ from $\mathsf{Hyb}_{10}^{(b)}$ are the additional checks the challenger performs when $\mathcal{A}$ makes its ind$^{\text{th}}$ random oracle query ξ_{ind}. Consider an execution of $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$. First, if $\xi_{\mathsf{ind}} \neq \xi^*$, then the challenger in both experiments outputs 0. Thus, it suffices to consider the case where

$$\xi_{\mathsf{ind}} = \xi^* = \big((\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)\big)$$

and $\mathsf{pk}_i^* = \big(\mathbf{W}_i^*, \{\mathbf{y}_{i,j}^*\}_{j \neq i}, \pi_i^*\big)$. Throughout the analysis, we define $\mathbf{B}_{f_i^*} := \mathsf{EvalF}(\mathbf{B}, f_i^*)$. We consider several possibilities:

- Suppose there exists some $i \in [N]$ where $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) \neq 1$. Then the challenger in $\mathsf{Hyb}_{10}^{(b)}$ always outputs 0. We claim this is also the case in $\mathsf{Hyb}_9^{(b)}$. By definition of ξ^* (see Eq. (5.10)), algorithm $\mathcal{A}$ must have submitted $(\mathsf{idx}_i, f_i^*, \mathsf{pk}_i^*)$ as the tuple for slot i during the challenge phase. We consider two possibilities:
  - If $\mathsf{idx}_i \in [\mathsf{ctr}]$, then the challenger looks up $\mathsf{D}[\mathsf{idx}_i] = (i', f', \mathsf{pk}')$ and checks that $i = i'$, $f_i^* = f'$ and $\mathsf{pk}_i^* = \mathsf{pk}'$. Otherwise, the challenger outputs 0. If all of these checks pass, then the challenger must have added the mapping $\mathsf{idx}_i \mapsto (i, f_i^*, \mathsf{pk}_i^*)$ to D in response to a key-generation query. In this case, if $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) = 0$, then the challenger outputs 0 (this is the abort condition in $\mathsf{Hyb}_2^{(b)}$).
  - If $\mathsf{idx}_i = \perp$, then the challenger outputs 0 if $\mathsf{IsValid}(\mathsf{idx}_i, i, f_i^*, \mathsf{pk}_i^*) = 0$.

  In both cases, if $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) = 0$, then the challenger in $\mathsf{Hyb}_9^{(b)}$ would also output 0.

- Suppose $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) = 1$ for all $i \in [N]$, and there exists an index $i \in [N]$ where the following holds:

- $(i, \mathbf{B}_{f_i^*}, \mathbf{W}_i^*) \notin \mathsf{D}_{\mathsf{sk}}$ at the time $\mathcal{A}$ queries ξ_{ind} to the random oracle; and
- $f_i^*(\mathbf{x}) = 0$.

Then, the challenger outputs 0 in $\mathsf{Hyb}_{10}^{(b)}$. We show that the same holds in $\mathsf{Hyb}_9^{(b)}$. By definition of ξ_{ind}, algorithm $\mathcal{A}$ must have submitted $(\mathsf{idx}_i, f_i^*, \mathsf{pk}_i^*)$ as the tuple for slot i in the challenge phase for some choice of $\mathsf{idx}_i \in [\mathsf{ctr}] \cup \{\perp\}$. We consider two cases:

- Suppose $\mathsf{idx}_i \in [\mathsf{ctr}]$. This means the challenger sampled $\mathsf{pk}_i^* = (\mathbf{W}_i^*, \{\mathbf{y}_{i,j}^*\}_{j \neq i}, \pi_i^*)$ in response to a key-generation query on (i, f_i^*). Moreover, since $(i, \mathbf{B}_{f_i^*}, \mathbf{W}_i^*) \notin \mathsf{D}_{\mathsf{sk}}$ at the time it queried the random oracle on ξ_{ind}, the challenger must have sampled $\mathbf{W}_i^*$ when responding to a key-generation query *after* algorithm $\mathcal{A}$ queried the random oracle on ξ_{ind}. But in this case, the challenger always outputs 0 (see the abort conditions from $\mathsf{Hyb}_7^{(b)}$).
- Suppose $\mathsf{idx}_i = \perp$. In this case, if $\mathcal{A}$ is admissible, it must be the case that $f_i^*(\mathbf{x}) = 1$. Thus this case does not happen for an admissible adversary.

Thus, as long as $\mathcal{A}$ is admissible, the challenger outputs 0 in this case in both $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$.

The only setting where the challenger's behavior in $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$ could differ is if the following occurs:

- $\xi_{\mathsf{ind}} = \xi^* = \big((\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)\big)$ where $\mathsf{pk}_i^* = \big(\mathbf{W}_i^*, \{\mathbf{y}_{i,j}^*\}_{j \neq i}, \pi_i^*\big)$.
- For all $i \in [N]$, $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) = 1$.
- There exists an index $i \in [N]$ where $(i, \mathbf{B}_{f_i^*}, \mathbf{W}_i^*) \notin \mathsf{D}_{\mathsf{sk}}$ at the time the adversary queries $H_\rho(\xi_{\mathsf{ind}})$, and moreover, $C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_{f_i^*}, \mathbf{W}_i^*, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_i^*) = 0$ and $\mathbf{y}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i^*}, \mathbf{W}_i^*, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}), \pi_i^*)$.

Let E to be the event that these condition are satisfied. By the above analysis, we have

$$|\Pr[\mathsf{Hyb}_9^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{10}^{(b)}(\mathcal{A}) = 1]| \leq \Pr[\mathsf{E}].$$

Suppose $\Pr[\mathsf{E}] = \varepsilon(\lambda)$ for some non-negligible ε. We now use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that breaks simulation-extractability of Π_{NIZK} as follows:

1. On input the security parameter 1^λ and the common reference string $\mathsf{crs}_{\mathsf{NIZK}}$, algorithm $\mathcal{B}$ starts by sampling $\mathsf{ind} \xleftarrow{\mathrm{R}} [Q_{\mathsf{ro}}]$. Then, it starts running $\mathcal{A}(1^\lambda)$. Whenever algorithm $\mathcal{A}$ makes a random oracle query, algorithm $\mathcal{B}$ responds with a random string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$.

2. Algorithm $\mathcal{B}$ constructs crs using the procedure described in $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$, except it uses $\mathsf{crs}_{\mathsf{NIZK}}$ from the challenger (instead of sampling it itself). It gives crs to $\mathcal{A}$. In addition, algorithm $\mathcal{B}$ initializes an empty dictionary D_{sk}. When sampling $\mathbf{t}_1, \ldots, \mathbf{t}_N \in \mathbb{Z}_q^n$, if there exists $i \neq j$ such that $\mathbf{t}_i = \mathbf{t}_j$, then algorithm $\mathcal{B}$ aborts with output 0 (as in $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$).

3. Whenever algorithm $\mathcal{A}$ makes a key-generation query on (i, f_i), algorithm $\mathcal{B}$ proceeds as follows:

   - First, algorithm $\mathcal{B}$ increments its counter $\mathsf{ctr} = \mathsf{ctr} + 1$. Then it computes $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$.
   - Next, it samples $\mathbf{W}_{i,\mathsf{ctr}}$, $\{\mathbf{y}_{i,j,\mathsf{ctr}}\}_{j \in [N]}$ using the procedure described in $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$ (i.e., according to Eq. (5.12)).
   - To generate the NIZK proof, algorithm $\mathcal{B}$ sends the circuit $C_{\mathcal{R}}$ and the statement $(\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}})$ to the challenger. The challenger replies with a (simulated) proof $\pi_{i,\mathsf{ctr}}$.
   - Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the public key $\mathsf{pk}_{\mathsf{ctr}} = \big(\mathbf{W}_{i,\mathsf{ctr}}, \{\mathbf{y}_{i,j,\mathsf{ctr}}\}_{j \neq i}, \pi_{i,\mathsf{ctr}}\big)$.
   - Finally, algorithm $\mathcal{B}$ adds the mapping $(i, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}) \mapsto (0, \mathbf{y}_{i,i,\mathsf{ctr}})$ to D_{sk}.

   As in $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$, if $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_{\mathsf{ctr}}) = 0$ or $\|\mathbf{y}_{i,i,\mathsf{ctr}}\| > \beta_{\mathsf{key}}$, algorithm $\mathcal{B}$ aborts and outputs $\perp$.

4. When algorithm $\mathcal{A}$ makes its ind$^{\text{th}}$ query ξ_{ind} to the random oracle, algorithm $\mathcal{B}$ attempts to parse $\xi_{\mathsf{ind}} = \big((\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)\big)$ where $\mathsf{pk}_i^* = \big(\mathbf{W}_i^*, \{\mathbf{y}_{i,j}^*\}_{j \neq i}, \pi_i^*\big)$. If ξ_{ind} does not have this form, algorithm $\mathcal{B}$ aborts with output $\perp$. Otherwise, algorithm $\mathcal{B}$ samples $i \xleftarrow{\mathrm{R}} [N]$ and outputs the relation $C_{\mathcal{R}}$, the statement $(\mathbf{A}, \mathbf{B}_{f_i^*}, \mathbf{W}_i^*, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}})$ and the proof π_i^*.

5. If $\mathcal{A}$ enters the challenge phase before making ind queries to the random oracle or if $\mathcal{A}$ aborts the experiment, then $\mathcal{B}$ aborts with output $\perp$.

By construction, algorithm $\mathcal{B}$ samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$. It constructs the proofs as $\pi_{i,\mathsf{ctr}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathsf{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathsf{key}}))$. Hence, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_9^{(b)}$ and $\mathsf{Hyb}_{10}^{(b)}$ for $\mathcal{A}$. Thus, with probability at least ε, event E occurs. This means there exists $i^* \in [N]$ where

- $(i^*, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*) \notin \mathsf{D}_{\mathsf{sk}}$ at the time the adversary queries $H_\rho(\xi_{\mathsf{ind}})$.
- $\mathsf{IsValid}(\mathsf{crs}, i^*, f_{i^*}^*, \mathsf{pk}_{i^*}^*) = 1$.
- $C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i^*}^*) = 0$ where $\mathbf{y}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}}), \pi_{i^*}^*)$.

Suppose $i = i^*$. Since algorithm $\mathcal{B}$ samples $i \xleftarrow{\mathrm{R}} [N]$, this occurs with probability $1/N$. We claim that in this case, algorithm $\mathcal{B}$ wins the simulation-extractability game:

- First, we argue that algorithm $\mathcal{B}$ did not submit the statement $(\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}})$ to the challenger and receive back the proof $\pi_{i^*}^*$. Since $\mathbf{t}_1, \ldots, \mathbf{t}_N$ are distinct, algorithm $\mathcal{B}$ would only request a simulated proof on the statement $(\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}})$ if the following occurs:
  - Algorithm $\mathcal{A}$ made a key-generation query on the pair (i^*, f^*) for some f^* where $\mathsf{EvalF}(\mathbf{B}, f^*) = \mathbf{B}_{f_{i^*}^*}$.
  - When responding to the key-generation query, algorithm $\mathcal{B}$ sampled the matrix $\mathbf{W}_{i^*}^*$ in response.

  By construction, if this happened, then algorithm $\mathcal{B}$ would have also added $(i^*, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*)$ to D_{sk}, which is a contradiction.
- Since $\mathsf{IsValid}(\mathsf{crs}, i^*, f_{i^*}^*, \mathsf{pk}_{i^*}^*) = 1$, this means $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}}), \pi_{i^*}^*) = 1$.
- Finally, we have $C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}}), \mathbf{y}_{i^*}^*) = 0$ where

$$\mathbf{y}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_{i^*}^*}, \mathbf{W}_{i^*}^*, \mathbf{r}_{i^*}, \mathbf{t}_{i^*}, \mathbf{p}, \beta_{\mathsf{key}}), \pi_{i^*}).$$

  This means algorithm $\mathcal{B}$ wins the simulation-extractability game.

Thus, as long as event E occurs and $i = i^*$, algorithm $\mathcal{B}$ wins the simulation-extractability game with overwhelming probability. As argued above, $\Pr[\mathsf{E} \wedge i = i^*] = \varepsilon/N$, which is non-negligible, and the claim holds. □

Lemma 5.20. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, and $\sigma_{\mathsf{crs}} \geq (m\ell_0 + m)\log(n\ell_0)$. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,* $|\Pr[\mathsf{Hyb}_{10}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{11}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. Since $\sigma_{\mathsf{crs}} \geq (m\ell_0 + m)\log(n\ell_0)$, by Lemma 4.5, the following distributions are statistically indistinguishable:

$$\left\{(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{crs}})\right\} \quad \text{and} \quad \left\{(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) : \begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{U}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\ell_0 \times m} \\ \mathbf{T}_0 \leftarrow [\mathbf{I}_{\ell_0} \otimes \mathbf{A} \mid \mathbf{U}_0]_{\sigma_{\mathsf{crs}}}^{-1}(\mathbf{G}_{n\ell_0}). \end{array}\right\}$$

Moreover, $\|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathsf{crs}}$ in the left distribution. By Lemma 4.7, $\|\mathbf{T}_{\mathbf{V}}\|, \|\mathbf{R}\| \leq \|\mathbf{T}_0\| \cdot \ell_0 m^2$ and the claim holds. □

Lemma 5.21. *Suppose $(\ell_0 m^{5/2} \cdot \sigma_{\mathsf{crs}}) \cdot \sigma_{\mathsf{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q) < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$ and Π_{DGS} is explainable (Definition 4.1). For every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,* $|\Pr[\mathsf{Hyb}_{11}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{12,p}^{(b)}(\mathcal{A}) = 1]| = 1/p(\lambda) + \mathsf{negl}(\lambda)$.

Proof. First, in $\mathsf{Hyb}_{11}^{(b)}$ and $\mathsf{Hyb}_{12,p}^{(b)}$, the outputs is 0 unless $\|\mathbf{T}_{\mathbf{V}}\| \leq \ell_0 m^2 \cdot \|\mathbf{T}_0\| \leq \ell_0 m^{5/2} \sigma_{\mathrm{crs}}$. Thus, it suffices to consider the case where $\|\mathbf{T}_{\mathbf{V}}\| \cdot \sigma_{\mathrm{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q) < \sigma_{\mathrm{agg}} < 2^{\lambda_{\mathsf{DGS}}}$. Moreover, $\|\mathbf{0}^{nN}\| \leq 2^{\lambda_{\mathsf{DGS}}}$. Thus, by the explainability of Π_{DGS}, the following distributions have $1/p(\lambda) + \mathsf{negl}(\lambda)$ statistical distance:

- $\mathcal{D}_{\mathsf{SamplePre}}$: Sample $\gamma^* \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\rho$ and $\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma_{\mathrm{agg}}; \gamma^*)$. Output $(\boldsymbol{\kappa}_0, \gamma^*)$.
- $\mathcal{D}_{\mathsf{Explain},p(\lambda)}$: Sample $\gamma \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\rho$ and $\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma; \gamma)$. Then resample the randomness $\gamma^* \leftarrow \mathsf{DGS.Explain}(1^{\lambda_{\mathsf{DGS}}}, 1^{p(\lambda)}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \boldsymbol{\kappa}_0, \sigma_{\mathrm{agg}})$. Output $(\boldsymbol{\kappa}_0, \gamma^*)$.

In $\mathsf{Hyb}_{11}^{(b)}$, the challenger samples γ^* (i.e., the value of $H_\rho(\xi_{\mathrm{ind}})$) according to the distribution $\mathcal{D}_{\mathsf{SamplePre}}$, whereas in $\mathsf{Hyb}_{12,p}^{(b)}$, the challenger samples γ^* according to the procedure in $\mathcal{D}_{\mathsf{Explain},p(\lambda)}$. The remainder of the experiment is unchanged so the claim follows. □

Lemma 5.22. *Suppose* $(\ell_0 m^{5/2} \cdot \sigma_{\mathrm{crs}}) \cdot \sigma_{\mathrm{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q) < \sigma_{\mathrm{agg}} < 2^{\lambda_{\mathsf{DGS}}}$ *and* Π_{DGS} *is correct (Definition 4.1). For every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $b \in \{0,1\}$ *and all* $\lambda \in \mathbb{N}$*,*

$$|\Pr[\mathsf{Hyb}_{12,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{13,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. As in the proof of Lemma 5.21, the outputs in $\mathsf{Hyb}_{12,p}^{(b)}$ and $\mathsf{Hyb}_{13,p}^{(b)}$ is 0 unless $\|\mathbf{T}_{\mathbf{V}}\| \leq \ell_0 m^2 \cdot \|\mathbf{T}_0\| \leq \ell_0 m^{5/2} \sigma_{\mathrm{crs}}$. Thus, it suffices to consider the case where $\|\mathbf{T}_{\mathbf{V}}\| \cdot \sigma_{\mathrm{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q) < \sigma_{\mathrm{agg}} < 2^{\lambda_{\mathsf{DGS}}}$. Then, by correctness of Π_{DGS}, the following two distributions are statistically indistinguishable:

$$\left\{\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma_{\mathrm{agg}})\right\} \quad \text{and} \quad \left\{\boldsymbol{\kappa}_0 \leftarrow \mathbf{V}_{\sigma_{\mathrm{agg}}}^{-1}(\mathbf{0}^{nN})\right\}.$$

In $\mathsf{Hyb}_{12,p}^{(b)}$, the challenger samples $\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \sigma_{\mathrm{agg}}; \gamma)$ where $\gamma \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\rho$. This corresponds to the left distribution. In $\mathsf{Hyb}_{13,p}^{(b)}$, the challenger samples $\boldsymbol{\kappa}_0 \leftarrow \mathbf{V}_{\sigma_{\mathrm{agg}}}^{-1}(\mathbf{0}^{nN})$, which corresponds to the right distribution. We conclude that the two distributions are statistically indistinguishable. □

Lemma 5.23. *Suppose* $n \geq \lambda$, $m \geq 2n \log q$, q *is prime, and* $\sigma_{\mathrm{agg}} \geq 4 \log(\ell_0 m)$*. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $b \in \{0,1\}$ *and all* $\lambda \in \mathbb{N}$*,*

$$|\Pr[\mathsf{Hyb}_{13,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{14,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. This follow from Lemma 3.7. Since $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathrm{agg}} \geq 4 \log(\ell_0 m)$, with overwhelming probability over the choice of $\mathbf{A} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}$, the statistical distance between the following distributions is negligible:

- Sample and output $\boldsymbol{\kappa}_0 \leftarrow \mathbf{V}_{\sigma_{\mathrm{agg}}}^{-1}(\mathbf{0}^{nN})$.
- Sample and output $\boldsymbol{\kappa}_0$ where

$$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,1} \\ \vdots \\ \mathbf{y}_{0,N} \\ \mathbf{d}_0 \end{bmatrix} \quad \text{where} \quad \mathbf{d}_0 \leftarrow D_{\mathbb{Z},\sigma_{\mathrm{agg}}}^k \text{ and } \begin{bmatrix} \mathbf{y}_{0,1} \\ \vdots \\ \mathbf{y}_{0,N} \end{bmatrix} \leftarrow (\mathbf{I}_N \otimes \mathbf{A})_{\sigma_{\mathrm{agg}}}^{-1}\left(\begin{bmatrix} \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1)\mathbf{d}_0 \\ \vdots \\ \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N)\mathbf{d}_0 \end{bmatrix}\right).$$

The first distribution is the distribution of $(\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{d}_0)$ in $\mathsf{Hyb}_{13,p}^{(b)}$, while the second is the distribution in $\mathsf{Hyb}_{14,p}^{(b)}$ since

$$\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{d}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)\mathbf{r}_i = \mathbf{W}_0 \mathbf{r}_i.$$

□

Lemma 5.24. *Suppose q is prime and* $\sigma_{\mathrm{agg}} \geq k \log nm$*. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $b \in \{0,1\}$ *and all* $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_{14,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{15,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. Since $\ell_0 \geq Nm'$, we appeal to Lemma 4.7 to conclude that the marginal distribution of $\mathbf{Z}$ (hence $\tilde{\mathbf{Z}}$) is statistically close to uniformly random, $\tilde{\mathbf{Z}}\mathbf{T}_{\tilde{\mathbf{Z}}} = \mathbf{G}_{nm}$, and $\|\mathbf{T}_{\tilde{\mathbf{Z}}}\| = 1$. Since $k = 3nm \log q > 2nm \log q$, q is prime, and $\sigma_{\mathsf{agg}} \geq k \log nm \geq \log k$, by Lemma 3.6, with overwhelming probability over the choice of $\tilde{\mathbf{Z}}$, the two following distributions are statistically close:

$$\left\{(\mathbf{d}_0, \tilde{\mathbf{Z}}\mathbf{d}_0) : \mathbf{d}_0 \leftarrow D^k_{\mathbb{Z},\sigma_{\mathsf{agg}}}\right\} \quad \text{and} \quad \left\{(\mathbf{d}_0, \mathsf{vec}(\mathbf{W}_0)) : \mathbf{W}_0 \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \mathbf{d}_0 \leftarrow \tilde{\mathbf{Z}}^{-1}_{\sigma_{\mathsf{agg}}}(\mathsf{vec}(\mathbf{W}_0))\right\}.$$

Moreover, note that if we define $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$, then $\mathsf{vec}(\mathbf{W}_0) = \tilde{\mathbf{Z}}\mathbf{d}_0$. Furthermore, by Lemma 3.8, given that $\tilde{\mathbf{Z}}\mathbf{T}_{\tilde{\mathbf{Z}}} = \mathbf{G}_{nm}$ and $\sigma_{\mathsf{agg}} \geq k \log nm = k\|\mathbf{T}_{\tilde{\mathbf{Z}}}\| \log(nm)$, the following two distributions are statistically indistinguishable:

$$\{\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathsf{vec}(\mathbf{W}_0), \sigma_{\mathsf{agg}})\} \quad \text{and} \quad \{\mathbf{d}_0 \leftarrow \tilde{\mathbf{Z}}^{-1}_{\sigma_{\mathsf{agg}}}(\mathsf{vec}(\mathbf{W}_0))\}.$$

Combining the two, we conclude that the following distributions are statistically indistinguishable:

- Sample $\mathbf{d}_0 \leftarrow D^k_{\mathbb{Z},\sigma_{\mathsf{agg}}}$ and set $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. This is the distribution in $\mathsf{Hyb}^{(b)}_{14,p}$.
- Sample $\mathbf{W}_0 \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}$ and set $\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathsf{vec}(\mathbf{W}_0), \sigma_{\mathsf{agg}})$. This is the distribution in $\mathsf{Hyb}^{(b)}_{15,p}$. □

Lemma 5.25. *For every polynomial p, all $b \in \{0,1\}$, and all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Hyb}^{(b)}_{15,p}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}^{(b)}_{16,p}(\mathcal{A}) = 1]$.*

Proof. In both experiments, the distributions of $\mathbf{U}_{\mathsf{ct}}$ and $\mathbf{W}_0$ are uniform over $\mathbb{Z}_q^{n\times m}$. Thus, these two experiments are identically distributed. □

Lemma 5.26. *Suppose $n \geq \lambda$, $m > 2n \log q$ and $q > 2$ is a prime. For every polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}^{(b)}_{16,p}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{17,p}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. The indistinguishability of the two hybrids follows from the generalized leftover hash lemma (Lemma 3.3). Given that $m > 2n \log q$ and $q > 2$ is a prime, the following pairs of distributions are statistically close for any fixed vector $\mathbf{e} \in \mathbb{Z}_q^m$:

- $\left\{(\mathbf{A}, \mathbf{A}\mathbf{K}_{\mathbf{U}}, \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{U}}) : \ \mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \ \mathbf{K}_{\mathbf{U}} \xleftarrow{\textsf{R}} \{0,1\}^{m\times m}\right\}$ and $\left\{(\mathbf{A}, \mathbf{U}^*_{\mathsf{ct}}, \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{U}}) : \begin{array}{c} \mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \ \mathbf{K}_{\mathbf{U}} \xleftarrow{\textsf{R}} \{0,1\}^{m\times m} \\ \mathbf{U}^*_{\mathsf{ct}} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m} \end{array}\right\}$.
- $\left\{(\mathbf{A}, \mathbf{A}\mathbf{k}_{\mathbf{p}}, \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}}) : \ \mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \ \mathbf{k}_{\mathbf{p}} \xleftarrow{\textsf{R}} \{0,1\}^{m}\right\}$ and $\left\{(\mathbf{A}, \mathbf{p}, \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}}) : \begin{array}{c} \mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \ \mathbf{k}_{\mathbf{p}} \xleftarrow{\textsf{R}} \{0,1\}^{m} \\ \mathbf{p} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n} \end{array}\right\}$.
- $\left\{(\mathbf{A}, \mathbf{A}\mathbf{K}_{\mathbf{W}}, \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}) : \ \mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \ \mathbf{K}_{\mathbf{W}} \xleftarrow{\textsf{R}} \{0,1\}^{m\times m}\right\}$ and $\left\{(\mathbf{A}, \mathbf{W}^*_0, \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}) : \begin{array}{c} \mathbf{A} \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m}, \ \mathbf{K}_{\mathbf{W}} \xleftarrow{\textsf{R}} \{0,1\}^{m\times m} \\ \mathbf{W}^*_0 \xleftarrow{\textsf{R}} \mathbb{Z}_q^{n\times m} \end{array}\right\}$.

The lemma now follows by a standard hybrid argument. □

Lemma 5.27. *For every polynomial p, all $b \in \{0,1\}$, and all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Hyb}^{(b)}_{17,p}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}^{(b)}_{18,p}(\mathcal{A}) = 1]$.*

Proof. The difference between these two experiments is purely syntactic. Let $\xi_{\mathsf{ind}} = ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j\neq i}, \pi_i)$. Let $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$. Recall the following invariants introduced in $\mathsf{Hyb}^{(b)}_{10}$. If the challenger does not terminate early, then $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_i) = 1$ for all $i \in [N]$, and moreover, every tuple $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$ associated with ξ_{ind} is contained in D_{sk}. In addition, the mappings in D_{sk} satisfy the following properties:

- If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}^*_i)$, then $\|\mathbf{y}^*_i\| \leq \beta_{\mathsf{key}}$ and $\mathbf{A}\mathbf{y}^*_i = \mathbf{W}_i\mathbf{r}_i + \mathbf{t}_i$.
- If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}^*_i)$, then $\|\mathbf{y}^*_i\| \leq \beta_{\mathsf{key}}$, $\mathbf{A}\mathbf{y}^*_i = \mathbf{W}_i\mathbf{r}_i + \mathbf{B}_{f_i}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p}$, and $f_i(\mathbf{x}) = 1$.

In $\mathsf{Hyb}^{(b)}_{17,p}$, the challenger samples $\mathbf{y}_{0,i} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{W}_0\mathbf{r}_i)$ for all $i \in [N]$. We show that this coincides with the challenger's behavior in $\mathsf{Hyb}^{(b)}_{18,p}$. We will use the following properties:

- Since $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_i) = 1$ this means that $\mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i \mathbf{r}_j$ for all $i \neq j$.
- In $\mathsf{Hyb}_{17,p}^{(b)}$ and $\mathsf{Hyb}_{18,p}^{(b)}$, the challenger sets $\mathbf{W}_0 = \mathbf{W}_0^* - \sum_{i \in [N]} \mathbf{W}_i = \mathbf{A}\mathbf{K}_\mathbf{W} - \sum_{i \in [N]} \mathbf{W}_i$.

We now consider the two possibilities:

- Suppose $\mathsf{D}_\mathsf{sk}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}_i^*)$. Then $\mathbf{A}\mathbf{y}_i^* = \mathbf{W}_i \mathbf{r}_i + \mathbf{t}_i$. This allows us to write

$$\mathbf{W}_0 \mathbf{r}_i = \mathbf{W}_0^* \mathbf{r}_i - \sum_{j \neq i} \mathbf{W}_j \mathbf{r}_i - (\mathbf{W}_i \mathbf{r}_i + \mathbf{t}_i) + \mathbf{t}_i = \mathbf{A}\mathbf{K}_\mathbf{W} \mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}_i^* + \mathbf{t}_i.$$

  This coincides with the challenger's behavior in $\mathsf{Hyb}_{18,p}^{(b)}$.

