Distributed Monotone-Policy Encryption for DNFs from Lattices

Jeffrey Champion
UT Austin
jchampion@utexas.edu

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

Abstract

Distributed monotone-policy encryption augments public-key encryption with fine-grained decryption capabilities in a trustless manner. In this scheme, users independently generate a public/private key-pair and post their public key to a public-key directory. Thereafter, anyone can encrypt a message to a set of public keys together with an access policy. Any set of users that satisfies the access policy can decrypt the ciphertext while the message should remain computationally hidden to any unsatisfying set of users. The primary efficiency requirement is succinctness: namely, the size of the ciphertext should be sublinear (or polylogarithmic) in the description length of the policy. Distributed monotone-policy encryption directly generalizes recent trustless cryptographic notions like threshold encryption with silent setup and distributed broadcast encryption.

In this work, we show how to construct distributed monotone-policy encryption for Boolean formulas in disjunctive normal form (DNF formulas) that supports an unbounded number of users. Security relies on the decomposed learning with errors (LWE) assumption, a simple and falsifiable lattice assumption, in the random oracle model. Previously, such a scheme was only known from plain witness encryption in the random oracle model. Our scheme has a transparent setup and the ciphertext size is $\mathsf{poly}(\lambda, \log N)$, where N is the number of variables in the DNF formula.

1 Introduction

Traditional public-key encryption has a central point of failure. Namely, if a user's secret key is ever compromised, then the attacker gains complete access to all of the data encrypted using that key. Threshold cryptography [Des87, Fra89, DF89, SDFY94] is a standard solution to distribute trust in cryptographic systems. In a threshold encryption scheme [DF89, SDFY94], the secret decryption key would be secret shared across N different users such that at least T of them must come together to decrypt a ciphertext. Moreover, the message associated with the ciphertext remains computationally hidden even if any subset of up to $T - 1$ users collude. Traditionally, threshold cryptography operates in a model where a trusted dealer generates the shares of the decryption key and distributes them to the users. While such a solution might be suitable if a single user wants to split their decryption key into multiple shares, this is less suitable for distributed settings where a group of (mutually distrusting) users would like to serve as a decryption committee for a particular system and there is no natural entity that could be trusted to generate and distribute the key. One solution then is for the N users to engage in an *interactive* distribute key-generation (DKG) protocol to generate their respective decryption shares. However, such protocols often incur high communication and computational costs [GJKR99, TCZ+20]. A natural question is whether we can take advantage of the benefits provided by threshold cryptography without requiring an interactive protocol.

Threshold encryption with silent setup. An alternative approach to having a trusted dealer generate a decryption key and then issue shares to different users is for each user to generate their own public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ *independently* of everyone else and publish their public key to a public-key directory. Thereafter, if a group of N users want to join together and form a decryption committee (with respect to some decryption policy), they can derive a joint public key associated with their group as a deterministic function of their individual public keys. Put another way, at encryption time, the encrypter can select a set of users S and encrypt a message to the set of public keys $\{\mathsf{pk}_i\}_{i \in S}$ associated with those users. The encrypter can also specify a decryption policy (e.g., at least T users from the set S must participate in order to decrypt). As in standard threshold encryption, we require the ciphertexts to be short (*independent* of the size of S). Now, given a ciphertext ct, each user $i \in S$ can use their secret key sk_i to issue a *short* decryption hint ht_i for the ciphertext. Once someone has collected enough decryption hints from all users $i \in S' \subseteq S$ where S' is an authorized set of decrypters, they can recover the underlying message. Schemes with this property include threshold encryption with silent setup [RSY21, GKPW24] as well as distributed or flexible broadcast encryption [WQZDF10, BZ14, FWW23] which corresponds to the special case where the threshold is always set to 1 (e.g., a user encrypts to a set of public keys, and any individual user in the set can decrypt). Recent works [DJWW25, WW26] have studied generalizations beyond threshold policies (e.g., these works can support policies like a conjunction of thresholds which capture settings where there are multiple decryption committees and one needs a majority from each committee to approve the decryption request before the message is revealed). Such schemes are referred to as distributed monotone-policy encryption schemes.

Constructions of distributed monotone-policy encryption. Thus far, much of the work on distributed monotone-policy encryption has focused on distributed or flexible broadcast encryption which correspond to the special case where the threshold policy is fixed at 1. Here, we have constructions from a broad set of assumptions including pairings [WQZDF10, KMW23], lattices [CW24, CHW25, WW25], as well as general tools like indistinguishability obfuscation [BZ14] and witness encryption [FWW23]. However, having a fixed threshold of 1 does not capture a primary aim of threshold cryptography which is to distribute the decryption capability across *multiple* users.

If we consider schemes that support decryption policies involving multiple independent decrypters, then we have schemes that support threshold policies [GKPW24, BCF+25, GWWW26] and Boolean formulas [WW26] from pairings, schemes that support CNF or DNF formulas from witness encryption [DJWW25], and schemes that support threshold policies [RSY21, ADM+24] or read-once bounded-space Turing machines [DJWW25] from indistinguishability obfuscation. Notably missing from this list are constructions based on lattice assumptions; we note that while we have lattice-based witness encryption schemes based on the (private-coin) evasive learning with errors (LWE) assumption [Tsa22, VWW22], a line of recent work has raised concerns on the plausibility of this assumption [VWW22, BÜW24, BDJ+25, DJM+25, AMYY25, HJL25, HHY25]. A natural question then is whether we can build any kind of distributed monotone-policy encryption scheme that can support threshold policies from falsifiable lattice assumptions. This is the focus of this work.

Our results. In this work, we show how to build distributed monotone-policy encryption from the decomposed LWE assumption [AMR25] in the random oracle model (alternatively, we could also instantiate with the stronger succinct LWE assumption from [Wee24]). Specifically, we give a construction of distributed monotone-policy encryption that support DNF formulas.[1] While DNF policies do not capture arbitrary

[1]Technically, our approach supports any policy family that has a linear secret sharing scheme with small reconstruction coefficients and a linear independence property (see Definition B.3; this coincides with the same properties needed for the lattice-based

threshold policies, they do include constant thresholds as a special case. On the flip side, DNF policies also capture decryption policies that cannot be naturally described by a threshold (e.g., the decryption set must include the chairman together with three out of five board members). More formally, we show the following:

Theorem 1.1 (Informal)**.** *Let λ be a security parameter. Under the ℓ-decomposed LWE assumption, where $\ell = \mathsf{poly}(\lambda)$, there exists a distributed monotone-policy encryption for access policies captured by DNF formulas in the random oracle model with the following properties:*

- **Public parameters:** *The public parameters have size $\mathsf{poly}(\lambda)$ and can be sampled using a* transparent *procedure (i.e., the public parameters can be described by a uniform random string).*
- **Public/secret key size:** *Each user's public and secret key has size $\mathsf{poly}(\lambda)$. In combination with the previous bullet, this means the scheme supports an unbounded number of users.*
- **Ciphertext size:** *An encryption of a bit $\mu \in \{0,1\}$ with respect to a DNF formula $P\colon \{0,1\}^N \to \{0,1\}$ has size $\mathsf{poly}(\lambda, \log N)$. We assume that each variable is used exactly once in P. We can extend to the case where each variable is used up to K times by having each user publish K public keys. This increases the size of the public keys by a factor of K, but does not affect the ciphertext size.*

Finally, our distributed monotone-policy encryption scheme supports a "one-round decryption" process. Namely, given a ciphertext ct, each user can use their secret key to generate an associated decryption hint ht_i. Given decryption hints from any subset of users that satisfy the policy P, it is possible to recover the message.

Comparison with [DJWW25]. To our knowledge, the only distributed monotone-policy encryption scheme that supports DNF formulas for an unbounded number of users is the scheme from [DJWW25], which relies on plain witness encryption (and function-binding hash functions) in the random oracle model. The pairing-based scheme from [WW26] can support general Boolean formulas in the plain model but assumes there is an a priori bound on the number of users N that can appear in a decryption policy, and moreover, the public parameters in their scheme scale *quadratically* with this bound. Other works have focused on threshold policies [RSY21, GKPW24, ADM+24] or a plain disjunction, which corresponds to distributed broadcast encryption [WQZDF10, BZ14, KMW23, GLWW23, FWW23, GKPW24, CW24, CHW25, WW25]. We now compare the parameter sizes of our scheme to that of [DJWW25]:

- **Ciphertext size:** The ciphertext size in the scheme of [DJWW25] grows with the maximum size of a min-term in the DNF. In our construction, the ciphertexts are fully succinct (i.e., scale polylogarithmically with the size of the policy).
- **User public key size:** An advantage of the [DJWW25] approach is that they support DNFs where input variables can be reused an arbitrary number of times. In our construction, we require an a priori bound K on the number of times a variable appears in the policy, and the size of the user public keys scale linearly with the bound K (the ciphertext size remains independent of K).

Another view: reusable succinct computational secret sharing. Distributed monotone-policy encryption is closely related to a reusable variant of succinct computational secret sharing. In succinct computational secret sharing [ABI+23], the goal is to share a message μ according to a policy P such that the size of the largest share is much smaller than the description complexity of P. Distributed monotone-policy encryption gives a solution to this problem in the model where each user is allowed to have some

multi-authority attribute-based encryption (ABE) from [DKW21]).

precomputed state that is allowed to grow with the bound on the policy size, but is otherwise independent of the message and the policy. Thereafter, a dealer can secret share a message μ according to any policy P with a succinct share (whose size is essentially independent of the description length of P).

1.1 Technical Overview

The starting point of this work is the recent distributed broadcast encryption scheme by Wee and Wu [WW25] based on Wee's matrix commitment scheme [Wee25]. We start with a general description of this scheme. Let n, m, q be lattice parameters where $m = O(n \log q)$. We use curly underlines to suppress low-norm error terms (e.g., we write $\underbrace{\mathbf{s}^\intercal \mathbf{A}}$ to denote $\mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal$ where $\mathbf{e}$ is a low-norm vector). Recall also that the learning with errors assumption (LWE) [Reg05] says that the following distributions are computationally indistinguishable:

$$(\mathbf{A}, \underbrace{\mathbf{s}^\intercal \mathbf{A}}) \quad \text{and} \quad (\mathbf{A}, \mathbf{u}^\intercal) \quad \text{where} \quad \mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m},\ \mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n,\ \mathbf{u} \xleftarrow{R} \mathbb{Z}_q^m$$

Matrix commitments. A matrix commitment [Wee25] to a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times N}$ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ where

$$\mathbf{C} \cdot \mathbf{V}_N = \mathbf{M} - \mathbf{B} \cdot \mathbf{Z},$$

where $\mathbf{B} \in \mathbb{Z}_q^{n \times 4m^5}, \mathbf{V}_N \in \mathbb{Z}_q^{m \times N}$ are *public* matrices and $\mathbf{Z} \in \mathbb{Z}_q^{4m^5 \times N}$ is a low-norm opening. Moreover, there are efficient, public, and deterministic algorithms that compute $\mathbf{C}$ and $\mathbf{Z}$ from $\mathbf{M}$ and a set of public parameters $\mathsf{pp}_{\mathsf{com}}$. The work of [WW25] observes that the algorithms for computing the commitment and the openings are local. Namely, even when $\mathbf{M}$ is exponentially-wide, but sparse (e.g., $N = 2^\lambda$, but $\mathbf{M}$ only contains $\mathsf{poly}(\lambda)$ non-zero columns), there is an efficient algorithm for computing a *local* opening for any column $\mathbf{m}_i$ of $\mathbf{M}$: namely, $\mathbf{z}_i \in \mathbb{Z}_q^{4m^5}$ such that

$$\mathbf{C} \cdot \mathbf{v}_{N,i} = \mathbf{m}_i - \mathbf{B} \cdot \mathbf{z}_i, \tag{1.1}$$

where $\mathbf{v}_{N,i}$ denotes the i^{th} column of V_N (there is an efficient algorithm for computing any individual column $\mathbf{v}_{N,i}$ of $\mathbf{V}_N$). Moreover, when using matrix commitments to construct encryption schemes, we also require that the LWE assumption holds with respect to the matrix $\mathbf{B}$ given the public parameters $\mathsf{pp}_{\mathsf{com}}$ for the matrix commitment scheme. As shown in [Wee25], this is implied by either the succinct LWE assumption [Wee24] or the decomposed LWE assumption [AMR25].

Distributed broadcast encryption using matrix commitments. We now recall the [WW25] distributed broadcast encryption scheme from matrix commitments. Recall first that in distributed broadcast encryption, each user is associated with an index $i \in [N]$, where N can be set to be $N = 2^\lambda$. Users can individually sample a public/private key-pair and publish their public key. Later, anyone can encrypt a message to an arbitrary set of public keys, and the requirement is that the size of the ciphertext scales sublinearly (ideally, polylogarithmically) with the size of the broadcast set. Anyone in the set should be able to decrypt the ciphertext and recover the message whereas outsiders should not be able to learn anything about the message. The distributed broadcast encryption scheme from [WW25] for N users works as follows:

- **Public parameters:** The public parameters for the distributed broadcast encryption scheme is a triple $\mathsf{pp} = (\mathsf{pp}_{\mathsf{com}}, \mathbf{A}, \mathbf{p})$, where $\mathsf{pp}_{\mathsf{com}}$ is the public parameters for the matrix commitment (which defines the matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times 4m^5}$), and $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $\mathbf{p} \xleftarrow{R} \mathbb{Z}_q^n$. The vector $\mathbf{p}$ is a dual Regev public key while the matrix $\mathbf{A}$ is used for re-randomization.

- **User keys:** To generate a key for slot $i \in [N]$, the user samples $\mathbf{y}_i \xleftarrow{R} \{0,1\}^{4m^5}$. The associated public key is $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_{N,i} \in \mathbb{Z}_q^n$, where $\mathbf{v}_{N,i} \in \mathbb{Z}_q^m$ is the vector from Eq. (1.1) computed from $\mathsf{pp}_{\mathsf{com}}$.
- **Encryption:** Let $S \subseteq [N]$ be a set of indices and let $\{(j, \mathbf{t}_j)\}_{j \in S}$ be a set of public keys. To encrypt a bit $\mu \in \{0,1\}$, the encrypter proceeds as follows:
  - Let $\mathbf{C}_j$ be the matrix commitment to $\mathbf{u}_j^\intercal \otimes \mathbf{t}_j$ where $\mathbf{u}_j \in \{0,1\}^N$ is the j^{th} standard basis vector.
  - Sample an LWE secret $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$ and output the ciphertext
$$\mathsf{ct} = (\mathbf{s}^\intercal \mathbf{B}\ ,\ \mathbf{s}^\intercal \left(\mathbf{A} + \textstyle\sum_{j \in S} \mathbf{C}_j\right)\ ,\ \mathbf{s}^\intercal \mathbf{p} + \mu \cdot \lfloor q/2 \rfloor)\ .$$
- **Decryption:** Suppose $i \in S$. To decrypt, the user first computes the opening $\mathbf{z}_{j,i} \in \mathbb{Z}_q^{4m^5}$ from Eq. (1.1):
$$\mathbf{C}_j \mathbf{v}_{N,i} = \delta_{j,i} \mathbf{t}_i - \mathbf{B}\mathbf{z}_{j,i}, \tag{1.2}$$
where $\delta_{j,i} = 1$ if $j = i$ and $\delta_{j,i} = 0$ otherwise. Recall that $\mathbf{C}_j$ is a commitment to $\mathbf{u}_j^\intercal \otimes \mathbf{t}_j$, so its i^{th} column is $\delta_{j,i}\mathbf{t}_i$. Now, the user can compute
$$\begin{aligned} \mathbf{s}^\intercal \left(\mathbf{A} + \textstyle\sum_{j \in S} \mathbf{C}_j\right) \cdot \mathbf{v}_{N,i} - \mathbf{s}^\intercal \mathbf{B} \cdot (\mathbf{y}_i - \textstyle\sum_{j \in S} \mathbf{z}_{j,i}) \\ \approx \mathbf{s}^\intercal \mathbf{A}\mathbf{v}_{N,i} + \mathbf{s}^\intercal \textstyle\sum_{j \in S} \mathbf{C}_j \mathbf{v}_{N,i} - \mathbf{s}^\intercal (\mathbf{B}\mathbf{y}_i - \textstyle\sum_{j \in S} (\delta_{j,i}\mathbf{t}_i - \mathbf{C}_j \mathbf{v}_{N,i})) \\ = \mathbf{s}^\intercal \mathbf{A}\mathbf{v}_{N,i} - \mathbf{s}^\intercal \mathbf{B}\mathbf{y}_i + \mathbf{s}^\intercal \mathbf{t}_i \\ = \mathbf{s}^\intercal \mathbf{p}, \end{aligned} \tag{1.3}$$
where the first equality follows from Eq. (1.2) and the last equality follows from the public key structure (i.e., $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_{N,i}$). Given $\mathbf{s}^\intercal \mathbf{p}$, the user can recover $\mu \cdot \lfloor q/2 \rfloor$ and round to obtain μ.