- Suppose $\mathsf{D}_\mathsf{sk}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}_i^*)$. Then $\mathbf{A}\mathbf{y}_i^* = \mathbf{W}_i \mathbf{r}_i + \mathbf{B}_{f_i} \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p}$ and $f_i(\mathbf{x}) = 1$. As in the proof of Theorem 5.8 (see Eq. (5.8)), when $\mathbf{B} = \mathbf{U}_\mathsf{ct} \mathbf{T}_\mathsf{fun} \in \mathbb{Z}_q^{n \times \ell m'}$, we have

$$[\mathbf{A} \mid \mathbf{U}_\mathsf{ct} - (\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_n)\mathbf{U}] \cdot \begin{bmatrix} -(\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_m)\mathbf{T}_\mathsf{in} \\ \mathbf{T}_\mathsf{fun} \end{bmatrix} = \mathbf{B} - \mathbf{x}^\mathsf{T} \otimes \mathbf{G}.$$

  By Theorem 3.9,

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

  Next, in $\mathsf{Hyb}_{17,p}^{(b)}$ and $\mathsf{Hyb}_{18,p}^{(b)}$, the challenger sets $\mathbf{U}_\mathsf{ct} = \mathbf{U}_\mathsf{ct}^* + (\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_n)\mathbf{U}$ and $\mathbf{U}_\mathsf{ct}^* = \mathbf{A}\mathbf{K}_\mathbf{U}$. Thus, we can now write

$$\begin{aligned}
\mathbf{B}_{f_i} &= (\mathbf{B} - \mathbf{x}^\mathsf{T} \otimes \mathbf{G}) \cdot \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} + \mathbf{G} \\
&= [\mathbf{A} \mid \mathbf{U}_\mathsf{ct} - (\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_n)\mathbf{U}] \cdot \begin{bmatrix} -(\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_m)\mathbf{T}_\mathsf{in} \\ \mathbf{T}_\mathsf{fun} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} + \mathbf{G} \\
&= [\mathbf{A} \mid \mathbf{U}_\mathsf{ct}^*] \cdot \begin{bmatrix} -(\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_m)\mathbf{T}_\mathsf{in} \\ \mathbf{T}_\mathsf{fun} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} + \mathbf{G} \\
&= [\mathbf{A} \mid \mathbf{A}\mathbf{K}_\mathbf{U}] \cdot \begin{bmatrix} -(\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_m)\mathbf{T}_\mathsf{in} \\ \mathbf{T}_\mathsf{fun} \end{bmatrix} \cdot \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} + \mathbf{G} \\
&= \mathbf{A} \underbrace{(\mathbf{K}_\mathbf{U} \mathbf{T}_\mathsf{fun} \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} - (\mathbf{x}^\mathsf{T} \otimes \mathbf{I}_m)\mathbf{T}_\mathsf{in} \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}})}_{\mathbf{K}_\mathbf{B}^{(i)}} + \mathbf{G} = \mathbf{A}\mathbf{K}_\mathbf{B}^{(i)} + \mathbf{G}.
\end{aligned}$$

  In $\mathsf{Hyb}_{17,p}^{(b)}$ and $\mathsf{Hyb}_{18,p}^{(b)}$, the challenger sets $\mathbf{p} = \mathbf{A}\mathbf{k}_\mathbf{p}$, so we can now write

$$\begin{aligned}
\mathbf{A}\mathbf{y}_i^* &= \mathbf{W}_i \mathbf{r}_i + \mathbf{B}_{f_i} \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p} \\
&= \mathbf{W}_i \mathbf{r}_i + (\mathbf{A}\mathbf{K}_\mathbf{B}^{(i)} + \mathbf{G})\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{A}\mathbf{k}_\mathbf{p} \\
&= \mathbf{W}_i \mathbf{r}_i + \mathbf{A}(\mathbf{K}_\mathbf{B}^{(i)} \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{k}_\mathbf{p}) + \mathbf{t}_i.
\end{aligned}$$

  Consider now the term $\mathbf{W}_0 \mathbf{r}_i$:

$$\begin{aligned}
\mathbf{W}_0 \mathbf{r}_i &= \mathbf{W}_0^* \mathbf{r}_i - \sum_{j \neq i} \mathbf{W}_j \mathbf{r}_i - \mathbf{W}_i \mathbf{r}_i \\
&= \mathbf{A}\mathbf{K}_\mathbf{W} \mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{W}_i \mathbf{r}_i \\
&= \mathbf{A}\mathbf{K}_\mathbf{W} \mathbf{r}_i - \sum_{j \neq i} \mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}_i^* + \mathbf{A}\mathbf{K}_\mathbf{B}^{(i)} \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{A}\mathbf{k}_\mathbf{p} + \mathbf{t}_i.
\end{aligned}$$

  Once again, this coincides with the challenger's behavior in $\mathsf{Hyb}_{18,p}^{(b)}$.

We conclude that the challenger samples $\mathbf{y}_{0,i}$ using identical procedures in the two experiments. □

Lemma 5.28. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} > 2^{\lambda}(\beta_{\mathsf{key}} + m^{O(d)}\sigma_{\mathsf{crs}})$. Then, for every polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}^{(b)}_{18,p}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{19,p}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. As in the proof of Lemma 5.27, parse $\xi_{\mathsf{ind}} = ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$ where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$. Let $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$. If the challenger does not terminate early, then $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_i) = 1$ for all $i \in [N]$, and moreover, every tuple $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$ associated with ξ_{ind} is contained in D_{sk}. In addition, the mappings in D_{sk} satisfy the following properties:

- If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}^*_i)$, then $\|\mathbf{y}^*_i\| \leq \beta_{\mathsf{key}}$ and $\mathbf{A}\mathbf{y}^*_i = \mathbf{W}_i\mathbf{r}_i + \mathbf{t}_i$.
- If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}^*_i)$, then $\|\mathbf{y}^*_i\| \leq \beta_{\mathsf{key}}$, $\mathbf{A}\mathbf{y}^*_i = \mathbf{W}_i\mathbf{r}_i + \mathbf{B}_{f_i}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{p}$, and $f_i(\mathbf{x}) = 1$.

We now show that $\mathsf{Hyb}^{(b)}_{18,p}$ and $\mathsf{Hyb}^{(b)}_{19,p}$ are statistically indistinguishable by Theorem 4.3. To do so, we first define the vector $\hat{\mathbf{y}}_{0,i} \in \mathbb{Z}^m_q$:

$$\hat{\mathbf{y}}_{0,i} = \begin{cases} \mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}^*_i & \mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}^*_i) \\ \mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}^*_i + \mathbf{K}^{(i)}_{\mathbf{B}}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{k}_{\mathbf{p}} & \mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}^*_i). \end{cases}$$

We start by bounding $\|\hat{\mathbf{y}}_{0,i}\|_2$ for all $i \in [N]$.

- By construction, $\|\mathbf{K}_{\mathbf{U}}\|, \|\mathbf{K}_{\mathbf{W}}\|, \|\mathbf{k}_{\mathbf{p}}\|, \|\mathbf{G}^{-1}(\mathbf{t}_i)\| \leq 1$.
- Since $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_i) = 1$, this means $\|\mathbf{y}_{i,j}\| \leq \beta_{\mathsf{key}}$ for all $i \neq j$.
- From the abort condition introduced in $\mathsf{Hyb}^{(b)}_{11}$, we have that $\|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathsf{crs}}$ and $\|\mathbf{R}\| \leq \ell_0 m^2 \cdot \|\mathbf{T}_0\| \leq \ell_0 m^{5/2} \cdot \sigma_{\mathsf{crs}}$. Combined with Lemma 4.6, we further have $\|\mathbf{T}_{\mathsf{fun}}\|, \|\mathbf{T}_{\mathsf{in}}\| \leq \|\mathbf{T}_{\mathsf{ct}}\| \leq \|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathsf{crs}}$.
- By Theorem 3.9, we have $\|\mathbf{H}_{\mathbf{B},f_i,\mathbf{x}}\| \leq m^{O(d)}$. Therefore,

$$\begin{aligned} \|\mathbf{K}^{(i)}_{\mathbf{B}}\| &= \|\mathbf{K}_{\mathbf{U}}\mathbf{T}_{\mathsf{fun}}\mathbf{H}_{\mathbf{B},f_i,\mathbf{x}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m)\mathbf{T}_{\mathsf{in}}\mathbf{H}_{\mathbf{B},f_i,\mathbf{x}}\| \\ &\leq m \cdot \sqrt{m}\sigma_{\mathsf{crs}} \cdot \ell m \cdot m^{O(d)} + \ell m \cdot \sqrt{m}\sigma_{\mathsf{crs}} \cdot \ell m \cdot m^{O(d)} \leq \ell_0{}^2 m^{O(d)}\sigma_{\mathsf{crs}}. \end{aligned}$$

We now consider the two cases:

- If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}^*_i)$, then

$$\left\|\hat{\mathbf{y}}_{0,i}\right\| = \left\|\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \textstyle\sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}^*_i\right\| \leq m \cdot \ell_0 m^{5/2}\sigma_{\mathsf{crs}} + N\beta_{\mathsf{key}} = \ell_0 m^{7/2}\sigma_{\mathsf{crs}} + N\beta_{\mathsf{key}}.$$

- If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}^*_i)$, then

$$\begin{aligned} \left\|\hat{\mathbf{y}}_{0,i}\right\| &= \left\|\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \textstyle\sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}^*_i + \mathbf{K}^{(i)}_{\mathbf{B}}\mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{k}_{\mathbf{p}}\right\| \\ &\leq m \cdot \ell_0 m^{5/2}\sigma_{\mathsf{crs}} + N\beta_{\mathsf{key}} + \ell_0^2 m^{O(d)}\sigma_{\mathsf{crs}} \cdot m + 1 \\ &\leq \ell_0^2 m^{O(d)}\sigma_{\mathsf{crs}} + N\beta_{\mathsf{key}} \end{aligned}$$

Thus, for all $i \in [N]$,

$$\|\hat{\mathbf{y}}_{0,i}\|_2 \leq \sqrt{m}\|\hat{\mathbf{y}}_{0,i}\| \leq \ell_0^2 m^{O(d)}\sigma_{\mathsf{crs}} + N\beta_{\mathsf{key}} \leq \ell_0^2 m^{(O(d)}\sigma_{\mathsf{crs}} + \ell_0\beta_{\mathsf{key}}.$$

Since $\sigma_{\mathsf{agg}} > 2^{\lambda}(m^{O(d)}\sigma_{\mathsf{crs}} + \beta_{\mathsf{key}})$ and $\ell_0 = \mathsf{poly}(\lambda)$, we conclude that $\sqrt{\|\hat{\mathbf{y}}_{0,i}\|_2/\sigma_{\mathsf{agg}}}$ is negligible. By Theorem 4.3, for all $i \in [N]$, the following two distributions are statistically close:

$$\left\{\mathbf{A}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{t}_i + \mathbf{A}\hat{\mathbf{y}}_{0,i})\right\} \quad \text{and} \quad \left\{\mathbf{A}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{t}_i) + \hat{\mathbf{y}}_{0,i}\right\}.$$

The left distribution corresponds to the sampling procedure of $\mathsf{Hyb}^{(b)}_{18,p}$, while the right distribution corresponds to the sampling procedure of $\mathsf{Hyb}^{(b)}_{19,p}$. The lemma now follows by a standard hybrid argument. □

Lemma 5.29. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} > \log m$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{19,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{20,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The challenger in $\mathsf{Hyb}_{19,p}^{(b)}$ and $\mathsf{Hyb}_{20,p}^{(b)}$ samples $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$. Since $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} > \log m$, we can appeal to Lemma 3.6 to conclude that the following distributions are statistically indistinguishable:

$$\left\{(\mathbf{k}_{\mathbf{t}_i}, \mathbf{A}\mathbf{k}_{\mathbf{t}_i}) \ : \ \mathbf{k}_{\mathbf{t}_i} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}^m\right\} \quad \text{and} \quad \left\{(\mathbf{k}_{\mathbf{t}_i}, \mathbf{t}_i) \ : \ \mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \ , \ \mathbf{k}_{\mathbf{t}_i} \leftarrow \mathbf{A}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{t}_i)\right\}.$$

The left distribution corresponds to how the challenger samples $(\mathbf{k}_{\mathbf{t}_i}, \mathbf{t}_i)$ in $\mathsf{Hyb}_{20,p}^{(b)}$ while the right distribution corresponds to how the challenger samples them in $\mathsf{Hyb}_{19,p}^{(b)}$. The claim now follows by a hybrid argument. □

Lemma 5.30. *Suppose $m \geq \lambda$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,* $|\Pr[\mathsf{Hyb}_{20,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{21,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. The only difference between these two experiments is if in the challenge phase, the challenger samples $\mathbf{e} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{LWE}}}^m$ where $\|\mathbf{e}\| > \sqrt{m}\sigma_{\mathsf{LWE}}$. By Lemma 3.5, this happens with negligible probability. □

Lemma 5.31. *Suppose the ℓ_0-succinct LWE assumption holds for parameter $(n, m, q, \sigma_{\mathsf{LWE}}, \sigma_{\mathsf{crs}})$. For every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$\Pr[\mathsf{Hyb}_{21,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{22,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. Suppose $|\Pr[\mathsf{Hyb}_{21,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{22,p}^{(b)}(\mathcal{A}) = 1]| = \varepsilon(\lambda)$ for some non-negligible ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the ℓ_0-succinct LWE assumption:

1. On input the security parameter 1^λ and the ℓ_0-succinct LWE challenge $(\mathbf{A}, \mathbf{c}^{\mathsf{T}}, \mathbf{U}_0, \mathbf{T}_0)$, algorithm $\mathcal{B}$ samples an index $\mathsf{ind} \xleftarrow{\mathrm{R}} [Q_{\mathsf{ro}}]$. It starts running algorithm $\mathcal{A}$ on input the security parameter 1^λ. Algorithm $\mathcal{A}$ outputs the slot count 1^N, the policy family 1^τ, and an attribute $\mathbf{x} \in \{0,1\}^\ell$.

2. Algorithm $\mathcal{B}$ simulates the setup phase by sampling

$$\begin{aligned}
(\mathbf{U}, \mathbf{T}_{\mathsf{ct}}) &\leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, [\ell]) \\
(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) &\leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N) \\
(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) &\leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda) \\
\mathbf{K}_{\mathbf{U}}, \mathbf{K}_{\mathbf{W}} &\xleftarrow{\mathrm{R}} \{0,1\}^{m \times m}, \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m \\
\mathbf{U}_{\mathsf{ct}} &= \mathbf{A}\mathbf{K}_{\mathbf{U}} + (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}, \mathbf{W}_0^* = \mathbf{A}\mathbf{K}_{\mathbf{W}}, \mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}} \\
\mathbf{k}_{\mathbf{t}_1}, \ldots, \mathbf{k}_{\mathbf{t}_N} &\leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}^m \\
\mathbf{t}_1 &= \mathbf{A}\mathbf{k}_{\mathbf{t}_1}, \ldots, \mathbf{t}_N = \mathbf{A}\mathbf{k}_{\mathbf{t}_N}.
\end{aligned}$$

   In addition, algorithm $\mathcal{B}$ parses $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$ and $\mathbf{T}_{\mathsf{ct}} = \begin{bmatrix} \mathbf{T}_{\mathsf{in}} \\ \mathbf{T}_{\mathsf{fun}} \end{bmatrix}$. It sets $\mathbf{B} = \mathbf{U}_{\mathsf{ct}}\mathbf{T}_{\mathsf{fun}}$. If there exists any $i \neq j$ where $\mathbf{t}_i = \mathbf{t}_j$, then algorithm $\mathcal{B}$ aborts with output 0. Algorithm $\mathcal{B}$ also aborts with output 0 if $\|\mathbf{T}_0\| > \sqrt{m}\sigma_{\mathsf{crs}}$, $\|\mathbf{T}_{\mathbf{V}}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|$, or $\|\mathbf{R}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|$. If all checks pass, then $\mathcal{B}$ constructs the common reference string

$$\mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \mathbf{U}, \mathbf{U}_{\mathsf{ct}}, \mathbf{T}_{\mathsf{ct}}, \{\mathbf{t}_i, \mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}).$$

   Algorithm $\mathcal{B}$ gives crs to $\mathcal{A}$ and then initializes a counter $\mathsf{ctr} = 0$ as well as empty dictionaries $\mathsf{D}, \mathsf{D}_{\mathsf{sk}}$.

3. Whenever $\mathcal{A}$ makes a key-generation query on a pair (i, f_i), algorithm $\mathcal{B}$ increments the counter ctr = ctr + 1 and computes $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$. Algorithm $\mathcal{B}$ then samples

$$\begin{bmatrix} \mathbf{y}_{i,1,\mathrm{ctr}} \\ \vdots \\ \mathbf{y}_{i,N,\mathrm{ctr}} \\ \mathbf{d}_{i,\mathrm{ctr}} \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes \mathbf{t}_i, \sigma_{\mathrm{key}})$$

It sets $\mathbf{W}_{i,\mathrm{ctr}} = \mathbf{Z}(\mathbf{d}_{i,\mathrm{ctr}} \otimes \mathbf{I}_m)$, computes $\pi_{i,\mathrm{ctr}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathrm{ctr}}, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathrm{key}}))$, and sets $\mathsf{pk}_{\mathrm{ctr}} = (\mathbf{W}_{i,\mathrm{ctr}}, \left\{\mathbf{y}_{i,j,\mathrm{ctr}}\right\}_{j \neq i}, \pi_{i,\mathrm{ctr}})$. Algorithm $\mathcal{B}$ then checks that $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_{\mathrm{ctr}}) = 1$ and that $\|\mathbf{y}_{i,i,\mathrm{ctr}}\| \leq \beta_{\mathrm{key}}$. If either condition does not hold, then it aborts with output 0. Otherwise, algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with $(\mathrm{ctr}, \mathsf{pk}_{\mathrm{ctr}})$ to $\mathcal{A}$. Algorithm $\mathcal{B}$ also adds the mapping $\mathrm{ctr} \mapsto (i, f_i, \mathsf{pk}_{\mathrm{ctr}})$ to D and the mapping $(i, \mathbf{B}_{f_i}, \mathbf{W}_{i,\mathrm{ctr}}) \mapsto (0, \mathbf{y}_{i,i,\mathrm{ctr}})$ to D_{sk} (if such a mapping is not already present).

4. When $\mathcal{A}$ makes a query to the random oracle, if it is not the ind^th query, then $\mathcal{B}$ responds with a uniform random string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^{\rho}$. If it is the ind^th query ξ_{ind}, then algorithm $\mathcal{B}$ proceeds as follows.

   - First, it parses $\xi_{\mathsf{ind}} = ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_N, f_N))$, where $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$ and $f_i \in \mathcal{F}_{\tau}$. If ξ_{ind} does not have this form, then algorithm $\mathcal{B}$ aborts with output 0.
   - Algorithm $\mathcal{B}$ checks that for all $i \in [N]$, $\mathsf{IsValid}(\mathsf{crs}, i, f_i, \mathsf{pk}_i) = 1$. If not, it aborts with output 0.
   - For each $i \in [N]$, algorithm $\mathcal{B}$ computes $\mathbf{B}_{f_i} = \mathsf{EvalF}(\mathbf{B}, f_i)$. If $(i, \mathbf{B}_{f_i}, \mathbf{W}_i)$ is not contained in D_{sk}, then it computes $\mathbf{y}_i^* = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathcal{R}}, (\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathrm{key}}), \pi_i)$. Algorithm $\mathcal{B}$ aborts and outputs 0 if
   $$C_{\mathcal{R}}((\mathbf{A}, \mathbf{B}_{f_i}, \mathbf{W}_i, \mathbf{r}_i, \mathbf{t}_i, \mathbf{p}, \beta_{\mathrm{key}}), \mathbf{y}_i^*) = 0 \quad \text{or} \quad f_i(\mathbf{x}) = 0.$$
   If all conditions are satisfied, then algorithm $\mathcal{B}$ adds the mapping $(i, \mathbf{B}_{f_i}, \mathbf{W}_i) \mapsto (1, \mathbf{y}_i^*)$ to D_{sk}.
   - Now, for each $i \in [N]$, algorithm $\mathcal{B}$ constructs $\mathbf{y}_{0,i}$ as follows:
     - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (0, \mathbf{y}_i^*)$, it computes $\mathbf{y}_{0,i} = \mathbf{K}_{\mathbf{W}} \mathbf{r}_i - \sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}_i^* + \mathbf{k}_{\mathbf{t}_i}$.
     - If $\mathsf{D}_{\mathsf{sk}}[(i, \mathbf{B}_{f_i}, \mathbf{W}_i)] = (1, \mathbf{y}_i^*)$, it computes $\mathbf{y}_{0,i} = \mathbf{K}_{\mathbf{W}} \mathbf{r}_i - \sum_{j \neq i} \mathbf{y}_{j,i} - \mathbf{y}_i^* + \mathbf{K}_{\mathbf{B}}^{(i)} \mathbf{G}^{-1}(\mathbf{t}_i) + \mathbf{k}_{\mathbf{p}} + \mathbf{k}_{\mathbf{t}_i}$, where $\mathbf{K}_{\mathbf{B}}^{(i)} = \mathbf{K}_{\mathbf{U}} \mathbf{T}_{\mathsf{fun}} \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_m) \mathbf{T}_{\mathsf{in}} \mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}}$ and $\mathbf{H}_{\mathbf{B}, f_i, \mathbf{x}} = \mathsf{EvalFX}(\mathbf{B}, f_i, \mathbf{x})$.
   - Next, algorithm $\mathcal{B}$ sets $\mathbf{W}_0 = \mathbf{W}_0^* - \sum_{i \in [N]} \mathbf{W}_i$ and $\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathrm{vec}(\mathbf{W}_0), \sigma_{\mathrm{agg}})$. It sets
   $$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,1} \\ \vdots \\ \mathbf{y}_{0,N} \\ \mathbf{d}_0 \end{bmatrix}$$
   and computes $\gamma^* \leftarrow \mathsf{DGS.Explain}(1^{\lambda_{\mathsf{DGS}}}, 1^{p(\lambda)}, \mathbf{V}, \mathbf{T}_{\mathbf{V}}, \mathbf{0}^{nN}, \boldsymbol{\kappa}_0, \sigma_{\mathrm{agg}})$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with γ^*.

5. In the challenge phase, algorithm $\mathcal{A}$ specifies a tuple $(\mathsf{idx}_i, f_i^*, \mathsf{pk}_i^*)$ for each $i \in [N]$. The challenger parses $\mathsf{pk}_i^* = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}, \pi_i)$ and checks the following:

   - If $\mathsf{idx}_i \in [\mathrm{ctr}]$, then algorithm $\mathcal{B}$ looks up the entry $\mathsf{D}[\mathsf{idx}_i] = (i', f', \mathsf{pk}')$. If $i \neq i'$ or $f_i^* \neq f'$ or $\mathsf{pk}_i^* \neq \mathsf{pk}'$, then algorithm $\mathcal{B}$ aborts with output 0.
   - If $\mathsf{idx}_i = \perp$, algorithm $\mathcal{B}$ checks that $\mathsf{IsValid}(\mathsf{crs}, i, f_i^*, \mathsf{pk}_i^*) = 1$. If not, algorithm $\mathcal{B}$ outputs 0.

   Finally, algorithm $\mathcal{B}$ checks that $\mathcal{A}$ has made at least ind queries to the random oracle, and moreover, its ind^th query satisfies
   $$\xi_{\mathsf{ind}} = \left((\mathsf{pk}_1^*, f_1^*), \ldots, (\mathsf{pk}_N^*, f_N^*)\right).$$
   If not, algorithm $\mathcal{B}$ aborts with output 0. Otherwise, algorithm $\mathcal{B}$ sets $\mathbf{c}_1^{\mathsf{T}} = \mathbf{c}^{\mathsf{T}}$, $\mathbf{c}_2^{\mathsf{T}} = \mathbf{c}^{\mathsf{T}} \mathbf{K}_{\mathbf{W}}$, $\mathbf{c}_3^{\mathsf{T}} = \mathbf{c}^{\mathsf{T}} \mathbf{K}_{\mathbf{U}}$, and $c_4 = \mathbf{c}^{\mathsf{T}} \mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor$. It gives the challenge ciphertext $\mathsf{ct}^* = (\mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, \mathbf{c}_3^{\mathsf{T}}, c_4)$ to $\mathcal{A}$.

6. At the end of the experiment, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which $\mathcal{B}$ also outputs.

First, we argue that algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_{21,p}^{(b)}$ or $\mathsf{Hyb}_{22,p}^{(b)}$. By definition, The ℓ_0-succinct LWE challenger samples $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0)$ as

$$\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \quad \mathbf{U}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell_0 n \times m}, \quad \mathbf{T}_0 \leftarrow [\mathbf{I}_{\ell_0} \otimes \mathbf{A} \mid \mathbf{U}]^{-1}_{\sigma_{\mathsf{crs}}}(\mathbf{G}_{n\ell_0}).$$

This is exactly the specification in $\mathsf{Hyb}_{21,p}^{(b)}$ and $\mathsf{Hyb}_{22,p}^{(b)}$. We conclude that algorithm $\mathcal{B}$ perfectly simulates the setup phase, the key-generation phase, and the random oracle queries exactly as in $\mathsf{Hyb}_{21,p}^{(b)}$ or $\mathsf{Hyb}_{22,p}^{(b)}$. Consider now the distribution of the challenge ciphertext. We consider two possibilities:

- Suppose $\mathbf{c}^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}$ where $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{LWE}}}^m$. In this case, the challenge ciphertext is of the form

  $$\mathsf{ct}^* = (\mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}\,,\ \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{K}_{\mathbf{W}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}\,,\ \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{K}_{\mathbf{U}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{U}}\,,\ \mathbf{s}^{\mathsf{T}}\mathbf{A}\mathbf{k}_{\mathbf{p}} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor).$$

  Now, by definition,

  $$\begin{aligned} \mathbf{A}\mathbf{K}_{\mathbf{W}} &= \mathbf{W}_0^* = \left(\mathbf{W}_0 + \sum_{i \in [N]} \mathbf{W}_i\right) = \widehat{\mathbf{W}} \\ \mathbf{A}\mathbf{K}_{\mathbf{U}} &= \mathbf{U}_{\mathsf{ct}} - (\mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U} \\ \mathbf{A}\mathbf{k}_{\mathbf{p}} &= \mathbf{p}. \end{aligned}$$

  Thus, we can alternatively write the challenge ciphertext ct^* as

  $$\mathsf{ct}^* = (\mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}\,,\ \mathbf{s}^{\mathsf{T}}\widehat{\mathbf{W}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}\,,\ \mathbf{s}^{\mathsf{T}}(\mathbf{U}_{\mathsf{ct}} - \mathbf{x}^{\mathsf{T}} \otimes \mathbf{I}_n)\mathbf{U}\,,\ \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor).$$

  This is precisely the distribution in $\mathsf{Hyb}_{21,p}^{(b)}$.

- Suppose $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$. In this case, the challenge ciphertext

  $$\mathsf{ct}^* = (\mathbf{c}^{\mathsf{T}}\,,\ \mathbf{c}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}\,,\ \mathbf{c}^{\mathsf{T}}\mathbf{K}_{\mathbf{U}}\,,\ \mathbf{c}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor),$$

  This is precisely the distribution in $\mathsf{Hyb}_{22,p}^{(b)}$.

We conclude that $\mathcal{B}$ breaks the ℓ_0-succinct LWE assumption with the same advantage ε. □

Lemma 5.32. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} > \log m$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{22,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{23,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. This lemma follows by the same argument as in the proof of Lemma 5.29. □

Lemma 5.33. *Suppose $n \geq \lambda$, $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} > 2^\lambda(\beta_{\mathsf{key}} + m^{O(d)}\sigma_{\mathsf{crs}})$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{23,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{24,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. This lemma follows by the same argument as in the proof of Lemma 5.28. □

Lemma 5.34. *Suppose $n \geq \lambda$, $m \geq 2(n+1) \log q$, and $q > 2$ is a prime. For every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_{24,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{25,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. This follows via the leftover hash lemma (Lemma 3.3). Since $n \geq \lambda$, $m \geq 2(n+1) \log q$ and $q > 2$ is a prime, the following pair of distributions are statistically indistinguishable:

- Sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{c}_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m, \mathbf{k}_\mathbf{p} \xleftarrow{\mathrm{R}} \{0,1\}^m$ and output $(\mathbf{A}, \mathbf{c}_1, \mathbf{A}\mathbf{k}_\mathbf{p}, \mathbf{c}_1^\mathsf{T}\mathbf{k}_\mathbf{p} + b \cdot \lfloor q/2 \rfloor)$.
- Sample $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{c}_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m, \mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n, c_4 \xleftarrow{\mathrm{R}} \mathbb{Z}_q$ and output $(\mathbf{A}, \mathbf{c}_1, \mathbf{p}, c_4 + b \cdot \lfloor q/2 \rfloor)$.

The first distribution corresponds to the distribution of $(\mathbf{A}, \mathbf{c}_1, \mathbf{p}, c_4)$ in $\mathsf{Hyb}_{24,p}^{(b)}$ while the second distribution corresponds to that in $\mathsf{Hyb}_{25,p}^{(b)}$. □

Lemma 5.35. *For every polynomial p and all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Hyb}_{25,p}^{(0)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_{25,p}^{(1)}(\mathcal{A}) = 1]$.*

Proof. This is immediate since the challenger's behavior in $\mathsf{Hyb}_{25,p}^{(b)}$ is independent of the challenge bit b. □

Proof of Theorem 5.9. To complete the proof of Theorem 5.9, suppose algorithm $\mathcal{A}$ wins the attribute-selective security game with non-negligible advantage ε. Then

$$|\Pr[\mathsf{Hyb}_0^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0^{(1)}(\mathcal{A}) = 1]| = \varepsilon(\lambda).$$

Let Q_{ro} be a bound on the number of random oracle queries algorithm $\mathcal{A}$ makes. By Lemma 5.10, this means

$$|\Pr[\mathsf{Hyb}_1^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1]| = \frac{\varepsilon(\lambda)}{Q_{\mathrm{ro}}}. \tag{5.14}$$

Since ε is non-negligible and $Q_{\mathrm{ro}} = \mathrm{poly}(\lambda)$, this means $\varepsilon(\lambda)/Q_{\mathrm{ro}}$ from Eq. (5.14) is also non-negligible. Thus, there exists a polynomial p' such that for infinitely many $\lambda \in \mathbb{N}$, it holds that $\varepsilon(\lambda)/Q_{\mathrm{ro}} \geq 1/p'(\lambda)$. Let $p(\lambda) = 3p'(\lambda)$. By Lemmas 5.11 to 5.35, we have for all $\lambda \in \mathbb{N}$ (and recalling that for $i \leq 11$, $\mathsf{Hyb}_{i,p}^{(b)}(\mathcal{A}) \equiv \mathsf{Hyb}_i^{(b)}(\mathcal{A})$),

$$\begin{aligned} |\Pr[\mathsf{Hyb}_1^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1]| \leq{} & \sum_{i=1}^{24} |\Pr[\mathsf{Hyb}_{i,p}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i+1,p}^{(0)}(\mathcal{A}) = 1]| \\ & + |\Pr[\mathsf{Hyb}_{25}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{25}^{(1)}(\mathcal{A}) = 1]| \\ & + \sum_{i=1}^{24} |\Pr[\mathsf{Hyb}_{i+1,p}^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i,p}^{(1)}(\mathcal{A}) = 1]| \\ \leq{} & 2/p(\lambda) + \delta(\lambda), \end{aligned}$$

where $\delta(\lambda) = \mathrm{negl}(\lambda)$ is a negligible function. Combined with Eq. (5.14), this means for all $\lambda \in \mathbb{N}$,

$$\frac{\varepsilon(\lambda)}{Q_{\mathrm{ro}}} \leq \frac{2}{p(\lambda)} + \delta(\lambda).$$

Now, by assumption, there are infinitely many $\lambda \in \mathbb{N}$ where

$$\frac{\varepsilon(\lambda)}{Q_{\mathrm{ro}}} \geq \frac{1}{p'(\lambda)} = \frac{3}{p(\lambda)},$$

which means $\delta(\lambda) > 1/p(\lambda)$ for infinitely-many $\lambda \in \mathbb{N}$. This contradicts the fact that δ is negligible. Therefore Construction 5.6 is attribute-selective secure without corruptions. □

Parameter instantiation. Let λ be a security parameter, N be a bound on the number of users, and τ be a policy parameter. We can instantiate the lattice parameters in Construction 5.6 to satisfy Theorems 5.7 to 5.9:

- We write $d = d(\tau)$, $\ell = \ell(\tau)$ and set the lattice dimension $n = (\lambda d \log \ell \log N)^{1/\varepsilon}$ for some constant $\varepsilon \in (0,1)$. We set $\lambda_{\mathsf{DGS}} = \tilde{O}(\lambda d \log \ell \log N)$, where $\tilde{O}$ suppresses $\mathrm{poly}(\log \lambda, \log d, \log \log N, \log \log \ell)$ factors. We assume $\lambda_{\mathsf{DGS}} \geq \log q$ in the following. We set $m = 3n \log q$. In the following, it will be the case that $\log q = \tilde{O}(n^\varepsilon)$. Therefore, $\log m \leq \tilde{O}(1)$ and $\log \ell_0 = \log \max(\ell, Nm) \leq \tilde{O}(\log \ell \log N)$.

- We upper bound $\sigma_{\text{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q)$ by $\tilde{O}({\ell_0}^2 m^3 \lambda_{\mathsf{DGS}}^2)$.
- We set $\sigma_{\mathsf{LWE}} = \mathsf{poly}(\lambda)$, $\sigma_{\text{crs}} = O({\ell_0}^2 m^2)$, $\sigma_{\text{key}} = O({\ell_0}^3 m^5) \cdot \sigma_{\text{crs}} = O({\ell_0}^5 m^7)$, $\beta_{\text{key}} = O(m) \cdot \sigma_{\text{key}} = O({\ell_0}^5 m^8)$, $\sigma_{\text{agg}} = 2^\lambda \cdot m^{O(d)} \cdot \tilde{O}({\ell_0}^5 \lambda_{\mathsf{DGS}}^2)$, $\beta_{\text{agg}} = O(m) \cdot \sigma_{\text{agg}} = 2^\lambda \cdot m^{O(d)} \cdot \tilde{O}({\ell_0}^5 \lambda_{\mathsf{DGS}}^2)$.
- We choose a prime modulus

  $$q = 2^\lambda m^{O(d)} \cdot \tilde{O}({\ell_0}^6 \lambda_{\mathsf{DGS}}^2) \cdot \mathsf{poly}(\lambda) = 2^{\tilde{O}(\lambda d \log m \log \ell_0 \log \lambda_{\mathsf{DGS}})} \leq 2^{\tilde{O}(\lambda d \log N \log \ell \log \lambda_{\mathsf{DGS}})} \leq 2^{\tilde{O}(\lambda_{\mathsf{DGS}})} = 2^{\tilde{O}(n^\varepsilon)}.$$

  Note that we can always set appropriate constant factors such that $\lambda_{\mathsf{DGS}} = \tilde{O}(\lambda d \log \ell \log N)$ and $\lambda_{\mathsf{DGS}} \geq \log q = \tilde{O}(\lambda d \log \ell \log N) \cdot \mathsf{polylog}(\lambda_{\mathsf{DGS}})$.

We now affirm that the parameter settings are sufficient for the construction. In particular,

- Theorem 4.2 gives an explainable discrete Gaussian preimage sampler with $\sigma_{\text{loss}} = O(m^{3/2} \log(m\lambda) \log \log q)$. Since all invocations of Π_{DGS} in the construction use security parameter λ_{DGS} and matrix $\mathbf{V} \in \mathbb{Z}_q^{nN \times (mN+k)}$, we can upper bound σ_{loss} by $O((mN + mn \log q)^{3/2} \log(m\lambda_{\mathsf{DGS}}) \log \log q) \leq \tilde{O}({\ell_0}^2 m^3 \lambda_{\mathsf{DGS}}^2)$
- We assume hardness of ℓ_0-succinct LWE with parameters $(n, m, q, \sigma_{\mathsf{LWE}}, \sigma_{\text{crs}})$, where $m \geq 2n \log q$ and $q = 2^{\tilde{O}(n^\varepsilon)}$. In particular, this corresponds to ℓ_0-succinct LWE with a sub-exponential modulus-to-noise ratio.
- The completeness requirements from Theorem 5.7 are satisfied since
  - $n \geq \lambda$, $m = 3n \log q$.
  - $\sigma_{\text{crs}} = O({\ell_0}^2 m^2)$, $\beta_{\text{key}} = O({\ell_0}^3 m^6) \cdot \sigma_{\text{crs}} > O({\ell_0}^2 m^3) \cdot \sigma_{\text{crs}}$.
- The correctness requirements from Theorem 5.8 are satisfied since
  - $n \geq \lambda$, $m = 3n \log q \geq 2n \log q$.
  - $\sigma_{\text{crs}} = O({\ell_0}^2 m^2)$, $\beta_{\text{key}} = O({\ell_0}^3 m^6) \cdot \sigma_{\text{crs}} > O({\ell_0}^2 m^3) \cdot \sigma_{\text{crs}}$.
  - $q = 2^\lambda m^{O(d)} \cdot \tilde{O}({\ell_0}^6 \lambda_{\mathsf{DGS}}^2) \cdot \mathsf{poly}(\lambda) \geq m^{O(d)} \cdot O({\ell_0}^2) \cdot O({\ell_0}^2 m^2) \cdot \mathsf{poly}(\lambda) \geq m^{O(d)} \cdot O({\ell_0}^2) \sigma_{\text{crs}} \sigma_{\mathsf{LWE}}$
  - $q = 2^\lambda m^{O(d)} \cdot \tilde{O}({\ell_0}^6 \lambda_{\mathsf{DGS}}^2) \cdot \mathsf{poly}(\lambda) \geq O(m^{3/2}) \cdot \mathsf{poly}(\lambda) \cdot (O(\ell_0) \cdot O({\ell_0}^5 m^8) + 2^\lambda \cdot m^{O(d)} \cdot \tilde{O}({\ell_0}^5 \lambda_{\mathsf{DGS}}^2)) \geq O(m^{3/2} \cdot \sigma_{\mathsf{LWE}}(N\beta_{\text{key}} + \beta_{\text{agg}}))$.
- The security requirement for Theorem 5.9 is satisfied since
  - $n \geq \lambda$, $m = 3n \log q$, $q > 2$.
  - $\sigma_{\text{crs}} = O({\ell_0}^2 m^2)$, $\sigma_{\text{key}} = O({\ell_0}^3 m^5) \cdot \sigma_{\text{crs}}$, $\beta_{\text{key}} = O(m) \cdot \sigma_{\text{key}}$, $\beta_{\text{agg}} = O(m) \cdot \sigma_{\text{agg}}$.
  - $\sigma_{\text{agg}} = 2^\lambda \cdot m^{O(d)} \cdot \tilde{O}({\ell_0}^5 \lambda_{\mathsf{DGS}}^2) \geq 2^\lambda O({\ell_0}^5 m^8) + 2^\lambda m^{O(d)} \cdot O({\ell_0}^2 m^2) \geq 2^\lambda (\beta_{\text{key}} + m^{O(d)} \sigma_{\text{crs}})$.
  - $\sigma_{\text{agg}} = 2^\lambda \cdot m^{O(d)} \cdot \tilde{O}({\ell_0}^5 \lambda_{\mathsf{DGS}}^2) \geq O(\ell_0 m^{5/2}) \cdot O({\ell_0}^2 m^2) \cdot \tilde{O}({\ell_0}^2 m^3 \lambda_{\mathsf{DGS}}^2) \geq O(\ell_0 m^{5/2}) \cdot \sigma_{\text{crs}} \cdot \sigma_{\text{loss}}$.
  - $2^{\lambda_{\mathsf{DGS}}} \geq q \geq \beta_{\text{agg}} > \sigma_{\text{agg}}$.

With this setting of parameters, we obtain a slotted key-policy registered ABE scheme with the following parameter sizes:

- **Common reference string size:** The common reference string

  $$\mathsf{crs} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathbf{A}, \mathbf{p}, \mathbf{U}, \mathbf{U}_{\text{ct}}, \mathbf{T}_{\text{ct}}, \{\mathbf{t}_i\}_{i \in [N]}, \{\mathbf{r}_i\}_{i \in [N]}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$$

  consists of the following components:
  - Matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{p} \in \mathbb{Z}_q^n$, with overall size $(n+1) m \log q = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.

- Homomorphic computation components $\mathbf{U} \in \mathbb{Z}_q^{\ell n \times m}$, $\mathbf{U}_{\mathsf{ct}} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{T}_{\mathsf{ct}} \in \mathbb{Z}_q^{(\ell+1)m \times \ell m'}$, and $\mathbf{t}_1, \ldots, \mathbf{t}_N \in \mathbb{Z}_q^n$, with overall size is bounded by

$$(\ell^2 + N) \cdot \mathsf{poly}(n, m, \log q) = (\ell^2 + N)\mathsf{poly}(\lambda, d, \log \ell, \log N).$$

- Key-generation components $\mathbf{r}_1, \ldots, \mathbf{r}_N \in \mathbb{Z}_q^m$, $\mathbf{V} \in \mathbb{Z}_q^{nN \times (mN+k)}$, $\mathbf{Z} \in \mathbb{Z}_q^{n \times mk}$, $\mathbf{T}_{\mathbf{V}} \in \mathbb{Z}_q^{(mN+k) \times m'N}$, $\mathbf{T}_{\tilde{\mathbf{Z}}} \in \mathbb{Z}_q^{nm \times mm'}$, where $k = 3nm \lceil \log q \rceil$. The overall size is bounded by

$$N^2 \mathsf{poly}(n, m, \log q) = N^2 \mathsf{poly}(\lambda, d, \log \ell, \log N).$$

- The remaining component $\mathsf{crs}_{\mathsf{NIZK}}$ has size $\mathsf{poly}(\lambda)$.

Thus the common reference string has size $|\mathsf{crs}| = (\ell^2 + N^2)\mathsf{poly}(\lambda, d, \log \ell, \log N)$.

- **Public key size:** Each user's public key pk consists of a matrix $\mathbf{W} \in \mathbb{Z}_q^{n \times m}$, $N - 1$ cross-terms $\mathbf{y}_j \in \mathbb{Z}_q^m$, and a proof π. Thus $|\mathsf{pk}| \leq (n + N)m \log q + \mathsf{poly}(n, m, \log q) = N \cdot \mathsf{poly}(\lambda, d, \log \ell, \log N)$.
- **Secret key size:** The secret key for user $i \in [N]$ consists of a vector $\mathbf{y}_i \in \mathbb{Z}_q^m$, so $|\mathsf{sk}_i| = O(m \log q) = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.
- **Master public key size:** The aggregated master public key mpk consists of a matrix $\widehat{\mathbf{W}} \in \mathbb{Z}_q^{n \times m}$ of size $nm \log q = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.
- **Helper decryption key size:** The aggregated helper decryption key for user $i \in [N]$ is a vector $\mathsf{hsk}_i \in \mathbb{Z}_q^m$, so $|\mathsf{hsk}_i| = O(m \log q) = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.
- **Ciphertext size:** The ciphertext consists of three vectors in $\mathbb{Z}_q^m$ and one $\mathbb{Z}_q$ element, with overall size $(3m + 1) \log q = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.

Putting everything together, we obtain the following corollary:

Corollary 5.36 (Slotted Key-Policy Registered ABE). *Let λ be a security parameter and $N = N(\lambda)$ be any polynomial. Let $\mathcal{F}$ be a family of decryption policies on attributes of length $\ell = \ell(\lambda)$ that can be computed by a Boolean circuit of depth at most $d = d(\lambda)$. Let $\ell_0 \geq \max(\ell, N \cdot \mathsf{poly}(\lambda, \log N))$ Then, assuming polynomial hardness of the ℓ_0-succinct LWE assumption with a sub-exponential modulus-to-noise ratio, there exists a slotted key-policy registered ABE scheme that supports up to N users and policy family $\mathcal{F}$ in the random oracle model. The scheme satisfies attribute-selective security without corruptions with the following efficiency properties:*

- *The size of the common reference string is $|\mathsf{crs}| = (\ell^2 + N^2)\mathsf{poly}(\lambda, d, \log \ell, \log N)$.*
- *The size of each user's public key is $|\mathsf{pk}_i| = N \cdot \mathsf{poly}(\lambda, d, \log \ell, \log N)$.*
- *The size of each user's secret key is $|\mathsf{sk}_i| = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.*
- *The size of the master public key is $|\mathsf{mpk}| = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.*
- *The size of the helper decryption key for each user is $|\mathsf{hsk}_i| = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.*
- *The size of the ciphertext is $|\mathsf{ct}| = \mathsf{poly}(\lambda, d, \log \ell, \log N)$.*

Remark 5.37 (Key-Policy Registered ABE in the Random Oracle Model)**.** As discussed in Remark 5.5, we can apply the results from [FWW23] to transform an attribute-selective slotted registered ABE scheme that does not support corruptions into an attribute-selective construction that does support corruptions in the random oracle model. The construction incurs constant overhead. Furthermore, as discussed in Theorem A.6, we can apply results from [HLWW23] to transform a slotted registered ABE scheme (with a long CRS) into a bounded registered ABE scheme with $\log N$ overhead. Therefore, combined with Corollary 5.36, we obtain an attribute-selective key-policy registered ABE scheme for general (bounded-depth) Boolean circuit policies that supports an a priori bounded number of users N. With complexity leveraging, we also obtain an adaptively-secure scheme. In this case, the parameters (notably, the ciphertext size) scale with the attribute length $|\mathbf{x}|$.

Remark 5.38 (Identity-Based Distributed Broadcast Encryption)**.** The works of [AY20, AWY20] show that an ABE scheme supporting log-depth policies with succinct ciphertexts directly immediately implies an identity-based broadcast encryption scheme. In identity-based broadcast encryption, each user has an identity (e.g., a username), and ciphertexts can be encrypted to an arbitrary set of identities. This generalizes standard broadcast encryption [FN93] which assumes that the user identities are the integers $1, 2, \ldots, N$. In particular, suppose we have either

- a key-policy ABE scheme where the ciphertext size is sublinear in the length of the attribute $\mathbf{x}$ (i.e., $|\mathsf{ct}| \leq o(|\mathbf{x}|)$) and which supports membership policies (i.e., on input a set S, the policy $P_y(S) = 1$ if and only if $y \in S$); or
- a ciphertext-policy ABE scheme where the ciphertext size is sublinear in the size of the policy P (i.e., $|\mathsf{ct}| \leq o(|P|)$) and which supports which supports membership policies (i.e., on input an element y, the policy $P_S(y) = 1$ if and only if $y \in S$).

Then, we one can construct an identity-based broadcast encryption scheme by setting the secret-key of user i to the ABE secret key for policy P_i (resp., the secret key for attribute i), and setting the ciphertext for a set S to be an ABE ciphertext with attribute S (resp., a ciphertext for the policy P_S). Furthermore, the broadcast encryption scheme is selectively secure if the underlying ABE scheme satisfies attribute-selective security (resp., policy-selective security). The scheme is adaptively secure if the underlying ABE scheme is adaptively secure.

This implication directly extends to the setting of *distributed* broadcast encryption [WQZDF10, BZ14] where each user chooses their own public keys (see Section 6 for a formal definition). In this setting, users choose their own public/private keys (just like in registered ABE) and post their public keys to a public-key directory. Afterwards, anyone can encrypt a message to an arbitrary set of public keys with a ciphertext whose size scales sublinearly with the size of the set. In the standard notion of distributed broadcast encryption (and in all existing constructions [WQZDF10, BZ14, KMW23, FWW23, CW24]), the encrypter and the decrypter needs to know the public keys of each user in the broadcast set in order to encrypt or decrypt, respectively.

In identity-based distributed broadcast encryption, the encrypter (and decrypter) only needs to know the *identities* of the users in the broadcast set (e.g., their usernames or email addresses) rather than their specific public keys. Notably, identity-based broadcast encryption eliminates the need to separately lookup user public keys at encryption or decryption time. It is straightforward to adapt the [AY20, AWY20] approach to obtain an identity-based distributed broadcast encryption scheme from any registered ABE scheme with succinct ciphertexts. For simplicity, we just sketch the construction assuming a key-policy registered ABE scheme, but a similar approach works starting from a ciphertext-policy registered ABE scheme.

- **Key-generation:** Each user samples their own public/private key for the underlying registered ABE scheme where the policy is tied to their identity (e.g., the key for the identity id is the function $P_{\mathsf{id}}(S)$ that takes as input a set S and outputs 1 if $\mathsf{id} \in S$ and 0 otherwise).
- **Aggregation:** The key-curator aggregate all of the users' public keys into a single (short) master public key. It gives helper decryption keys to each of the registered users.
- **Encryption:** To encrypt a message to a set of identities S, the encrypter only needs to know the master public key for the scheme and the set S. The encrypter constructs a registered ABE ciphertext that encrypts the message with attribute S. If the registered ABE scheme has succinct ciphertexts, then the size of the ciphertext is sublinear in the size of the attribute (i.e., the size of the broadcast set S) as well as the total number of users N in the system. Note that the encrypter only needs to know the recipient set S, but *not* the individual public keys for the users in S.
- **Decryption:** To decrypt a message associated with a set of identities S, the decrypter just applies the decryption procedure for registered ABE. Similar to the case with encryption, the decrypter only needs to know the recipient set S, but *not* the individual public keys for the users in S.

Furthermore, the identity-based distributed broadcast encryption scheme supports unbounded number of users if the underlying registered ABE scheme is unbounded.

Combining Corollary 5.36 and Remark 5.37, we thus obtain the following corollary:

Corollary 5.39 (Identity-Based Distributed Broadcast Encryption)**.** *Let λ be a security parameter. Take any polynomial $N = N(\lambda)$. Then, under the ℓ-succinct LWE assumption (where $\ell \geq N \cdot \mathsf{poly}(\lambda, \log N)$) with a sub-exponential modulus-to-noise ratio, there exists a selectively-secure identity-based distributed broadcast encryption scheme that supports up to N users in the random oracle model.*

Remark 5.40 (Very-Selective Security)**.** As discussed at the end of Section 1.1, the recent work of [AMR25] show how to construct a registered ABE scheme without random oracles and which achieves a notion called "very-selective" security where the adversary has to commit to the randomness of its key-generation queries (as well as the challenge attribute) ahead of time. While our focus in this work is on the standard security definition of registered ABE which allows the adversary to register its own keys, for sake of comparison, we note here that it is straightforward to achieve very-selective security in the *plain* model with a small tweak to Construction 5.6.

Since the adversary in the very-selective security commits to all of the keys in advance, we can simply include the re-randomization components $(\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{d}_0)$ in the CRS instead of generating it at aggregation time. As such, we no longer require the random oracle to compress this tuple. In this setting, we no longer need the NIZK proofs of well-formedness or the explainable sampling procedure (we just publish the sample $(\mathbf{y}_{0,1}, \ldots, \mathbf{y}_{0,N}, \mathbf{d}_0)$ in the CRS).

6 Adaptively-Secure Distributed Broadcast Encryption

In this section, we show that the same re-randomization technique we used to construct registered ABE can be used to construct an *adaptively-secure* (distributed) broadcast encryption scheme from the ℓ-succinct LWE assumption in the random oracle model. We do this by first constructing a distributed broadcast encryption (DBE) scheme that is semi-statically secure from the ℓ-succinct assumption in the random oracle model. Then, using existing transformations [GW09, KMW23], we can generically compile the semi-statically-secure scheme into an adaptively-secure construction (with only constant overhead). While this transformation only works in the random oracle model, our base broadcast encryption scheme is already in the random oracle model, so there is essentially no additional cost for realizing adaptive security. Previously, the only construction of adaptively-secure (distributed) broadcast encryption relied on witness encryption in the random oracle model [FWW23]. All other lattice-based constructions of (centralized or distributed) broadcast encryption [BV22, Wee22, Wee24, CW24] only achieved selective security. Note that an (adaptively-secure) distributed broadcast encryption directly implies an (adaptively-secure) centralized broadcast encryption scheme (with linear-size public parameters); namely, the public key for the centralized broadcast encryption scheme can simply be a list of the public keys for each user.

Distributed broadcast encryption. We now give the formal definition of distributed broadcast encryption from [BZ14, KMW23]. We define both adaptive security and semi-static security [GW09, KMW23].

Definition 6.1 (Distributed Broadcast Encryption [BZ14, KMW23])**.** Let λ be the security parameter and N be the number of users. An N-user distributed broadcast encryption scheme Π_{DBE} is a tuple of efficient algorithms $\Pi_{\mathsf{DBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^N) \to \mathsf{pp}$: On input the security parameter λ and the number of users N, the setup algorithm outputs the public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}, i) \to (\mathsf{pk}_i, \mathsf{sk}_i)$: On input the public parameters pp and an index $i \in [N]$, the key-generation algorithm outputs a public key and secret key $(\mathsf{pk}_i, \mathsf{sk}_i)$.
- $\mathsf{IsValid}(\mathsf{pp}, i, \mathsf{pk}_i) \to b$: On input the public parameters pp, an index $i \in [N]$, and a public key pk_i, the validity-checking algorithm outputs a bit $b \in \{0, 1\}$.
- $\mathsf{Encrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mu) \to \mathsf{ct}$: On input the public parameters pp, a collection of public keys pk_j and a message $\mu \in \{0, 1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mathsf{ct}, (i, \mathsf{sk}_i)) \to \mu$: On input the public parameters pp, a collection of public keys pk_j, 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:

- **Completeness:** For all $\lambda, N \in \mathbb{N}$, and all indices $i \in [N]$, it holds that

$$\Pr\left[\mathsf{IsValid}(\mathsf{pp}, i, \mathsf{pk}_i) = 1 : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N) \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i) \end{array}\right] = 1.$$

- **Correctness:** We say Π_{DBE} is correct if for all $\lambda, N \in \mathbb{N}$, all indices $i \in [N]$, all sets $S \subseteq [N]$ where $i \in S$, all pp in the support of $\mathsf{Setup}(1^\lambda, 1^N)$, all $(\mathsf{pk}_i, \mathsf{sk}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, i)$, all collections of tuples $\{(j, \mathsf{pk}_j)\}_{j \in S \setminus \{i\}}$ where $\mathsf{IsValid}(\mathsf{pp}, j, \mathsf{pk}_j) = 1$, and all messages $\mu \in \{0, 1\}$, we have

$$\Pr[\mathsf{Decrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mathsf{ct}, (i, \mathsf{sk}_i)) = \mu : \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mu)] = 1.$$

- **Adaptive security:** For a security parameter λ, a bound N on the number of users, and a bit $b \in \{0, 1\}$, we define the adaptive security game between an adversary $\mathcal{A}$ and a challenger as follows:
  - **Setup phase:** The challenger begins by sampling $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$. and gives the security parameter 1^λ, the number of users 1^N, and the public parameters pp to $\mathcal{A}$. The challenger also initializes an empty dictionary D and a counter $\mathsf{ctr} = 0$ for keeping track of key-generation queries as well as an empty set C for keeping track of corrupted keys.
  - **Query phase:** Algorithm $\mathcal{A}$ can now make the following queries:
    - **Key-generation query:** On input a slot index $i \in [N]$, the challenger samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$ and replies to $\mathcal{A}$ with pk_i. The challenger also increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$ and adds the mapping $\mathsf{ctr} \mapsto (i, \mathsf{pk}_i, \mathsf{sk}_i)$ to $\mathcal{A}$.
    - **Corruption query:** On input an index $i \in [\mathsf{ctr}]$, the challenger responds with $\mathsf{D}[\mathsf{ctr}]$. The challenger also adds ctr to C.
  - **Challenge phase:** After $\mathcal{A}$ is done making queries, it specifies a challenge set $S \subseteq [\mathsf{ctr}]$. The challenger checks that $S \cap C = \varnothing$ (i.e., that the adversary did not corrupt any index in the challenge set). For each $j \in S$, let $\mathsf{D}[j] = (i_j, \mathsf{pk}_j, \mathsf{sk}_j)$. The challenger additionally checks that the indices i_j are all distinct (i.e., at most one public key associated with each slot). If either check fails, the challenger halts with output 0. Otherwise, the challenger computes $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(i_j, \mathsf{pk}_j)\}_{j \in S}, b)$ and sends ct_b to $\mathcal{A}$.
  - **Output phase:** At the end of the game, algorithm $\mathcal{A}$ outputs $b' \in \{0, 1\}$, which is the output of the experiment.

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

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda) \tag{6.1}$$

  in the adaptive security game. We say that the scheme is adaptively secure for up to N users if Eq. (6.1) holds for the specific value of N.

- **Succinctness:** There exists a fixed polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda, N \in \mathbb{N}$, all subsets $S \subseteq [N]$, all public parameters pp in the support of $\mathsf{Setup}(1^\lambda, 1^N)$, all key-pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, i)$ for $i \in S$, 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 $|\mathsf{ct}| \leq \mathsf{poly}(\lambda + \log N)$.

Definition 6.2 (Semi-Static Security)**.** Let $\Pi_{\mathsf{DBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a distributed broadcast encryption scheme. For a security parameter λ, a bound N on the number of users, and a bit $b \in \{0, 1\}$, we define the semi-static security game between an adversary $\mathcal{A}$ and a challenger as follows:

- **Setup phase:** On input the security parameter 1^λ and the number of users 1^N, the adversary outputs a set $S^* \subseteq [N]$.

- **Key-generation phase:** The challenger samples $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$. Then for each $i \in S^*$, the challenger samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$. It gives $\big(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S^*}\big)$ to $\mathcal{A}$.
- **Challenge phase:** Algorithm $\mathcal{A}$ chooses a challenge set $S \subseteq S^*$. The challenger replies with the challenge ciphertext $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, b)$.
- **Output phase:** At the end of the game, algorithm $\mathcal{A}$ outputs $b' \in \{0,1\}$, which is the output of the experiment.

We say the distributed broadcast encryption scheme is semi-statically secure if for all polynomials $N = N(\lambda)$, and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| = \mathsf{negl}(\lambda) \tag{6.2}$$

in the semi-static security game. We say that the scheme is semi-statically secure for up to N users if Eq. (6.2) holds for the specific value of N.

Theorem 6.3 (Semi-Static to Adaptive Security [GW09, KMW23])**.** *Suppose Π_{DBE} is a distributed broadcast encryption scheme that satisfies semi-static security. Then, there exists an adaptively-secure distributed broadcast encryption scheme Π'_{DBE} that satisfies adaptive security in the random oracle model.*

6.1 Semi-Statically-Secure Distributed Broadcast Encryption from Lattices

In this section, we give our construction of a semi-static secure distributed broadcast encryption scheme from ℓ-succinct LWE in the random oracle model, where $\ell = N \cdot \mathsf{poly}(\lambda, \log N)$, λ is a security parameter, and N is a bound on the number of users. We use the same re-randomization technique as in our registered ABE scheme from Section 5 and Construction 5.6. In distributed broadcast encryption, there is no aggregation algorithm, so the re-randomization of the "aggregated" key happens at *encryption* time instead. In addition, in distributed broadcast encryption, the challenge ciphertext is always encrypted to a set of honestly-generated public keys, so we no longer need to extract the secret keys for adversarially-chosen public keys in the security reduction. This means we do not need to include NIZK proofs of knowledge in the base scheme.