To prove selective security (where the adversary declares upfront the set of users $S \subseteq [N]$ associated with the challenge ciphertext), the [WW25] argument proceeds as follows:

- When generating the keys for honest users $j \in S$, the reduction samples $\mathbf{t}_j \xleftarrow{R} \mathbb{Z}_q^n$. By the leftover hash lemma, this is statistically indistinguishable from the real keys. Observe now that the honest keys $\mathbf{t}_j$ are *independent* of $\mathbf{A}$.
- The reduction algorithm computes the matrix commitments $\mathbf{C}_j$ for each $j \in S$ as above and then programs $\mathbf{A} = \mathbf{B}\mathbf{K}_{\mathbf{A}} - \sum_{j \in S} \mathbf{C}_j$ where $\mathbf{K}_{\mathbf{A}} \xleftarrow{R} \{0,1\}^{4m^5 \times m}$. It also sets $\mathbf{p} = \mathbf{B}\mathbf{k}_{\mathbf{p}}$ where $\mathbf{k}_{\mathbf{p}} \xleftarrow{R} \{0,1\}^{4m^5}$. By the leftover hash lemma, the distributions of $\mathbf{A}$ and $\mathbf{p}$ remain statistically close to uniform.
- With this setting of parameters, the challenger ciphertext can be written as
$$\mathsf{ct} = (\mathbf{s}^\intercal \mathbf{B}\ ,\ \mathbf{s}^\intercal \mathbf{B}\mathbf{K}_{\mathbf{A}}\ ,\ \mathbf{s}^\intercal \mathbf{B}\mathbf{k}_{\mathbf{p}} + \mu \cdot \lfloor q/2 \rfloor)\ .$$
Security reduces to LWE with respect to the matrix $\mathbf{B}$ (given the commitment public parameters $\mathsf{pp}_{\mathsf{com}}$). This in turn relies on either the succinct LWE assumption or the decomposed LWE assumption.

Extending to linear secret sharing polices. We now show how to extend the above distributed broadcast encryption scheme to support more general policies. In particular, we consider policies that can be described by a linear secret sharing scheme with small (e.g., $\{0,1\}$) reconstruction coefficients.[2]

The high-level idea is to broadcast to the set S of all users who appear in the policy. Unlike distributed broadcast encryption, a user $i \in S$ in the broadcast set should not simply recover the message from the ciphertext (only an authorized *quorum* of users should be able to decrypt). Instead, we aim for the following invariant: if $i \in S$, then the user can use their decryption key to recover a *secret share* $\hat{\mu}_i$ of the message. A group of users who collectively satisfy the decryption policy can then combine their shares together to recover the message μ. The question is how to realize a decryption procedure where each user's key decrypts the ciphertext to *different* values. Specifically, each user in the broadcast set should be able to recover only their share $\hat{\mu}_i$ of the message and nothing more.

In the above encryption scheme, during decryption, a user essentially combines their secret key with the ciphertext components to recover $\mathbf{s}^\mathsf{T}\mathbf{p}$. The value $\mathbf{s}^\mathsf{T}\mathbf{p}$ can be viewed as the blinding factor for the message (recall the ciphertext component c_3 is $c_3 = \mathbf{s}^\mathsf{T}\mathbf{p} + \mu \cdot \lfloor q/2 \rfloor$.) In some sense, a user's secret key in the [WW25] encryption scheme provides a mechanism to recode from $\mathbf{s}^\mathsf{T}\mathbf{B}$ to $\mathbf{s}^\mathsf{T}\mathbf{p}$ whenever the user is in the broadcast set. This recoding is possible because the user's public key is of the form $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i + \mathbf{p} - \mathbf{A}\mathbf{v}_{N,i}$. Observe that we can substitute $\mathbf{p}$ for $\hat{\mathbf{p}}_i$ and the same decryption mechanism can now be used to transform $\mathbf{s}^\mathsf{T}\mathbf{B}$ into $\mathbf{s}^\mathsf{T}\hat{\mathbf{p}}_i$. This lends itself to a natural strategy for users to recover a share of the message rather than the message itself:

- Let P be a target policy (over a set of N parties). First, we secret share $\mathbf{p}$ according to the policy P to obtain shares $\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N$. The assumption is that any subset of users $S \subseteq [n]$ who satisfy the policy can recover $\mathbf{p}$ by taking a $\{0,1\}$-linear combination of their shares.
- The ciphertext will be an encryption with respect to the full set of users (associated with the policy).
- Each user's public key is now of the form $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i + \hat{\mathbf{p}}_i - \mathbf{A}\mathbf{v}_{N,i}$. By the same decryption mechanism described above, user i can use their secret key $\mathbf{y}_i$ together with the ciphertext components to learn $\mathbf{s}^\mathsf{T}\hat{\mathbf{p}}_i$. Since the reconstruction algorithm of the secret sharing scheme is linear with small coefficients, any set $S \subseteq [n]$ of users where $P(S) = 1$ can combine their shares $\{\mathbf{s}^\mathsf{T}\hat{\mathbf{p}}_i\}_{i \in S}$ to learn $\mathbf{s}^\mathsf{T}\mathbf{p}$, and from there, recover the message.

The wrinkle with this approach is that each user's public key depends on the policy, and specifically the share $\hat{\mathbf{p}}_i$. After all, the user samples their public key with the intent to recode to the specific vector $\hat{\mathbf{p}}_i$. This is feasible in the setting of distributed broadcast encryption because the policy there is fixed: it is *always* a disjunction (alternatively, a threshold policy with threshold 1). In distributed monotone-policy encryption however, the policy is determined at encryption time, not key-generation time. The natural solution then is to move the choice of $\hat{\mathbf{p}}_i$ to encryption time rather than key-generation time. Namely, each user's public key will be $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i$ and at encryption time, instead of committing $\mathbf{u}_i^\mathsf{T} \otimes \mathbf{t}_i$ as before, the encrypter now commits to $\mathbf{u}_i^\mathsf{T} \otimes (\mathbf{t}_i + \hat{\mathbf{p}}_i)$. From a functionality standpoint (and ignoring the fact that we dropped the $\mathbf{A}\mathbf{v}_{N,i}$ component here), the user here is encrypting to the effective public key $(\mathbf{t}_i + \hat{\mathbf{p}}_i)$, exactly as required in our basic approach described above.[3] To summarize, the basic structure of our distributed monotone-policy encryption is now as follows:

[2] For security, we will also require the linear secret sharing scheme satisfy a certain linear independence property (in fact, the same property needed for lattice-based multi-authority ABE schemes [DKW21]). This additional linear independence requirement limits our approach to DNF policies.

[3] We remark here that the original [WW25] scheme could have chosen the public keys to be $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i + \mathbf{A}\mathbf{v}_{N,i}$ and had the encrypter commit to $\mathbf{u}_i^\mathsf{T} \otimes (\mathbf{t}_i + \mathbf{p})$. Their security proof would still apply equally well to this scheme. Of course, in the setting of distributed broadcast encryption, introducing the $\mathbf{p}$ vector at encryption time seems to be unnecessary complication. But in our setting, this is a key ingredient to achieve functionality.

- **Public parameters:** The public parameters $\mathsf{pp} = (\mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ are the same as before, except we drop the matrix $\mathbf{A}$. The $\mathbf{A}$ matrix was used in [WW25] to program the challenge set into the public parameters in the selective security proof. As we discuss below, we do not know how to implement the [WW25] proof strategy in our setting and require a different approach (based on the ciphertext re-randomization strategy from [CHW25]).
- **User keys:** To generate a key for an index $i \in [N]$, the user samples $\mathbf{y}_i \xleftarrow{R} \{0,1\}^{4m^5}$. The public key is now $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i$ and the secret key remains $\mathbf{y}_i$.
- **Encryption:** Given a list of public keys $\mathbf{t}_1, \ldots, \mathbf{t}_N$ together with a policy $P\colon \{0,1\}^N \to \{0,1\}$ that can be described by a linear secret sharing scheme, the encrypter does the following:
  - First, the encrypter secret shares the target vector $\mathbf{p} \in \mathbb{Z}_q^n$ according to the policy P. Let $\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N \in \mathbb{Z}_q^n$ be the resulting shares. Since decryption will require knowledge of the individual shares, we assume that the randomness used for deriving the shares $\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N$ is included as part of the ciphertext. To preserve succinctness, we require that the randomness has a short description. We compress this by working in the random oracle model. Note that our security proof also relies on programming the randomness, which is why we do not replace this with a short seed for a pseudorandom generator.
  - Next, the encrypter computes a matrix commitment $\mathbf{C}_j$ to $\mathbf{u}_j^\intercal \otimes (\mathbf{t}_j + \hat{\mathbf{p}}_j)$ for $j \in [N]$.
  - Finally, the encrypter samples an LWE secret $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and outputs the ciphertext
$$\mathsf{ct} = (\mathbf{s}^\intercal \mathbf{B}\ ,\ \mathbf{s}^\intercal \left(\textstyle\sum_{j\in[N]} \mathbf{C}_j\right)\ ,\ \mathbf{s}^\intercal \mathbf{p} + \mu \cdot \lfloor q/2 \rfloor\ ). \tag{1.4}$$
  - Note that if we are encrypting multiple messages to the same set of public keys $\{\mathbf{t}_j\}_{j\in[N]}$ and the same policy P, then we can compute $\sum_{j\in[N]} \mathbf{C}_j$ once and treat that as the "public key" associated with this set of public keys and policy P. Given this initial preprocessing, the running time of the encryption algorithm no longer depends on the number of users or the size of the policy. The ability to preprocess the public keys for a group of users into a succinct key is a requirement in the notion of threshold encryption with silent setup [GKPW24, WW26].
- **Decryption:** As argued earlier, using the same mechanism as the distributed broadcast encryption scheme (see Eq. (1.3)), each user $i \in [N]$ can compute a share $\hat{\mu}_i = \mathbf{s}^\intercal \hat{\mathbf{p}}_i$ using their secret key together with the ciphertext components derive a secret share of the message. From here, any set $T \subseteq [N]$ of users where $P(T) = 1$ can use the reconstruction algorithm for the linear secret sharing scheme to compute $\mathbf{s}^\intercal \mathbf{p}$ and from here, recover the message.

Proving security. The next technical challenge is the security proof. In this work, we consider a semi-selective security notion where the adversary commits to the challenge policy $P\colon \{0,1\}^N \to \{0,1\}$ and the set of honest indices $\mathcal{H} \subseteq [N]$ (which in particular, determine the honest key queries) *after* seeing the public parameters (but before seeing any individual user's public key). Once the adversary is given the honest public keys $\{\mathsf{pk}_i\}_{i\in\mathcal{H}}$, it can then choose the public keys for the corrupted users itself.

To prove security from decomposed LWE, we need to show how to simulate the challenge ciphertext (Eq. (1.4)) given $(\mathsf{pp}_{\mathsf{com}}, \mathbf{s}^\intercal \mathbf{B})$, where $\mathsf{pp}_{\mathsf{com}}$ defines the matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times 4m^5}$. As in the distributed broadcast encryption proof strategy, the reduction can sample $\mathbf{k}_{\mathbf{p}} \leftarrow \{0,1\}^{4m^5}$, set $\mathbf{p} = \mathbf{B}\mathbf{k}_{\mathbf{p}}$, and simulate the message-blinding component as $\mathbf{s}^\intercal \mathbf{p} \approx \mathbf{s}^\intercal \mathbf{B} \cdot \mathbf{k}_{\mathbf{p}}$. The main technical challenge is handling the term $\mathbf{s}^\intercal \left(\sum_{j\in[N]} \mathbf{C}_j\right)$. There are two technical hurdles here:

- First, unlike distributed broadcast encryption, some of the commitments $\mathbf{C}_j$ are now commitments to *adversarially-chosen* public keys $\mathbf{t}_j$. In the setting of distributed broadcast encryption, the users in the challenge set are necessarily honest, and thus, the commitments $\mathbf{C}_j$ are all under the control of the reduction. This enables the simple partitioning argument described earlier where the commitments to the honest parties can be programmed into the public parameters (via the $\mathbf{A}$ matrix). It is not clear how to implement this strategy when the adversary has the ability to choose public keys.
- Second, unlike the single-input setting (i.e. registered ABE), adversarial users have the ability to *partially decrypt* the challenge ciphertext. In our scheme, this means the adversary can learn unauthorized shares of the message (e.g., values of the form $\mathbf{s}^\mathsf{T}\hat{\mathbf{p}}_j$). If the unauthorized shares are correlated (e.g., $\hat{\mathbf{p}}_i = \hat{\mathbf{p}}_j$ for a pair of corrupted indices $i \neq j$), this can leak information about the LWE error (and jeopardize security).

We address these challenges using the ciphertext re-randomization technique from [CHW25] (specifically, the instantiation used in [WW25] in their registered ABE construction) and a stronger linear independence guarantee on the $\{0,1\}$-linear secret sharing scheme (analogous to the properties previously used to construct lattice-based multi-authority ABE [DKW21]). The additional linear independence requirement we rely on to tackle the second challenge is the reason our scheme is currently limited to DNF policies (as opposed to monotone Boolean formulas).

Ciphertext re-randomization. The re-randomization procedure from [CHW25, WW25] shows how to sample a (uniform random) matrix $\mathbf{C}_0 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times m}$ together with low-norm vectors $\mathbf{z}_{0,i} \in \mathbb{Z}_q^{4m^5}$ such that

$$\forall i \in [N] : \mathbf{B}\mathbf{z}_{0,i} = -\mathbf{C}_0\mathbf{v}_{N,i}. \tag{1.5}$$

We can view $\mathbf{C}_0$ as a (random) matrix commitment to the all-zeroes matrix $\mathbf{0}^{n\times N}$ and $\mathbf{z}_{0,i}$ as the respective openings. Using this random matrix commitment, we modify the broadcast component in the ciphertext to $\mathbf{s}^\mathsf{T}\left(\mathbf{C}_0 + \sum_{j\in[N]}\mathbf{C}_j\right)$. We can view $\mathbf{C}_0$ as a "virtual user" that re-randomizes the ciphertext. Correctness is preserved as long as the tuple $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$ is included in the ciphertext. Of course, including all of the openings $\mathbf{z}_{0,i}$ in the ciphertext would mean the ciphertext size scales with the number of inputs to the policy, which violates succinctness. Similar to [CHW25, WW25], we again rely on the random oracle to compress the randomness used to sample $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$. The ciphertext would just contain the seed to the random oracle used to sample this re-randomizing tuple. As we discuss more below, the security reduction will also need to program the specific tuple $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$ into the output of the random oracle. Such programming can be facilitated by relying on an explainability property as is done in [CHW25, WW25].

Now, when constructing the challenge ciphertext, the reduction sets $\mathbf{C}_0 = \mathbf{B}\mathbf{K}_\mathbf{C} - \sum_{j\in[N]}\mathbf{C}_j$ for $\mathbf{K}_\mathbf{C} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5\times m}$. By the leftover hash lemma [HILL99, DORS08], the marginal distribution of $\mathbf{C}_0$ is still uniform. For this setting of $\mathbf{C}_0$, the relevant term in the challenge ciphertext now becomes

$$\mathbf{s}^\mathsf{T}\left(\mathbf{C}_0 + \sum_{j\in[N]}\mathbf{C}_j\right) = \mathbf{s}^\mathsf{T}\mathbf{B}\mathbf{K}_\mathbf{C} \approx \mathbf{s}^\mathsf{T}\mathbf{B}\cdot\mathbf{K}_\mathbf{C}.$$

However, when changing the distribution of $\mathbf{C}_0$, the reduction must also simulate the openings $\mathbf{z}_{0,i}$ (to satisfy Eq. (1.5)). Doing so will require special properties on the $\{0,1\}$-linear secret sharing scheme.

Simulating the openings $\mathbf{z}_{0,i}$. To simulate the openings $\mathbf{z}_{0,i}$ when $\mathbf{C}_0 = \mathbf{B}\mathbf{K}_\mathbf{C} - \sum_{j\in[N]}\mathbf{C}_j$, we first observe that for all $i \in [N]$, the reduction already knows a low-norm vector $\mathbf{z}'_{0,i}$ where

$$(\mathbf{t}_i + \hat{\mathbf{p}}_i) - \mathbf{B}\mathbf{z}'_{0,i} = -\mathbf{C}_0\mathbf{v}_{N,i}.$$

This is because $\mathbf{C}_j$ is a matrix commitment to $\mathbf{u}_j^\intercal \otimes (\mathbf{t}_j + \hat{\mathbf{p}}_j)$ and so $\mathbf{C}_j \mathbf{v}_{N,i} = \delta_{j,i}(\mathbf{t}_j + \hat{\mathbf{p}}_j) - \mathbf{B}\mathbf{z}_{j,i}$. This means

$$-\mathbf{C}_0 \mathbf{v}_{N,i} = -(\mathbf{B}\mathbf{K}_\mathbf{C} - \textstyle\sum_{j\in[N]} \mathbf{C}_j)\mathbf{v}_{N,i} = -\mathbf{B} \cdot \underbrace{(\mathbf{K}_\mathbf{C}\mathbf{v}_{N,i} + \textstyle\sum_{j\in[N]} \mathbf{z}_{j,i})}_{\mathbf{z}'_{0,i}} + (\mathbf{t}_i + \hat{\mathbf{p}}_i). \tag{1.6}$$

Thus, all we need is a strategy that allows the reduction to obtain low-norm vectors that recode from $\mathbf{B}$ to $\mathbf{t}_i + \hat{\mathbf{p}}_i$ for each $i \in [N]$. Moreover, these vectors will be chosen from a discrete Gaussian distribution with sufficient width to smudge out $\mathbf{z}'_{0,i}$. The idea is to embed these smudging vectors in the public keys $\mathbf{t}_i$ for the honest users and in the adversarial shares $\hat{\mathbf{p}}_i$ for the corrupted users. Specifically, we do the following:

- For an honest user $i \in \mathcal{H}$, the reduction samples the public key as $\mathbf{t}_i = \mathbf{d}_i - \hat{\mathbf{p}}_i$, where $\mathbf{d}_i = \mathbf{B}\hat{\mathbf{z}}_{0,i}$ and $\hat{\mathbf{z}}_{0,i}$ is a smudging vector. We note that the marginal distribution of $\mathbf{t}_i$ remains uniform (as in the real scheme), and thus, this change does not affect the view of the adversary. Here, we also critically rely on the fact that the adversary commits to the policy P before it can make key-generation queries. This means the reduction algorithm can sample the shares $\hat{\mathbf{p}}_i$ for the honest parties and embed them into the public keys for the honest users. With this choice of $\mathbf{t}_i$, Eq. (1.6) now becomes

  $$-\mathbf{C}_0 \mathbf{v}_{N,i} = (\mathbf{t}_i + \hat{\mathbf{p}}_i) - \mathbf{B}\mathbf{z}'_{0,i} = \mathbf{B}(\hat{\mathbf{z}}_{0,i} - \mathbf{z}'_{0,i}).$$

  The reduction takes $\mathbf{z}_{0,i} = \hat{\mathbf{z}}_{0,i} - \mathbf{z}'_{0,i}$ to be the low-norm opening.

- For a corrupted user $i \in [N] \setminus \mathcal{H}$ (with adversarially-chosen public key $\mathbf{t}_i$), the reduction needs to first extract an associated (low-norm) secret key $\mathbf{y}_i \in \{0,1\}^{4m^5}$ where $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i$. This can be accomplished by requiring the adversary to include a (simulation-sound) non-interactive zero-knowledge (NIZK) proof of knowledge of the associated secret key $\mathbf{y}_i$ as part of their public key. Next, in the security reduction, the reduction programs the share $\hat{\mathbf{p}}_i$ associated with the malicious party to be $\hat{\mathbf{p}}_i = \mathbf{B}\hat{\mathbf{z}}_{0,i}$. This step relies on the property that the marginal distribution of the shares associated with an unauthorized group of users is statistically close to uniform. This property holds whenever the linear secret sharing scheme satisfies a linear independence property (see Appendix B). Then, for all $i \in [N] \setminus \mathcal{H}$, Eq. (1.6) becomes

  $$-\mathbf{C}_0 \mathbf{v}_{N,i} = (\mathbf{t}_i + \hat{\mathbf{p}}_i) - \mathbf{B}\mathbf{z}'_{0,i} = \mathbf{B}(\mathbf{y}_i + \hat{\mathbf{z}}_{0,i} - \mathbf{z}'_{0,i}).$$

  The reduction takes $\mathbf{z}_{0,i} = \mathbf{y}_i + \hat{\mathbf{z}}_{0,i} - \mathbf{z}'_{0,i}$ to be the low-norm opening. Furthermore, in order to simulate the share generation randomness for sharing $\mathbf{p}$ (since it is part of the ciphertext), we require that the linear secret sharing scheme be explainable. That is, given a collection of shares, there is an efficient method to sample appropriately-distributed share-generation randomness. Such a property is straightforward to achieve for linear secret sharing schemes.

In either case, the reduction can simulate the virtual user's openings $\mathbf{z}_{0,i}$, meaning the reduction can now simulate the challenge ciphertext as a whole. Note that we additionally have to ensure that the openings $\mathbf{z}_{0,i}$ have the "right" distribution (i.e., match those that would be output by the sampler). We can achieve this by relying on a smudging lemma from [CHW25]; namely, as long as the reduction algorithm samples $\hat{\mathbf{z}}_{0,i}$ from a sufficiently-wide discrete Gaussian, we can argue that the resulting distribution of $\mathbf{z}_{0,i}$ is statistically close to the distribution output by the sampler in the real scheme. It is also important to observe that in the above construction, for each user $i \in [N]$ appearing in the challenge ciphertext, there is a unique and independent variable where the reduction can introduce the required smudging term to simulate the openings in the challenge ciphertext: for honest users, this is in the public keys $\mathbf{t}_i$ while for dishonest users, this is in the shares $\hat{\mathbf{p}}_i$.

One-round distributed decryption. Finally, we show how to modify our scheme to support a one-round distributed decryption protocol. This means that there is a hint-generation algorithm which given a ciphertext ct and secret key sk_i, outputs a hint ht_i. The correctness requirement says that a set of hints from an authorized decryption quorum is sufficient to reconstruct the message. Security says that the ciphertext ct should computationally hide the message if the adversary has decryption hints on ct from any unauthorized set of users *and* any collection of decryption hints for other ciphertexts $\mathsf{ct}' \neq \mathsf{ct}$. This is formalized via a "CCA-style" security game where the adversary can query the challenger on ciphertext-index pairs (ct', i) and the challenger replies with the decryption hint ht_i from user i on ct'. Thus, to prove security in this model, we need to augment the reduction with a method to answer these hint-computation queries.

The structure of the above scheme directly lends itself to supporting a one-round decryption algorithm. Namely, we can take the decryption hint for a user with secret key $\mathbf{y}_i$ to simply be $\underset{\sim}{\mathbf{s}}^\mathsf{T}\underset{\sim}{\mathbf{B}} \cdot \mathbf{y}_i$ (where $\underset{\sim}{\mathbf{s}}^\mathsf{T}\underset{\sim}{\mathbf{B}}$ is the vector from the ciphertext). Recall from Eq. (1.3) that this is the only component in the decryption relation that depends on the user's secret key $\mathbf{y}_i$. Given the decryption hints from an authorized set of users, one can recover each user's share of the message via Eq. (1.3), and from there, the message.

However, this basic approach cannot be secure, as the hint-generation algorithm would essentially be taking a vector $\mathbf{c} \in \mathbb{Z}_q^m$ as input (from the ciphertext) and outputting the inner product $\mathbf{c}^\mathsf{T}\mathbf{y}_i$, where $\mathbf{y}_i$ is the user's secret key. Since linear functions are easy to learn, the hints generated using the basic procedure would completely leak the user's secret key. Thus, we need to modify this basic hint generation procedure to not leak anything about the secret key. Looking towards the security reduction, we in fact aim for the stronger property that the output of the hint-generation algorithm can be simulated *without* knowledge of the user's secret key. Our approach for doing this requires making two small modifications to the construction:

- First, we require that the ciphertext additionally include a NIZK proof of knowledge that the vector $\mathbf{c}^\mathsf{T} = \underset{\sim}{\mathbf{s}}^\mathsf{T}\underset{\sim}{\mathbf{B}}$ in the ciphertext is a well-formed LWE instance with respect to the matrix $\mathbf{B}$. Specifically, the encrypter must prove knowledge of a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a low-norm vector $\mathbf{e}^\mathsf{T}$ such that $\mathbf{c}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}$. The hint-generation algorithm will output $\perp$ if the NIZK proof component of the ciphertext does not verify.

- Next, the hint-generation algorithm takes the input ciphertext component $\mathbf{c}^\mathsf{T} = \underset{\sim}{\mathbf{s}}^\mathsf{T}\underset{\sim}{\mathbf{B}}$ and outputs $\mathbf{c}^\mathsf{T}\mathbf{y}_i + \tilde{e}$, where $\tilde{e}$ is freshly-sampled smudging term. Namely, it now outputs a *noisy* inner product rather than the exact inner product.

If the NIZK is valid, then $\mathbf{c}^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}$. Next, the fact that $\mathbf{B}\mathbf{y}_i = \mathbf{t}_i$, where $\mathbf{t}_i$ is the user's public key, we have

$$\mathbf{c}^\mathsf{T}\mathbf{y}_i + \tilde{e} = \mathbf{s}^\mathsf{T}\mathbf{B}\mathbf{y}_i + \mathbf{e}^\mathsf{T}\mathbf{y}_i + \tilde{e} = \mathbf{s}^\mathsf{T}\mathbf{t}_i + \mathbf{e}^\mathsf{T}\mathbf{y}_i + \tilde{e},$$

If $\mathbf{e}$ is low-norm (as ensured by the NIZK proof), then $\mathbf{e}^\mathsf{T}\mathbf{y}_i$ is also low-norm. As long as $\tilde{e}$ is sampled from a sufficiently-wide distribution so as to smudge out $\mathbf{e}^\mathsf{T}\mathbf{y}_i$, then the distribution of $\mathbf{c}^\mathsf{T}\mathbf{y}_i + \tilde{e}$ is statistically indistinguishable from the distribution of $\mathbf{s}^\mathsf{T}\mathbf{t}_i + \tilde{e}$. The key observation now is that the distribution of $\mathbf{s}^\mathsf{T}\mathbf{t}_i + \tilde{e}$ is a function of only the public key $\mathbf{t}_i$. Correspondingly, the adversary (who constructs the ciphertext component $\mathbf{c}$ from $\mathbf{s}$ and $\mathbf{e}$) could have simulated the output of the hint-generation algorithm itself. This ensures that the decryption hints do not leak information about the underlying secret key. We note that the use of noise smudging to support distributed one-round decryption is a common technique in lattice-based cryptography (cf., [MW16, BGG+18]).

Formally, in the security analysis, the reduction algorithm will answer hint-generation queries by first extracting the encryption randomness $\mathbf{s}$ from the ciphertext and then responding with $\mathbf{s}^\mathsf{T}\mathbf{t}_i + \tilde{e}$, where $\mathbf{t}_i$ is the public key for user i. This procedure no longer requires knowledge of the secret key $\mathbf{y}_i$ (which the reduction cannot know). We provide the full details of our construction and analysis in Section 3.

An open problem: beyond DNFs from lattices. Our construction in this work supports the class of DNF policies, and it is an open problem to support policies beyond DNFs. A natural target is general threshold policies (recall that DNFs can only represent threshold polices where the threshold is constant). The reason our current scheme does not support general threshold policies is because we do not have a linear secret sharing scheme for threshold policies with the requisite properties. Recall that for our construction, we require that the linear secret sharing scheme associated with the policy satisfy two properties: (1) the reconstruction coefficients are binary, or more generally, small; and (2) the rows belonging to an unauthorized set of parties are linearly independent. To our knowledge, existing linear secret sharing schemes for threshold policies fall short on at least one of the two properties:

- **Shamir secret sharing.** The classic secret sharing scheme by Shamir [Sha79] over $\mathbb{Z}_q$ (for prime q) satisfies the second property, but not the first. Namely, the reconstruction coefficients in Shamir's secret sharing scheme are large values over $\mathbb{Z}_q$. While there are techniques such as "clearing out the denominators" [BGG+18] that allow integrating Shamir secret sharing with lattice-based constructions, the resulting construction no longer satisfy succinctness. Specifically, the approach from [BGG+18] implements Shamir reconstruction over the *integers*, and necessitates working with a lattice modulus whose bit-length scales with the number of users. In the context of our work, such an instantiation would yield a scheme with the ciphertext size scales linearly with the number of users. The primary goal in distributed monotone-policy encryption is to achieve sublinear-size ciphertexts. (Otherwise, a trivial construction of distributed monotone-policy encryption with linear-size ciphertexts is to share the message using a linear secret sharing scheme and individually encrypt each share to each user.[4])
- **Secret sharing with small reconstruction coefficients.** The work of [DDP+25] uses the (ramp) threshold secret sharing scheme with small reconstruction coefficients from [ANP23] to build a threshold traitor tracing scheme. The linear secret sharing scheme from [ANP23] is based on erasure codes. To the best of our knowledge, we do not currently know of an explicit instantiation of this template that simultaneously supports small reconstruction coefficients and satisfies the additional linear independence property we require.

It is an interesting question to design linear secret sharing schemes for threshold policies that satisfy both requirements. Such a scheme would not only yield distributed monotone-policy encryption for a broader policy family via our framework, but would also find applications to other settings such as lattice-based multi-authority attribute-based encryption [DKW21] as well as optimal threshold traitor tracing [DDP+25].[5] It would also be interesting to develop alternative paths for constructing lattice-based distributed monotone-policy encryption that do not rely on linear secret sharing schemes with these additional properties.

Concurrent work. In a concurrent and independent work, Afshar, Goyal, and Yadugiri [AGY26] also show how to construct a distributed monotone-policy encryption scheme supporting DNF policies from the decomposed LWE assumption in the random oracle model. Their results and construction follow a similar approach as our work. Here, we describe the main differences between the two:

[4]Note that this basic construction does not support the ability to preprocess a set of public keys into a succinct master public key that represents the set of users and the respective policy (as required in the notion of threshold encryption with silent setup [GKPW24]). However, we can easily extend this basic approach by replacing the public-key encryption scheme with a registration-based encryption scheme [GHMR18, DKL+23]. This now yields a scheme that supports preprocessing (yielding a short master public key) and linear-size ciphertexts.

[5]The security analysis of [DDP+25, Theorem 8] implicitly relies on the additional property that the shares for any unauthorized set of users are independent and uniformly random (which corresponds to the linear independence property we define).

- **Public-key aggregation.** The work of [AGY26] requires an explicit preprocessing algorithm that takes a set of users together with the policy and outputs a succinct encryption key. This is a standard requirement in silent threshold cryptography [GKPW24, WW26]. Given the preprocessed key, the encryption algorithm can run in time that is independent of the number of parties and the policy. Since preprocessing is not an explicit requirement in notions like distributed broadcast encryption [WQZDF10, BZ14] and distributed monotone-policy encryption [DJWW25], we did not explicitly require it. We do note that our scheme can support this type of preprocessing (see the discussion immediately following Eq. (1.4)). In particular, it inherits it from the underlying distributed broadcast encryption scheme.
- **Security.** In this work, we consider semi-selective security where the adversary has to declare the challenge policy as well as the indices of the honest users *after* it sees the CRS and before it makes key-generation queries. Our definition allows the adversary to choose the public keys for the corrupted users as well as make hint-generation queries on arbitrary non-challenge ciphertexts of the adversary's choosing. The work of [AGY26] considers a weaker selective security notion where the adversary commits to the policy and the indices of the honest users ahead of time. The adversary in the selective security model from [AGY26] cannot choose the public keys for the corrupted users or make any hint-generation queries in the security game. Neither work allows the adversary to corrupt an honest user (i.e., learn the secret key associated with an honest user; in Definition 2.15, we refer to this as a "static" model). Both works prove security from the decomposed LWE assumption in the random oracle model.