Construction 6.4 (Distributed Broadcast Encryption)**.** Let $\lambda \in \mathbb{N}$ be a security parameter, $N \in \mathbb{N}$ be the number of users, and n, m, q be lattice parameters. Let $\sigma_{\mathsf{pp}}, \sigma_{\mathsf{key}}, \sigma_{\mathsf{agg}}, \sigma_{\mathsf{LWE}}$ be Gaussian width parameters and $\beta_{\mathsf{key}}, \beta_{\mathsf{agg}}$ be norm bounds. Let $k = 3nm \log q$, $m' = n \lceil \log q \rceil$, and $\ell_0 = Nm'$ be fixed dimensions. All the above parameters are functions of λ and N. Our construction relies on the following additional primitives:

- Let $\Pi_{\mathsf{DGS}} = (\mathsf{DGS.SamplePre}, \mathsf{DGS.Explain})$ be a $(\rho', \sigma_{\mathsf{loss}})$-explainable discrete Gaussian preimage sampler. Let $\lambda_{\mathsf{DGS}}(\lambda, N)$ be the security parameter for the sampler. Additionally, let $\rho = \rho(\lambda_{\mathsf{DGS}}, \lambda, N)$ be a randomness length that upper-bounds ρ' for all sampler instances in the construction.
- Let $\big\{H_\rho \colon \{0,1\}^* \to \{0,1\}^\lambda\big\}_{\lambda \in \mathbb{N}}$ be a family of hash functions which we model as a random oracle in the security analysis.

We construct our distributed broadcast encryption scheme $\Pi_{\mathsf{DBE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda, 1^N)$: On input the security parameter λ and the bound on the number of users N, the setup algorithm proceeds as follows:
  1. Sample trapdoor $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{pp}})$.
  2. Compute the trapdoor $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N)$ using Lemma 4.7, where

$$\mathbf{V} = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] = [\mathbf{I}_N \otimes \mathbf{A} \mid \mathbf{M}_{\mathbf{Z},\mathbf{R}}] \in \mathbb{Z}_q^{nN \times (mN+k)}. \tag{6.3}$$

3. Sample vectors $\mathbf{p}, \mathbf{t}_1, \ldots, \mathbf{t}_N \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$.

Output $\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$ where $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$.

- $\mathsf{KeyGen}(\mathsf{pp}, i)$: On input the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$ and an index $i \in [N]$, the key-generation algorithm samples

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{d} \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes \mathbf{p}, \sigma_{\mathsf{key}}), \tag{6.4}$$

where $\boldsymbol{\eta}_i \in \{0,1\}^N$ is the the i^{th} standard basis vector, $\mathbf{y}_i \in \mathbb{Z}^m$ for each $i \in [N]$, and $\mathbf{d} \in \mathbb{Z}^k$. If $\|\mathbf{y}_i\| > \beta_{\mathsf{key}}$ for any $i \in [N]$, it sets

$$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{d} \end{bmatrix} = \mathbf{T}_{\mathbf{V}} \cdot \mathbf{G}_{nN}^{-1}(\boldsymbol{\eta}_i \otimes \mathbf{p}).$$

Finally, it sets $\mathbf{W} = \mathbf{Z}(\mathbf{d} \otimes \mathbf{I}_m) \in \mathbb{Z}_q^{n \times m}$ and outputs the public key $\mathsf{pk} = (\mathbf{W}, \{\mathbf{y}_j\}_{j \neq i})$ and the secret key $\mathsf{sk} = \mathbf{y}_i$.

- $\mathsf{IsValid}(\mathsf{pp}, i, \mathsf{pk}_i)$: On input the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$, an index $i \in [N]$, and a public key $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i})$, the validity-checking algorithm outputs 1 if the following holds:

$$\forall j \neq i : \mathbf{A}\mathbf{y}_{i,j} = \mathbf{W}_i \mathbf{r}_j \quad \text{and} \quad \|\mathbf{y}_{i,j}\| \leq \beta_{\mathsf{key}}.$$

Otherwise, the algorithm outputs 0.

- $\mathsf{Encrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mu)$: On input the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$, a collection of public keys $\mathsf{pk}_j = (\mathbf{W}_j, \{\mathbf{y}_{j,j'}\}_{j' \neq j})$ for each $j \in S = \{i_1, \ldots, i_{|S|}\}$, and a message $\mu \in \{0,1\}$, the encryption algorithm samples

$$\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n \,,\; \mathbf{e} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{LWE}}}^m \,,\; \xi \xleftarrow{\mathrm{R}} \{0,1\}^\lambda \,,\; \mathbf{K}_{\mathbf{W}} \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m} \,,\; \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^m.$$

If $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$, it sets $\mathbf{e} = \mathbf{0}^m$ instead. It computes $(\mathbf{M}_S, \mathbf{T}_S) \leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{M}_{\mathbf{Z},\mathbf{R}}, \mathbf{T}_{\mathbf{V}}, S)$, sets $\mathbf{V}_S = [\mathbf{I}_{|S|} \otimes \mathbf{A} \mid \mathbf{M}_S]$, and samples

$$\begin{bmatrix} \mathbf{y}_{0,i_1} \\ \vdots \\ \mathbf{y}_{0,i_{|S|}} \\ \mathbf{d}_0 \end{bmatrix} \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; H_\rho(\xi)), \tag{6.5}$$

where $\mathbf{y}_{0,j} \in \mathbb{Z}^m$ for each $j \in S$, $\mathbf{d}_0 \in \mathbb{Z}^k$. If $\|\mathbf{y}_{0,j}\| > \beta_{\mathsf{agg}}$ for any $j \in S$, it sets $\mathbf{W}_0 = \mathbf{0}^{n \times m}$ and $\mathbf{y}_{0,j} = \mathbf{0}^m$ for all $j \in S$. Otherwise, it leaves $\mathbf{y}_{0,j}$ unchanged and sets $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. It sets $\mathbf{W}_S = \sum_{j \in S} \mathbf{W}_j$ and outputs

$$\mathsf{ct} = \left(\xi, \mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}, \mathbf{s}^{\mathsf{T}}(\mathbf{W}_0 + \mathbf{W}_S) + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}, \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + \mu \cdot \lfloor q/2 \rfloor \right).$$

- $\mathsf{Decrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mathsf{ct}, (i, \mathsf{sk}_i))$: On input the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$, a collection of public keys $\mathsf{pk}_j = (\mathbf{W}_j, \{\mathbf{y}_{j,j'}\}_{j' \neq j})$ for each $j \in S$, a ciphertext $\mathsf{ct} = (\xi, \mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, c_3)$, and a secret key $\mathsf{sk}_i = \mathbf{y}_{i,i} \in \mathbb{Z}_q^m$ for an index $i \in S$, the decryption algorithm computes $(\{\mathbf{y}_{0,j}\}_{j \in S}, \mathbf{d}_0)$ from ξ as in Eq. (6.5) and computes

$$z = c_3 + \mathbf{c}_2^{\mathsf{T}}\mathbf{r}_i - \mathbf{c}_1^{\mathsf{T}}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) \in \mathbb{Z}_q,$$

and outputs $\lfloor z \rceil$ where $\lfloor z \rceil$ outputs 0 if $-q/4 \leq z < q/4$ and 1 otherwise. If $i \notin S$ it outputs 0.

Theorem 6.5 (Completeness)**.** *Suppose q is prime, $n \geq \lambda$, $m \geq 3n \log q$, $\sigma_{\mathrm{pp}} \geq O({\ell_0}^2 m^2)$, and $\beta_{\mathrm{key}} \geq \sigma_{\mathrm{pp}} \cdot O({\ell_0}^2 m^3)$. Then, Construction 6.4 is complete.*

Proof. Let $\lambda, N \in \mathbb{N}$ and take any index $i \in [N]$. Let

$$\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N),$$

and sample $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$. Then, we can write

$$\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}) \quad \text{and} \quad \mathsf{sk}_i = \mathbf{y}_{i,i}.$$

We show that $\mathsf{IsValid}(\mathsf{pp}, i, \mathsf{pk}_i) = 1$ with probability 1:

- Since $\sigma_{\mathrm{pp}} \geq O({\ell_0}^2 m^2) \geq (m\ell_0 + m) \cdot \log(n\ell_0)$, Lemma 4.5 implies that $\|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathrm{pp}}$. By Lemma 4.7, we have $\|\mathbf{T}_{\mathbf{V}}\| \leq \sqrt{m}\sigma_{\mathrm{pp}} \cdot \ell_0 m^2 \leq \sigma_{\mathrm{pp}} \cdot O(\ell_0 m^3)$.
- Next, $\|\mathbf{T}_{\mathbf{V}} \cdot \mathbf{G}_{nN}^{-1}(\boldsymbol{\eta}_i \otimes \mathbf{p})\| \leq m' N \|\mathbf{T}_{\mathbf{V}}\| \leq \sigma_{\mathrm{pp}} \cdot O({\ell_0}^2 m^3) < \beta_{\mathrm{key}}$ by definition of ℓ_0 and the assumption on β_{key}. Thus, $\|\mathbf{y}_{i,j}\| \leq \beta_{\mathrm{key}}$ for all $j \in [N]$.
- By construction of $\mathbf{V}$ (Eq. (6.3)) and the fact that $\mathbf{V} \cdot \mathbf{T}_{\mathbf{V}} = \mathbf{G}_{nN}$, Lemma 3.8 and Eq. (3.1) ensure that

  $$\mathbf{0} = \mathbf{A}\mathbf{y}_{i,j} - \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)\mathbf{d}_i = \mathbf{A}\mathbf{y}_{i,j} - \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)(\mathbf{d}_i \otimes 1) = \mathbf{A}\mathbf{y}_{i,j} - \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)\mathbf{r}_j$$

  holds for all $j \neq i$. By definition, KeyGen sets $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$, implying

  $$\mathbf{A}\mathbf{y}_{i,j} = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)\mathbf{r}_j = \mathbf{W}_i \mathbf{r}_j.$$

Thus, $\mathsf{IsValid}(\mathsf{pp}, i, \mathsf{pk}_i)$ always outputs 1. □

Theorem 6.6 (Correctness)**.** *Suppose $q \geq 4m^{3/2}\sigma_{\mathsf{LWE}}(N\beta_{\mathrm{key}} + \beta_{\mathrm{agg}}) + 8\ell_0 m^5 \sigma_{\mathsf{LWE}}\sigma_{\mathrm{pp}}$, q is prime, $n \geq \lambda$, $m \geq 2n \log q$, $\sigma_{\mathrm{pp}} \geq O({\ell_0}^2 m^2)$, $\beta_{\mathrm{key}} > \sigma_{\mathrm{pp}} \cdot O({\ell_0}^2 m^3)$, and Π_{DGS} is correct. Then, Construction 6.4 satisfies correctness.*

Proof. Let $\lambda, N \in \mathbb{N}$ and take any index $i \in [N]$. Let $\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$ and $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$. Parse $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i})$, and $\mathsf{sk}_i = \mathbf{y}_{i,i}$. By the analysis in Theorem 6.5, we always have

$$\|\mathbf{y}_{i,i}\| \leq \beta_{\mathrm{key}} \quad \text{and} \quad \mathbf{A}\mathbf{y}_{i,i} = \mathbf{W}_i \mathbf{r}_i + \mathbf{p}. \tag{6.6}$$

Take any set $S \subseteq [N]$ and any collection of public keys $\{\mathsf{pk}_j\}_{j \in S \setminus \{i\}}$ where pk_j satisfies $\mathsf{IsValid}(\mathsf{pp}, \mathsf{pk}_i, i) = 1$. This means that for all $j \in S \setminus \{i\}$,

$$\mathbf{A}\mathbf{y}_{j,i} = \mathbf{W}_j \mathbf{r}_i \quad \text{and} \quad \|\mathbf{y}_{j,i}\| \leq \beta_{\mathrm{key}}. \tag{6.7}$$

Take any message $\mu \in \{0, 1\}$ and let $\mathsf{ct} = (\xi, \mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, c_3) \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{j, \mathsf{pk}_j\}_{j \in S}, \mu)$. Let $\mathbf{s} \in \mathbb{Z}_q^n, \mathbf{e} \in \mathbb{Z}_q^m, \mathbf{K}_{\mathbf{W}} \in \{0, 1\}^{m \times m}, \mathbf{k}_{\mathbf{p}} \in \{0, 1\}^m$ be the components sampled by encryption, and let $(\{\mathbf{y}_{0,j}\}_{j \in S}, \mathbf{d}_0)$ be computed as in Eq. (6.5) from ξ. By the structure of $\mathbf{V}$ (Eq. (6.3)) and correctness of Π_{DGS}, the vectors $\mathbf{y}_{0,i}$ and $\mathbf{d}_0$ from Eq. (6.5) satisfy the relation $\mathbf{A}\mathbf{y}_{0,i} - \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{d}_0 = \mathbf{0}^n$. This means

$$\mathbf{A}\mathbf{y}_{0,i} = \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_i)\mathbf{d}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)\mathbf{r}_i = \mathbf{W}_0 \mathbf{r}_i.$$

By definition of $\mathsf{Encrypt}$, we conclude that $(\mathbf{W}_0, \{\mathbf{y}_{0,j}\}_{j \in S})$ satisfy

$$\|\mathbf{y}_{0,i}\| \leq \beta_{\mathrm{agg}} \quad \text{and} \quad \mathbf{A}\mathbf{y}_{0,i} = \mathbf{W}_0 \mathbf{r}_i \tag{6.8}$$

Consider the output of the decryption algorithm $\mathsf{Decrypt}(\mathsf{pp}, \{(j, \mathsf{pk}_j)\}_{j \in S}, \mathsf{ct}, (i, \mathsf{sk}_i))$. First,

$$\mathbf{c}_1^{\mathsf{T}}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) = \mathbf{s}^{\mathsf{T}}\mathbf{A}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) + \underbrace{\mathbf{e}^{\mathsf{T}}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right)}_{\tilde{e}_1}.$$

Combined with Eqs. (6.6) to (6.8), this becomes

$$\begin{aligned}\mathbf{c}_1^{\mathsf{T}}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) &= \mathbf{s}^{\mathsf{T}}(\mathbf{W}_i\mathbf{r}_i + \mathbf{p} + \mathbf{W}_0\mathbf{r}_i) + \sum_{j \in S \setminus \{i\}} \mathbf{s}^{\mathsf{T}}\mathbf{W}_j\mathbf{r}_i + \tilde{e}_1 \\ &= \mathbf{s}^{\mathsf{T}}\left(\mathbf{W}_S\mathbf{r}_i + \mathbf{p} + \mathbf{W}_0\mathbf{r}_i\right) + \tilde{e}_1,\end{aligned}$$

using the fact that $\mathbf{W}_S = \sum_{i \in S} \mathbf{W}_i$ and $i \in S$. Next,

$$c_3 + \mathbf{c}_2^{\mathsf{T}}\mathbf{r}_i = \mu \cdot \lfloor q/2 \rceil + \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{s}^{\mathsf{T}}(\mathbf{W}_0 + \mathbf{W}_S)\mathbf{r}_i + \underbrace{\mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i}_{\tilde{e}_2}.$$

Putting everything together, we have

$$c_3 + \mathbf{c}_2^{\mathsf{T}}\mathbf{r}_i - \mathbf{c}_1^{\mathsf{T}}\left(\mathbf{y}_{i,i} + \mathbf{y}_{0,i} + \sum_{j \in S \setminus \{i\}} \mathbf{y}_{j,i}\right) = \mu \cdot \lfloor q/2 \rceil - \tilde{e}_1 + \tilde{e}_2.$$

It suffices to show that $|\tilde{e}_1 - \tilde{e}_2| < q/4$ always holds:

- By construction, $\|\mathbf{e}\| \leq \sqrt{m}\sigma_{\mathsf{LWE}}$.
- Since $\|\mathbf{y}_{j,i}\| \leq \beta_{\mathsf{key}}$ for $j \in S$ and $\|\mathbf{y}_{0,i}\| \leq \beta_{\mathsf{agg}}$, it follows that

$$|\tilde{e}_1| \leq Nm^{3/2}\beta_{\mathsf{key}}\sigma_{\mathsf{LWE}} + m^{3/2}\beta_{\mathsf{agg}}\sigma_{\mathsf{LWE}} \leq m^{3/2}\sigma_{\mathsf{LWE}}(N\beta_{\mathsf{key}} + \beta_{\mathsf{agg}}).$$

- Next, $\mathbf{k}_{\mathbf{p}} \in \{0,1\}^m$, $\mathbf{K}_{\mathbf{W}} \in \{0,1\}^{m \times m}$, and $\|\mathbf{r}_i\| \leq \|\mathbf{T}_0\| \cdot \ell_0 m^2$ by Lemma 4.7. Since $\|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathsf{pp}}$ by Lemma 4.5, we have

$$|\tilde{e}_2| \leq \left|\mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}}\right| + \left|\mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i\right| \leq m^{3/2}\sigma_{\mathsf{LWE}} + m^{5/2}\sigma_{\mathsf{LWE}} \cdot \|\mathbf{r}_i\| \leq 2\ell_0 m^5 \sigma_{\mathsf{LWE}}\sigma_{\mathsf{pp}}.$$

Correctness holds when $q \geq 4\,|\tilde{e}_1 - \tilde{e}_2|$, so it suffices to set

$$q \geq 4m^{3/2}\sigma_{\mathsf{LWE}}(N\beta_{\mathsf{key}} + \beta_{\mathsf{agg}}) + 8\ell_0 m^5 \sigma_{\mathsf{LWE}}\sigma_{\mathsf{pp}}.$$ □

Theorem 6.7 (Semi-Static Security)**.** *Suppose the following constraints hold:*

- *Lattice parameters: $q > 2$ and q is prime, $n \geq \lambda$, $m \geq 3n \log q$.*
- *Width parameters: $\sigma_{\mathsf{pp}} \geq O(\ell_0{}^2 m^2)$, $\sigma_{\mathsf{key}} \geq O(\ell_0{}^3 m^5) \cdot \sigma_{\mathsf{pp}}$, $\beta_{\mathsf{key}} \geq \sqrt{m}\sigma_{\mathsf{key}}$, $\beta_{\mathsf{agg}} \geq \sqrt{m}\sigma_{\mathsf{agg}}$, and*

$$2^{\lambda_{\mathsf{DGS}}} > \sigma_{\mathsf{agg}} \geq \max\{2^{\lambda}(\ell_0 m^4 \sigma_{\mathsf{pp}}\beta_{\mathsf{key}})\,,\ O(\ell_0 m^{5/2}) \cdot \sigma_{\mathsf{pp}}\sigma_{\mathsf{loss}}\}.$$

Suppose also that Π_{DGS} is correct and explainable. Then, under the ℓ_0-succinct LWE assumption (Assumption 3.10) with parameters $(n, m, q, \sigma_{\mathsf{LWE}}, \sigma_{\mathsf{pp}})$, Construction 6.4 is semi-statically secure for up to N users in the random oracle model.

Proof. Take any polynomial $N = N(\lambda)$ and any efficient adversary $\mathcal{A}$ for the semi-static security game, and suppose $\mathcal{A}$ wins the game with non-negligible advantage ε. For ease of exposition, we assume that $\mathcal{A}$ never queries the random oracle on the same input more than once; this is without loss of generality since we can generically transform any adversary that does not satisfy this property into one that does by simply maintaining a table of random oracle input/outputs corresponding to the queries the adversary made.

Hybrid sequence. We now define a sequence of hybrid experiments. Our hybrids are parameterized by a bit $b \in \{0,1\}$ and a polynomial $p \in \mathsf{poly}(\lambda)$. We omit the index p when the hybrid definition is independent of the choice of p.

- $\mathsf{Hyb}_0^{(b)}$: This is the semi-static security game with challenge bit $b \in \{0,1\}$. At the beginning of the game, the adversary $\mathcal{A}$ declares the set $S^* \subseteq [N]$. The challenger then samples $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N)$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$ for each $i \in S^*$, and sends $(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S^*})$ to $\mathcal{A}$. Adversary $\mathcal{A}$ then declares the set $S \subseteq S^*$ and the challenger replies with $\mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{pp}, \{(i, \mathsf{pk}_i)\}_{i \in S}, b)$. To recall, the challenger samples the components as follows:
  - The challenger starts by sampling
  $$\begin{aligned} (\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) &\leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{pp}}) \\ (\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_\mathbf{V}, \mathbf{T}_{\tilde{\mathbf{Z}}}) &\leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N) \\ \mathbf{p}, \mathbf{t}_1, \ldots, \mathbf{t}_N &\xleftarrow{\text{R}} \mathbb{Z}_q^n \end{aligned}$$
  It parses $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$ and sets the public parameters to be
  $$\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i \in [N]}, \mathbf{T}_\mathbf{V}, \mathbf{T}_{\tilde{\mathbf{Z}}}).$$
  - To generate the public key for $i \in S^*$, the challenger samples
  $$\boldsymbol{\kappa}_i = \begin{bmatrix} \mathbf{y}_{i,1} \\ \vdots \\ \mathbf{y}_{i,N} \\ \mathbf{d}_i \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_\mathbf{V}, \boldsymbol{\eta}_i \otimes \mathbf{p}, \sigma_{\mathsf{key}}), \tag{6.9}$$
  where $\mathbf{y}_{i,j} \in \mathbb{Z}_q^m$ and $\mathbf{d}_i \in \mathbb{Z}_q^k$. If $\|\mathbf{y}_i\| > \beta_{\mathsf{key}}$ for any $i \in [N]$, it sets
  $$\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \\ \mathbf{d} \end{bmatrix} = \mathbf{T}_\mathbf{V} \cdot \mathbf{G}_{nN}^{-1}(\boldsymbol{\eta}_i \otimes \mathbf{p}). \tag{6.10}$$
  It sets $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$ and $\mathsf{pk}_i = \big(\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{j \neq i}\big)$.
  - Finally, to generate the challenge ciphertext for the set S, the challenger samples $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{LWE}}}^m$, $\xi^* \xleftarrow{\text{R}} \{0,1\}^\lambda$, $\mathbf{K}_\mathbf{W} \xleftarrow{\text{R}} \{0,1\}^{m \times m}$, and $\mathbf{k}_\mathbf{p} \xleftarrow{\text{R}} \{0,1\}^m$. If $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$, it sets $\mathbf{e} = \mathbf{0}^m$. It computes $(\mathbf{M}_S, \mathbf{T}_S) \leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{M}_{\mathbf{Z},\mathbf{R}}, \mathbf{T}_\mathbf{V}, S)$ for $\mathbf{M}_{\mathbf{Z},\mathbf{R}}$ as in Eq. (6.3), and sets $\mathbf{V}_S = [\mathbf{I}_{|S|} \otimes \mathbf{A} \mid \mathbf{M}_S]$. Finally, it computes and $\gamma^* = H_\rho(\xi^*)$
  $$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,i_1} \\ \vdots \\ \mathbf{y}_{0,i_{|S|}} \\ \mathbf{d}_0 \end{bmatrix} = \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma^*), \tag{6.11}$$
  where $\mathbf{y}_{0,j} \in \mathbb{Z}^m$ for each $j \in S$, $\mathbf{d}_0 \in \mathbb{Z}^k$, and $S = \{i_1, \ldots, i_{|S|}\}$. If $\|\mathbf{y}_{0,j}\| > \beta_{\mathsf{agg}}$ for any $j \in S$, it sets $\mathbf{W}_0 = \mathbf{0}^{n \times m}$ and $\mathbf{y}_{0,j} = \mathbf{0}^m$ for all $j \in S$. Otherwise, it leaves $\mathbf{y}_{0,j}$ unchanged and sets $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$. It sets $\mathbf{W}_S = \sum_{j \in S} \mathbf{W}_j$ and constructs the challenge ciphertext as
  $$\mathsf{ct}_b = (\xi^*, \mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, c_3) = \big(\xi^*, \mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}, \mathbf{s}^{\mathsf{T}}(\mathbf{W}_0 + \mathbf{W}_S) + \mathbf{e}^{\mathsf{T}}\mathbf{K}_\mathbf{W}, \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_\mathbf{p} + b \cdot \lfloor q/2 \rfloor\big).$$

  At the end of the experiment, 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 for all $i \in S^*$, the challenger samples the public key for user i as

$$\mathbf{W}_i \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}, \quad \mathbf{d}_i \leftarrow \tilde{\mathbf{Z}}_{\sigma_{\mathsf{key}}}^{-1}(\mathsf{vec}(\mathbf{W}_i)), \quad \forall j \in [N] \setminus \{i\}, \, \mathbf{y}_{i,j} \leftarrow \mathbf{A}_{\sigma_{\mathsf{key}}}^{-1}(\mathbf{W}_i \mathbf{r}_j).$$

  If $\|\mathbf{y}_{i,j}\| > \beta_{\mathsf{key}}$ for any $j \in [N]$, it still sets $\mathbf{y}_{i,j}$ and $\mathbf{d}_i$ according to Eq. (6.10). Note that in this experiment, the challenger does *not* sample the "secret key" $\mathbf{y}_{i,i}$.

- $\mathsf{Hyb}_2^{(b)}$: Same as $\mathsf{Hyb}_1^{(b)}$, except for all $i \in S^*$ the challenger samples a vector $\mathbf{y}_{i,i} \leftarrow \mathbf{A}_{\sigma_{\mathsf{key}}}^{-1}(\mathbf{W}_i \mathbf{r}_i + \mathbf{t}_i)$ and outputs 0 if $\|\mathbf{y}_{i,j}\| > \beta_{\mathsf{key}}$ for any $j \in [N]$ instead of setting $\mathbf{y}_{i,j}$ as in Eq. (6.10) The challenger also no longer sets $\mathbf{e} = \mathbf{0}^m$ if $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$ in the challenge phase. Note that the adversary's view does not depend on $\mathbf{y}_{i,i}$.

- $\mathsf{Hyb}_3^{(b)}$: Same as $\mathsf{Hyb}_2^{(b)}$, except for all $i \in S^*$, the challenger samples

$$\boldsymbol{\kappa}_i \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes \mathbf{t}_i, \sigma_{\mathsf{key}}),$$

  derives $\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,N}, \mathbf{d}_i$ as in Eq. (6.9), and sets $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$.

- $\mathsf{Hyb}_4^{(b)}$: Same as $\mathsf{Hyb}_3^{(b)}$, except the challenger samples the seed $\xi^* \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\lambda$ for the challenge ciphertext at the start of the experiment. The challenger also samples $\gamma^* \overset{\mathrm{R}}{\leftarrow} \{0,1\}^\rho$ at the start of the experiment. If the adversary queries H on ξ^* before the challenge phase, the challenger aborts and outputs 0. If the adversary queries H on ξ^* after the challenge phase, the challenger replies with γ^*.

- $\mathsf{Hyb}_5^{(b)}$: Same as $\mathsf{Hyb}_4^{(b)}$, except in the setup phase, the challenger constructs $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0)$ as

$$\mathbf{A} \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}, \quad \mathbf{U}_0 \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{\ell_0 n \times m}, \quad \mathbf{T}_0 \leftarrow [\mathbf{I}_{\ell_0} \otimes \mathbf{A} \mid \mathbf{U}_0]_{\sigma_{\mathsf{pp}}}^{-1}(\mathbf{G}_{n\ell_0}).$$

  After computing $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N)$, the challenger aborts and outputs 0 if

$$\|\mathbf{T}_0\| > \sqrt{m}\sigma_{\mathsf{pp}}, \quad \|\mathbf{T}_{\mathbf{V}}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|, \quad \text{or } \|\mathbf{R}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|.$$

- $\mathsf{Hyb}_{6,p}^{(b)}$: Same as $\mathsf{Hyb}_5^{(b)}$, except after generating the challenge ciphertext, the challenger now computes

$$\gamma' \leftarrow \mathsf{DGS.Explain}(1^{\lambda_{\mathsf{DGS}}}, 1^{p(\lambda)}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \boldsymbol{\kappa}_0, \sigma_{\mathsf{agg}}).$$

  As in $\mathsf{Hyb}_5^{(b)}$, the experiment aborts and outputs 0 if the adversary queries H on ξ^* before the challenge phase. If the adversary queries H on ξ^* after the challenge phase, the challenger now replies with γ'.

- $\mathsf{Hyb}_{7,p}^{(b)}$: Same as $\mathsf{Hyb}_{6,p}^{(b)}$, except the challenger samples $\boldsymbol{\kappa}_0 \leftarrow (\mathbf{V}_S)_{\sigma_{\mathsf{agg}}}^{-1}(0^{nN})$.

- $\mathsf{Hyb}_{8,p}^{(b)}$: Same as $\mathsf{Hyb}_{7,p}^{(b)}$, except the challenger samples $\mathbf{d}_0 \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}^k$, sets $\mathbf{W}_0 = \mathbf{Z}(\mathbf{d}_0 \otimes \mathbf{I}_m)$, and samples $\mathbf{y}_{0,j} \leftarrow \mathbf{A}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{W}_0 \mathbf{r}_j)$ for $j \in S$ to construct $\boldsymbol{\kappa}_0$. The challenger outputs 0 if $\|\mathbf{y}_{0,j}\| > \beta_{\mathsf{agg}}$ for any $j \in S$ instead of setting $\mathbf{y}_{0,j} = \mathbf{0}^m$ and $\mathbf{W}_0 = \mathbf{0}^{n \times m}$.

- $\mathsf{Hyb}_{9,p}^{(b)}$: Same as $\mathsf{Hyb}_{8,p}^{(b)}$, except the challenger samples $\mathbf{W}_0 \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}$ and $\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathsf{vec}(\mathbf{W}_0), \sigma_{\mathsf{agg}})$.

- $\mathsf{Hyb}_{10,p}^{(b)}$: Same as $\mathsf{Hyb}_{9,p}^{(b)}$, except the challenger samples $\mathbf{W}_0^* \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q^{n \times m}$ and sets $\mathbf{W}_0 = \mathbf{W}_0^* - \mathbf{W}_S$.

- $\mathsf{Hyb}_{11,p}^{(b)}$: Same as $\mathsf{Hyb}_{10,p}^{(b)}$, except the challenger sets $\mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}}, \mathbf{W}_0^* = \mathbf{A}\mathbf{K}_{\mathbf{W}}$.

- $\mathsf{Hyb}_{12,p}^{(b)}$: Same as $\mathsf{Hyb}_{11,p}^{(b)}$, except the challenger samples $\mathbf{k}_{\mathbf{t}_i} \leftarrow \mathbf{A}_{\sigma_{\mathsf{agg}}}^{-1}(\mathbf{t}_i)$ for $i \in [N]$ and for each $i \in S$ it sets $\mathbf{y}_{0,i} = \mathbf{K}_{\mathbf{W}} \mathbf{r}_i - \sum_{j \in S} \mathbf{y}_{j,i} + \mathbf{k}_{\mathbf{t}_i}$.

- $\mathsf{Hyb}_{13,p}^{(b)}$: Same as $\mathsf{Hyb}_{12,p}^{(b)}$, except the challenger samples $\mathbf{k}_{\mathbf{t}_1}, \ldots, \mathbf{k}_{\mathbf{t}_N} \overset{\mathrm{R}}{\leftarrow} D_{\mathbb{Z},\sigma_{\mathsf{agg}}}^m$ and sets $\mathbf{t}_i = \mathbf{A}\mathbf{k}_{\mathbf{t}_i}$ for all $i \in [N]$.

- $\mathsf{Hyb}_{14,p}^{(b)}$: Same as $\mathsf{Hyb}_{13,p}^{(b)}$, except the challenger samples $\mathbf{c}_1 \xleftarrow{\text{R}} \mathbb{Z}_q^m$ and sets $\mathbf{c}_2^{\mathsf{T}} = \mathbf{c}_1^{\mathsf{T}} \mathbf{K}_{\mathbf{W}}, c_3 = \mathbf{c}_1^{\mathsf{T}} \mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor$.
- $\mathsf{Hyb}_{15,p}^{(b)}$: Same as $\mathsf{Hyb}_{14,p}^{(b)}$, except the challenger samples $\mathbf{p} \xleftarrow{\text{R}} \mathbb{Z}_q^n$ and $c_3 \xleftarrow{\text{R}} \mathbb{Z}_q$.

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

Lemma 6.8. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$, and $\sigma_{\mathsf{key}} \geq 3{\ell_0}^3 m^{9/2} \cdot \sigma_{\mathsf{pp}}$. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_0^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The statistical indistinguishability of the two hybrids follows directly from Corollary 4.10. Given $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$, $\sigma_{\mathsf{key}} \geq 3{\ell_0}^3 m^{9/2} \cdot \sigma_{\mathsf{pp}}$, by Corollary 4.10 with overwhelming probability over honestly generated $(\mathbf{A}, \mathbf{V}, \mathbf{Z}, \mathbf{T}_{\mathbf{V}})$, the following two distributions have negligible statistical distance

- Sample
$$\begin{bmatrix} \mathbf{y}_{i,1} \\ \vdots \\ \mathbf{y}_{i,N} \\ \mathbf{d}_i \end{bmatrix} \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes \mathbf{t}_i, \sigma_{\mathsf{key}}),$$
and output $\left(\{\mathbf{y}_{i,j}\}_{j \neq i}, \mathbf{d}_i\right)$.
- Sample $\mathbf{W}_i \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$ $\mathbf{d}_i \leftarrow \tilde{\mathbf{Z}}_{\sigma_{\mathsf{key}}}^{-1}(\mathsf{vec}(\mathbf{W}_i))$, and for each $j \neq i$, sample $\mathbf{y}_{i,j} \leftarrow \mathbf{A}_{\sigma_{\mathsf{key}}}^{-1}(\mathbf{W}_i \mathbf{r}_j)$. Output $\left(\{\mathbf{y}_{i,j}\}_{j \neq i}, \mathbf{d}_i\right)$.

The first distribution corresponds to $\mathsf{Hyb}_0^{(b)}$ and the second distribution corresponds to $\mathsf{Hyb}_1^{(b)}$. □

Lemma 6.9. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$, and $\beta_{\mathsf{key}} \geq \sqrt{m}\sigma_{\mathsf{key}}$. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,* $|\Pr[\mathsf{Hyb}_1^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.

Proof. First by Lemma 4.5, given $n \geq \lambda$, $m \geq 3n \log q$, and $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$, the distribution of $\mathbf{A}$ is statistically close to uniform. Since $m \geq 3n \log q$, we can appeal to Lemma 3.5 and a union bound to get that with overwhelming probability $\|\mathbf{y}_{i,j}\| \leq \sqrt{m}\sigma_{\mathsf{key}} \leq \beta_{\mathsf{key}}$ for all $i \in S^*$ and $j \in [N]$. Also by Lemma 3.5, we can conclude that $\|\mathbf{e}\| \leq \sqrt{m}\sigma_{\mathsf{LWE}}$. □

Lemma 6.10. *Suppose $n \geq \lambda$, $m \geq 3n \log q$, q is prime, $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$, and $\sigma_{\mathsf{key}} \geq 3{\ell_0}^3 m^{9/2} \cdot \sigma_{\mathsf{pp}}$. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. This lemma follows by the same argument as in the proof of Lemma 6.8. □

Lemma 6.11. *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. In $\mathsf{Hyb}_3^{(b)}$, the challenger samples $\xi^* \xleftarrow{\text{R}} \{0,1\}^\lambda$ *after* the adversary submits its challenge query. Let $Q_{\mathsf{ro}} = \mathsf{poly}(\lambda)$ be a bound on the number of random oracle queries algorithm $\mathcal{A}$ makes. By a union bound, with probability $1 - Q_{\mathsf{ro}}/2^\lambda$ over the choice of ξ^*, the adversary does not query the random oracle on ξ^* prior to the challenge phase. Conditioned on ξ^* not being queries prior to the challenger phase (which happens with overwhelming probability), the distribution of $\gamma^* = H_\rho(\xi^*)$ is uniform over $\{0,1\}^\rho$. This is the distribution of γ^* in $\mathsf{Hyb}_4^{(b)}$ and thus the claim holds. □

Lemma 6.12. *Suppose $m \geq 3n \log q$, q is prime, and $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_4^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_5^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. Since $\sigma_{\mathsf{pp}} \geq (m\ell_0 + m)\log(n\ell_0)$, by Lemma 4.5, the following distributions are statistically indistinguishable:

$$\left\{(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) \leftarrow \mathsf{SuccinctTrapGen}(1^n, 1^{\ell_0}, q, m, \sigma_{\mathsf{pp}})\right\} \quad \text{and} \quad \left\{(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0) : \begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{U}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\ell_0 \times m} \\ \mathbf{T}_0 \leftarrow [\mathbf{I}_{\ell_0} \otimes \mathbf{A} \mid \mathbf{U}_0]^{-1}_{\sigma_{\mathsf{pp}}}(\mathbf{G}_{n\ell_0}). \end{array}\right\}$$

Moreover, $\|\mathbf{T}_0\| \leq \sqrt{m}\sigma_{\mathsf{pp}}$ in the left distribution. By Lemma 4.7, $\|\mathbf{T}_\mathbf{V}\|, \|\mathbf{R}\| \leq \|\mathbf{T}_0\| \cdot \ell_0 m^2$ and the claim holds. □

Lemma 6.13. *Suppose $(\ell_0 m^{5/2} \cdot \sigma_{\mathsf{pp}}) \cdot \sigma_{\mathsf{loss}} < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$ and Π_{DGS} is explainable (Definition 4.1). Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_5^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{6,p}^{(b)}(\mathcal{A}) = 1]| = 1/p(\lambda) + \mathsf{negl}(\lambda).$$

Proof. By the abort condition in $\mathsf{Hyb}_5^{(b)}$ and Lemma 4.6, in the encryption phase the challenger always ensures

$$\|\mathbf{T}_S\| \leq \|\mathbf{T}_\mathbf{V}\| \leq \|\mathbf{T}_0\| \cdot \ell_0 m^2 \leq \ell_0 m^{5/2} \cdot \sigma_{\mathsf{pp}}.$$

Now, given $\|\mathbf{T}_S\| \cdot \sigma_{\mathsf{loss}} < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$ and $\left\|\mathbf{0}^{nN}\right\| \leq 2^{\lambda_{\mathsf{DGS}}}$, by the explainability of Π_{DGS}, the following two distributions have $1/p(\lambda) + \mathsf{negl}(\lambda)$ statistical distance:

- $\mathcal{D}_{\mathsf{SamplePre}}$: Sample $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$, $\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma)$, output $(\boldsymbol{\kappa}_0, \gamma)$.
- $\mathcal{D}_{\mathsf{Explain},p(\lambda)}$: Sample $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^\rho$, $\boldsymbol{\kappa}_0 \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}_S, \mathbf{T}_s, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma)$. Then resample the randomness $\gamma' \xleftarrow{\mathrm{R}} \mathsf{DGS.Explain}(1^{\lambda_{\mathsf{DGS}}}, 1^{p(\lambda)}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \boldsymbol{\kappa}_0, \sigma_{\mathsf{agg}})$. Output $(\boldsymbol{\kappa}_0, \gamma')$.

In $\mathsf{Hyb}_5^{(b)}$, the challenger sets $H(\xi^*) := \gamma^*$ as in $\mathcal{D}_{\mathsf{SamplePre}}$ whereas in $\mathsf{Hyb}_{6,p}^{(b)}$, the challenger sets $H(\xi^*) := \gamma'$ as in $\mathcal{D}_{\mathsf{Explain},p(\lambda)}$. The remainder of the experiment is unchanged so the lemma follows. □

Lemma 6.14. *Suppose $(\ell_0 m^{5/2} \cdot \sigma_{\mathsf{pp}}) \cdot \sigma_{\mathsf{loss}} < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$ and Π_{DGS} is correct (Definition 4.1). For every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{6,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{7,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. By the abort condition in $\mathsf{Hyb}_5^{(b)}$ and Lemma 4.6, in the encryption phase the challenger always ensures

$$\|\mathbf{T}_S\| \leq \|\mathbf{T}_\mathbf{V}\| \leq \|\mathbf{T}_0\| \cdot \ell_0 m^2 \leq \ell_0 m^{5/2} \cdot \sigma_{\mathsf{pp}}.$$

Now given $\|\mathbf{T}_S\| \cdot \sigma_{\mathsf{loss}} < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$ and $\left\|\mathbf{0}^{nN}\right\| \leq 2^{\lambda_{\mathsf{DGS}}}$, by correctness of Π_{DGS}, the following two distributions have $\mathsf{negl}(\lambda)$ statistical distance:

$$\left\{\mathbf{x} \leftarrow \mathsf{DGS.SamplePre}(1^{\lambda_{\mathsf{DGS}}}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \sigma_{\mathsf{agg}}; \gamma)\right\} \quad \text{and} \quad \left\{\mathbf{x} \leftarrow (\mathbf{V}_S)^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{0}^{nN})\right\}.$$

The left and right distributions correspond to $\mathsf{Hyb}_{6,p}^{(b)}$ and $\mathsf{Hyb}_{7,p}^{(b)}$ respectively. □

Lemma 6.15. *Suppose $m \geq 2n \log q$, q is prime, $\sigma_{\mathsf{agg}} \geq 4\log(\ell_0 m)$, and $\beta_{\mathsf{agg}} \geq \sqrt{m}\sigma_{\mathsf{agg}}$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{7,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{8,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The indistinguishability of the two hybrids follows from Lemmas 3.5 and 3.7. Since $m \geq 2n \log q$, q is prime, and $\sigma_{\mathsf{agg}} \geq 4\log(\ell_0 m)$, with overwhelming probability over random $\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$, the statistical distance between the following distributions is $\mathsf{negl}(n)$ by Lemma 3.7:

- Sample and output $\boldsymbol{\kappa}_0 \leftarrow (\mathbf{V}_S)^{-1}_{\sigma_{\mathrm{agg}}}(\mathbf{0}^{nN})$.
- Sample and output $\boldsymbol{\kappa}_0$ where
$$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,i_1} \\ \vdots \\ \mathbf{y}_{0,i_{|S|}} \\ \mathbf{d}_0 \end{bmatrix} \quad \text{where} \quad \mathbf{d}_0 \leftarrow D^k_{\mathbb{Z},\sigma_{\mathrm{agg}}} \text{ and } \begin{bmatrix} \mathbf{y}_{0,i_1} \\ \vdots \\ \mathbf{y}_{0,i_{|S|}} \end{bmatrix} \leftarrow (\mathbf{I}_{|S|} \otimes \mathbf{A})^{-1}_{\sigma_{\mathrm{agg}}}\left(\begin{bmatrix} \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_{i_1})\mathbf{d}_0 \\ \vdots \\ \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_{i_{|S|}})\mathbf{d}_0 \end{bmatrix} \right),$$
and $S = \{i_1, \ldots, i_{|S|}\}$.

The first distribution corresponds to $\mathsf{Hyb}^{(b)}_{7,p}$. By Lemma 3.5 and a union bound, $\|\mathbf{y}_{0,j}\| \le \sqrt{m}\sigma_{\mathrm{agg}} \le \beta_{\mathrm{agg}}$ for all $j \in S$ with overwhelming probability in the second distribution. Thus, the second distribution is statistically close to $\mathsf{Hyb}^{(b)}_{8,p}$ since $\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_j)\mathbf{d}_0 = \mathbf{W}_0\mathbf{r}_j$. □

Lemma 6.16. *Suppose q is prime, and $\sigma_{\mathrm{agg}} \ge k \log nm$. Then, for every polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}^{(b)}_{8,p}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{9,p}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. First by Lemma 4.7, given $\ell_0 \ge Nm'$, the marginal distribution of $\mathbf{Z}$ (hence $\tilde{\mathbf{Z}}$) is statistically close to uniformly random, $\tilde{\mathbf{Z}}\mathbf{T}_{\tilde{\mathbf{Z}}} = \mathbf{G}_{nm}$, and $\left\|\mathbf{T}_{\tilde{\mathbf{Z}}}\right\| = 1$. Hence by Lemma 3.6, given that $\tilde{\mathbf{Z}} \in \mathbb{Z}_q^{nm \times k}$ satisfies $k = 3nm \log q > 2nm \log q$, q is prime, and $\sigma_{\mathrm{agg}} \ge k \log nm \ge \log k$, with overwhelming probability over the choice of $\tilde{\mathbf{Z}}$, the following distributions have negligible statistical distance:
$$\left\{(\mathbf{d}_0, \mathsf{vec}(\mathbf{W}_0) = \tilde{\mathbf{Z}}\mathbf{d}_0) : \mathbf{d}_0 \leftarrow D^k_{\mathbb{Z},\sigma_{\mathrm{agg}}}\right\} \quad \text{and} \quad \left\{(\mathbf{d}_0, \mathsf{vec}(\mathbf{W}_0)) : \begin{array}{c} \mathbf{W}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \\ \mathbf{d}_0 \leftarrow \tilde{\mathbf{Z}}^{-1}_{\sigma_{\mathrm{agg}}}(\mathsf{vec}(\mathbf{W}_0)) \end{array}\right\}.$$
Furthermore, by Lemma 3.8, given that $\tilde{\mathbf{Z}}\mathbf{T}_{\tilde{\mathbf{Z}}} = \mathbf{G}_{nm}$ and $\sigma_{\mathrm{agg}} \ge k \log nm = k\|\mathbf{T}_{\tilde{\mathbf{Z}}}\| \log nm$, the following distributions have negligible statistical distance:
$$\{\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathsf{vec}(\mathbf{W}_0), \sigma_{\mathrm{agg}})\} \quad \text{and} \quad \{\mathbf{d}_0 \leftarrow \tilde{\mathbf{Z}}^{-1}_{\sigma_{\mathrm{agg}}}(\mathsf{vec}(\mathbf{W}_0))\}.$$
The lemma now follows by a hybrid argument. □

Lemma 6.17. *For every polynomial p, all $b \in \{0,1\}$, and all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Hyb}^{(b)}_{9,p}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}^{(b)}_{10,p}(\mathcal{A}) = 1]$.*

Proof. In both experiments, $\mathbf{W}_0^*$ is uniformly distributed. Thus, these two experiments are identically distributed. □

Lemma 6.18. *Suppose $n \ge \lambda$, $m \ge 2n \log q$ and $q > 2$ is prime. Then, for every polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}^{(b)}_{10,p}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{11,p}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. By Lemma 3.3, for all $\mathbf{e} \in \mathbb{Z}_q^m$, the following two distributions are statistically indistinguishable:
$$\left\{(\mathbf{A}, \mathbf{A}\mathbf{R}^*, \mathbf{e}^{\mathsf{T}}\mathbf{R}^*) : \begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m} \\ \mathbf{R}^* \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m+1} \end{array}\right\} \text{ and } \left\{(\mathbf{A}, [\mathbf{p} \mid \mathbf{W}_0^*], \mathbf{e}^{\mathsf{T}}\mathbf{R}^*) : \begin{array}{c} \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \\ \mathbf{R}^* \xleftarrow{\mathrm{R}} \{0,1\}^{m \times m+1}, \\ [\mathbf{p} \mid \mathbf{W}_0^*] \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m+1} \end{array}\right\}.$$
By setting $\mathbf{R}^* = [\mathbf{k}_{\mathbf{p}} \mid \mathbf{K}_{\mathbf{W}}]$, the left and right distributions correspond to $\mathsf{Hyb}^{(b)}_{11,p}$ and $\mathsf{Hyb}^{(b)}_{10,p}$, respectively. □

Lemma 6.19. *Suppose $m \ge 2n \log q$, q is prime, and $\sigma_{\mathrm{agg}} > 2^{\lambda}(\ell_0 m^4 \sigma_{\mathrm{pp}} \beta_{\mathrm{key}})$. Then, for every polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*
$$|\Pr[\mathsf{Hyb}^{(b)}_{11,p}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{12,p}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. First, for $i \in S$ we observe

$$\mathbf{W}_0\mathbf{r}_i = \mathbf{W}_0^*\mathbf{r}_i - \sum_{j\in S\setminus i}\mathbf{W}_j\mathbf{r}_i - (\mathbf{W}_i\mathbf{r}_i + \mathbf{t}_i) + \mathbf{t}_i = \mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in S\setminus i}\mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}_{i,i} + \mathbf{t}_i.$$

This means $\mathbf{y}_{0,i} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathrm{agg}}}(\mathbf{A}\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in S\setminus i}\mathbf{A}\mathbf{y}_{j,i} - \mathbf{A}\mathbf{y}_{i,i} + \mathbf{t}_i)$ in $\mathsf{Hyb}_{11}^{(b)}$. Now, for $i \in S$, let

$$\hat{\mathbf{y}}_{0,i} = \mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in[N]\setminus i}\mathbf{y}_{j,i} - \mathbf{y}_{i,i}.$$

We bound $\|\hat{\mathbf{y}}_{0,i}\|_2$ for all $i \in S$:

- By the abort condition introduced in $\mathsf{Hyb}_2^{(b)}$, $\|\mathbf{y}_{i,j}\| \le \beta_{\mathrm{key}}$ holds for all $i \in S, j \in [N]$.
- By the abort condition introduced in $\mathsf{Hyb}_5^{(b)}$, $\|\mathbf{T}_0\| \le \sqrt{m}\sigma_{\mathrm{pp}}$ and $\|\mathbf{R}\| \le \ell_0 m^2 \cdot \|\mathbf{T}_0\| \le \ell_0 m^{5/2}\sigma_{\mathrm{pp}}$.
- Additionally, $\|\mathbf{K}_{\mathbf{W}}\| \le 1$ by definition. Thus, for $i \in S$ we have

$$\|\hat{\mathbf{y}}_{0,i}\| = \left\|\mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in[N]\setminus i}\mathbf{y}_{j,i} - \mathbf{y}_{i,i}\right\| \le \ell_0 m^{7/2}\sigma_{\mathrm{pp}} + N\beta_{\mathrm{key}}.$$

Thus, we also have $\|\hat{\mathbf{y}}_{0,i}\|_2 \le \sqrt{m}\|\hat{\mathbf{y}}_{0,i}\| \le \ell_0 m^4\sigma_{\mathrm{pp}} + \sqrt{m}N\beta_{\mathrm{key}}$. Since $\sigma_{\mathrm{agg}} > 2^\lambda(\ell_0 m^4 \sigma_{\mathrm{pp}}\beta_{\mathrm{key}})$, we conclude that $\sqrt{\|\hat{\mathbf{y}}_{0,i}\|_2/\sigma_{\mathrm{agg}}}$ is negligible. By Theorem 4.3, for each $i \in S$, the following distributions are also statistically close:

$$\left\{\mathbf{A}^{-1}_{\sigma_{\mathrm{agg}}}(\mathbf{t}_i + \mathbf{A}\hat{\mathbf{y}}_{0,i})\right\} \quad \text{and} \quad \left\{\mathbf{A}^{-1}_{\sigma_{\mathrm{agg}}}(\mathbf{t}_i) + \hat{\mathbf{y}}_{0,i}\right\}.$$

The left and right distributions correspond to $\mathsf{Hyb}_{11,p}^{(b)}$ and $\mathsf{Hyb}_{12,p}^{(b)}$ respectively. The lemma then follows by a hybrid argument. □

Lemma 6.20. *Suppose $m \ge 2n\log q$, q is prime, and $\sigma_{\mathrm{agg}} > \log m$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$, $|\Pr[\mathsf{Hyb}_{12,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{13,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. By Lemma 3.6, given that $\mathbf{A}$ is sampled uniform randomly, $m \ge 2n\log q$, q is prime, and $\sigma_{\mathrm{agg}} > \log m$, the following two distributions have negligible statistical distance:

$$\left\{(\mathbf{k}_{\mathbf{t}_i}, \mathbf{A}\mathbf{k}_{\mathbf{t}_i}) \;:\; \mathbf{k}_{\mathbf{t}_i} \leftarrow D^m_{\mathbb{Z},\sigma_{\mathrm{agg}}}\right\} \quad \text{and} \quad \left\{(\mathbf{k}_{\mathbf{t}_i}, \mathbf{t}_i) \;:\; \mathbf{t}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n, \mathbf{k}_{\mathbf{t}_i} \leftarrow \mathbf{A}^{-1}_{\sigma_{\mathrm{agg}}}(\mathbf{t}_i)\right\}.$$

Since the sampling procedure is identical for all $i \in [N]$, the lemma follows by a hybrid argument. □

Lemma 6.21. *Suppose the ℓ_0-succinct LWE assumption (Assumption 3.10) holds with parameters $(n, m, q, \sigma_{\mathsf{LWE}}, \sigma_{\mathrm{pp}})$. Then, for every polynomial p, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{13,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{14,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. Suppose there exists a bit $b \in \{0,1\}$ such that $|\Pr[\mathsf{Hyb}_{13,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{14,p}^{(b)}(\mathcal{A}) = 1]| \ge \varepsilon$ for some non-negligible ε. We use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ that breaks the ℓ_0-succinct LWE assumption with parameters $(n, m, q, \sigma_{\mathsf{LWE}}, \sigma_{\mathrm{pp}})$:

1. At the beginning of the game, algorithm $\mathcal{B}$ receives a tuple $(\mathbf{A}, \mathbf{c}_1^{\mathsf{T}}, \mathbf{U}_0, \mathbf{T}_0)$ from the ℓ_0-succinct LWE challenger and runs $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{Transform}(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0, N)$. It parses $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$. Algorithm $\mathcal{B}$ aborts with output 0 if $\|\mathbf{T}_0\| > \sqrt{m}\sigma_{\mathrm{pp}}$, $\|\mathbf{T}_{\mathbf{V}}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|$, or $\|\mathbf{R}\| > \ell_0 m^2 \cdot \|\mathbf{T}_0\|$.

2. Algorithm $\mathcal{B}$ samples $\mathbf{K}_{\mathbf{W}} \leftarrow \{0,1\}^{m\times m}, \mathbf{k}_{\mathbf{p}} \leftarrow \{0,1\}^m$, and $\mathbf{k}_{\mathbf{t}_i} \leftarrow D^m_{\mathbb{Z},\sigma_{\mathrm{agg}}}$ for $i \in [N]$. It sets $\mathbf{t}_i = \mathbf{A}\mathbf{k}_{\mathbf{t}_i}$ for $i \in [N]$, $\mathbf{p} = \mathbf{A}\mathbf{k}_{\mathbf{p}}$, $\mathbf{W}_0^* = \mathbf{A}\mathbf{K}_{\mathbf{W}}$, $\mathbf{c}_2^{\mathsf{T}} = \mathbf{c}_1^{\mathsf{T}}\mathbf{K}_{\mathbf{W}}$, $c_3 = \mathbf{c}_1^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor$, and
$$\mathsf{pp} = (\mathbf{A}, \mathbf{p}, \mathbf{V}, \mathbf{Z}, \{\mathbf{r}_i, \mathbf{t}_i\}_{i\in[N]}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}}).$$

3. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ to get $S^* \subseteq [N]$. For each $i \in S^*$, algorithm $\mathcal{B}$ samples $\boldsymbol{\kappa}_i \leftarrow \mathsf{SamplePre}(\mathbf{V}, \mathbf{T}_{\mathbf{V}}, \boldsymbol{\eta}_i \otimes \mathbf{t}_i, \sigma_{\mathrm{key}})$ and sets $\mathbf{W}_i = \mathbf{Z}(\mathbf{d}_i \otimes \mathbf{I}_m)$, where $\mathbf{y}_{i,j}$ and $\mathbf{d}_i$ are derived as in Eq. (6.9). If $\|\mathbf{y}_{i,j}\| > \beta_{\mathrm{key}}$ for any $j \in [N]$, algorithm $\mathcal{B}$ aborts and outputs 0. Otherwise, algorithm $\mathcal{B}$ sets $\mathsf{pk}_i = (\mathbf{W}_i, \{\mathbf{y}_{i,j}\}_{i\neq j})$ for $i \in S^*$ and gives $(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in S^*})$ to $\mathcal{A}$ to get $S \subseteq S^*$.

4. Algorithm $\mathcal{B}$ samples $\xi^* \xleftarrow{\mathrm{R}} \{0,1\}^{\lambda}$. If algorithm $\mathcal{A}$ queries the random oracle on ξ^* prior to the challenge phase, then $\mathcal{B}$ outputs 0. On all other random oracle queries, algorithm $\mathcal{B}$ responds with a random string $\gamma \xleftarrow{\mathrm{R}} \{0,1\}^{\rho}$.

5. Algorithm $\mathcal{B}$ runs $(\mathbf{M}_S, \mathbf{T}_S) \leftarrow \mathsf{DimRed}(\mathbf{A}, \mathbf{M}_{\mathbf{Z},\mathbf{R}}, \mathbf{T}_{\mathbf{V}}, S)$ for $\mathbf{M}_{\mathbf{Z},\mathbf{R}}$ as in Eq. (6.3), sets $\mathbf{V}_S = [\mathbf{I}_{|S|} \otimes \mathbf{A} \mid \mathbf{M}_S]$, and sets $\mathbf{W}_0 = \mathbf{W}_0^* - \mathbf{W}_S$. Next, algorithm $\mathcal{B}$ samples $\mathbf{d}_0 \leftarrow \mathsf{SamplePre}(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}, \mathsf{vec}(\mathbf{W}_0), \sigma_{\mathrm{agg}})$ and sets $\mathbf{y}_{0,i} = \mathbf{K}_{\mathbf{W}}\mathbf{r}_i - \sum_{j\in S} \mathbf{y}_{j,i} + \mathbf{k}_{\mathbf{t}_i}$ for $i \in S$. Algorithm $\mathcal{B}$ aborts and outputs 0 if $\|\mathbf{y}_{0,j}\| > \beta_{\mathrm{agg}}$ for any $j \in S$. If the checks pass, algorithm $\mathcal{B}$ sets
$$\boldsymbol{\kappa}_0 = \begin{bmatrix} \mathbf{y}_{0,i_1} \\ \vdots \\ \mathbf{y}_{0,i_{|S|}} \\ \mathbf{d}_0 \end{bmatrix},$$
and computes $\gamma' \leftarrow \mathsf{DGS.Explain}(1^{\lambda_{\mathsf{DGS}}}, 1^{\rho(\lambda)}, \mathbf{V}_S, \mathbf{T}_S, \mathbf{0}^{nN}, \boldsymbol{\kappa}_0, \sigma_{\mathrm{agg}})$. If $\mathcal{A}$ queries the random oracle on ξ^*, then algorithm $\mathcal{B}$ responds with γ'.