2 Preliminaries

We begin with with some basic notation and definitions. Our presentation here is taken largely verbatim from [CHW25, §3]. We write λ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n] := \{1, \ldots, n\}$. For a set S, we write 2^S to denote the power set of S (i.e., the set of all subsets of S). For a finite set S, we write $x \xleftarrow{R} S$ to denote that x is sampled uniformly from S, and 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 2.1 (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 $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$. 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. On each query, the challenger proceeds as follows:
    - The challenger first checks if $C(x,w) = 1$. If not, the challenger responds with $\perp$.
    - 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 game as follows:

- The challenger samples $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{TrapSetup}(1^\lambda)$ 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} is simulation extractable 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] shows how to construct a simulation-sound extractable NIZK for NP from any NIZK for NP and a public-key encryption scheme. 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 2.2 (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.

2.1 Lattice Preliminaries

We now recall some basic facts about lattices. Parts of this section are taken verbatim from [CHW25, §3.1]. 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 $\|\mathbf{V}\| = \max_{i,j} |V_{i,j}|$. We write $\|\mathbf{v}\|_2$ to denote the ℓ_2-norm of $\mathbf{v}$ (i.e., $\|\mathbf{v}\|_2^2 := \sum_{i \in [n]} v_i^2$). 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 $(-q/2, q/2]$.

Kronecker 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 Kronecker product. 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 the (generalized) leftover hash lemma [HILL99, DORS08, ABB10]:

Lemma 2.3 (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 $k = \mathrm{poly}(n)$, the statistical distance between the following distributions is $\mathsf{negl}(n)$:*

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

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 $\mathbf{A}\mathbf{x} = \mathbf{y} \bmod q$. 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 $\mathbf{G}_n = \mathbf{I}_n \otimes \mathbf{g}^\intercal \in \mathbb{Z}_q^{n\times m'}$ be the gadget matrix [MP12] where $\mathbf{I}_n$ is the identity matrix of dimension n, $\mathbf{g}^\intercal = [1, 2, \ldots, 2^{\lceil \log q \rceil - 1}]$, and $m' = n\lceil \log q \rceil$. For $m > m'$, we overload $\mathbf{G}_n$ to denote the padded gadget matrix $\mathbf{G}_n = [\mathbf{I}_n \otimes \mathbf{g}^\intercal \mid \mathbf{0}^{n\times(m-m')}]$. We now recall some basic properties of the discrete Gaussian distribution.

Lemma 2.4 (Gaussian Tail Bound [MP12, Lemma 2.6, adapted]). *For all $\sigma > 0$ and any $\lambda \in \mathbb{N}$,*

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

Lemma 2.5 (Gaussian Samples [GPV08, adapted]). *Let n, m, q, σ be lattice parameters such that $\sigma \ge \log m$, $m \ge 2n \log q$, 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 $\mathsf{negl}(n)$:*

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

Smudging lemmas. We now recall a standard noise smudging lemma along with the Gaussian preimage smudging lemma from [CHW25]:

Lemma 2.6 (Noise Smudging [BDE+18]). *Let λ be a security parameter, $a \in \mathbb{Z}$ be an integer such that $|a| \le B$, and $\sigma \ge B \cdot \lambda^{\omega(1)}$. Then, the statistical distance between the following distributions is $\mathsf{negl}(\lambda)$:*

$$\{e : e \leftarrow D_{\mathbb{Z},\sigma}\} \quad \textit{and} \quad \{a + e : e \leftarrow D_{\mathbb{Z},\sigma}\}.$$

Lemma 2.7 (Gaussian Preimage Smudging [CHW25, Theorem 4.3]). *Let n, m, q be lattice parameters such that $m \ge 2n \log q$ 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 $O(\sqrt{\|\mathbf{z}\|_2 / \sigma})$:*

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

Decomposed LWE. The learning with errors (LWE) assumption [Reg05] with parameters (n, m, q, σ) states that the following distributions are computationally indistinguishable:

$$(\mathbf{B}, \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}^\intercal) \text{ and } (\mathbf{B}, \mathbf{v}^\intercal) \quad \text{where} \quad \mathbf{B} \xleftarrow{R} \mathbb{Z}_q^{n\times m},\ \mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n,\ \mathbf{e} \xleftarrow{R} D_{\mathbb{Z},\sigma}^m,\ \mathbf{v} \xleftarrow{R} \mathbb{Z}_q^m.$$

For a parameter $\ell \in \mathbb{N}$, the ℓ-decomposed LWE assumption from [AMR25] asserts that LWE is hard when the matrix $\mathbf{B}$ is sampled in a structured manner:

$$\mathbf{B} := \mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) + \mathrm{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G} \quad \text{where} \quad \mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m},\ \mathbf{R} \leftarrow D_{\mathbb{Z}}^{m\times \ell m}.$$

We now give the formal statement of the assumption:

Assumption 2.8 (Decomposed LWE [AMR25]). Let λ be a security parameter and let $n = n(\lambda), m = m(\lambda), q = q(\lambda), \sigma = \sigma(\lambda)$ be lattice parameters. Let $s > 0$ be a Gaussian width parameter and $\ell \in \mathbb{N}$ be a

dimension. We say that the (ℓ, s)-decomposed LWE assumption with parameters (n, m, q, σ) holds if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$:

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

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times \ell^2 m}$, $\mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m}$, $\mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m}$ $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\sigma}^m$, and $\mathbf{v} \xleftarrow{\text{R}} \mathbb{Z}_q^{\ell^2 m}$. We will sometimes write "ℓ-decomposed LWE" as shorthand to refer to an instance of the (ℓ, s)-decomposed LWE assumption for some (implicitly-defined) width parameter s.

We now show that the analogs of Lemmas 2.3, 2.5 and 2.7 hold even for the structured matrix $\mathbf{B}$ used in the decomposed LWE assumption. These properties will be needed in our security proof. We provide the formal proofs of these statements in Appendix A.

Corollary 2.9 (Leftover Hash Lemma for Decomposed LWE Matrix)**.** *Let n, m, q be integers such that $m \geq 2n \log q$ and q is prime. Let $\ell \in \mathbb{N}$ and suppose $s \geq \log m$. Then, for all fixed vectors $\mathbf{e} \in \mathbb{Z}_q^{\ell^2 m}$ and all $k = \mathsf{poly}(n)$, the statistical distance between the following distributions is $\mathsf{negl}(n)$:*

- $\left\{(\mathbf{W}, \mathbf{R}, \mathbf{BK}, \mathbf{e}^\intercal\mathbf{K}) : \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m},\ \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m},\ \mathbf{K} \xleftarrow{\text{R}} \{0,1\}^{\ell^2 m \times k}\right\}$*; and*
- $\left\{(\mathbf{W}, \mathbf{R}, \mathbf{U}, \mathbf{e}^\intercal\mathbf{K}) : \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m},\ \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m},\ \mathbf{U} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times k},\ \mathbf{K} \xleftarrow{\text{R}} \{0,1\}^{\ell^2 m \times k}\right\}$,

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times \ell^2 m}$ in both distributions.

Corollary 2.10 (Gaussian Samples for Decomposed LWE Matrix)**.** *Let n, m, q, σ be lattice parameters such that $\sigma \geq \log m$, $m \geq 2n \log q$, and q is prime. Let $\ell \in \mathbb{N}$ and suppose $s \geq \log m$. Then, the statistical distance between the following distributions is $\mathsf{negl}(n)$:*

- $\left\{(\mathbf{W}, \mathbf{R}, \mathbf{x}, \mathbf{Bx}) : \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m}, \mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}^{\ell^2 m}\right\}$*; and*
- $\left\{(\mathbf{W}, \mathbf{R}, \mathbf{x}, \mathbf{y}) : \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m}, \mathbf{y} \xleftarrow{\text{R}} \mathbb{Z}_q^n, \mathbf{x} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{y})\right\}$,

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times \ell^2 m}$ in both distributions.

Corollary 2.11 (Gaussian Preimage Smudging for Decomposed LWE Matrix)**.** *Let n, m, q be lattice parameters such that $m \geq 2n \log q$ and q is prime. Let $\ell \in \mathbb{N}$ and $s \geq \log m$. Then for all vectors $\mathbf{y} \in \mathbb{Z}_q^n$, $\mathbf{z} \in \mathbb{Z}_q^{\ell^2 m}$, and all width parameters $\sigma > \lambda^{\omega(1)} m \|\mathbf{z}\|_2$, the statistical distance between the following distributions is $\mathsf{negl}(\lambda)$:*

$$\left\{(\mathbf{W}, \mathbf{R}, \mathbf{x}) : \begin{array}{c} \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m} \\ \mathbf{x} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{y} + \mathbf{Bz}) \end{array}\right\} \quad \textit{and} \quad \left\{(\mathbf{W}, \mathbf{R}, \mathbf{x}) : \begin{array}{c} \mathbf{W} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m \times \ell m} \\ \mathbf{x} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{y}) + \mathbf{z} \end{array}\right\},$$

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times \ell^2 m}$.

Matrix commitments. The main building block we use in this work is the matrix commitments by Wee [Wee25]. In Wee's construction, a commitment to a matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ is a matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ such that

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

where $\mathbf{B}, \mathbf{V}_L$ are matrices defined by a set of *public* parameters pp, and $\mathbf{Z}$ is a low-norm opening. The matrix $\mathbf{V}_L$ also has low norm. As first observed in [WW25], the running time of the commit algorithm only depends on the number of non-zero columns of $\mathbf{M}$ (and only logarithmically with the width of $\mathbf{M}$). This allows one to commit to *sparse* matrices $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$ where $L = 2^\lambda$, provided that $\mathbf{M}$ contains only a polynomial number of non-zero columns. In this setting, there are efficient algorithms for computing individual columns of the matrix $\mathbf{V}_L$ and opening $\mathbf{Z}$. We summarize the main properties from [Wee25, WW25] below. Specifically, we describe the version from [Wee25, Appendix C.2] where the public parameters pp can be described by a uniform random string. This latter property will enable constructions with a transparent setup.

Theorem 2.12 (Sparse Matrix Commitment [Wee25, WW25, adapted])**.** *Let n, m, q be lattice parameters where $m \geq 2n \log q$. There exists a triple of efficient and* deterministic *algorithm* $(\mathsf{Com}, \mathsf{Ver}, \mathsf{Open})$ *with the following syntax and properties:*

- $\mathsf{Com}(\mathsf{pp}_{\mathsf{com}}, \mathbf{M}) \to \mathbf{C}$*: On input the public parameters* $\mathsf{pp}_{\mathsf{com}}$ *and a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$*, the* Com *algorithm outputs a commitment matrix* $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ *in time* $\mathrm{poly}(K, m, \log q, \log L)$*, where K is the number of non-zero columns in* $\mathbf{M}$*.*[6]
- $\mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, L, i) \to \mathbf{v}_{L,i}$*: On input the public parameters* $\mathsf{pp}_{\mathsf{com}}$*, the length parameter $L \in \mathbb{N}$ (in* binary*), and an index $i \in [L]$, the* Ver *algorithm outputs a vector* $\mathbf{v}_{L,i} \in \mathbb{Z}_q^m$ *in time* $\mathrm{poly}(m, \log q, \log L)$*.*
- $\mathsf{Open}(\mathsf{pp}_{\mathsf{com}}, \mathbf{M}, i) \to \mathbf{z}_i$*: On input the public parameters* $\mathsf{pp}_{\mathsf{com}}$*, a matrix* $\mathbf{M} \in \mathbb{Z}_q^{n \times L}$*, and an index $i \in [L]$, the* Open *algorithm outputs a vector* $\mathbf{z}_i \in \mathbb{Z}_q^{4m^5}$ *in time* $\mathrm{poly}(K, m, \log q, \log L)$*, where K is the number of non-zero columns in* $\mathbf{M}$*.*

For all $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$ *where* $\mathbf{W} \in \mathbb{Z}_q^{n \times 2m^3}$, $\mathbf{R} \in \mathbb{Z}_q^{m \times 2m^3}$, *and* $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathrm{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times 4m^5}$*, all parameters $L \in \mathbb{N}$, all matrices $\mathbf{M} = [\mathbf{m}_1 \mid \cdots \mid \mathbf{m}_L] \in \mathbb{Z}_q^{n \times L}$, all indices $i \in [L]$, and setting*

$$\begin{aligned} \mathbf{C} &= \mathsf{Com}(\mathsf{pp}_{\mathsf{com}}, \mathbf{M}) && \in \mathbb{Z}_q^{n \times m} \\ \mathbf{v}_{L,i} &= \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, L, i) && \in \mathbb{Z}_q^{m} \\ \mathbf{z}_i &= \mathsf{Open}(\mathsf{pp}_{\mathsf{com}}, \mathbf{M}, i) && \in \mathbb{Z}_q^{4m^5}, \end{aligned}$$

the following hold:

$$\mathbf{C}\mathbf{v}_{L,i} = \mathbf{m}_i - \mathbf{B}\mathbf{z}_i \quad \text{and} \quad \|\mathbf{v}_{L,i}\| \leq O(\|\mathbf{R}\| \cdot m^4 \log q) \quad \text{and} \quad \|\mathbf{z}_i\| \leq O(\|\mathbf{R}\| \cdot m^7 \log q \log L).$$

Explainable re-randomizer. The security analysis of our construction relies on the re-randomization technique from [CHW25]. Specifically, the security proof relies on being able to take a sample from a discrete Gaussian distribution (over a lattice coset) and produce a set of random coins (for the discrete Gaussian sampler) that would explain the sample. We use the explainable sampling syntax from [LW22] who showed how to construct an explainable sampler for the discrete Gaussian distribution over the integers. In this work, we use the specific formulation from [WW25] based on matrix commitments, which we recall below (after adapting the statement to the setting with a structured matrix $\mathbf{B}$):

Theorem 2.13 (Explainable Re-Randomizer [WW25, adapted])**.** *Let λ be a security parameter, n, m, q be lattice parameters, and ρ be a randomness parameter. There exists a pair of efficient algorithms* $(\mathsf{Sample}, \mathsf{Explain})$ *with the following syntax:*

[6] Here, we assume that the description of $\mathbf{M}$ is provided in the form of a list of non-zero columns.

- $\mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^\lambda, 1^N, \sigma; r) \to (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$: *On input the public parameters* $\mathsf{pp}_{\mathsf{com}}$*, the security parameter* λ*, the dimension* N*, a width parameter* $\sigma > 0$*, and randomness* $r \in \{0,1\}^\rho$*, the sampler algorithm outputs* $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$ *together with* $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^{4m^5}$*.*

- $\mathsf{Explain}(\mathsf{pp}_{\mathsf{com}}, 1^\lambda, 1^\kappa, (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N), \sigma)$: *On input the public parameters* $\mathsf{pp}_{\mathsf{com}}$*, a security parameter* 1^λ*, a precision parameter* κ*, a matrix* $\mathbf{C} \in \mathbb{Z}_q^{n \times m}$*, vectors* $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{Z}_q^{4m^5}$*, and a width parameter* $\sigma > 0$*, the explain algorithm outputs randomness* $r \in \{0,1\}^\rho$*.*

There exists a polynomial $\rho = \mathsf{poly}(\lambda, N, n, m, \log q)$ *such that for all* $n \geq \lambda$ *and* $m \geq 2n \log q$*, all* $N \in \mathbb{N}$*, all* $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$ *where* $\mathbf{W} \in \mathbb{Z}_q^{n \times 2m^3}$*,* $\mathbf{R} \in \mathbb{Z}_q^{m \times 2m^3}$*, and* $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times 4m^5}$*, the following two properties hold:*

- **Sampling distribution:** *For all width parameters* $\|\mathbf{R}\| \cdot O(m^{12}N^3) < \sigma < 2^\lambda$*, the statistical distance between the following distributions is bounded by* $\mathsf{negl}(\lambda)$*:*
  - $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^\lambda, 1^N, \sigma; r)$ *where* $r \xleftarrow{\mathrm{R}} \{0,1\}^\rho$*.*
  - $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N)$ *where* $\mathbf{C} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$ *and for all* $i \in [N]$*,* $\mathbf{y}_i \leftarrow \mathbf{B}_\sigma^{-1}(-\mathbf{C}\mathbf{v}_{N,i})$*,* $\mathbf{v}_{N,i} = \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, N, i)$*, and* $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$*.*

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

- **Explainability:** *There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all precision parameters* κ*, dimensions* $N \in \mathbb{N}$*, and width parameters* $\|\mathbf{R}\| \cdot O(m^{12}N^3) < \sigma < 2^\lambda$*, the statistical distance between the following distributions of* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N, r)$ *is bounded by* $1/\kappa + \mathsf{negl}(\lambda)$*:*
  - $\mathcal{D}_{\mathsf{Sample}}$*: Sample* $r \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ *and* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^\lambda, 1^N, \sigma; r)$*.*
  - $\mathcal{D}_{\mathsf{Explain}}$*: Sample* $r' \xleftarrow{\mathrm{R}} \{0,1\}^\rho$ *and* $(\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N) \leftarrow \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^\lambda, 1^N, \sigma; r')$*. Then, sample* $r \leftarrow \mathsf{Explain}(\mathsf{pp}_{\mathsf{com}}, 1^\lambda, 1^\kappa, (\mathbf{C}, \mathbf{y}_1, \ldots, \mathbf{y}_N), \sigma)$*.*

2.2 Distributed Monotone-Policy Encryption

In this section, we recall the notion of distributed monotone-policy encryption with one-round distributed decryption from [DJWW25, Appendix B]. Relative to [DJWW25], we introduce an explicit IsValid predicate for checking whether a public key is well-formed or not. Then, in the security definition, the adversary is restricted to choosing public keys that satisfy the IsValid predicate. In addition, much like prior work on distributed broadcast encryption [WQZDF10, BZ14, KMW23], we also associate each key with an index $\mathsf{id} \in [2^\lambda]$ and users encrypt to a set of public keys associated with *distinct* indices. In our construction (see Section 3), we allow the indices to be an arbitrary λ-bit string (e.g., they can be the user's name in an identity-based setting or just sampled uniformly at random). Note that there are generic compilers based on cuckoo hashing [FKdP23] or bipartite matching [GLWW23] that can be used to remove the need to associate public keys with indices. We now give the full definition:

Definition 2.14 (Distributed Monotone-Policy Encryption [DJWW25, adapted])**.** Let λ be a security parameter and $\mathcal{P}$ be a family of monotone access policies. We model each policy $P \in \mathcal{P}$ as a monotone Boolean function. When $P\colon \{0,1\}^n \to \{0,1\}$ is a function on n-bit inputs and $T \subset [n]$, we will sometimes overload notation and define $P(T)$ to be $P(\beta_1, \ldots, \beta_n)$ where $\beta_i = 1$ if $i \in T$ and $\beta_i = 0$ otherwise. A distributed monotone-policy encryption scheme with policy space $\mathcal{P}$ is a tuple of efficient algorithms $\Pi_{\mathsf{MPE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Encrypt}, \mathsf{GetHint}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: On input the security parameter $\lambda \in \mathbb{N}$, the setup algorithm outputs a set of public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{id}) \to (\mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}})$: On input the public parameters pp and an index $\mathsf{id} \in [2^\lambda]$, the key-generation algorithm outputs a public key $\mathsf{pk}_{\mathsf{id}}$ and a secret key $\mathsf{sk}_{\mathsf{id}}$. We assume without loss of generality that $\mathsf{pk}_{\mathsf{id}}$ includes the associated index id.
- $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to b$: On input the public parameters pp, an index $\mathsf{id} \in [2^\lambda]$, and a public key pk, the validity-checking algorithm outputs a bit $b \in \{0,1\}$.
- $\mathsf{Encrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \mu) \to \mathsf{ct}$: On input the public parameters pp, a policy $P \in \mathcal{P}$ (on n-bit inputs), a tuple of n public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, and a message $\mu \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct. Note that the input length n is determined by P and is *not* fixed by the scheme.
- $\mathsf{GetHint}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct}) \to \mathsf{ht}_i$: On input the public parameters pp, a secret key sk_i, and a ciphertext ct, the hint generation algorithm outputs a decryption hint ht_i.
- $\mathsf{Decrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \{(i, \mathsf{ht}_i)\}_{i \in T}, \mathsf{ct}) \to \mu$: On input the public parameters pp, a policy $P \in \mathcal{P}$, a tuple of n public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, a set $T \subseteq [n]$, decryption hints ht_i for $i \in T$, and a ciphertext ct, the decryption algorithm outputs a message $\mu \in \{0,1\}$.

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

- **Completeness:** For all security parameters $\lambda \in \mathbb{N}$, all indices $\mathsf{id} \in [2^\lambda]$, all public parameters pp in the support of $\mathsf{Setup}(1^\lambda)$, and all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{id})$, we have $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}, \mathsf{pk}) = 1$.
- **Correctness:** For all security parameters $\lambda \in \mathbb{N}$, all policies $P \in \mathcal{P}$ (on n-bit inputs), all sets $T \subseteq [n]$ where $P(T) = 1$, all tuples of distinct indices $\mathsf{id}_1, \ldots, \mathsf{id}_n \in [2^\lambda]$, all messages $\mu \in \{0,1\}$, all public parameters pp in the support of $\mathsf{Setup}(1^\lambda)$, and all public keys pk_i for $i \notin T$,

$$\Pr\left[\mathsf{Decrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \{(i, \mathsf{ht}_i)\}_{i \in T}, \mathsf{ct}) = \mu\right] = 1,$$

  where $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{id}_i)$ for every $i \in T$, $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \mu)$, and $\mathsf{ht}_i \leftarrow \mathsf{GetHint}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$ for all $i \in T$.[7]
- **Security:** For a security parameter λ, an adversary $\mathcal{A}$, and a bit $b \in \{0,1\}$, we define the security game as follows:
  - **Setup:** At the beginning of the game, the challenger samples the public parameters $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and initializes a counter $\mathsf{ctr} = 0$ and an (empty) list $C = \emptyset$. The list C is used to keep track of corrupted keys. The challenger gives pp to $\mathcal{A}$.
  - **Pre-challenge query phase:** The adversary can now make the following queries:
    - **Key-generation query:** In a key-generation query on an index $\mathsf{id} \in [2^\lambda]$, the challenger increments the counter $\mathsf{ctr} = \mathsf{ctr} + 1$, samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{id})$, and responds with $\mathsf{pk}_{\mathsf{ctr}}$.

[7] Note that we can also define a more restricted version of correctness that only requires correctness to hold for all public keys pk_i where $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}_i, \mathsf{pk}_i) = 1$ for all $i \notin T$.

* **Corruption query:** In a corruption query, the adversary specifies a counter value $\mathsf{ctr}' \leq \mathsf{ctr}$, and the challenger replies with $\mathsf{sk}_{\mathsf{ctr}'}$. The adversary adds ctr' to C.
* **Hint query:** In a hint query, the adversary specifies a ciphertext ct and a counter value $\mathsf{ctr}' \leq \mathsf{ctr}$. The challenger replies with $\mathsf{GetHint}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ctr}'}, \mathsf{ct})$.

– **Challenge phase:** In the challenge phase, the adversary specifies a challenge policy $P \in \mathcal{P}$ together with a tuple of distinct indices $(\mathsf{id}_1, \ldots, \mathsf{id}_n)$. In addition, for each $i \in [n]$, algorithm $\mathcal{A}$ either specifies a public key pk_i or a counter value ctr_i. For each $i \in [n]$ where algorithm $\mathcal{A}$ specifies a counter value $\mathsf{ctr}_i \leq \mathsf{ctr}$, the challenger sets $\mathsf{pk}_i = \mathsf{pk}_{\mathsf{ctr}_i}$. The challenger then checks the following conditions for each $i \in [n]$:

* If algorithm $\mathcal{A}$ specifies a counter value ctr_i, then the challenger checks that id_i is the index associated with $\mathsf{pk}_i = \mathsf{pk}_{\mathsf{ctr}_i}$.
* If algorithm $\mathcal{A}$ specifies a public key pk_i, then the challenger checks $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}_i, \mathsf{pk}_i) = 1$.

If any check fails for an index $i \in [n]$, the challenger halts with output 0. Otherwise, the challenger replies to $\mathcal{A}$ with the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Encrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), b)$.

– **Post-challenge query phase:** The adversary can continue to make corruption and hint queries in this phase. If the adversary asks for a hint on the challenge ciphertext ct^*, the counter value ctr' is added to C. (Note that post-challenge key-generation queries are not useful).

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

Let $\beta_i = 1$ if the adversary specified a public key pk_i or a corrupted counter value ctr_i where $\mathsf{ctr}_i \in C$ during the challenge phase. For indices $i \in [n]$ where $\mathcal{A}$ specified an uncorrupted counter value $\mathsf{ctr}_i \notin C$, let $\beta_i = 0$. We say that $\mathcal{A}$ is admissible if $P(\beta_1, \ldots, \beta_n) = 0$. We say that Π_{MPE} is secure if for all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

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

in the above security game.

- **Succinctness:** There exists a polynomial $\mathsf{poly}(\cdot, \cdot)$ such that for all $\lambda \in \mathbb{N}$, public parameters pp in the support of $\mathsf{Setup}(1^\lambda)$, policies $P \in \mathcal{P}$ (on n-bit inputs), public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$, and messages $\mu \in \{0, 1\}$, the size of the ciphertext ct output by $\mathsf{Encrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \mu)$ is $o(|P|) \cdot \mathsf{poly}(\lambda, \log n)$.

Definition 2.15 (Static Security)**.** Let Π_{MPE} be a distributed monotone-policy encryption scheme with policy space $\mathcal{P}$. We say that Π_{MPE} satisfies static security if Π_{MPE} is secure against adversaries that does not make any corruption queries during the query phase and moreover, the adversary cannot make any hint queries on the challenge ciphertext in the post-challenge query phase. Importantly, the adversary can still choose public keys itself (for any non-accepting subset of parties) during the challenge phase.

Definition 2.16 (Semi-Policy-and-Query-Selective Security)**.** We say that a distributed monotone-policy scheme Π_{MPE} is *semi-policy-and-query-selective secure in the static model* if the adversary in Definition 2.14 is static and moreover, the adversary declares its challenge policy P, the challenge set of indices $(\mathsf{id}_1, \ldots, \mathsf{id}_n)$, and the indices associated with the uncorrupted users at the beginning of the pre-challenge phase (but *after* seeing the public parameters). In this model, the adversary is still allowed to (adaptively) choose the public keys for the corrupted users and make hint queries (subject to the same restriction of not making hint queries on the challenge ciphertext in the post-challenge query phase).

3 Distributed Monotone-Policy Encryption for DNFs

In this section, we give our construction of a distributed monotone-policy encryption scheme for DNF formulas under the decomposed LWE assumption (Assumption 2.8).

3.1 Building Block: Explainable Secret Sharing

Our construction uses an explainable linear secret sharing scheme with small reconstruction coefficients. The specific properties we require are twofold:

- **Independence of unauthorized shares:** The first requirement is that the shares for any unauthorized set of users are uniform and independent values from the share space.
- **Explainability:** The second requirement is an explainability property that says that there is an efficient algorithm that takes as input a set of shares output by the (honest) sharing algorithm and outputs a uniform random string r representing the share-generation randomness.

We introduce the abstraction here, and in Appendix B, show that both properties immediately follow from standard linear secret sharing schemes with a linear independence property [LW11, DKW21].

Definition 3.1 (Explainable $\{0,1\}$-Linear Secret Sharing)**.** Let $\mathcal{P}$ be a family of policies, $\mathcal{M}$ be a message space, and $\mathcal{S}$ be a share space (where $\mathcal{S}$ is a group under addition). An explainable $\{0,1\}$-linear secret sharing scheme is a triple of efficient algorithms $\Pi_{\mathsf{SS}} = (\mathsf{Share}, \mathsf{Reconst}, \mathsf{ExplainShares})$ with the following syntax:

- $\mathsf{Share}(P, \mu) \to (s_1, \ldots, s_n)$: On input a policy $P \in \mathcal{P}$ (over n parties) and a message $\mu \in \mathcal{M}$, the sharing algorithm outputs a tuple of shares $(s_1, \ldots, s_n) \in \mathcal{S}^n$ for each party.
- $\mathsf{Reconst}(P, T) \to T^*$: On input a policy $P \in \mathcal{P}$ (over n parties) and a set $T \subseteq [n]$, the reconstruction algorithm outputs a subset $T^* \subseteq T$.
- $\mathsf{ExplainShares}(P, \mu, (s_1, \ldots, s_n)) \to \gamma$: On input a policy $P \in \mathcal{P}$ (over n parties), a message $\mu \in \mathcal{M}$, and shares $(s_1, \ldots, s_n) \in \mathcal{S}^n$, the share-explanation algorithm outputs a string $\gamma \in \{0,1\}^*$.

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

- **Correctness:** For all policies $P \in \mathcal{P}$ (over n parties), messages $\mu \in \mathcal{M}$, and subsets $T \subseteq [n]$ where $P(T) = 1$, we have
$$\Pr\left[\sum_{i \in T^*} s_i = \mu : \begin{array}{c}(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(P, \mu) \\ T^* = \mathsf{Reconst}(P, T)\end{array}\right] = 1.$$
- **Independence of unauthorized shares:** There exists an efficient algorithm Complete such that for all policies $P \in \mathcal{P}$ (over n parties), all messages $\mu \in \mathcal{M}$, and all subsets $T \subseteq [n]$ where $P(T) = 0$, the following two distributions are identical:
  - Output $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(P, \mu)$.
  - Sample $s_i \xleftarrow{R} \mathcal{S}$ for all $i \in T$ and $\{(i, s_i)\}_{i \notin T} \leftarrow \mathsf{Complete}(P, \mu, \{(i, s_i)\}_{i \in T})$. Output $(s_1, \ldots, s_n)$.
- **Explainability:** Let $\rho_{\mathsf{SS}} = \rho_{\mathsf{SS}}(n)$ be the length of the randomness needed to sample shares for a policy over n parties. For all policies $P \in \mathcal{P}$ (over n parties) and messages $\mu \in \mathcal{M}$, the following two distributions are identical:

- $\mathcal{D}_{\mathsf{Sample}}$: Sample $r \xleftarrow{R} \{0,1\}^{\rho_{\mathsf{SS}}}$ and $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(P, \mu; r)$. Output $(\mu, s_1, \ldots, s_n, r)$.
- $\mathcal{D}_{\mathsf{Explain}}$: Sample $r' \xleftarrow{R} \{0,1\}^{\rho_{\mathsf{SS}}}$ and $(s_1, \ldots, s_n) \leftarrow \mathsf{Share}(P, \mu; r')$. Then, re-sample the randomness $r \leftarrow \mathsf{ExplainShares}(P, \mu, s_1, \ldots, s_n)$. Output $(\mu, s_1, \ldots, s_n, r)$.

Explainable linear secret sharing for DNFs. In Appendix B, we show that any $\{0,1\}$-linear secret sharing scheme satisfying a linear independence property on unauthorized shares [DKW21] implies an explainable $\{0,1\}$-linear secret sharing scheme. The construction from [LW11] (Fact B.4) immediately yields an explainable linear secret sharing scheme for policies that can be captured by DNF formulas:

Corollary 3.2 (Linear Secret Sharing for DNF Formulas)**.** *Let $q \in \mathbb{N}$ be a prime and $t \in \mathbb{N}$ be a dimension. Let $\mathcal{P}$ be the set of DNF formulas where each input variable is used exactly once. There exists an explainable $\{0,1\}$-linear secret sharing scheme with message space $\mathbb{Z}_q^t$ and share space $\mathbb{Z}_q^t$ for $\mathcal{P}$.*

3.2 Distributed Monotone-Policy Encryption Construction

We now give our main construction of a distributed monotone-policy encryption scheme for policies that can be captured by an explainable $\{0,1\}$-linear secret sharing scheme (e.g., which includes the class of DNF policies where each variable is used exactly once). While the basic scheme only supports DNFs where each variable appears exactly once in the policy, it is straightforward to support formulas where each variable appears at most K times by having each user generate K independent public keys, one for each copy of the variable. A similar "one-use" restriction also arises in pairing-based attribute-based encryption schemes and a standard solution (e.g., see [LOS+10]) is to simply create multiple independent copies of an attribute to represent multiple copies.