6. Algorithm $\mathcal{B}$ gives $\mathsf{ct}_b = (\xi, \mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, c_3)$ to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

First, we argue that algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_{13,p}^{(b)}$ or $\mathsf{Hyb}_{14,p}^{(b)}$. By definition, The ℓ_0-succinct LWE challenger samples $(\mathbf{A}, \mathbf{U}_0, \mathbf{T}_0)$ as
$$\mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}, \quad \mathbf{U}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{\ell_0 n \times m}, \quad \mathbf{T}_0 \leftarrow [\mathbf{I}_{\ell_0} \otimes \mathbf{A} \mid \mathbf{U}]^{-1}_{\sigma_{\mathrm{pp}}}(\mathbf{G}_{n\ell_0}).$$

This is exactly the specification in $\mathsf{Hyb}_{13,p}^{(b)}$ and $\mathsf{Hyb}_{14,p}^{(b)}$. We conclude that algorithm $\mathcal{B}$ perfectly simulates the setup phase, the public keys, and the random oracle queries exactly as in $\mathsf{Hyb}_{13,p}^{(b)}$ or $\mathsf{Hyb}_{14,p}^{(b)}$. If $\mathbf{c}_1^{\mathsf{T}} = \mathbf{s}^{\mathsf{T}}\mathbf{A} + \mathbf{e}^{\mathsf{T}}$ where $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D^m_{\mathbb{Z},\sigma_{\mathrm{LWE}}}$, then algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_{13,p}^{(b)}(\mathcal{A})$. If $\mathbf{c}_1^{\mathsf{T}} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^m$, then algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_{14,p}^{(b)}(\mathcal{A})$. Thus, algorithm $\mathcal{B}$ breaks ℓ_0-succinct LWE with the same advantage ε. □

Lemma 6.22. *Suppose $n \geq \lambda$, $m \geq 2(n+1)\log q$, and $q > 2$ is a prime. Then, for every polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*
$$|\Pr[\mathsf{Hyb}_{14,p}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{15,p}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. This follows from Lemma 3.3 applied to the matrix $\begin{bmatrix} \mathbf{A} \\ \mathbf{c}_1^{\mathsf{T}} \end{bmatrix} \in \mathbb{Z}_q^{(n+1)\times m}$. □

Lemma 6.23. *For every polynomial p and all $\lambda \in \mathbb{N}$, $\Pr[\mathsf{Hyb}_{15,p}^{(0)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_{15,p}^{(1)}(\mathcal{A}) = 1]$.*

Proof. By construction, the challenger's behavior in $\mathsf{Hyb}_{15,p}^{(b)}$ is *independent* of the challenge bit $b \in \{0,1\}$, so the adversary's view in the two distributions is identical. □

Proof of Theorem 6.7. To finish the proof of Theorem 6.7, we show that Construction 6.4 is semi-static secure by combining Lemmas 6.8 to 6.23. By assumption, there exists an adversary $\mathcal{A}$ that wins with advantage $\varepsilon(\lambda)$, which means for all $\lambda \in \mathbb{N}$ we have $|\Pr[\mathsf{Hyb}_0^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0^{(1)}(\mathcal{A}) = 1]| = \varepsilon(\lambda)$. Therefore, there exists some polynomial p' such that for infinitely many $\lambda \in \mathbb{N}$, $\varepsilon(\lambda) \geq 1/p'(\lambda)$. Let $p(\lambda) = 3p'(\lambda)$. By Lemmas 6.8 to 6.23, we have for all $\lambda \in \mathbb{N}$ (and recalling that for $i \leq 5$, $\mathsf{Hyb}_{i,p}^{(b)}(\mathcal{A}) \equiv \mathsf{Hyb}_i^{(b)}(\mathcal{A})$),

$$\begin{aligned} |\Pr[\mathsf{Hyb}_1^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1^{(1)}(\mathcal{A}) = 1]| &\leq \sum_{i=0}^{14} |\Pr[\mathsf{Hyb}_{i,p}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i+1,p}^{(0)}(\mathcal{A}) = 1]| \\ &\quad + |\Pr[\mathsf{Hyb}_{15}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{15}^{(1)}(\mathcal{A}) = 1]| \\ &\quad + \sum_{i=0}^{14} |\Pr[\mathsf{Hyb}_{i+1,p}^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i,p}^{(1)}(\mathcal{A}) = 1]| \\ &\leq 2/p(\lambda) + \delta(\lambda), \end{aligned}$$

where $\delta(\lambda) = \mathsf{negl}(\lambda)$ is a negligible function. Thus, for infinitely many $\lambda \in \mathbb{N}$, $2/p(\lambda) + \delta(\lambda) \geq 1/p'(\lambda) = 3/p(\lambda)$. Hence $\delta(\lambda) \geq 1/p(\lambda)$ for infinitely many λ, contradicting the fact that δ is negligible, which proves the theorem. □

Parameter instantiation. Let λ be a security parameter and N be a bound on the number of users. We can instantiate the lattice parameters in Construction 6.4 to satisfy Theorems 6.5 to 6.7:

- We set the lattice dimension $n = (\lambda \log N)^{1/\varepsilon}$ for some constant $\varepsilon \in (0, 1)$. and $m = 3n \log q$. Recall that $\ell_0 = Nm' \leq Nm$.
- We can bound $\sigma_{\mathsf{loss}}(\lambda_{\mathsf{DGS}}, nN, mN + k, q)$ by $\tilde{O}({\ell_0}^2 m^3 \lambda_{\mathsf{DGS}}^2)$. Below, we show that we can set $\lambda_{\mathsf{DGS}} = \tilde{O}(\lambda \log N)$. Here $\tilde{O}(\cdot)$ suppresses $\mathsf{poly}(\log \lambda, \log\log N)$ terms.
- We set $\sigma_{\mathsf{LWE}} = \mathsf{poly}(n)$, $\sigma_{\mathsf{pp}} = O({\ell_0}^2 m^2)$, $\sigma_{\mathsf{key}} = O({\ell_0}^3 m^5) \cdot \sigma_{\mathsf{pp}} = O({\ell_0}^5 m^7)$, $\beta_{\mathsf{key}} = m\sigma_{\mathsf{key}} = O({\ell_0}^5 m^8)$, $\sigma_{\mathsf{agg}} = 2^\lambda \ell_0 m^4 \sigma_{\mathsf{pp}} \beta_{\mathsf{key}} \sigma_{\mathsf{loss}} = 2^\lambda \cdot \tilde{O}({\ell_0}^{10} m^{17} \lambda_{\mathsf{DGS}}^2)$, and $\beta_{\mathsf{agg}} = \sqrt{m}\sigma_{\mathsf{agg}}$.
- We set the modulus q to be prime such that

  $$q = 2^\lambda \cdot \tilde{O}({\ell_0}^{10} m^{19} \lambda_{\mathsf{DGS}}^2) \cdot \mathsf{poly}(n) = 2^{\tilde{O}(\lambda \log m \log \ell_0 \log \lambda_{\mathsf{DGS}})} = 2^{\tilde{O}(\lambda \log N \log \lambda_{\mathsf{DGS}})} \leq 2^{\lambda_{\mathsf{DGS}}} = 2^{\tilde{O}(n^\varepsilon)},$$

  where the third equality comes from $\log m = \tilde{O}(1)$, and we set λ_{DGS} such that $\lambda_{\mathsf{DGS}} = \tilde{O}(\lambda \log N)$ and $\lambda_{\mathsf{DGS}} \geq \log q = \tilde{O}(\lambda \log N) \cdot \mathsf{polylog}(\lambda_{\mathsf{DGS}})$.

With this setting of parameters, we obtain a semi-statically-secure distributed broadcast encryption scheme with the following parameters (for simplicity, we assume $N \geq \lambda$):

- **Public parameter size:** The public parameters pp have size $|\mathsf{pp}| = N^2 \cdot \mathsf{poly}(\lambda, \log N)$.
- **Public key size:** Each user's public key pk consists of a matrix $\mathbf{W} \in \mathbb{Z}_q^{n \times m}$ and $N - 1$ cross-terms $\mathbf{y}_j \in \mathbb{Z}_q^m$, so $|\mathsf{pk}| \leq (n + N)m \log q = N \cdot \mathsf{poly}(\lambda, \log N)$.
- **Secret key size:** The secret key for user $i \in [N]$ consists of a vector $\mathbf{y}_i \in \mathbb{Z}_q^m$, so $|\mathsf{sk}_i| = O(m \log q) = \mathsf{poly}(\lambda, \log N)$.
- **Ciphertext size:** The ciphertext for any set $S \subseteq [N]$ and message $\mu \in \{0, 1\}$ consists of of $2m + 1$ elements of $\mathbb{Z}_q$ and 2λ bits, so $|\mathsf{ct}| = \mathsf{poly}(\lambda, \log N)$.

Combined with Theorem 6.3, we now obtain an adaptively-secure distributed broadcast encryption scheme:

Corollary 6.24 (Adaptively-Secure Distributed Broadcast Encryption)**.** *Let λ be a security parameter and $N = N(\lambda)$ be any polynomial. Let $\ell \geq N \cdot \mathsf{poly}(\lambda, \log N)$. Under the ℓ-succinct LWE assumption with a sub-exponential modulus-to-noise ratio, there exists an adaptively secure distributed broadcast scheme in the random oracle model. The size of the ciphertext and a user's secret key is $\mathsf{poly}(\lambda, \log N)$. The the size of a user's public key is $N \cdot \mathsf{poly}(\lambda, \log N)$ and the size of the public parameters is $N^2 \cdot \mathsf{poly}(\lambda, \log N)$.*

Remark 6.25 (Adaptively-Secure Centralized Broadcast from ℓ-Succinct LWE in the ROM)**.** A distributed broadcast encryption scheme generically implies a centralized broadcast encryption scheme (with a long public key). Namely, the master public key for the centralized broadcast encryption scheme will consist of the public parameters pp for the distributed broadcast encryption scheme together with public keys for the N users. Thus, Construction 6.4 also implies an adaptively-secure centralized broadcast encryption scheme with $O(N^2)$-sized master public key.

7 Explainable Discrete Gaussian Preimage Sampler

In this section, we show how we can combine the preimage sampling algorithm of Gentry, Peikert, and Vaikuntanathan [GPV08] and the explainable discrete Gaussian sampler by Lu and Waters [LW22] to obtain an explainable discrete Gaussian preimage sampler as defined in Definition 4.1.

Notation. Throughout this section, we write $\mathbb{R}^+$ to denote the set of positive real numbers. Throughout this section, we will often describe algorithms (parameterized by a security parameter) as having real-valued inputs for ease of notation. In these cases, we assume that the input is represented as a value with $\Theta(\lambda)$ bits of precision. As usual, for $\sigma > 0$, we write $D_{\mathbb{Z},\sigma}$ to denote the discrete Gaussian distribution with width σ. For $c \in \mathbb{R}$ and $\sigma > 0$, we write $D_{\mathbb{Z},c,\sigma}$ to denote the discrete Gaussian distribution with center c and width σ.

Explainable discrete Gaussian sampler. We begin by recalling the explainable discrete Gaussian sampler from [LW22] that supports sampling a discrete Gaussian over the integers.

Theorem 7.1 (Explainable Discrete Gaussian Sampler over $\mathbb{Z}$ [LW22, Appendix B])**.** *Let λ be a security parameter. There exists a polynomial $\rho = \rho(\lambda)$ and a pair of efficient algorithms $(\mathsf{SampleDG}, \mathsf{ExplainDG})$ with the following syntax:*

- $\mathsf{SampleDG}(1^\lambda, \sigma, c; r) \to x$*: On input the security parameter λ, a width parameter $\sigma \in \mathbb{R}^+$, a center $c \in \mathbb{R}$, and randomness $r \in \{0,1\}^\rho$, the discrete Gaussian sampling algorithm outputs a sample $x \in \mathbb{Z}$.*
- $\mathsf{ExplainDG}(1^\lambda, 1^\kappa, \sigma, c, x) \to r$*: On input the security parameter λ, a precision parameter κ, a width parameter $\sigma \in \mathbb{R}^+$, a center $c \in \mathbb{R}$, and a value $x \in \mathbb{Z}$, the explain algorithm outputs randomness $r \in \{0,1\}^\rho$.*

Moreover, the algorithms satisfy the following properties:

- **Correctness:** *There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all $\log \lambda < \sigma < 2^\lambda$, and all $c \in \mathbb{R}$ where $|c| < 2^\lambda$, the statistical distance between the following two distributions is $\mathsf{negl}(\lambda)$:*

$$\left\{ x : \begin{array}{c} r \xleftarrow{\mathrm{R}} \{0,1\}^{\rho(\lambda)} \\ x \leftarrow \mathsf{SampleDG}(1^\lambda, \sigma, c; r) \end{array} \right\} \quad \text{and} \quad \left\{ x : x \leftarrow D_{\mathbb{Z},\sigma,c} \right\}.$$

  Moreover, for all $\lambda \in \mathbb{N}$, and all $c, \sigma \in \mathbb{R}$,

$$\Pr\left[|z - c| \leq \sigma\sqrt{\lambda} : z \leftarrow \mathsf{SampleDG}(1^\lambda, \sigma, c; r) \right] = 1.$$

- **Explainable:** *There exists a negligible function $\mathsf{negl}(\cdot)$ such for all $\lambda \in \mathbb{N}$, all $\log \lambda < \sigma < 2^\lambda$, and all $c \in \mathbb{R}$ where $|c| < 2^\lambda$, the statistical distance between the following distributions is $1/\kappa + \mathsf{negl}(\lambda)$:*

$$\left\{ (x, r) : \begin{array}{c} r \xleftarrow{\mathrm{R}} \{0,1\}^{\rho(\lambda)} \\ x \leftarrow \mathsf{SampleDG}(1^\lambda, \sigma, c; r) \end{array} \right\} \quad \text{and} \quad \left\{ (x, r) : \begin{array}{c} r' \xleftarrow{\mathrm{R}} \{0,1\}^{\rho(\lambda)} \\ x \leftarrow \mathsf{SampleDG}(1^\lambda, \sigma, c; r') \\ r \leftarrow \mathsf{ExplainDG}(1^\lambda, 1^\kappa, \sigma, c, x) \end{array} \right\}.$$

Ajtai trapdoors. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix and $\mathbf{y} \in \mathbb{Z}_q^n$ be a vector in the column-span of $\mathbf{A}$. Previously, Gentry, Peikert, Vaikuntanathan [GPV08] showed how to sample from the distribution $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ given a short basis $\mathbf{T} \in \mathbb{Z}^{m \times m}$ where $\mathbf{AT} = \mathbf{0} \bmod q$ and $\mathbf{T}$ is full rank over the *reals* (i.e., a short basis for the lattice $\Lambda^\perp(\mathbf{A}) := \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{0} \bmod q\}$). We give the formal definition below:

Definition 7.2 (Ajtai Trapdoor [Ajt96, adapted])**.** Let n, m, q be lattice parameters and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix. We say that a matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is an Ajtai-trapdoor for $\mathbf{A}$ if $\mathbf{AT} = \mathbf{0} \bmod q$ and $\mathbf{T}$ is full rank over $\mathbb{R}$.

Ajtai trapdoors from gadget trapdoors. Micciancio and Peikert [MP12, Lemma 5.3] showed that a gadget trapdoor for a matrix $\mathbf{A}$ directly implies an Ajtai trapdoor for the same matrix $\mathbf{A}$ of comparable quality. Technically, their work considers a slightly different formulation of gadget trapdoors (i.e., a short matrix $\mathbf{T}$ where $\mathbf{A}\left[\begin{smallmatrix}\mathbf{T}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}_n$) whereas in this work, we adopt the convention of taking a gadget trapdoor to be a short matrix $\mathbf{T}$ where $\mathbf{AT} = \mathbf{G}_n$ (without the extra identity matrix). Nonetheless, their approach still applies. We state the lemma below, and for completeness, include a proof of this statement in Appendix B.3.

Lemma 7.3 (Ajtai Trapdoor for Gadget Matrix [MP12, §4.2])**.** *Let n, q be lattice parameters and $m' = n\lceil \log q \rceil$. Then the gadget matrix $\mathbf{G}_n \in \mathbb{Z}_q^{n \times m'}$ has an Ajtai trapdoor $\mathbf{S}_n \in \mathbb{Z}^{m' \times m'}$ where $\|\mathbf{S}\| = 2$. Moreover, there is an efficient, explicit, and deterministic algorithm that computes $\mathbf{S}_n$ in $\mathrm{poly}(n, \log q)$ time.*

Lemma 7.4 (Gadget Trapdoor Implies Ajtai Trapdoor [MP12])**.** *Let n, q be lattice parameters and let $m' = n\lceil \log q \rceil$. Take any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T} \in \mathbb{Z}_q^{m \times m'}$ where $\mathbf{AT} = \mathbf{G}_n$. Let $\mathbf{T}' = [\mathbf{I}_m - \mathbf{T}\mathbf{G}_n^{-1}(\mathbf{A}) \mid \mathbf{T}\mathbf{S}_n] \in \mathbb{Z}_q^{m \times (m+m')}$, where $\mathbf{S}_n \in \mathbb{Z}_q^{m' \times m'}$ is an Ajtai trapdoor for $\mathbf{G}_n$ Then $\mathbf{AT}' = \mathbf{0} \bmod q$ and $\mathbf{T}'$ is full rank over $\mathbb{R}$.*

Preimage sampling using Ajtai trapdoors. The work of [GPV08] describes an efficient algorithm SamplePreGPV to efficiently sample from the distribution $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ given an Ajtai trapdoor for $\mathbf{A}$. From Lemma B.1, the distribution of $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ is precisely the distribution $\mathbf{x} + D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{x}}$, where $\mathbf{x}$ is an arbitrary solution to $\mathbf{Ax} = \mathbf{y}$. The work of [GPV08, §4.2] describe how to sample from the distribution $D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{x}}$ given an Ajtai trapdoor for $\mathbf{A}$, which immediately implies an algorithm for sampling from $\mathbf{A}_\sigma^{-1}(\mathbf{y})$:

Algorithm 3: The preimage sampling algorithm SamplePreGPV from [GPV08, §4.2, adapted].

```
SamplePreGPV($1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma$):
  1. If $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is not an Ajtai trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, abort.
  2. Use Gaussian elimination to compute a vector $\mathbf{x}^* \in \mathbb{Z}_q^m$ such that $\mathbf{Ax}^* = \mathbf{y}$. Compute the Gram-Schmidt
     orthogonalization $\tilde{\mathbf{T}}$ of matrix $\mathbf{T}$ (from left to right). Both of these steps are *deterministic*. Parse $\mathbf{T} =$
     $[\mathbf{t}_1 \mid \cdots \mid \mathbf{t}_m]$ and $\tilde{\mathbf{T}} = [\tilde{\mathbf{t}}_1 \mid \cdots \mid \tilde{\mathbf{t}}_m]$.
  3. Let $\mathbf{u}_m = \mathbf{0}$ and $\mathbf{c}_m = -\mathbf{x}^*$. For $i = m, m-1, \ldots, 1$, do:
     (a) Let $c_i' = \langle \mathbf{c}_i, \tilde{\mathbf{t}}_i \rangle / \langle \tilde{\mathbf{t}}_i, \tilde{\mathbf{t}}_i \rangle \in \mathbb{R}$ and $\sigma_i' = \sigma / \|\tilde{\mathbf{t}}_i\|_2 > 0$.
     (b) Sample $z_i \leftarrow D_{\mathbb{Z},\sigma_i',c_i'}$.
     (c) Let $\mathbf{c}_{i-1} = \mathbf{c}_i - z_i\mathbf{t}_i$ and $\mathbf{u}_{i-1} = \mathbf{u}_i + z_i\mathbf{t}_i$.
  4. Output $\mathbf{x}^* + \mathbf{u}_0$.
```

Theorem 7.5 (Preimage Sampling [GPV08])**.** *Let n, m, q be lattice parameters. There exist a negligible function $\mathrm{negl}(\cdot)$ such that for all $(\mathbf{A}, \mathbf{T})$ where $\mathbf{T} \in \mathbb{Z}^{m \times m}$ is an Ajtai trapdoor for $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and all targets $\mathbf{y}$ in the column space of $\mathbf{A}$, the following hold:*

- *For all $\sigma > 0$, the output $\mathbf{x} \leftarrow \mathsf{SamplePreGPV}(\mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$ satisfies $\mathbf{Ax} = \mathbf{y}$.*
- *For all $\sigma \geq \|\mathbf{T}\| \cdot \sqrt{m} \log m$, the statistical distance between the following distributions is $\mathrm{negl}(m)$:*

$$\{\mathbf{x} \leftarrow \mathsf{SamplePreGPV}(\mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)\} \quad \text{and} \quad \left\{\mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})\right\}.$$

Explainable sampling for the distribution $\mathbf{A}_\sigma^{-1}(\mathbf{y})$. Algorithm 3 essentially reduces the problem of sampling $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ to the problem of discrete Gaussian sampling over the integers. Thus, we can directly combine Algorithm 3 with the Lu-Waters explainable discrete Gaussian sampler over the integers (Theorem 7.1) to obtain an explainable discrete Gaussian sampler for sampling from $\mathbf{A}_\sigma^{-1}(\mathbf{y})$. We now describe the construction.

Construction 7.6 (Explainable Sampler for $\mathbf{A}_\sigma^{-1}(\mathbf{y})$)**.** Let (SampleDG, ExplainDG) be the explainable discrete Gaussian sampling algorithms described from Theorem 7.1, and let ρ_0 be the associated randomness bound. Let $\rho(\lambda, n, m, q) = m \cdot \rho_0(32\lambda m^3 \log q)$ and $\sigma_{\mathsf{loss}}(\lambda, n, m, q) = 18 m^{3/2} \log(m\lambda) \log\log q$. We construct an $(\rho, \sigma_{\mathsf{loss}})$-explainable discrete Gaussian preimage sampler as follows:

- $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r)$: 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 $\mathbf{y} \in \mathbb{Z}_q^n$, a width parameter $\sigma > 0$, and randomness $r \in \{0,1\}^\rho$, the sampler algorithm proceeds as follows:
  - Let $\lambda_0 = 32\lambda m^3 \log q$.
  - Let $\mathbf{T}_{\mathsf{Ajtai}} \in \mathbb{Z}_q^{m \times m}$ be the first m linearly-independent columns of $[\mathbf{I}_m - \mathbf{T}\mathbf{G}_n^{-1}(\mathbf{A}) \mid \mathbf{T}\mathbf{S}_n]$ where $\mathbf{S}_n$ is the Ajtai trapdoor for $\mathbf{G}_n$ from Lemma 7.3. Here, we consider linear independence over $\mathbb{R}$.
  - Let $r = r_1 \| \cdots \| r_m$ where $r_i \in \{0,1\}^{\rho/m}$. Run $\mathsf{SamplePreGPV}(1^{\lambda_0}, \mathbf{A}, \mathbf{T}_{\mathsf{Ajtai}}, \mathbf{y}, \sigma)$, except in Step 3b of Algorithm 3, sample $z_i \leftarrow \mathsf{SampleDG}(1^{\lambda_0}, \sigma_i', c_i'; r_i)$ for all $i \in [m]$.
- $\mathsf{Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{T}, \mathbf{y}, \mathbf{x}, \sigma)$: On input the security parameter λ, a precision parameter κ, the matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, a trapdoor $\mathbf{T} \in \mathbb{Z}_q^{m \times m'}$, a target $\mathbf{y} \in \mathbb{Z}_q^n$, a preimage $\mathbf{x} \in \mathbb{Z}^m$, and a width parameter $\sigma > 0$, the explain algorithm proceeds as follows:
  - If $\mathbf{A}\mathbf{x} \neq \mathbf{y} \bmod q$, output $\perp$. Otherwise, let $\lambda_0 = 32\lambda m^3 \log q$ and $\kappa_0 = m\kappa$.
  - Let $\mathbf{T}_{\mathsf{Ajtai}} \in \mathbb{Z}_q^{m \times m}$ be the first m linearly independent columns of $[\mathbf{I}_m - \mathbf{T}\mathbf{G}_n^{-1}(\mathbf{A}) \mid \mathbf{T}\mathbf{S}_n]$ where $\mathbf{S}_n$ is the Ajtai trapdoor for $\mathbf{G}_n$ from Lemma 7.3. Here, we consider linear independence over $\mathbb{R}$.
  - As in SamplePreGPV (Algorithm 3), deterministically compute the Gram-Schmidt orthogonalization $\tilde{\mathbf{T}}_{\mathsf{Ajtai}}$ of matrix $\mathbf{T}$ and the vector $\mathbf{x}^* \in \mathbb{Z}_q^m$ where $\mathbf{A}\mathbf{x}^* = \mathbf{y}$. Parse $\mathbf{T}_{\mathsf{Ajtai}} = [\mathbf{t}_1 \mid \cdots \mid \mathbf{t}_m]$ and $\tilde{\mathbf{T}}_{\mathsf{Ajtai}} = [\tilde{\mathbf{t}}_1 \mid \cdots \mid \tilde{\mathbf{t}}_m]$.
  - Let $\mathbf{u}_0 = \mathbf{x} - \mathbf{x}^*$. Compute $\mathbf{z} = \mathbf{T}_{\mathsf{Ajtai}}^{-1}\mathbf{u}_0$ over the *real* numbers. Abort if there exists any i where $z_i \notin \mathbb{Z}$. Otherwise, write $\mathbf{u}_0 = \sum_{i \in [m]} z_i \mathbf{t}_i$.
  - Let $\mathbf{c}_m = -\mathbf{x}^*$. For $i = m, m-1, \ldots, 1$, do:
    - Let $c_i' = \langle \mathbf{c}_i, \tilde{\mathbf{t}}_i \rangle / \langle \tilde{\mathbf{t}}_i, \tilde{\mathbf{t}}_i \rangle \in \mathbb{R}$ and $\sigma_i' = \sigma / \|\tilde{\mathbf{t}}_i\|_2 > 0$.
    - Compute $r_i \leftarrow \mathsf{ExplainDG}(1^{\lambda_0}, 1^{\kappa_0}, c_i', \sigma_i', z_i)$.
    - Let $\mathbf{c}_{i-1} = \mathbf{c}_i - z_i \mathbf{t}_i$.
  - Output $r = r_1 \| \cdots \| r_m$.

Theorem 7.7 (Correctness)**.** *For all functions $n \geq \lambda$, $m = \mathsf{poly}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T}$ where $\mathbf{A}\mathbf{T} = \mathbf{G}_n$, and all targets $\mathbf{y} \in \mathbb{Z}_q^n$ where $\|\mathbf{y}\| \leq 2^\lambda$, the following hold:*

- *For all $\sigma > 0$, the output $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$, satisfies $\mathbf{A}\mathbf{x} = \mathbf{y}$.*
- *For all width parameters $2^\lambda \geq \sigma \geq \|\mathbf{T}\| \cdot 18m^{3/2} \log(m\lambda) \log\log q$, the statistical distance between the following distributions is bounded by $\mathsf{negl}(\lambda)$:*

$$\left\{\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)\right\} \quad \textit{and} \quad \left\{\mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})\right\}$$

Proof. Take matrices $\mathbf{A}$ and $\mathbf{T}$ where $\mathbf{A}\mathbf{T} = \mathbf{G}$. Take any target $\mathbf{y} \in \mathbb{Z}_q^n$ where $\|\mathbf{y}\| \leq 2^\lambda$ and any width parameter $2^\lambda \geq \sigma \geq \|\mathbf{T}\| \cdot 18m^{3/2} \log(m\lambda) \log\log q$. Consider the output distribution of $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$. First, consider the matrix $\mathbf{T}_{\mathsf{Ajtai}}$ computed by SamplePre. By Lemma 7.4, $\mathbf{A} \cdot \mathbf{T}_{\mathsf{Ajtai}} = \mathbf{0} \bmod q$, and moreover, $\mathbf{T}_{\mathsf{Ajtai}}$ is linearly independent over $\mathbb{R}$. Thus $\mathbf{T}_{\mathsf{Ajtai}}$ is an Ajtai trapdoor for $\mathbf{A}$. The first requirement now follows by Theorem 7.5. For the second requirement, we start by showing that the intermediate variable c_i' and σ_i' chosen by SamplePre are properly bounded:

Lemma 7.8. *Take any matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and suppose* $\mathbf{T}$ *is a gadget trapdoor for* $\mathbf{A}$*. Take any target* $\mathbf{y} \in \mathbb{Z}_q^n$ *where* $\|\mathbf{y}\| \le 2^\lambda$ *and any width parameter* σ *where* $2^\lambda \ge \sigma \ge \|\mathbf{T}\| \cdot 18m^{3/2}\log(m\lambda)\log\log q$*. Then the centers* $c_i' \in \mathbb{R}$ *and width parameters* $\sigma_i' \in \mathbb{R}$ *for* $i \in [m]$ *chosen by* $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$ *satisfy* $|c_i'| < 2^{\lambda_0}$ *and* $\log\lambda_0 < \sigma_i' < 2^{\lambda_0}$.

Proof. First, we bound $\|\mathbf{T}_{\mathsf{Ajtai}}\|$. By Lemma 7.3, $\|\mathbf{S}\| \le 2$, so we conclude that $\|\mathbf{T}_{\mathsf{Ajtai}}\| \le 2m'\|\mathbf{T}\| \le 2m\|\mathbf{T}\|$. Let $\tilde{\mathbf{T}}_{\mathsf{Ajtai}} = [\tilde{\mathbf{t}}_1 \mid \cdots \mid \tilde{\mathbf{t}}_m]$ be the Gram-Schmidt orthogonalization of $\mathbf{T}_{\mathsf{Ajtai}}$. First, we bound σ_i'. By construction, $\sigma_i' = \sigma/\|\tilde{\mathbf{t}}_i\|_2$.