Construction 3.3 (Distributed Monotone-Policy Encryption)**.** Let λ be a security parameter. We define the following:

- Let $(n, m, q, \sigma_{\mathsf{LWE}})$ be LWE parameters (which may be functions of λ). Let $\sigma_{\mathsf{pp}}, \sigma_{\mathsf{agg}} > 0$ be Gaussian width parameters.
- Let $\mathcal{P}$ be a family of monotone Boolean functions on up to 2^λ-bit inputs that has an explainable $\{0,1\}$-linear secret sharing scheme $(\mathsf{Share}, \mathsf{Reconst}, \mathsf{ExplainShares})$ (Definition 3.1). Let $\rho_{\mathsf{SS}} = \rho_{\mathsf{SS}}(\lambda, N)$ be a bound on the number of bits of randomness needed for $\mathsf{Share}(P, \cdot)$ for a policy P on N-bit inputs.
- Let λ_{DGS} be the security parameter for the explainable re-randomizer $(\mathsf{Sample}, \mathsf{Explain})$ from Theorem 2.13. Let $\rho = \mathsf{poly}(\lambda_{\mathsf{DGS}}, N, n, m, \log q)$ be the randomness length from Theorem 2.13.
- Let $\Pi_{\mathsf{NIZK}} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.TrapSetup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify}, \mathsf{NIZK.Sim}, \mathsf{NIZK.Extract})$ be a simulation-sound extractable NIZK argument for NP.
  - Let C_{sk} be the Boolean circuit that takes a statement $(\mathsf{id}, \mathbf{B}, \mathbf{t})$, a witness $\mathbf{y}$, and outputs 1 if $\mathbf{t} = \mathbf{B}\mathbf{y}$ and $\mathbf{y} \in \{0,1\}^*$.
  - Let C_{ct} be the Boolean circuit that takes a statement $(\mathbf{B}, \mathbf{c}^\intercal, \beta, \mathsf{ct}_{\mathsf{rest}})$, a witness $(\mathbf{s}, \mathbf{e})$, and outputs 1 if $\mathbf{c}^\intercal = \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}^\intercal$ and $\|\mathbf{e}\| \leq \beta$.

  Note that C_{sk} technically ignores the index id and C_{ct} ignores $\mathsf{ct}_{\mathsf{rest}}$. However, since the NIZK is simulation-extractable, a proof for C_{sk} (resp., C_{ct}) still binds to the index id (resp., $\mathsf{ct}_{\mathsf{rest}}$). This property will be helpful in our security analysis.

- Let $H_{\mathsf{SS}}\colon \{0,1\}^* \to \{0,1\}$ and $H_{\mathsf{virt}}\colon \{0,1\}^\lambda \times \mathbb{N} \to \{0,1\}$ be hash functions (which are modeled as random oracles in the security analysis).

We construct a distributed monotone-policy encryption scheme $\Pi_{\mathsf{MPE}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Encrypt}, \mathsf{GetHint}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm samples

$$\mathsf{crs}_{\mathsf{NIZK}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$$
$$\mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n,\ \mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{pp}}}^{m\times 2m^3}.$$

  If $\|\mathbf{R}\| > \sqrt{m}\sigma_{\mathsf{pp}}$, set $\mathbf{R} = \mathbf{0}^{m\times 2m^3}$. Let

$$\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\mathsf{T} \otimes \mathbf{G} \in \mathbb{Z}_q^{n\times 4m^5} \quad \text{and} \quad \mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B}).$$

  Finally, it outputs $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$.

- $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{id})$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$ and an index $\mathsf{id} \in [2^\lambda]$, the key-generation algorithm samples $\mathbf{y} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5}$ and computes

$$\mathbf{t} = \mathbf{B}\mathbf{y} \in \mathbb{Z}_q^n \quad \text{and} \quad \pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}, \mathbf{B}, \mathbf{t}), \mathbf{y}).$$

  It outputs the public key $\mathsf{pk} = (\mathbf{t}, \pi)$ and the secret key $\mathsf{sk} = \mathbf{y}$.

- $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}, \mathsf{pk})$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$, an index $\mathsf{id} \in [2^\lambda]$, and a public key $\mathsf{pk} = (\mathbf{t}, \pi)$, the validity-checking algorithm computes and outputs $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}, \mathbf{B}, \mathbf{t}), \pi)$.

- $\mathsf{Encrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \mu)$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$, a policy $P \in \mathcal{P}$, a collection of public keys $\mathsf{pk}_j = (\mathbf{t}_j, \pi_j)$ for $j \in [N]$, and a message $\mu \in \{0,1\}$, the encryption algorithm samples

$$\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n,\ \mathbf{e} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{LWE}}}^{4m^5},\ \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}} \xleftarrow{\mathrm{R}} \{0,1\}^\lambda,\ \mathbf{K}_{\mathbf{C}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5\times m},\ \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5}.$$

  If $\|\mathbf{e}\| > \sqrt{m}\cdot\sigma_{\mathsf{LWE}}$, it sets $\mathbf{e} = \mathbf{0}^m$. Next, it computes the public shares

$$(\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N) = \mathsf{Share}\big(P, \mathbf{p}; H_{\mathsf{SS}}(\tau_{\mathsf{SS}}, P, 1)\| \cdots \| H_{\mathsf{SS}}(\tau_{\mathsf{SS}}, P, \rho_{\mathsf{SS}})\big), \tag{3.1}$$

  and for each $j \in [N]$, it computes commitments

$$\mathbf{C}_j = \mathsf{Com}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_j^\mathsf{T} \otimes (\mathbf{t}_j + \hat{\mathbf{p}}_j)) \in \mathbb{Z}_q^{n\times m},$$

  where $\mathbf{u}_j \in \mathbb{Z}^N$ is the j^{th} unit vector. Then, it computes the re-randomization component

$$(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) = \mathsf{Sample}\big(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathrm{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; H_{\mathsf{virt}}(\tau_{\mathsf{virt}}, 1)\| \cdots \| H_{\mathsf{virt}}(\tau_{\mathsf{virt}}, \rho)\big), \tag{3.2}$$

  and sets $\mathbf{c}_1^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}$, $\mathbf{c}_2^\mathsf{T} = \mathbf{s}^\mathsf{T}\left(\mathbf{C}_0 + \sum_{j\in[N]} \mathbf{C}_j\right) + \mathbf{e}^\mathsf{T}\mathbf{K}_{\mathbf{C}}$, and $c_3 = \mathbf{s}^\mathsf{T}\mathbf{p} + \mathbf{e}^\mathsf{T}\mathbf{k}_{\mathbf{p}} + \mu\cdot\lfloor q/2\rfloor$. Finally, it sets $\mathsf{ct}_{\mathsf{rest}} = (\tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_2^\mathsf{T}, c_3)$ and computes the proof

$$\pi_{\mathsf{ct}} \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\mathsf{T}, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \mathsf{ct}_{\mathsf{rest}}), (\mathbf{s}, \mathbf{e})),$$

  and outputs $\mathsf{ct} = (\pi_{\mathsf{ct}}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_1^\mathsf{T}, \mathbf{c}_2^\mathsf{T}, c_3)$.

- $\mathsf{GetHint}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$: On input the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ where $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$, a secret key $\mathsf{sk}_i = \mathbf{y}_i$, and a ciphertext $\mathsf{ct} = (\pi_{\mathsf{ct}}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_1^\intercal, \mathbf{c}_2^\intercal, c_3)$, the hint-generation algorithm first sets $\mathsf{ct}_{\mathsf{rest}} = (\tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_2^\intercal, c_3)$ and then checks that

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \mathsf{ct}_{\mathsf{rest}}), \pi_{\mathsf{ct}}) = 1.$$

  If the check fails, it outputs $\perp$. Otherwise, it samples $e \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{agg}}}$, sets $e = 0$ if $|e| > \sqrt{m} \cdot \sigma_{\mathsf{agg}}$, and outputs $\mathbf{c}_1^\intercal \mathbf{y}_i + e$.

- $\mathsf{Decrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \{(i, \mathsf{ht}_i)\}_{i \in T}, \mathsf{ct})$: On input the parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$, a policy $P \in \mathcal{P}$, a collection of public keys $\mathsf{pk}_j = (\mathbf{t}_j, \pi_j)$ for $j \in [N]$, a list of decryption hints $\mathsf{ht}_i = h_i$ for $i \in T \subseteq [N]$, and a ciphertext $\mathsf{ct} = (\pi_{\mathsf{ct}}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_1^\intercal, \mathbf{c}_2^\intercal, c_3)$, the decryption algorithm first computes the shares $(\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N)$ and re-randomization tuple $(\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N})$ from τ_{SS} and τ_{virt} as in Eqs. (3.1) and (3.2). Then, the algorithm computes the following for $i \in T$:

$$\begin{aligned} \mathbf{v}_i &= \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, N, i) \\ \forall j \in [N] : \mathbf{z}_{j,i} &= \mathsf{Open}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_j^\intercal \otimes (\mathbf{t}_j + \hat{\mathbf{p}}_j), i). \end{aligned}$$

  Then, for $i \in T$, the algorithm computes the shares of the message $\hat{\mu}_i$ as follows:

$$\hat{\mu}_i = \mathbf{c}_1^\intercal \left(\mathbf{z}_{0,i} + \sum_{j \in [N]} \mathbf{z}_{j,i} \right) + \mathbf{c}_2^\intercal \mathbf{v}_i - h_i. \tag{3.3}$$

  Finally, it computes $T^* = \mathsf{Reconst}(P, T)$ and outputs 0 if $-q/4 < \left(c_3 - \sum_{i \in T^*} \hat{\mu}_i\right) < q/4$ and 1 otherwise.

Theorem 3.4 (Completeness)**.** *If Π_{NIZK} is complete, then Construction 3.3 is complete.*

Proof. Take any $\lambda \in \mathbb{N}$ and any $\mathsf{id} \in [2^\lambda]$. Take any $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ in the support of $\mathsf{Setup}(1^\lambda)$ and any $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{id})$. By construction, we can parse $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$ and $\mathsf{pk} = (\mathbf{t}, \pi)$ where $\mathbf{t} = \mathbf{B}\mathbf{y} \in \mathbb{Z}_q^n$, $\mathbf{y} \in \{0,1\}^{4m^5}$, and $\pi \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}, \mathbf{B}, \mathbf{t}), \mathbf{y})$. By inspection $C_{\mathsf{sk}}((\mathsf{id}, \mathbf{B}, \mathbf{t}), \mathbf{y}) = 1$, so completeness of Π_{NIZK} guarantees that $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}, \mathbf{B}, \mathbf{t}), \pi) = 1$. By definition, $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}, \mathsf{pk})$ outputs 1, as desired. □

Theorem 3.5 (Correctness)**.** *Suppose $q > N^2 \cdot O(m^{13} \sigma_{\mathsf{pp}} \sigma_{\mathsf{agg}} \sigma_{\mathsf{LWE}} \log q \log N)$ and the explainable linear secret sharing scheme* $(\mathsf{Share}, \mathsf{Reconst}, \mathsf{ExplainShares})$ *is correct. Then, Construction 3.3 is correct.*

Proof. Take any $\lambda \in \mathbb{N}$, any $N \leq 2^\lambda$, any $P \in \mathcal{P}$ on N-bit inputs, any set $T \subseteq [N]$ where $P(T) = 1$, any tuple of indices $\mathsf{id}_1, \ldots, \mathsf{id}_N \in [2^\lambda]$, any message $\mu \in \{0,1\}$, any pp in the support of $\mathsf{Setup}(1^\lambda)$, any $(\mathsf{pk}_i, \mathsf{sk}_i)$ in the support of $\mathsf{KeyGen}(\mathsf{pp}, \mathsf{id}_i)$ for $i \in T$, and any pk_i for $i \notin T$. Let $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \mu)$ and for each $i \in T$, let $\mathsf{ht}_i \leftarrow \mathsf{GetHint}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{ct})$. Now, consider the output of

$$\mathsf{Decrypt}(\mathsf{pp}, P, (\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \{(i, \mathsf{ht}_i)\}_{i \in T}, \mathsf{ct}).$$

We analyze each step individually:

- Let $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ and $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$. By definition of Setup, we have $\|\mathbf{R}\| \leq \sqrt{m}\sigma_{\mathsf{pp}}$.
- For $i \in T$, we have $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ where $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i$ for some $\mathbf{y}_i \in \{0,1\}^{4m^5}$. For $i \notin T$, we have $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ for $\mathbf{t}_i \in \mathbb{Z}_q^n$.

- By definition, Encrypt samples $\mathbf{s}, \mathbf{e}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{K}_{\mathsf{C}}, \mathbf{k}_{\mathbf{p}}$ such that $\|\mathbf{e}\| \le \sqrt{m}\sigma_{\mathsf{LWE}}$ and $\|\mathbf{K}_{\mathsf{C}}\|, \left\|\mathbf{k}_{\mathbf{p}}\right\| \le 1$. The ciphertext has the form

$$\begin{aligned}\mathsf{ct} &= \left(\pi_{\mathsf{ct}}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{s}^{\mathsf{T}}\mathbf{B} + \mathbf{e}^{\mathsf{T}}\ ,\ \mathbf{s}^{\mathsf{T}}\left(\mathbf{C}_0 + \sum_{j\in[N]} \mathbf{C}_j\right) + \mathbf{e}^{\mathsf{T}}\mathbf{K}_{\mathsf{C}}\ ,\ \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + \mu\cdot\lfloor q/2\rfloor\right)\\ &= (\pi_{\mathsf{ct}}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_1^{\mathsf{T}}, \mathbf{c}_2^{\mathsf{T}}, c_3),\end{aligned}$$

  where we have

$$\begin{aligned}(\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N) &= \mathsf{Share}(P, \mathbf{p}; H_{\mathsf{SS}}(\tau_{\mathsf{SS}}, P, 1)\| \cdots \|H_{\mathsf{SS}}(\tau_{\mathsf{SS}}, P, \rho_{\mathsf{SS}}))\\ \mathbf{C}_j &= \mathsf{Com}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_j^{\mathsf{T}} \otimes (\mathbf{t}_j + \hat{\mathbf{p}}_j))\\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; H_{\mathsf{virt}}(\tau_{\mathsf{virt}}, 1)\| \cdots \|H_{\mathsf{virt}}(\tau_{\mathsf{virt}}, \rho)).\end{aligned}$$

  Furthermore, by construction we have $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^{\mathsf{T}}, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \mathsf{ct}_{\mathsf{rest}}), \pi_{\mathsf{ct}}) = 1$.

- By definition of GetHint, for all $i \in T$,

$$\mathsf{ht}_i = h_i = \mathbf{c}_1^{\mathsf{T}}\mathbf{y}_i + e_i = \mathbf{s}^{\mathsf{T}}\mathbf{B}\mathbf{y}_i + \mathbf{e}^{\mathsf{T}}\mathbf{y}_i + e_i = \mathbf{s}^{\mathsf{T}}\mathbf{t}_i + \mathbf{e}^{\mathsf{T}}\mathbf{y}_i + e_i,$$

  for some $|e_i| < \sqrt{m}\cdot\sigma_{\mathsf{agg}}$.

- For each $i \in T$, the Decrypt algorithm computes

$$\begin{aligned}\mathbf{v}_i &= \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, N, i)\\ \mathbf{z}_{j,i} &= \mathsf{Open}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_j^{\mathsf{T}} \otimes (\mathbf{t}_j + \hat{\mathbf{p}}_j), i)\end{aligned}$$

  for all $j \in [N]$. By Theorem 2.12, we have $\mathbf{B}\mathbf{z}_{j,i} = -\mathbf{C}_j\mathbf{v}_i$ for $i \ne j$ and $\mathbf{B}\mathbf{z}_{i,i} = \mathbf{t}_i + \hat{\mathbf{p}}_i - \mathbf{C}_i\mathbf{v}_i$. By Theorem 2.13 we also have $\mathbf{B}\mathbf{z}_{0,i} = -\mathbf{C}_0\mathbf{v}_i$. Thus for $i \in T$,

$$\begin{aligned}\hat{\mu}_i &= \mathbf{c}_1^{\mathsf{T}}\left(\mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{z}_{j,i}\right) + \mathbf{c}_2^{\mathsf{T}}\mathbf{v}_i - h_i\\ &= \mathbf{s}^{\mathsf{T}}(\mathbf{t}_i + \hat{\mathbf{p}}_i) - \mathbf{s}^{\mathsf{T}}\left(\mathbf{C}_0 + \sum_{j\in[N]} \mathbf{C}_j\right)\mathbf{v}_i + \mathbf{s}^{\mathsf{T}}\left(\mathbf{C}_0 + \sum_{j\in[N]} \mathbf{C}_j\right)\mathbf{v}_i + \mathbf{e}^{\mathsf{T}}\left(\mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{z}_{j,i} + \mathbf{K}_{\mathsf{C}}\mathbf{v}_i\right) - h_i\\ &= \mathbf{s}^{\mathsf{T}}(\mathbf{t}_i + \hat{\mathbf{p}}_i) + \mathbf{e}^{\mathsf{T}}\left(\mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{z}_{j,i} + \mathbf{K}_{\mathsf{C}}\mathbf{v}_i\right) - (\mathbf{s}^{\mathsf{T}}\mathbf{t}_i + \mathbf{e}^{\mathsf{T}}\mathbf{y}_i + e_i)\\ &= \mathbf{s}^{\mathsf{T}}\hat{\mathbf{p}}_i + \mathbf{e}^{\mathsf{T}}\left(\mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{z}_{j,i} + \mathbf{K}_{\mathsf{C}}\mathbf{v}_i - \mathbf{y}_i\right) - e_i.\end{aligned}$$

  Since the algorithm computes $T^* = \mathsf{Reconst}(P, T)$, and $P(T) = 1$, by secret sharing correctness, we have $\sum_{i\in T^*} \hat{\mathbf{p}}_i = \mathbf{p}$. Let $\hat{e}_i = \mathbf{e}^{\mathsf{T}}(\mathbf{z}_{0,i} + \sum_{j\in[N]} \mathbf{z}_{j,i} + \mathbf{K}_{\mathsf{C}}\mathbf{v}_i - \mathbf{y}_i) - e_i$. Then,

$$c_3 - \textstyle\sum_{i\in T^*} \hat{\mu}_i = \mathbf{s}^{\mathsf{T}}\mathbf{p} + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} + \mu\cdot\lfloor q/2\rfloor - (\mathbf{s}^{\mathsf{T}}\mathbf{p} + \textstyle\sum_{i\in T^*} \hat{e}_i) = \mu\cdot\lfloor q/2\rfloor + \mathbf{e}^{\mathsf{T}}\mathbf{k}_{\mathbf{p}} - \textstyle\sum_{i\in T^*} \hat{e}_i.$$

- Correctness holds as long as $\left|\mathbf{e}^\intercal \mathbf{k}_{\mathbf{p}} - \sum_{i \in T^*} \hat{e}_i\right| < q/4$, so it remains to bound $\left|\mathbf{e}^\intercal \mathbf{k}_{\mathbf{p}} - \sum_{i \in T^*} \hat{e}_i\right|$. First, note that $\left|\mathbf{e}^\intercal \mathbf{k}_{\mathbf{p}}\right| \le 4m^6 \sigma_{\mathsf{LWE}}$. By Theorems 2.12 and 2.13, we have the following for $i \in T$

$$\begin{aligned}
\|\mathbf{v}_i\| &\le O(\|\mathbf{R}\| \cdot m^4 \log q) = O(\sigma_{\mathsf{pp}} \cdot m^{9/2} \log q) \\
\|\mathbf{z}_{j,i}\| &\le O(\|\mathbf{R}\| \cdot m^7 \log q \log N) = O(\sigma_{\mathsf{pp}} \cdot m^{15/2} \log q \log N) \\
\|\mathbf{z}_{0,i}\| &\le \sqrt{m}\sigma_{\mathsf{agg}}.
\end{aligned}$$

  A (very) loose upper bound of $|\hat{e}_i|$ is then $|\hat{e}_i| \le N \cdot O(\sigma_{\mathsf{pp}} \sigma_{\mathsf{agg}} \sigma_{\mathsf{LWE}} \cdot m^{13} \log q \log N)$ and thus

$$\left|\mathbf{e}^\intercal \mathbf{k}_{\mathbf{p}} - \textstyle\sum_{i \in T^*} \hat{e}_i\right| \le N^2 \cdot O(\sigma_{\mathsf{pp}} \sigma_{\mathsf{agg}} \sigma_{\mathsf{LWE}} \cdot m^{13} \log q \log N).$$

By setting $q > N^2 \cdot O(\sigma_{\mathsf{pp}} \sigma_{\mathsf{agg}} \sigma_{\mathsf{LWE}} \cdot m^{13} \log q \log N)$, correctness follows. □

Theorem 3.6 (Semi-Policy-and-Query-Selective Security)**.** *Suppose Π_{NIZK} is complete, zero-knowledge, and simulation-extractable, and that the $(2m^2, \sigma_{\mathsf{pp}})$-decomposed LWE assumption holds with lattice parameters $(n, m, q, \sigma_{\mathsf{LWE}})$. Suppose also that $n \ge \lambda$, $m \ge 2n \log q$, $q > 2$ is prime, $\sigma_{\mathsf{pp}}, \sigma_{\mathsf{LWE}} > \log m$, $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot O(\sigma_{\mathsf{pp}} \sigma_{\mathsf{LWE}} \cdot m^{13} N^3)$, and $\lambda_{\mathsf{DGS}} > \log \sigma_{\mathsf{agg}}$. Then, Construction 3.3 satisfies semi-policy-and-query-selective security in the static security model (Definition 2.16).*

Proof. Take any efficient adversary $\mathcal{A}$ for the semi-policy-and-query-selective security game in the static model (Definition 2.16), 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 precision parameter $\kappa = \kappa(\lambda)$ (used for the explainable sampling procedure). We omit the index κ when the hybrid definition is independent of the choice of κ.

- $\mathsf{Hyb}_0^{(b)}$: This is the semi-policy-and-query-selective security game with challenge bit $b \in \{0, 1\}$:
  - **Setup phase:** On input the security parameter 1^λ, the challenger samples

$$\begin{aligned}
\mathsf{crs}_{\mathsf{NIZK}} &\leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\
\mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n,\ \mathbf{W} &\xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times 2m^3},\ \mathbf{R} \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{pp}}}^{m \times 2m^3}.
\end{aligned}$$

    If $\|\mathbf{R}\| > \sqrt{m}\sigma_{\mathsf{pp}}$, the challenger sets $\mathbf{R} = \mathbf{0}^{m \times 2m^3}$ instead. Let

$$\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times 4m^5}$$

    and $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$. The challenger gives the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ to $\mathcal{A}$. Additionally, whenever algorithm $\mathcal{A}$ queries H_{SS} on a tuple $(\tau_{\mathsf{SS}}, P, i) \in \{0, 1\}^*$, the challenger replies with a string $\gamma_{\mathsf{SS}} \xleftarrow{\mathrm{R}} \{0, 1\}$. Whenever algorithm $\mathcal{A}$ queries H_{virt} on a pair $(\tau_{\mathsf{virt}}, i) \in \{0, 1\}^\lambda \times \mathbb{N}$, the challenger replies with a string $\gamma_{\mathsf{virt}} \xleftarrow{\mathrm{R}} \{0, 1\}$.

- **Key-generation queries:** Algorithm $\mathcal{A}$ outputs a policy P, a set of indices $(\mathsf{id}_1, \ldots, \mathsf{id}_N)$, and a set of honest slots $\mathcal{H} \subseteq [N]$. If $P([N] \setminus \mathcal{H}) = 1$, the challenger outputs 0. Otherwise, for $i \in \mathcal{H}$, the challenger samples $\mathbf{y}_i \xleftarrow{R} \{0,1\}^{4m^5}$, sets $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i \in \mathbb{Z}_q^n$, and computes the NIZK proof $\pi_i \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i), \mathbf{y}_i)$. The challenger replies to the adversary with $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ for all $i \in \mathcal{H}$.
- **Hint queries:** When algorithm $\mathcal{A}$ makes a hint query $(\hat{\mathsf{ct}}, i)$ for $i \in \mathcal{H}$, the challenger parses $\hat{\mathsf{ct}} = (\hat{\pi}_{\mathsf{ct}}, \hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_1^\mathsf{T}, \hat{\mathbf{c}}_2^\mathsf{T}, \hat{c}_3)$ and defines $\hat{\mathsf{ct}}_{\mathsf{rest}} = (\hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_2^\mathsf{T}, \hat{c}_3)$. Then it checks that

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \hat{\mathbf{c}}_1^\mathsf{T}, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}}) = 1,$$

  and replies with $\perp$ if the check fails. If the check passes, the challenger samples $e \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}$, sets $e = 0$ if $|e| > \sqrt{m} \cdot \sigma_{\mathsf{agg}}$, and replies with $\hat{\mathbf{c}}_1^\mathsf{T}\mathbf{y}_i + e$.
- **Challenge phase:** In the challenge phase, algorithm $\mathcal{A}$ starts by outputting the public keys $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ for each $i \in [N] \setminus \mathcal{H}$ to complete the challenge tuple. If $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}_i, \mathsf{pk}_i) = 0$ for any $i \in [N] \setminus \mathcal{H}$, the challenger outputs 0. Otherwise, the challenger constructs the challenge ciphertext ct^* by first sampling

$$\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n,\ \mathbf{e} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{LWE}}}^{4m^5},\ \tau_{\mathsf{SS}}^*, \tau_{\mathsf{virt}}^* \xleftarrow{R} \{0,1\}^\lambda,\ \mathbf{K}_{\mathbf{C}} \xleftarrow{R} \{0,1\}^{4m^5 \times m},\ \mathbf{k}_{\mathbf{p}} \xleftarrow{R} \{0,1\}^{4m^5}.$$

  If $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$, the challenger sets $\mathbf{e} = \mathbf{0}^m$ instead. Next, the challenger computes the public shares of $\mathbf{p}$ as

$$\begin{aligned} \gamma_{\mathsf{SS}}^* &= H_{\mathsf{SS}}(\tau_{\mathsf{SS}}^*, P, 1) \| \cdots \| H_{\mathsf{SS}}(\tau_{\mathsf{SS}}^*, P, \rho_{\mathsf{SS}}) \\ (\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N) &= \mathsf{Share}(P, \mathbf{p}; \gamma_{\mathsf{SS}}^*). \end{aligned} \tag{3.4}$$

  Next, for each $i \in [N]$, the challenger computes commitments

$$\mathbf{C}_i = \mathsf{Com}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^\mathsf{T} \otimes (\mathbf{t}_i + \hat{\mathbf{p}}_i)) \in \mathbb{Z}_q^{n \times m},$$

  where $\mathbf{u}_i \in \mathbb{Z}^N$ is the i^{th} unit vector. Then, the challenger computes the re-randomization components as

$$\begin{aligned} \gamma_{\mathsf{virt}}^* &= H_{\mathsf{virt}}(\tau_{\mathsf{virt}}^*, 1) \| \cdots \| H_{\mathsf{virt}}(\tau_{\mathsf{virt}}^*, \rho) \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; \gamma_{\mathsf{virt}}^*), \end{aligned} \tag{3.5}$$

  and sets $\mathbf{c}_1^\mathsf{T} = \mathbf{s}^\mathsf{T}\mathbf{B} + \mathbf{e}^\mathsf{T}$, $\mathbf{c}_2^\mathsf{T} = \mathbf{s}^\mathsf{T}\left(\mathbf{C}_0 + \sum_{j \in [N]} \mathbf{C}_j\right) + \mathbf{e}^\mathsf{T}\mathbf{K}_{\mathbf{C}}$, and $c_3 = \mathbf{s}^\mathsf{T}\mathbf{p} + \mathbf{e}^\mathsf{T}\mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor$. Finally, the challenger sets $\mathsf{ct}_{\mathsf{rest}}^* = (\tau_{\mathsf{SS}}^*, \tau_{\mathsf{virt}}^*, \mathbf{c}_2^\mathsf{T}, c_3)$ and computes the proof

$$\pi_{\mathsf{ct}}^* \leftarrow \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\mathsf{T}, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \mathsf{ct}_{\mathsf{rest}}^*), (\mathbf{s}, \mathbf{e})),$$

  and gives $\mathsf{ct}^* = (\pi_{\mathsf{ct}}^*, \tau_{\mathsf{SS}}^*, \tau_{\mathsf{virt}}^*, \mathbf{c}_1^\mathsf{T}, \mathbf{c}_2^\mathsf{T}, c_3)$ to $\mathcal{A}$.
- **Post-challenge hint queries:** Algorithm $\mathcal{A}$ can continue to make hint queries that do not include ct^*. These are answered exactly like the pre-challenge hint queries.
- **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.

- $\mathsf{Hyb}_1^{(b)}$: Same as $\mathsf{Hyb}_0^{(b)}$ except for the following modifications:

– The challenger samples the seeds $\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}} \xleftarrow{R} \{0,1\}^\lambda$ for the challenge ciphertext at the start of the experiment.

– The challenger samples the associated strings $\gamma^*_{\mathsf{SS}} \xleftarrow{R} \{0,1\}^{\rho_{\mathsf{SS}}}, \gamma^*_{\mathsf{virt}} \xleftarrow{R} \{0,1\}^{\rho}$ after P (and thus N) is declared by the adversary.

Next, the challenger answers queries to the random oracle H_{SS} and H_{virt} using the following modified procedure:

– If the adversary queries H_{SS} on a triple $(\tau^*_{\mathsf{SS}}, \cdot, \cdot)$ or queries H_{virt} on a pair $(\tau^*_{\mathsf{virt}}, \cdot)$ before the challenge phase, the challenger halts the experiment with output 0.

– If the adversary queries $(\tau^*_{\mathsf{SS}}, P, i)$ for $i \in [\rho_{\mathsf{SS}}]$ or $(\tau^*_{\mathsf{virt}}, i)$ for $i \in [\rho]$ after the challenge phase, the challenger replies with the i^{th} bit of γ^*_{SS} or γ^*_{virt}, respectively.

Finally, the challenger also halts with output 0 if any pre-challenge hint query includes τ^*_{SS} as part of the ciphertext.

- $\mathsf{Hyb}_2^{(b)}$: Same as $\mathsf{Hyb}_1^{(b)}$ except the challenger replaces the NIZK proofs in the honest users' keys as well as for the challenge ciphertext with simulated NIZK proofs:

  – During setup, the challenger samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$.

  – During key-generation, the challenger computes $\pi_i \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i))$ for $i \in \mathcal{H}$.

  – When constructing the challenge ciphertext, after computing $\mathbf{c}_2$ and c_3, the challenger sets $\mathsf{ct}^*_{\mathsf{rest}} = (\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}}, \mathbf{c}_2^\intercal, c_3)$. Then, it computes

  $$\pi^*_{\mathsf{ct}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}})).$$

- $\mathsf{Hyb}_3^{(b)}$: Same as $\mathsf{Hyb}_2^{(b)}$ except the challenger extracts the secret keys for adversarially-generated public keys as well as the encryption randomness for hint queries:

  – When the adversary makes a hint query $(\hat{\mathsf{ct}}, i)$ for an index $i \in \mathcal{H}$, the challenger parses $\hat{\mathsf{ct}} = (\hat{\pi}_{\mathsf{ct}}, \hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_1^\intercal, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$ and sets $\hat{\mathsf{ct}}_{\mathsf{rest}} = (\hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$. Then, it computes

  $$(\mathbf{s}, \mathbf{e}) = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}}).$$

  If $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$ or $\hat{\mathbf{c}}_1^\intercal \neq \mathbf{s}^\intercal \mathbf{B} + \mathbf{e}^\intercal$, the challenger outputs 0.

  – In the challenge phase, after confirming that all public keys are valid, the challenger computes

  $$\mathbf{y}_i = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i), \pi_i)$$

  for each $i \in [N] \setminus \mathcal{H}$. If $\mathbf{y}_i \notin \{0,1\}^{4m^5}$ or $\mathbf{B}\mathbf{y}_i \neq \mathbf{t}_i$ for any $i \in [N] \setminus \mathcal{H}$, the challenger outputs 0. In particular, the challenger now knows a vector $\mathbf{y}_i \in \{0,1\}^{4m^5}$ such that $\mathbf{B}\mathbf{y}_i = \mathbf{t}_i$ for all $i \in [N]$ prior to constructing the challenge ciphertext.