- For all $i \in [m]$, we have
$$\|\tilde{\mathbf{t}}_i\| \le \|\tilde{\mathbf{T}}_{\mathsf{Ajtai}}\| \le \|\mathbf{T}_{\mathsf{Ajtai}}\| \le 2m \cdot \|\mathbf{T}\| \le 2^{\lambda+1} \cdot m \tag{7.1}$$
This yields an upper bound
$$\|\tilde{\mathbf{t}}_i\|_2 \le \sqrt{m} \cdot \|\tilde{\mathbf{t}}_i\| \le \sqrt{m} \cdot \|\mathbf{T}_{\mathsf{Ajtai}}\| \le 2 \cdot \|\mathbf{T}\| \cdot m^{3/2} \le 2^{\lambda+1} \cdot m^{3/2}. \tag{7.2}$$
Thus, for all $i \in [m]$, we have
$$\sigma_i' = \frac{\sigma}{\|\tilde{\mathbf{t}}_i\|_2} \ge \frac{18\|\mathbf{T}\| \cdot m^{3/2}\log(m\lambda)\log\log q}{2\|\mathbf{T}\| \cdot m^{3/2}} \ge 9\log(m\lambda)\log\log q > \log\lambda_0 = \log(m\lambda) + 2\log m + \log\log q + 5.$$

- Next, $\tilde{\mathbf{T}}$ is an orthogonal basis for $\mathbb{R}^m$ and $\mathbf{T}$ is an integer matrix. Thus, $\prod_{i\in[m]} \|\tilde{\mathbf{t}}_i\|_2 = |\det(\tilde{\mathbf{T}})| = |\det(\mathbf{T})| \ge 1$. By Eq. (7.2), we now obtain the lower bound
$$\|\tilde{\mathbf{t}}_i\|_2 = \frac{|\det(\tilde{\mathbf{T}})|}{\prod_{j\ne i}\|\tilde{\mathbf{t}}_j\|_2} \ge \frac{1}{(2^{\lambda+1}m^{3/2})^{m-1}}. \tag{7.3}$$
Thus, for all $i \in [m]$, we have,
$$\sigma_i' = \frac{\sigma}{\|\tilde{\mathbf{t}}_i\|_2} \le 2^\lambda \cdot (2^{\lambda+1}m^{3/2})^{m-1} \le 2^{2m\lambda} \cdot 2^{3/2m\log m} \le 2^{4m^2\lambda} < 2^{\lambda_0}. \tag{7.4}$$

Next, we bound the center c_i' for each $i \in [m]$. Then, we have the following:

- First, $\mathbf{c}_m = -\mathbf{x}^*$, where $\mathbf{x}^* \in \mathbb{Z}_q^m$. Thus, $\|\mathbf{c}_m\| = \|\mathbf{x}^*\| \le q$.

- Next, for each $i \in [m]$, the sampler algorithm samples $z_i \leftarrow \mathsf{SampleDG}(1^{\lambda_0}, \sigma_i', c_i'; r_i)$. By Theorem 7.1, for all $i \in [m]$, it holds that $|z_i - c_i'| \le \sigma_i'\sqrt{\lambda_0}$.

- By construction, $\mathbf{c}_i = -\mathbf{x}^* - \sum_{j>i} z_i\mathbf{t}_i$. From Eqs. (7.1) and (7.4), $\|\mathbf{t}_j\| \le 2^{\lambda+1}m$ and $\sigma_j' \le 2^\lambda \cdot (2^{\lambda+1}m^{3/2})^{m-1}$ for all $j \in [m]$. This means
$$\begin{aligned}\|\mathbf{c}_i\| \le \|\mathbf{x}^*\| + \sum_{j>i}|z_j| \cdot \|\mathbf{t}_j\| &\le q + \sum_{j>i}(|c_j'| + \sigma_j'\sqrt{\lambda_0}) \cdot (2^{\lambda+1}m) \\ &\le q + m \cdot 2^\lambda \cdot \sqrt{\lambda_0} \cdot (2^{\lambda+1}m^{3/2})^m + \sum_{j=i+1}^{m}|c_j'| \cdot (2^{\lambda+1}m).\end{aligned}$$

- By definition of c_i' and using Eq. (7.3), we have for all $i \in [m]$,
$$\begin{aligned}|c_i'| = \frac{\langle \mathbf{c}_i, \tilde{\mathbf{t}}_i\rangle}{\langle \tilde{\mathbf{t}}_i, \tilde{\mathbf{t}}_i\rangle} &\le m \cdot \|\mathbf{c}_i\| \cdot \|\mathbf{T}_{\mathsf{Ajtai}}\| \cdot (2^{\lambda+1}m^{3/2})^{2(m-1)} \\ &\le m \cdot \|\mathbf{c}_i\| \cdot (2^{\lambda+1}m) \cdot (2^{\lambda+1}m^{3/2})^{2(m-1)} \\ &\le (q + 2^\lambda m\sqrt{\lambda_0}) \cdot (2^{\lambda+1}m^{3/2})^{3m-1} + \sum_{j=i+1}^{m}|c_j'| \cdot m \cdot (2^{\lambda+1}m)^2 \cdot (2^{\lambda+1}m^{3/2})^{2(m-1)} \\ &\le q \cdot \sqrt{\lambda_0} \cdot (2^{\lambda+1}m^{3/2})^{3m} + \sum_{j=i+1}^{m}|c_j'| \cdot (2^{\lambda+1}m^{3/2})^{2m}.\end{aligned}$$

In particular, this means that for all $i \in [m]$,

$$|c_i'| \leq q \cdot \sqrt{\lambda_0} \cdot (2^{\lambda+1} m^{3/2})^{3m} \cdot \sum_{j=i}^{m} (2^{\lambda+1} m^{3/2})^{2m(m-i)}.$$

Therefore, for all $i \in [m]$, we can bound

$$\begin{aligned}
|c_i'| &\leq q \cdot \sqrt{\lambda_0} \cdot (2^{\lambda+1} m^{3/2})^{3m} \cdot m \cdot (2^{\lambda+1} m^{3/2})^{2m^2-2m} \\
&\leq q\sqrt{\lambda_0} \cdot 2^{(\lambda+1)(2m^2+m)} m^{3m^2+1.5m} \\
&\leq q \cdot 6\lambda m^2 \log q \cdot 2^{6\lambda m^2} \cdot m^{4.5m^2} \\
&\leq 6q^2 \cdot 2^{7\lambda m^2} \cdot m^{4.5m^2} \\
&\leq 2^{3+7m^2\lambda+4.5m^2 \log m+2\log q} \\
&< 2^{32\lambda m^3 \log q} = 2^{\lambda_0}.
\end{aligned}$$

The lemma follows. □

Completing the proof of Theorem 7.7. Theorem 7.7 now follows from Lemma 7.8 and Theorems 7.1 and 7.5:

- By Lemma 7.8, for all targets $\mathbf{y} \in \mathbb{Z}_q^n$ where $\|\mathbf{y}\| \leq 2^\lambda$ and all width parameters σ where $2^\lambda \geq \sigma \geq \|\mathbf{T}\| \cdot 18m^{3/2} \log(m\lambda) \log\log q$, the centers $c_i' \in \mathbb{R}$ and width parameters $\sigma_i' \in \mathbb{R}$ for $i \in [m]$ chosen by $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$ satisfy $|c_i'| < 2^{\lambda_0}$ and $\log \lambda_0 < \sigma_i' < 2^{\lambda_0}$.
- By Theorem 7.1 this means that the statistical distance between the distributions $z_i \leftarrow D_{\mathbb{Z}, c_i, \sigma_i}$ and $z_i \leftarrow \mathsf{SampleDG}(1^{\lambda_0}, c_i, \sigma_i; r_i)$ can be bounded by a negligible function $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$. Note here that $\lambda_0 > \lambda$.
- Since SamplePreGPV samples z_i for each $i \in [m]$, the statistical distance between the output distribution of $\mathsf{SamplePreGPV}(\mathbf{A}, \mathbf{T}_{\mathsf{Ajtai}}, \mathbf{y}, \sigma)$ and that of $\mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma)$ is at most $m \cdot \varepsilon(\lambda)$, which is negligible when $m = \mathsf{poly}(\lambda)$.
- Finally, $\mathbf{T}_{\mathsf{Ajtai}}$ is an Ajtai trapdoor for $\mathbf{A}$ and $\|\mathbf{T}_{\mathsf{Ajtai}}\| < 2m\,\|\mathbf{T}\|$. By Theorem 7.5, for all $\sigma > \|\mathbf{T}\| \cdot 2m^{3/2} \log m \geq \left\|\mathbf{T}_{\mathsf{Ajtai}}\right\| \cdot \sqrt{m} \log m$, the distribution $\mathsf{SamplePreGPV}(\mathbf{A}, \mathbf{T}_{\mathsf{Ajtai}}, \mathbf{y}, \sigma)$ and $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ are statistically close. □

Theorem 7.9 (Explainability)**.** *There exist a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda, \kappa \in \mathbb{N}$*, all matrices* $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ *and* $\mathbf{T}$ *where* $\mathbf{AT} = \mathbf{G}_n$*, all targets* $\mathbf{y} \in \mathbb{Z}_q^n$ *where* $\|\mathbf{y}\| \leq 2^\lambda$*, and all width parameters* $2^\lambda > \sigma > \|\mathbf{T}\| \cdot 18m^{3/2} \log(m\lambda) \log\log q$*, the statistical distance between the following distributions is bounded by* $1/\kappa + \mathsf{negl}(\lambda)$*.*

- $\mathcal{D}_{\mathsf{SamplePre}}$*: Sample* $r \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ *and* $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r)$*. Output* $(\mathbf{x}, r)$*.*
- $\mathcal{D}_{\mathsf{Explain}}$*: Sample* $r' \xleftarrow{\mathrm{R}} \{0,1\}^\rho$*,* $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r')$*, and* $r \xleftarrow{\mathrm{R}} \mathsf{Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{T}, \mathbf{y}, \mathbf{x}, \sigma)$*. Output* $(\mathbf{x}, r)$*.*

Proof. Consider an execution of $\mathbf{x} \leftarrow \mathsf{SamplePre}(1^\lambda, \mathbf{A}, \mathbf{T}, \mathbf{y}, \sigma; r')$ and $r \xleftarrow{\mathrm{R}} \mathsf{Explain}(1^\lambda, 1^\kappa, \mathbf{A}, \mathbf{T}, \mathbf{y}, \mathbf{x}, \sigma)$. We start by arguing that the two algorithm compute the exact set of centers $c_1', \ldots, c_m'$ and widths $\sigma_1', \ldots, \sigma_m'$.

- Both SamplePre and Explain compute the *same* Gram-Schmidt orthogonalized basis $\tilde{\mathbf{T}}_0$ and solution $\mathbf{x}^*$ (since these steps are deterministic).
- By construction, SamplePre outputs $\mathbf{x} = \mathbf{x}^* + \mathbf{u}_0$ and $\mathbf{u}_0 = \sum_{i \in [n]} z_i \mathbf{t}_i$, where z_i are the coefficients it sampled. Since $\mathbf{T}_{\mathsf{Ajtai}}$ is a basis for $\mathbb{R}^m$, given $\mathbf{u}_0 \in \mathbb{R}^m$, the decomposition $\mathbf{u}_0 = \sum_{i \in [m]} z_i \mathbf{t}_i$ is unique. On the other hand, the Explain algorithm computes the coefficients z_i such that $\mathbf{x} - \mathbf{x}^* = \sum_{i \in [m]} z_i \mathbf{t}_i$. Therefore Explain computes the same coefficients $z_1, \ldots, z_m$ as those originally sampled by SamplePre.

- Since SamplePre and Explain both set $\mathbf{c}_m^* = -\mathbf{x}^*$, and moreover, the values of $\mathbf{c}_1, \ldots, \mathbf{c}_m$ are fully determined by $z_1, \ldots, z_m$ and $\mathbf{T}_{\mathsf{Ajtai}}$, we conclude that the two algorithms also compute the same set of $\mathbf{c}_1, \ldots, \mathbf{c}_m$.
- Since the centers c_i' and the widths σ_i' are completely defined by $\mathbf{c}_i$ and $\tilde{\mathbf{T}}_0$, the variables are computed in an identical manner in SamplePre and Explain.

By Lemma 7.8, we have that $|c_i'| \le 2^{\lambda_0}$ and $\log \lambda_0 < \sigma_i' < 2^{\lambda_0}$ for all $i \in [m]$. By Theorem 7.1, there exists a negligible function $\mathsf{negl}'(\cdot)$ such that the following two distributions have at most $1/\kappa_0 + \mathsf{negl}'(\lambda_0)$ statistical distance.

- Sample $r_i \xleftarrow{\mathrm{R}} \{0,1\}^{\rho/m}$ and $z_i \leftarrow \mathsf{SampleDG}(1^\lambda, c_i', \sigma_i'; r_i)$ and output (r_i, z_i).
- Sample $r_i' \xleftarrow{\mathrm{R}} \{0,1\}^{\rho/m}$, $z_i \leftarrow \mathsf{SampleDG}(1^\lambda, c_i', \sigma_i'; r_i')$, $r_i \xleftarrow{\mathrm{R}} \mathsf{ExplainDG}(1^\lambda, 1^{\kappa'}, c_i', \sigma_i', z_i)$, and output (r_i, z_i).

The first distribution corresponds to the joint distribution of (r_i, z_i) in SamplePre while the second distribution corresponds to that of (r_i, z_i) in Explain. Since there are m such pairs, the statistical distance between the distribution of $(r_1, \ldots, r_m, z_1, \ldots, z_m)$ from SamplePre and $(r_1, \ldots, r_m, z_1, \ldots, z_m)$ from Explain is at most $m/\kappa_0 + m \cdot \mathsf{negl}'(\lambda_0) = 1/\kappa + \mathsf{negl}(\lambda)$. Since the distribution of $(r_1, \ldots, r_m, z_1, \ldots, z_m)$ in the two distributions uniquely determine $(\mathbf{x}, r)$ and vice versa, the claim follows. □

Acknowledgements

We thank Brent Waters and Hoeteck Wee for many helpful discussions on the ℓ-succinct LWE assumption and registered ABE. 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. *IACR Cryptol. ePrint Arch.*, 2025.

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

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In *TCC*, 2020.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In *EUROCRYPT*, 2020.

[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.

[BCTW16] Zvika Brakerski, David Cash, Rotem Tsabary, and Hoeteck Wee. Targeted homomorphic attribute-based encryption. In *TCC*, 2016.

[BDJ$^+$24] Pedro Branco, Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Pseudorandom obfuscation and applications. *IACR Cryptol. ePrint Arch.*, 2024.

[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.

[BTVW17] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *TCC*, 2017.

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In *ASIACRYPT*, 2024.

[BV15] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions - or: How to secretly embed a circuit in your PRF. In *TCC*, 2015.

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. In *ITCS*, 2022.

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

[CES21] Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. Optimizing registration based encryption. In *Cryptography and Coding*, 2021.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, 2003.

[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.

[DKL⁺23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In *EUROCRYPT*, 2023.

[DKW21] Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority ABE for dnfs from LWE. In *EUROCRYPT*, 2021.

[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.

[FKdP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In *ASIACRYPT*, 2023.

[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.

[GHM⁺19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In *PKC*, 2019.

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

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In *ACM CCS*, 2023.

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

[GMPW20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In *PKC*, 2020.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2), 1988.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO*, 1986.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *STOC*, 2021.

[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.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In *CRYPTO*, 2020.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

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

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4), 1999.

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *FOCS*, 2023.

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

[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 (2)*, Lecture Notes in Computer Science, 2022.

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

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. In *FOCS*, 2004.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, 1999.

[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.

[RW15] Yannis Rouselakis and Brent Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In *Financial Cryptography and Data Security*, 2015.

[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.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In *CRYPTO*, 2022.

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-io from evasive LWE. In *ASIACRYPT*, 2022.

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

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In *EUROCRYPT*, 2022.

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

[WQZDF10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption. In *ACM CCS*, 2010.

[WW23a] Hoeteck Wee and David J. Wu. Lattice-based functional commitments: Fast verification and cryptanalysis. In *ASIACRYPT*, 2023.

[WW23b] Hoeteck Wee and David J. Wu. Succinct vector, polynomial, and functional commitments from lattices. In *EUROCRYPT*, 2023.

[WWW22] Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In *TCC*, 2022.

[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.

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

A Registered Attribute-Based Encryption Definitions

In this section, we give the formal definition of key-policy registered ABE adapted from [HLWW23]. For full generality, we decouple the policy-family parameter from the security parameter (i.e., allow these to be set independently). We consider the same relaxations from Section 5.1 (where the key-generation algorithm is allowed to depend on the policy).

Definition A.1 (Registered Attribute-Based Encryption [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 $\mathcal{P} = \{\mathcal{P}_\tau\}_{\tau \in \mathbb{N}}$ be a set of policies on $\mathcal{X}$ (where each $P \in \mathcal{P}_\tau$ is a mapping $P \colon \mathcal{X}_\tau \to \{0, 1\}$). A registered *key-policy* attribute-based encryption scheme with attribute space $\mathcal{X}$ and policy space $\mathcal{P}$ consists of a tuple of efficient algorithms $\Pi_{\mathsf{RABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Register}, \mathsf{Encrypt}, \mathsf{Update}, \mathsf{Decrypt})$ with the following properties:

- $\mathsf{Setup}(1^\lambda, 1^\tau) \to \mathsf{crs}$: On input the security parameter λ and the policy-family parameter τ, the setup algorithm outputs a common reference string crs. We assume crs contains an implicit description of 1^λ and 1^τ.

- $\mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux}, P) \to (\mathsf{pk}, \mathsf{sk})$: On input the common reference string crs, auxiliary state aux, and a decryption policy $P \in \mathcal{P}_\tau$, the key-generation algorithm outputs a public key pk and a secret key sk.

- $\mathsf{Register}(\mathsf{crs}, \mathsf{aux}, P, \mathsf{pk}) \to (\mathsf{mpk}, \mathsf{aux}')$: On input the common reference string crs, auxiliary state aux, a decryption policy $P \in \mathcal{P}_\tau$, and a public key pk, the registration algorithm *deterministically* outputs the master public key mpk and an updated state aux′. We assume mpk also contains an implicit description of 1^λ and 1^τ.

- $\mathsf{Encrypt}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$: 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.

- $\mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}) \to \mathsf{hsk}$: On input the common reference string crs, auxiliary state aux, and a public key pk, the update algorithm *deterministically* outputs a helper decryption key hsk.

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{hsk}, x, \mathsf{ct}) \to \mu$: On input the master public key mpk, a secret key sk, a helper decryption key hsk, an attribute $x \in \mathcal{X}_\tau$, and a ciphertext ct, the decryption algorithm either outputs a message $\mu \in \{0, 1\}$ or a special flag $\mu = \mathsf{GetUpdate}$ to indicate an updated helper decryption key is needed to decrypt. This algorithm is *deterministic.*

Definition A.2 (Bounded Registered ABE [HLWW23, Definition 4.4])**.** We say that a registered ABE scheme Π_{RABE} is *bounded* if there is an a priori bound on the number of registered users in the system. In this setting, the Setup algorithm takes as input a bound parameter 1^N which specifies the maximum number of registered users the scheme supports. In the correctness and security definitions (Definitions A.3 and A.4), the adversary specifies the bound 1^N at the beginning of the correctness or security game, and moreover, the adversary in the game can make a maximum of N registration queries.

Correctness and security requirements. We now define the correctness and efficiency requirements of a registered ABE scheme. Our definitions are essentially the same as those from [HLWW23], just adapted to the key-policy setting.

Definition A.3 (Correctness and Efficiency of Registered ABE)**.** Let $\Pi_{\mathsf{RABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Register}, \mathsf{Encrypt}, \mathsf{Update}, \mathsf{Decrypt})$ be a registered key-policy ABE scheme with attribute space $\mathcal{X} = \{\mathcal{X}_\tau\}_{\tau \in \mathbb{N}}$ and policy space $\mathcal{P} = \{\mathcal{P}_\tau\}_{\tau \in \mathbb{N}}$. For a security parameter λ and an adversary $\mathcal{A}$, we define the correctness experiment as follows:

- **Setup phase:** On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs the policy family parameter 1^τ. The challenger samples the common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\tau)$ and gives crs to $\mathcal{A}$. The challenger also initializes $\mathsf{aux} = \perp$. and two counters $\mathsf{ctr}[\mathsf{reg}] = 0$ to keep track of the number of registration queries and $\mathsf{ctr}[\mathsf{enc}] = 0$ to keep track of the number of encryption queries. Finally, it initializes $\mathsf{ctr}[\mathsf{reg}]^* = \infty$ as the index for the target key (which will also be updated during the game).

- **Query phase:** During the query phase, the adversary $\mathcal{A}$ is able to make the following queries:
  - **Register non-target key query:** In a non-target-key registration query, the adversary $\mathcal{A}$ specifies a public key pk and a policy $P \in \mathcal{P}_\tau$. The challenger increments the counter $\mathsf{ctr}[\mathsf{reg}] = \mathsf{ctr}[\mathsf{reg}] + 1$ and registers the key by computing $(\mathsf{mpk}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux}') = \mathsf{Register}(\mathsf{crs}, \mathsf{aux}, P, \mathsf{pk})$. The challenger updates its auxiliary data by setting $\mathsf{aux} = \mathsf{aux}'$ and replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{reg}], \mathsf{mpk}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux})$.
  - **Register target key query:** In a target-key registration query, the adversary specifies a policy $P^* \in \mathcal{P}_\tau$. If $\mathsf{ctr}[\mathsf{reg}]^* \neq \infty$, then the challenger replies with $\perp$. Otherwise, the challenger increments the counter $\mathsf{ctr}[\mathsf{reg}] = \mathsf{ctr}[\mathsf{reg}] + 1$, samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux}, P^*)$, and registers $(\mathsf{mpk}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux}') = \mathsf{Register}(\mathsf{crs}, \mathsf{aux}, P^*, \mathsf{pk})$. It computes the helper decryption key $\mathsf{hsk}^* = \mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$. The challenger updates its auxiliary data by setting $\mathsf{aux} = \mathsf{aux}'$, stores the index of the target identity $\mathsf{ctr}[\mathsf{reg}]^* = \mathsf{ctr}[\mathsf{reg}]$, and replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{reg}], \mathsf{mpk}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux}, \mathsf{pk}^*, \mathsf{hsk}^*, \mathsf{sk}^*)$.

  - **Encryption query:** In an encryption query, the adversary submits the index $\mathsf{ctr}[\mathsf{reg}]^* \le i \le \mathsf{ctr}[\mathsf{reg}]$ of a public key, a message $\mu_{\mathsf{ctr}[\mathsf{enc}]} \in \{0,1\}$, and an attribute $x_{\mathsf{ctr}[\mathsf{enc}]} \in \mathcal{X}_\tau$. If the adversary has not yet registered a target key, or if $P^*(x_{\mathsf{ctr}[\mathsf{enc}]}) = 0$, then the challenger replies with $\perp$. Otherwise, the challenger increments the counter $\mathsf{ctr}[\mathsf{enc}] = \mathsf{ctr}[\mathsf{enc}]+1$ and computes $\mathsf{ct}_{\mathsf{ctr}[\mathsf{enc}]} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}_i, x_{\mathsf{ctr}[\mathsf{enc}]}, \mu_{\mathsf{ctr}[\mathsf{enc}]})$. The challenger replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{enc}], \mathsf{ct}_{\mathsf{ctr}[\mathsf{enc}]})$.
  - **Decryption query:** In a decryption query, the adversary submits a ciphertext index $1 \le j \le \mathsf{ctr}[\mathsf{enc}]$. The challenger computes $\mu'_j = \mathsf{Decrypt}(\mathsf{sk}^*, \mathsf{hsk}^*, x_j, \mathsf{ct}_j)$. If $\mu'_j = \mathsf{GetUpdate}$, then the challenger computes $\mathsf{hsk}^* = \mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$ and recomputes $\mu'_j = \mathsf{Decrypt}(\mathsf{sk}^*, \mathsf{hsk}^*, x_j, \mathsf{ct}_j)$. If $\mu'_j \ne \mu_j$, the experiment halts with output $b = 1$.

  If the adversary has finished making queries and the experiment has not halted (as a result of a decryption query), then the experiment outputs $b = 0$.

We say that Π_{RABE} is correct and efficient if for all (possibly unbounded) adversaries $\mathcal{A}$ making at most a polynomial number of queries, the following properties hold:

- **Correctness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the correctness game.
- **Compactness:** Let N be the number of registration queries the adversary makes in the above game. There exists a universal polynomial $\mathsf{poly}(\cdot)$ such that for all $i \in [N]$, $|\mathsf{mpk}_i| = \mathsf{poly}(\lambda + \log i)$. We also require that the size of the helper decryption key hsk^* satisfy $|\mathsf{hsk}^*| = \mathsf{poly}(\lambda + \log N)$ (at *all* points in the game).
- **Update efficiency:** Let N be the number of registration queries the adversary makes in the above game. Then, in the course of the above game, the challenger invokes the update algorithm Update at most $O(\log N)$ times, where each invocation runs in $\mathsf{poly}(\log N)$ time in the RAM model of computation. Specifically, we model Update as a RAM program that has *random* access to its input; thus, the running time of Update in the RAM model can be *smaller* than the input length.

Definition A.4 (Security of Registered ABE). Let $\Pi_{\mathsf{RABE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Register}, \mathsf{Encrypt}, \mathsf{Update}, \mathsf{Decrypt})$ be a registered key-policy ABE scheme with attribute space $\mathcal{X} = \{\mathcal{X}_\tau\}_{\tau\in\mathbb{N}}$ and policy space $\mathcal{P} = \{\mathcal{P}_\tau\}_{\tau\in\mathbb{N}}$. For a security parameter λ, an adversary $\mathcal{A}$, and a bit $b \in \{0,1\}$, we define the following game between $\mathcal{A}$ and the challenger:

- **Setup phase:** On input the security parameter 1^λ, the adversary $\mathcal{A}$ outputs the policy family parameter 1^τ. The challenger samples the common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\tau)$ and gives crs to $\mathcal{A}$. It then initializes the auxiliary input $\mathsf{aux} = \perp$, a counter $\mathsf{ctr} = 0$ for the number of honest-key-registration queries the adversary has made, an empty set of keys $C = \varnothing$ (to keep track of corrupted public keys), and an empty dictionary mapping public keys to registered attribute sets $\mathrm{D} = \varnothing$. For notational convenience, if $\mathsf{pk} \notin \mathrm{D}$, then we define $\mathrm{D}[\mathsf{pk}] := \varnothing$. to be the empty set.
- **Query phase:** Adversary $\mathcal{A}$ can now issue the following queries:
  - **Register corrupted key query:** In a corrupted-key-registration query, the adversary $\mathcal{A}$ specifies a public key pk and a policy $P \in \mathcal{P}_\tau$. The challenger registers the key by computing $(\mathsf{mpk}', \mathsf{aux}') = \mathsf{Register}(\mathsf{crs}, \mathsf{aux}, P, \mathsf{pk})$. The challenger updates its copy of the public key $\mathsf{mpk} = \mathsf{mpk}'$, its auxiliary data $\mathsf{aux} = \mathsf{aux}'$, and adds pk to C. Finally, it updates $D[\mathsf{pk}] = D[\mathsf{pk}] \cup \{P\}$. It replies to $\mathcal{A}$ with $(\mathsf{mpk}', \mathsf{aux}')$.
  - **Register honest key query:** In an honest-key-registration query, the adversary specifies a policy $P \in \mathcal{P}_\tau$. The challenger increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$ and samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux}, P)$, and registers $(\mathsf{mpk}', \mathsf{aux}') = \mathsf{Register}(\mathsf{crs}, \mathsf{aux}, P, \mathsf{pk}_{\mathsf{ctr}})$. The challenger updates its public key $\mathsf{mpk} = \mathsf{mpk}'$, its auxiliary data $\mathsf{aux} = \mathsf{aux}'$, and $D[\mathsf{pk}_{\mathsf{ctr}}] = D[\mathsf{pk}_{\mathsf{ctr}}] \cup \{P\}$. It replies to $\mathcal{A}$ with $(\mathsf{ctr}, \mathsf{mpk}', \mathsf{aux}', \mathsf{pk}_{\mathsf{ctr}})$.
  - **Corrupt honest key query:** In a corrupt-honest-key query, the adversary specifies an index $1 \le i \le \mathsf{ctr}$. Let $(\mathsf{pk}_i, \mathsf{sk}_i)$ be the i^{th} public/secret key the challenger samples when responding to the i^{th} honest-key-registration query. The challenger adds pk_i to C and replies to $\mathcal{A}$ with sk_i.

- **Challenge phase:** The adversary $\mathcal{A}$ specifies an attribute $x^* \in \mathcal{X}_\tau$ and the challenger replies with the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, x^*, b)$.
- **Output phase:** At the end of the game, algorithm $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

Let $S = \{P \in \mathsf{D}[\mathsf{pk}] : \mathsf{pk} \in C\}$ be the set of policies associated with corrupted public keys. We say that an adversary $\mathcal{A}$ is admissible if for all $P \in S$, it holds that $P(x^*) = 0$. We say that a registered ABE scheme is secure if for all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, we have that $|\Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1]| = \mathsf{negl}(\lambda)$ in the above security game.

Remark A.5 (Weaker Notions of Security)**.** Similar to Definitions 5.3 and 5.4, we can consider weaker security notions such as attribute-selective security (where the adversary in Definition A.4 has to commit to its challenge attribute x^* at the beginning of the security game) and security without corruptions (where the adversary in Definition A.4 is not allowed to make any corruption queries). Moreover, the transformations described in Remark 5.5 can be leveraged to achieve full adaptive security.

The [HLWW23] transformation. As mentioned above, the work of [HLWW23] shows how to generically compile a slotted registered ABE scheme (e.g., Definition 5.1) into a standard registered ABE scheme (Definition A.1). The transformation still applies with the relaxation of registered ABE we consider (Remark 5.2) where we allow the key-generation algorithm to take the policy as input, provided that we apply the relaxation to both the slotted registered ABE scheme and the normal registered ABE scheme. We state the main theorem below:

Theorem A.6 (Registered ABE from Slotted Registered ABE [HLWW23, §6])**.** *Suppose there exists a slotted registered ABE scheme with attribute space $\mathcal{X}$ and policy space $\mathcal{P}$. Then, there is a registered ABE scheme with the same attribute space $\mathcal{X}$ and policy space $\mathcal{P}$. The transformation preserves the security properties (e.g., adaptive security, attribute-selective security, or security without corruption queries) of the slotted scheme. If the CRS size of the slotted scheme is polylogarithmic in the number of slots, then the transformed scheme supports an unbounded number of users; otherwise, the transformed scheme supports an a priori bounded number of users (Definition A.2).*

B Additional Lattice Properties

In this section, we recall some additional lattice preliminaries and then give the proofs of Theorem 4.3 and Lemma 7.4.

Lattices. For a positive integer m, a lattice $\Lambda \subset \mathbb{R}^m$ is a discrete additive subgroup of $\mathbb{R}^m$. We say Λ is full-rank if Λ is generated as the set of all integer linear combinations of m linearly-independent basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^m$. For a lattice $\Lambda \subset \mathbb{R}^m$, the dual lattice is defined to be $\Lambda^* = \{\mathbf{w} \in \mathbb{R}^m \mid \forall \mathbf{x} \in \Lambda : \mathbf{w}^\mathsf{T}\mathbf{x} \in \mathbb{Z}\}$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define the full-rank q-ary lattices

$$\Lambda^\perp(\mathbf{A}) := \left\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\right\} \subseteq \mathbb{Z}^m$$
$$\Lambda(\mathbf{A}) := \left\{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{A}^\mathsf{T}\mathbf{x} \bmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^m\right\}.$$

By definition, $\Lambda(\mathbf{A}) = q \cdot (\Lambda^\perp(\mathbf{A}))^*$. For a vector $\mathbf{x} \in \mathbb{R}^m$ and a lattice $\Lambda \subset \mathbb{R}^m$, we write $\mathbf{x} + \Lambda$ to denote the coset $\{\mathbf{x} + \mathbf{y} : \mathbf{y} \in \Lambda\}$ of Λ associated with $\mathbf{x}$. We write $\lambda_1^\infty(\Lambda) := \min_{\mathbf{0} \neq \mathbf{v} \in \Lambda} \|\mathbf{v}\|$ to denote the ℓ_∞-norm of the shortest non-zero vector in Λ.

Discrete Gaussians over lattices. For a Gaussian width parameter $\sigma > 0$ and a center $\mathbf{c} \in \mathbb{R}^m$ we write $\rho_\sigma : \mathbb{R}^m \to \mathbb{R}$ to denote the Gaussian function $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) := \exp(-\pi\|\mathbf{x} - \mathbf{c}\|_2^2/\sigma^2)$. When $\mathbf{c} = \mathbf{0}$, we simply write $\rho_\sigma(\mathbf{x}) := \rho_{\sigma,\mathbf{0}}(\mathbf{x})$. For a lattice coset $\mathbf{x} + \Lambda$, we define $\rho_{\sigma,\mathbf{c}}(\mathbf{x} + \Lambda) := \sum_{\mathbf{y} \in \mathbf{x}+\Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{y})$. The discrete Gaussian distribution $D_{\mathbb{Z},\sigma,c}$ with width $\sigma > 0$ and center $c \in \mathbb{R}$ is defined to be $D_{\mathbb{Z},\sigma,c}(x) := \rho_{\sigma,c}(x)/\rho_{\sigma,c}(\mathbb{Z})$ for all $x \in \mathbb{Z}$. We write $D_{\mathbb{Z},\sigma} := D_{\mathbb{Z},\sigma,0}$. In particular, $D_{\mathbb{Z}^m,\sigma} \equiv D_{\mathbb{Z},\sigma}^m$. More generally, we define the discrete Gaussian distribution $D_{\mathbf{x}+\Lambda,\sigma,\mathbf{c}}$ over a lattice coset $\mathbf{x} + \Lambda$ with width σ and center $\mathbf{c} \in \mathbb{R}^m$ to be the distribution

$$D_{\mathbf{x}+\Lambda,\sigma,\mathbf{c}}(\mathbf{y}) := \begin{cases} \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(\mathbf{y}+\Lambda)} & \mathbf{y} \in \mathbf{x} + \Lambda \\ 0 & \text{otherwise.} \end{cases}$$

As usual, when $\mathbf{c} = \mathbf{0}$, we simply write $D_{\mathbf{x}+\Lambda,\sigma} := D_{\mathbf{x}+\Lambda,\sigma,\mathbf{0}}$.

Lemma B.1 (Distribution $\mathbf{A}_\sigma^{-1}(\mathbf{y})$ [GPV08, Lemma 5.2])**.** *Take any matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, any $\mathbf{y} \in \mathbb{Z}_q^n$ in the column-span of $\mathbf{A}$, and any $\mathbf{x}^* \in \mathbb{Z}_q^m$ where $\mathbf{A}\mathbf{x}^* = \mathbf{y}$. Then $\mathbf{A}_\sigma^{-1}(\mathbf{y}) \equiv \mathbf{x}^* + D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{x}^*}$.*

The smoothing parameter. Next, we recall the notion of the smoothing parameter [MR04]. For an m-dimensional lattice Λ and a positive real number $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest real value $\sigma > 0$ such that $\rho_{1/\sigma}(\Lambda^*) \le 1 + \varepsilon$. We now state some properties on the smoothing parameter:

Lemma B.2 (Smoothing Parameter [MR04, Lemma 4.4, implicit])**.** *Let $\Lambda \subset \mathbb{R}^m$ be a lattice. Then, for all $\varepsilon \in (0,1)$, all $\sigma \ge \eta_\varepsilon(\Lambda)$, and all $\mathbf{c} \in \mathbb{R}^m$, $\rho_{s,\mathbf{c}}(\Lambda) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_s(\Lambda)$.*

Lemma B.3 (Smoothing Parameter Bound [GPV08, Lemma 5.3])**.** *Let n, m, q be lattice parameters with q prime and $m \ge 2n \log q$. Then, there is a negligible function $\varepsilon(m) = \mathsf{negl}(m)$ such that for all but a q^{-n}-fraction of matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, it holds that $\eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \le \log m$.*

Subgaussian random variables. Next, we recall the concept of a subgaussian random variable; our presentation is adapted from [MP12, §2.4]. For $\delta \ge 0$, a random variable X with values in $\mathbb{R}$ is δ-subgaussian with parameter $\sigma > 0$ if for all $t \in \mathbb{R}$,

$$\mathbb{E}[\exp(2\pi t X)] \le \exp(\delta) \cdot \exp(\pi\sigma^2 t^2).$$

By Markov's inequality, if X is δ-subgaussian, then

$$\Pr[|X| \ge t] \le 2\exp(\delta)\exp(-\pi t^2/\sigma^2). \tag{B.1}$$

A vector-valued random variable $\mathbf{x} \in \mathbb{R}^m$ is δ-subgaussian with parameter σ if for all vectors $\mathbf{v} \in \mathbb{R}^m$ where $\|\mathbf{v}\|_2 = 1$, the distribution $\mathbf{x}^\intercal\mathbf{v}$ is δ-subgaussian with parameter σ. The work of [MP12] showed that the discrete Gaussian distribution over any lattice coset is subgaussian:

Lemma B.4 (Discrete Gaussians are Subgaussian [MP12, Lemma 2.8])**.** *Let $\Lambda \subset \mathbb{R}^m$ be a full-rank lattice and suppose $\sigma \ge \eta_\varepsilon(\Lambda)$ for some $\varepsilon \in (0,1)$. Then, for all $\mathbf{c} \in \mathbb{R}^m$, the distribution $D_{\mathbf{c}+\Lambda,\sigma}$ is $\ln\left(\frac{1+\varepsilon}{1-\varepsilon}\right)$-subgaussian with parameter σ.*

Kullback-Leibler divergence and Pinsker's inequality. For discrete probability distributions $\mathcal{D}, \mathcal{D}'$ over a common support $\mathcal{X}$, their Kullback-Leibler divergence is defined to be $D_{\mathsf{KL}}(\mathcal{D}\|\mathcal{D}') := \sum_{x\in\mathcal{X}} \mathcal{D}(x) \ln \frac{\mathcal{D}(x)}{\mathcal{D}'(x)}$. Next, Pinsker's inequality relates the statistical distance between two distributions to their Kullback-Leibler divergence:

Fact B.5 (Pinsker's Inequality)**.** Let $\mathcal{D}, \mathcal{D}'$ be discrete probability distributions with a common support. Then,

$$\Delta(\mathcal{D}, \mathcal{D}') \le \sqrt{\frac{1}{2} D_{\mathsf{KL}}(\mathcal{D}\|\mathcal{D}')},$$

where $\Delta(\mathcal{D}, \mathcal{D}')$ denotes the statistical distance between $\mathcal{D}$ and $\mathcal{D}'$.

B.1 Proof of Theorem 4.3 (Smudging Lemma)

We now give the proof of Theorem 4.3. By Lemma B.3, there exists a negligible function $\varepsilon(m) = \mathsf{negl}(m)$ such that for all but a q^{-n}-fraction of matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, $\eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \le \log m$. In the remainder of this proof, we restrict our attention to matrices $\mathbf{A}$ where $\eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \le \log m$. Take any vector $\mathbf{y} \in \mathbb{Z}_q^n$ in the column span of $\mathbf{A}$, any vector $\mathbf{z} \in \mathbb{Z}_q^m$, and $\sigma \ge \log m$. Let $\mathbf{t} \in \mathbb{Z}_q^m$ be an arbitrary vector where $\mathbf{A}\mathbf{t} = \mathbf{y}$. By Lemma B.1,

$$\begin{aligned} \mathbf{A}_\sigma^{-1}(\mathbf{y} + \mathbf{A}\mathbf{z}) &\equiv \mathbf{t} + \mathbf{z} + D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{t}-\mathbf{z}} \\ \mathbf{z} + \mathbf{A}_\sigma^{-1}(\mathbf{y}) &\equiv \mathbf{z} + \mathbf{t} + D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{t}}. \end{aligned}$$

The statistical distance between $\mathbf{A}_\sigma^{-1}(\mathbf{y} + \mathbf{A}\mathbf{z})$ and $\mathbf{z} + \mathbf{A}_\sigma^{-1}(\mathbf{y})$ is thus the statistical distance between the following distributions:

$$D_1 \coloneqq D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{t}} \quad \text{and} \quad D_2 \coloneqq D_{\Lambda^\perp(\mathbf{A}),\sigma,-\mathbf{t}-\mathbf{z}}.$$

We start by computing the Kullback-Leibler divergence between these two distributions:

$$\begin{aligned}
D_{\mathsf{KL}}(D_1 \| D_2) &= \sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} D_1(\mathbf{x}) \log \frac{D_1(\mathbf{x})}{D_2(\mathbf{x})} \\
&= \sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} D_1(\mathbf{x}) \log \frac{\rho_{\sigma,-\mathbf{t}}(\mathbf{x}) / \rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))}{\rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\mathbf{x}) / \rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\Lambda^\perp(\mathbf{A}))} \\
&= \frac{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))}{\rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\Lambda^\perp(\mathbf{A}))} \sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} D_1(\mathbf{x}) \log \frac{\exp(-\pi \left\|\mathbf{x} + \mathbf{t}\right\|_2^2 / \sigma^2)}{\exp(-\pi \left\|\mathbf{x} + \mathbf{t} + \mathbf{z}\right\|_2^2 / \sigma^2)} \\
&= \frac{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))}{\rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\Lambda^\perp(\mathbf{A}))} \frac{\pi}{\sigma^2} \sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} D_1(\mathbf{x}) \left(2(\mathbf{x} + \mathbf{t})^\mathsf{T}\mathbf{z} + \mathbf{z}^\mathsf{T}\mathbf{z}\right) \\
&= \frac{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))}{\rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\Lambda^\perp(\mathbf{A}))} \left(\frac{\pi \left\|\mathbf{z}\right\|_2^2}{\sigma^2} + \frac{2\pi \left\|\mathbf{z}\right\|_2}{\sigma^2} \sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} \frac{\rho_{\sigma,-\mathbf{t}}(\mathbf{x})}{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))} (\mathbf{x} + \mathbf{t})^\mathsf{T} \tilde{\mathbf{z}} \right),
\end{aligned} \tag{B.2}$$

where $\tilde{\mathbf{z}} = \mathbf{z} / \|\mathbf{z}\|_2$ is a unit vector. Since $\sigma \geq \log m \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{A}))$, we can appeal to Lemma B.2 to conclude that

$$\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A})) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_\sigma(\Lambda^\perp(\mathbf{A})) \quad \text{and} \quad \rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\Lambda^\perp(\mathbf{A})) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_\sigma(\Lambda^\perp(\mathbf{A})).$$

Next,

$$\frac{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))}{\rho_{\sigma,-\mathbf{t}-\mathbf{z}}(\Lambda^\perp(\mathbf{A}))} \leq \frac{\rho_\sigma(\Lambda^\perp(\mathbf{A}))}{\frac{1-\varepsilon}{1+\varepsilon} \cdot \rho_\sigma(\Lambda^\perp(\mathbf{A}))} \leq \frac{1+\varepsilon}{1-\varepsilon} \leq 1 + \frac{2\varepsilon}{1-\varepsilon} = 1 + \delta, \tag{B.3}$$

where $\delta = \frac{2\varepsilon}{1-\varepsilon}$. Next,

$$\begin{aligned}
\sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} \frac{\rho_{\sigma,-\mathbf{t}}(\mathbf{x})}{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))} (\mathbf{x} + \mathbf{t})^\mathsf{T} \tilde{\mathbf{z}} &= \sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} \frac{\rho_\sigma(\mathbf{x} + \mathbf{t})}{\rho_\sigma(\mathbf{t} + \Lambda^\perp(\mathbf{A}))} (\mathbf{x} + \mathbf{t})^\mathsf{T} \tilde{\mathbf{z}} \\
&= \sum_{\mathbf{u} \in \mathbf{t} + \Lambda^\perp(\mathbf{A})} \frac{\rho_\sigma(\mathbf{u})}{\rho_\sigma(\mathbf{t} + \Lambda^\perp(\mathbf{A}))} \mathbf{u}^\mathsf{T} \tilde{\mathbf{z}} = \mathbb{E}_{\mathbf{u} \leftarrow D_{\mathbf{t}+\Lambda^\perp(\mathbf{A}),\sigma}} \mathbf{u}^\mathsf{T} \tilde{\mathbf{z}}
\end{aligned} \tag{B.4}$$

Since $\sigma \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{A}))$, by Lemma B.4, the distribution $D_{\Lambda^\perp(\mathbf{A})+\mathbf{t},\sigma}$ is δ'-subgaussian with parameter σ where $\delta' = \ln \frac{1+\varepsilon}{1-\varepsilon}$. Since $\|\tilde{\mathbf{z}}\|_2 = 1$, this means the random variable $\mathbf{u}^\mathsf{T}\tilde{\mathbf{z}}$ is δ'-subgaussian with parameter σ. Thus, for all $t \geq 0$, by Eq. (B.1),

$$\Pr[|\mathbf{u}^\mathsf{T}\tilde{\mathbf{z}}| \geq t : \mathbf{u} \leftarrow D_{\mathbf{t}+\Lambda^\perp(\mathbf{A}),\sigma}] \leq 2\exp(\delta') \exp(-\pi t^2/\sigma^2) = 2 \cdot \frac{1+\varepsilon}{1-\varepsilon} \cdot \exp(-\pi t^2/\sigma^2).$$

We can now compute the expectation as

$$\mathbb{E}_{\mathbf{u} \leftarrow D_{\mathbf{t}+\Lambda^\perp(\mathbf{A}),\sigma}} \mathbf{u}^\mathsf{T}\tilde{\mathbf{z}} = \sum_{t=1}^{\infty} \Pr[|\mathbf{u}^\mathsf{T}\tilde{\mathbf{z}}| \geq t] \leq 2 \cdot \frac{1+\varepsilon}{1-\varepsilon} \int_0^\infty \exp(-\pi t^2/\sigma^2)\, dt = \frac{1+\varepsilon}{1-\varepsilon} \cdot \sigma.$$

Substituting back into Eq. (B.4), we have

$$\sum_{\mathbf{x} \in \Lambda^\perp(\mathbf{A})} \frac{\rho_{\sigma,-\mathbf{t}}(\mathbf{x})}{\rho_{\sigma,-\mathbf{t}}(\Lambda^\perp(\mathbf{A}))} (\mathbf{x} + \mathbf{t})^\mathsf{T}\tilde{\mathbf{z}} = \mathbb{E}_{\mathbf{u} \leftarrow D_{\mathbf{t}+\Lambda^\perp(\mathbf{A}),\sigma}} \mathbf{u}^\mathsf{T}\tilde{\mathbf{z}} \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \sigma.$$

In combination with Eqs. (B.2) and (B.3), we have

$$D_{\mathsf{KL}}(D_1 \| D_2) \leq (1+\delta)\left(\frac{\pi \left\|\mathbf{z}\right\|_2^2}{\sigma^2} + \frac{2\pi \left\|\mathbf{z}\right\|_2}{\sigma} \cdot \frac{1+\varepsilon}{1-\varepsilon}\right).$$

Since $\varepsilon = \mathsf{negl}(m)$, we can bound $(1+\varepsilon)/(1-\varepsilon) \leq O(1)$ and similarly, $\delta = 2\varepsilon/(1-\varepsilon) \leq O(1)$. Moreover, when $\sigma \geq \left\|\mathbf{z}\right\|_2$, $\left\|\mathbf{z}\right\|_2/\sigma \leq 1$ so we conclude that $D_{\mathsf{KL}}(D_1 \| D_2) \leq O\left(\left\|\mathbf{z}\right\|_2/\sigma\right)$. By Pinsker's Inequality (Fact B.5), we conclude that

$$\Delta(\mathbf{A}_\sigma^{-1}(\mathbf{y}+\mathbf{A}\mathbf{z}), \mathbf{z}+\mathbf{A}_\sigma^{-1}(\mathbf{y})) = \Delta(D_1, D_2) \leq O\big(\sqrt{\left\|\mathbf{z}\right\|_2/\sigma}\big)$$

B.2 Proof of Lemma 4.7 (ℓ-Succinct LWE Trapdoor Transformation)

The proof is implicit in [CW24, Theorem 5.1]. We reconstruct the algorithm here for completeness (and make its properties explicit). We start by defining the $\mathsf{Transform}(\mathbf{A}, \mathbf{U}, \mathbf{T}, N)$ algorithm:

- Sample $(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{TrapGen}(1^{nm}, q, k)$. Let $\tilde{\mathbf{Z}} = [\tilde{\mathbf{z}}_1 \mid \cdots \mid \tilde{\mathbf{z}}_{nm}] \in \mathbb{Z}_q^{nm \times k}$. Let $\mathbf{Z} = [\mathbf{Z}_1 \mid \cdots \mid \mathbf{Z}_k] \in \mathbb{Z}_q^{n \times mk}$ where $\mathsf{vec}(\mathbf{Z}_i) = \tilde{\mathbf{z}}_i$ for all $i \in [k]$.
- Parse the matrix $\mathbf{U}$ and the gadget matrix $\mathbf{G}_{nN}$ as follows:

$$\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_\ell \end{bmatrix} \quad \text{and} \quad \mathbf{G}_{nN} = \begin{bmatrix} \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{1,Nm'} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{N,1} & \cdots & \mathbf{x}_{N,Nm'} \end{bmatrix},$$

  where $\mathbf{U}_i \in \mathbb{Z}_q^{n \times m}$ for all $i \in [\ell]$ and $\mathbf{x}_{i,j} \in \mathbb{Z}_q^n$ for all $i \in [N], j \in [Nm']$. For all $i \in [N]$, set

$$\hat{\mathbf{x}}_i = \begin{bmatrix} \mathbf{x}_{i,1} \\ \mathbf{x}_{i,2} \\ \vdots \\ \mathbf{x}_{i,\ell} \end{bmatrix} \in \mathbb{Z}_q^{\ell n}$$

- For all $i \in [N]$ and $j \in [Nm']$, compute $\mathbf{d}_j = \mathbf{T}_{\tilde{\mathbf{Z}}}\mathbf{G}_{nm}^{-1}(\mathsf{vec}(\mathbf{U}_j))$ and

$$\begin{bmatrix} \mathbf{y}_{i,1} \\ \mathbf{y}_{i,2} \\ \vdots \\ \mathbf{y}_{i,\ell} \\ \mathbf{r}_i \end{bmatrix} = \mathbf{T}\mathbf{G}_{\ell n}^{-1}(\hat{\mathbf{x}}_i)$$

  where $\mathbf{y}_{i,j}, \mathbf{r}_i \in \mathbb{Z}_q^m$, $\mathbf{d}_j \in \mathbb{Z}_q^k$.
- Define the matrices

$$\mathbf{V} = \left[\begin{array}{ccc|c} \mathbf{A} & & & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1) \\ & \ddots & & \vdots \\ & & \mathbf{A} & -\mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N) \end{array}\right] \quad \text{and} \quad \mathbf{T}_{\mathbf{V}} = \begin{bmatrix} \mathbf{y}_{1,1} & \cdots & \mathbf{y}_{1,Nm'} \\ \vdots & \ddots & \vdots \\ \mathbf{y}_{N,1} & \cdots & \mathbf{y}_{N,Nm'} \\ -\mathbf{d}_1 & \cdots & -\mathbf{d}_{Nm'}. \end{bmatrix}$$

  Let $\mathbf{R} = [\mathbf{r}_1 \mid \cdots \mid \mathbf{r}_N]$ and output $(\mathbf{V}, \mathbf{Z}, \mathbf{R}, \mathbf{T}_{\mathbf{V}}, \mathbf{T}_{\tilde{\mathbf{Z}}})$

We verify the above algorithm satisfies the desired conditions as follows.

- Since $(\tilde{\mathbf{Z}}, \mathbf{T}_{\tilde{\mathbf{Z}}}) \leftarrow \mathsf{TrapGen}(1^{nm}, q, k)$, by Lemma 3.8, $\tilde{\mathbf{Z}}\mathbf{T}_{\tilde{\mathbf{Z}}} = \mathbf{G}_{nm}$, $\|\mathbf{T}_{\tilde{\mathbf{Z}}}\| = 1$, and $\mathbf{Z}$ is statistically close to uniform. Moreover,

  $$\tilde{\mathbf{Z}}\mathbf{d}_j = \tilde{\mathbf{Z}} \cdot \mathbf{T}_{\tilde{\mathbf{Z}}}\mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{U}_j)) = \mathbf{G}_{nm} \cdot \mathbf{G}_{nm}^{-1}(\mathrm{vec}(\mathbf{U}_j)) = \mathrm{vec}(\mathbf{U}_j).$$

  Since $\tilde{\mathbf{Z}} = [\mathrm{vec}(\mathbf{Z}_1) \mid \cdots \mid \mathrm{vec}(\mathbf{Z}_k)]$, this means $\mathbf{Z}(\mathbf{d}_j \otimes \mathbf{I}_m) = \mathbf{U}_j$.
- Similarly, since $[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T} = \mathbf{G}_{\ell n}$, we have that

  $$[\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \begin{bmatrix} \mathbf{y}_{i,1} \\ \mathbf{y}_{i,2} \\ \vdots \\ \mathbf{y}_{i,\ell} \\ \mathbf{r}_i \end{bmatrix} = [\mathbf{I}_\ell \otimes \mathbf{A} \mid \mathbf{U}] \cdot \mathbf{T}\mathbf{G}_{\ell n}^{-1}(\hat{\mathbf{x}}_i) = \hat{\mathbf{x}}_i.$$

  By construction of $\hat{\mathbf{x}}_i$, this means

  $$\mathbf{A}\mathbf{y}_{i,j} + \mathbf{U}_j\mathbf{r}_i = \mathbf{x}_{i,j}.$$
- Putting the pieces together, we now have

  $$\begin{aligned} \mathbf{V} \cdot \mathbf{T}_{\mathbf{V}} &= \begin{bmatrix} \mathbf{A}\mathbf{y}_{1,1} + \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1)\mathbf{d}_1 & \cdots & \mathbf{A}\mathbf{y}_{1,Nm'} + \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_1)\mathbf{d}_{Nm'} \\ \vdots & \ddots & \vdots \\ \mathbf{A}\mathbf{y}_{N,1} + \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N)\mathbf{d}_1 & \cdots & \mathbf{A}\mathbf{y}_{N,Nm'} + \mathbf{Z}(\mathbf{I}_k \otimes \mathbf{r}_N)\mathbf{d}_{Nm'} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}\mathbf{y}_{1,1} + \mathbf{Z}(\mathbf{d}_1 \otimes \mathbf{I}_m)\mathbf{r}_1 & \cdots & \mathbf{A}\mathbf{y}_{1,Nm'} + \mathbf{Z}(\mathbf{d}_{Nm'} \otimes \mathbf{I}_m)\mathbf{r}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{A}\mathbf{y}_{N,1} + \mathbf{Z}(\mathbf{d}_1 \otimes \mathbf{I}_m)\mathbf{r}_N & \cdots & \mathbf{A}\mathbf{y}_{N,Nm'} + \mathbf{Z}(\mathbf{d}_{Nm'} \otimes \mathbf{I}_m)\mathbf{r}_N \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}\mathbf{y}_{1,1} + \mathbf{U}_1\mathbf{r}_1 & \cdots & \mathbf{A}\mathbf{y}_{1,Nm'} + \mathbf{U}_{Nm'}\mathbf{r}_1 \\ \vdots & \ddots & \vdots \\ \mathbf{A}\mathbf{y}_{N,1} + \mathbf{U}_1\mathbf{r}_N & \cdots & \mathbf{A}\mathbf{y}_{N,Nm'} + \mathbf{U}_{Nm'}\mathbf{r}_N \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{1,Nm'} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{N,1} & \cdots & \mathbf{x}_{N,Nm'} \end{bmatrix} = \mathbf{G}_{nN}. \end{aligned}$$
- From Lemma 3.8, $\|\mathbf{T}_{\tilde{\mathbf{Z}}}\| = 1$. Thus, $\|\mathbf{d}_j\| \le mm'$. Next, $\|\mathbf{y}_{i,j}\|, \|\mathbf{r}_i\| \le \|\mathbf{T}\| \cdot \ell m'$. Since $m' \le m$, we can bound $\|\mathbf{R}\|, \|\mathbf{T}_{\mathbf{V}}\| \le \|\mathbf{T}\| \cdot \ell m^2$ and the claim holds. □

B.3 Proof of Lemma 7.4 (Gadget Trapdoor Implies Ajtai Trapdoor)

The proof follows via the same construction as that used in [MP12, Lemma 5.3]. We include the proof here for completeness. It is easy to see that $\mathbf{A}\mathbf{T}' = \mathbf{0} \bmod q$:

$$\mathbf{A}\mathbf{T}' = \mathbf{A} \cdot [\mathbf{I}_m - \mathbf{T}\mathbf{G}_n^{-1}(\mathbf{A}) \mid \mathbf{T}\mathbf{S}_n] = [\mathbf{A} - \mathbf{A}\mathbf{T}\mathbf{G}_n^{-1}(\mathbf{A}) \mid \mathbf{A}\mathbf{T}\mathbf{S}_n] = [\mathbf{A} - \mathbf{G}_n\mathbf{G}_n^{-1}(\mathbf{A}) \mid \mathbf{G}_n\mathbf{S}_n] = \mathbf{0}^{n \times (m+m')} \bmod q,$$

using the fact that $\mathbf{A}\mathbf{T} = \mathbf{G}_n$ and $\mathbf{G}_n\mathbf{S}_n = \mathbf{0} \bmod q$ (since $\mathbf{S}_n$ is an Ajtai trapdoor for $\mathbf{G}_n$). Next, we show that $\mathbf{T}'$ has full rank. To do so, consider the matrix $\tilde{\mathbf{S}}$ from [MP12, Lemma 5.3]:

$$\tilde{\mathbf{S}} = \begin{bmatrix} \mathbf{I}_m & \mathbf{T} \\ \mathbf{0} & \mathbf{I}_{m'} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ -\mathbf{G}_n^{-1}(\mathbf{A}) & \mathbf{S}_n \end{bmatrix} = \begin{bmatrix} \mathbf{I}_m - \mathbf{T}\mathbf{G}_n^{-1}(\mathbf{A}) & \mathbf{T}\mathbf{S}_n \\ -\mathbf{G}_n^{-1}(\mathbf{A}) & \mathbf{S}_n. \end{bmatrix} \in \mathbb{Z}^{(m+m') \times (m+m')}.$$

Since $\mathbf{S}_n$ is full rank (over $\mathbb{R}$), $\det(\mathbf{S}_n) \neq 0$. Next, $\det(\tilde{\mathbf{S}}) = \det(\mathbf{S}_n) \neq 0$, so $\tilde{\mathbf{S}}$ is also full rank over $\mathbb{R}$. This means that the first m rows of $\tilde{\mathbf{S}}$ (i.e., the matrix $\mathbf{T}'$) are linearly independent over $\mathbf{R}$. This means $\mathbf{T}'$ is full rank (over $\mathbb{R}$).