- $\mathsf{Hyb}_4^{(b)}$: Same as $\mathsf{Hyb}_3^{(b)}$ except the challenger now simulates the hints using the extracted ciphertext validity witnesses. In particular, for a hint query $(\hat{\mathsf{ct}}, i)$, the challenger first extracts $(\mathbf{s}, \mathbf{e})$ as in $\mathsf{Hyb}_3^{(b)}$. Then it samples $e \leftarrow D_{\mathbb{Z}, \sigma_{\mathsf{agg}}}$, and replies with $\mathbf{s}^\intercal \mathbf{t}_i + e$. At this point, the hint queries no longer depend on the vector $\mathbf{y}_i$.

- $\mathsf{Hyb}_5^{(b)}$: Same as $\mathsf{Hyb}_4^{(b)}$ except the challenger now samples $\hat{\mathbf{p}}_i \xleftarrow{R} \mathbb{Z}_q^n$ for $i \in [N] \setminus \mathcal{H}$ and computes $\{\hat{\mathbf{p}}_i\}_{i\in\mathcal{H}} \leftarrow \mathsf{Complete}(P, \mathbf{p}, \{\hat{\mathbf{p}}_i\}_{i\in[N]\setminus\mathcal{H}})$ before generating the honest keys. The challenger also computes
$$\gamma^*_{\mathsf{SS}} \leftarrow \mathsf{ExplainShares}(P, \mathbf{p}, (\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N)).$$
As in the previous experiments, it holds that $H_{\mathsf{SS}}(\tau^*_{\mathsf{SS}}, P, 1) \| \cdots \| H_{\mathsf{SS}}(\tau^*_{\mathsf{SS}}, P, \rho_{\mathsf{SS}}) = \gamma^*_{\mathsf{SS}}$.

- $\mathsf{Hyb}_6^{(b)}$: Same as $\mathsf{Hyb}_5^{(b)}$ except the challenger samples $\mathbf{k}_{\mathbf{p}} \xleftarrow{R} \{0,1\}^{4m^5}$ at setup time and sets $\mathbf{p} = \mathbf{B}\mathbf{k}_{\mathbf{p}}$. When generating honest keys, the challenger samples $\mathbf{d}_i \xleftarrow{R} \mathbb{Z}_q^n$ and sets $\mathbf{t}_i = \mathbf{d}_i - \hat{\mathbf{p}}_i$ for $i \in \mathcal{H}$.

- $\mathsf{Hyb}_{7,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_6^{(b)}$ except the adversary now derives the randomness γ^*_{virt} for the "virtual user" using the Explain algorithm. Specifically, the challenger samples
$$\begin{aligned} \gamma_{\mathsf{virt}} &\xleftarrow{R} \{0,1\}^{\rho} \\ (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}) &= \mathsf{Sample}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^N, \sigma_{\mathsf{agg}}; \gamma_{\mathsf{virt}}) \\ \gamma^*_{\mathsf{virt}} &\leftarrow \mathsf{Explain}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathsf{DGS}}}, 1^{\kappa}, (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}), \sigma_{\mathsf{agg}}). \end{aligned}$$
As in $\mathsf{Hyb}_1^{(b)}$, the challenger aborts if the adversary queries H_{virt} on τ^*_{virt} before the challenge phase. If the adversary queries $H_{\mathsf{virt}}(\tau^*_{\mathsf{virt}}, i)$ after the challenge phase, the challenger still replies with the i^{th} bit of γ^*_{virt}.

- $\mathsf{Hyb}_{8,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{7,\kappa}^{(b)}$ except the challenger now samples
$$\mathbf{C}_0 \xleftarrow{R} \mathbb{Z}_q^{n\times m} \quad \text{and} \quad \forall i \in [N] : \mathbf{z}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}(-\mathbf{C}_0\mathbf{v}_i)$$
where $\mathbf{v}_i = \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, N, i)$.

- $\mathsf{Hyb}_{9,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{8,\kappa}^{(b)}$ except the challenger now sets $\mathbf{C}_0 = \mathbf{B}\mathbf{K}_{\mathbf{C}} - \sum_{i\in[N]} \mathbf{C}_i$. Additionally, the challenger sets $\mathbf{z}_{i,j} = \mathsf{Open}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^{\intercal} \otimes (\mathbf{t}_i + \hat{\mathbf{p}}_i), j)$ and $\mathbf{v}_i = \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, N, i)$ for $i, j \in [N]$. For $i \in \mathcal{H}$, the challenger samples
$$\mathbf{z}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}\left(\mathbf{d}_i - \mathbf{B}\left(\mathbf{K}_{\mathbf{C}}\mathbf{v}_i + \textstyle\sum_{j\in[N]} \mathbf{z}_{j,i}\right)\right).$$
For $i \in [N] \setminus \mathcal{H}$, the challenger samples
$$\mathbf{z}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}\left(\hat{\mathbf{p}}_i - \mathbf{B}\left(\mathbf{K}_{\mathbf{C}}\mathbf{v}_i - \mathbf{y}_i + \textstyle\sum_{j\in[N]} \mathbf{z}_{j,i}\right)\right),$$
where $\mathbf{y}_i$ is the key extracted as in $\mathsf{Hyb}_3^{(b)}$.

- $\mathsf{Hyb}_{10,\kappa}^{(b)}$: Same as $\mathsf{Hyb}_{9,\kappa}^{(b)}$ except the challenger changes the distribution of $\mathbf{z}_{0,i}$. In particular, for $i \in \mathcal{H}$, the challenger samples $\hat{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{d}_i)$ and sets
$$\mathbf{z}_{0,i} = \hat{\mathbf{z}}_{0,i} - \mathbf{K}_{\mathbf{C}}\mathbf{v}_i - \sum_{j\in[N]} \mathbf{z}_{j,i}.$$
For $i \in [N] \setminus \mathcal{H}$, the challenger samples $\hat{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}(\hat{\mathbf{p}}_i)$ and sets
$$\mathbf{z}_{0,i} = \hat{\mathbf{z}}_{0,i} - \mathbf{K}_{\mathbf{C}}\mathbf{v}_i + \mathbf{y}_i - \sum_{j\in[N]} \mathbf{z}_{j,i}.$$

- $\mathsf{Hyb}^{(b)}_{11,\kappa}$: Same as $\mathsf{Hyb}^{(b)}_{10,\kappa}$ except for $i \in [N]$, the challenger now samples $\hat{\mathbf{z}}_{0,i} \leftarrow D^{4m^5}_{\mathbb{Z},\sigma_{\mathsf{agg}}}$. For $i \in \mathcal{H}$, the challenger sets $\mathbf{d}_i = \mathbf{B}\hat{\mathbf{z}}_{0,i}$. For $i \in [N] \setminus \mathcal{H}$, the challenger sets $\hat{\mathbf{p}}_i = \mathbf{B}\hat{\mathbf{z}}_{0,i}$. When constructing the challenge ciphertext, the challenger uses these values of $\hat{\mathbf{z}}_{0,i}$ instead of sampling them.
- $\mathsf{Hyb}^{(b)}_{12,\kappa}$: Same as $\mathsf{Hyb}^{(b)}_{11,\kappa}$ except in the setup phase, the challenger no longer checks if $\|\mathbf{R}\| > \sqrt{m}\sigma_{\mathsf{pp}}$, and when constructing the challenge ciphertext ct^*, the challenger no longer checks if $\|\mathbf{e}\| > \sqrt{m}\sigma_{\mathsf{LWE}}$.
- $\mathsf{Hyb}^{(b)}_{13,\kappa}$: Same as $\mathsf{Hyb}^{(b)}_{12,\kappa}$ except the challenger samples the component $\mathbf{c}_1 \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{4m^5}$, then computes $\mathbf{c}_2^\intercal = \mathbf{c}_1^\intercal \mathbf{K}_{\mathbf{C}}$ and $c_3 = \mathbf{c}_1^\intercal \mathbf{k}_{\mathbf{p}} + b \cdot \lfloor q/2 \rfloor$.
- $\mathsf{Hyb}^{(b)}_{14,\kappa}$: Same as $\mathsf{Hyb}^{(b)}_{13,\kappa}$ except the challenger samples $\mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $c_3 \xleftarrow{\mathrm{R}} \mathbb{Z}_q$.

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

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

Proof. Let Q_{ro} be a bound on the total number of random oracle queries algorithm $\mathcal{A}$ makes. By a union bound, with probability at most $2Q_{\mathsf{ro}}/2^\lambda$ over the choice of τ^*_{SS} and τ^*_{virt}, the adversary queries the random oracle on at least one seed prior to the challenge phase. Conditioned on neither seed being queried prior to the challenge phase, the distributions of γ^*_{SS} and γ^*_{virt} are uniform random, which would make $\mathsf{Hyb}^{(b)}_1$ identical to $\mathsf{Hyb}^{(b)}_0$. Similarly, if Q_{ht} bounds the number of hint queries, the probability of the challenger aborting after seeing any hint query is at most $Q_{\mathsf{ht}}/2^\lambda$. The adversary is efficient so $(2Q_{\mathsf{ro}} + Q_{\mathsf{ht}})/2^\lambda$ is negligible and the lemma follows. □

Lemma 3.8. *Suppose* Π_{NIZK} *satisfies zero-knowledge. Then, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $b \in \{0,1\}$ *and all* $\lambda \in \mathbb{N}$,

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

Proof. Suppose $|\Pr[\mathsf{Hyb}^{(b)}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_2(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks zero-knowledge of Π_{NIZK}:

- **Setup phase:** At the beginning of the game, algorithm $\mathcal{B}$ gets the common reference string $\mathsf{crs}_{\mathsf{NIZK}}$ from the zero-knowledge challenger and starts running $\mathcal{A}(1^\lambda)$. Algorithm $\mathcal{B}$ samples $\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}} \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$, $\mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times 2m^3}$, and $\mathbf{R} \leftarrow D^{m \times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}}$. If $\|\mathbf{R}\| > \sqrt{m}\sigma_{\mathsf{pp}}$, algorithm $\mathcal{B}$ sets $\mathbf{R} = \mathbf{0}^{m \times 2m^3}$ instead. Let $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times 4m^5}$ and $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$. Algorithm $\mathcal{B}$ gives the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ to $\mathcal{A}$. Additionally, whenever algorithm $\mathcal{A}$ queries H_{SS} on a tuple $(\tau_{\mathsf{SS}}, P, i) \in \{0,1\}^*$, algorithm $\mathcal{B}$ replies with a string $\gamma_{\mathsf{SS}} \xleftarrow{\mathrm{R}} \{0,1\}$. Whenever algorithm $\mathcal{A}$ queries H_{virt} on a pair $(\tau_{\mathsf{virt}}, i) \in \{0,1\}^\lambda \times \mathbb{N}$, algorithm $\mathcal{B}$ replies with a string $\gamma_{\mathsf{virt}} \xleftarrow{\mathrm{R}} \{0,1\}$. If either τ^*_{SS} or τ^*_{virt} are queried, algorithm $\mathcal{B}$ aborts.
- **Key-generation queries:** When algorithm $\mathcal{A}$ outputs a policy P, a set of indices $(\mathsf{id}_1, \ldots, \mathsf{id}_N)$, and a set of honest slots $\mathcal{H} \subseteq [N]$, algorithm $\mathcal{B}$ outputs 0 if $P([N] \setminus \mathcal{H}) = 1$. Otherwise, for $i \in \mathcal{H}$ algorithm $\mathcal{B}$ samples $\mathbf{y}_i \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5}$, sets $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i \in \mathbb{Z}_q^n$, and submits $(C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i), \mathbf{y}_i)$ to the

zero-knowledge challenger to get π_i. Algorithm $\mathcal{B}$ also samples $\gamma^*_{\mathsf{virt}} \xleftarrow{R} \{0,1\}^\rho$ and $\gamma^*_{\mathsf{SS}} \xleftarrow{R} \{0,1\}^{\rho_{\mathsf{SS}}}$ and uses them to answer post-challenge random oracle queries as in $\mathsf{Hyb}_1^{(b)}$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with $\{\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)\}_{i \in \mathcal{H}}$.

- **Hint queries:** Algorithm $\mathcal{B}$ answers hint queries exactly as in $\mathsf{Hyb}_1^{(b)}$.
- **Challenge phase:** Algorithm $\mathcal{B}$ executes the challenge phase exactly as in $\mathsf{Hyb}_1^{(b)}$ except it submits $(C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}}), (\mathbf{s}, \mathbf{e}))$ to the zero-knowledge challenger to get the π^*_{ct} component of the challenge ciphertext.
- **Post-challenge hint queries:** Algorithm $\mathcal{B}$ answers post-challenge hint queries exactly as in $\mathsf{Hyb}_1^{(b)}$.
- **Output phase:** At the end of the game, if algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, $\mathcal{B}$ also outputs b'.

By construction, for $i \in \mathcal{H}$ we have $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i$ and $\mathbf{y} \in \{0,1\}^{4m^5}$, so it holds that $C_{\mathsf{sk}}((\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i), \mathbf{y}_i) = 1$ for all $i \in \mathcal{H}$. Similarly, $C_{\mathsf{ct}}((\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}}), (\mathbf{s}, \mathbf{e})) = 1$. If the zero-knowledge challenger sampled the CRS and answered proof queries honestly, then algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_1^{(b)}$. If the zero-knowledge challenger sampled the CRS with a trapdoor and answered proof queries with simulated proofs, then algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}_2^{(b)}$. Thus, algorithm $\mathcal{B}$ breaks zero-knowledge with the same advantage δ. □

Lemma 3.9. *Suppose* Π_{NIZK} *satisfies simulation-sound extractability. 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. Suppose $|\Pr[\mathsf{Hyb}_2^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3^{(b)}(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks simulation-sound extractability of Π_{NIZK}:

- **Setup phase:** At the beginning of the game, algorithm $\mathcal{B}$ gets the common reference string $\mathsf{crs}_{\mathsf{NIZK}}$ from the simulation-sound extractability challenger and starts running $\mathcal{A}(1^\lambda)$. Algorithm $\mathcal{B}$ samples $\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}} \xleftarrow{R} \{0,1\}^\lambda$, $\mathbf{p} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n \times 2m^3}$, and $\mathbf{R} \leftarrow D^{m \times 2m^3}_{\mathbb{Z}, \sigma_{\mathsf{pp}}}$. If $\|\mathbf{R}\| > \sqrt{m}\sigma_{\mathsf{pp}}$, algorithm $\mathcal{B}$ sets $\mathbf{R} = \mathbf{0}^{m \times 2m^3}$ instead. Let $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times 4m^5}$ and $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$. Algorithm $\mathcal{B}$ gives the public parameters $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ to $\mathcal{A}$. Additionally, whenever algorithm $\mathcal{A}$ queries H_{SS} on a tuple $(\tau_{\mathsf{SS}}, P, i) \in \{0,1\}^*$, algorithm $\mathcal{B}$ replies with a string $\gamma_{\mathsf{SS}} \xleftarrow{R} \{0,1\}$. Whenever algorithm $\mathcal{A}$ queries H_{virt} on a pair $(\tau_{\mathsf{virt}}, i) \in \{0,1\}^\lambda \times \mathbb{N}$, algorithm $\mathcal{B}$ replies with a string $\gamma_{\mathsf{virt}} \xleftarrow{R} \{0,1\}$. If either τ^*_{SS} or τ^*_{virt} are queried, algorithm $\mathcal{B}$ aborts.
- **Key-generation queries:** When algorithm $\mathcal{A}$ outputs a policy P, a set of indices $(\mathsf{id}_1, \ldots, \mathsf{id}_N)$, and a set of honest slots $\mathcal{H} \subseteq [N]$, algorithm $\mathcal{B}$ outputs 0 if $P([N] \setminus \mathcal{H}) = 1$. Otherwise, for $i \in \mathcal{H}$ algorithm $\mathcal{B}$ samples $\mathbf{y}_i \xleftarrow{R} \{0,1\}^{4m^5}$, sets $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i \in \mathbb{Z}_q^n$, and submits $(C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i))$ to the simulation-sound extractability challenger to get π_i. Algorithm $\mathcal{B}$ also samples $\gamma^*_{\mathsf{virt}} \xleftarrow{R} \{0,1\}^\rho$ and $\gamma^*_{\mathsf{SS}} \xleftarrow{R} \{0,1\}^{\rho_{\mathsf{SS}}}$ and uses them to answer post-challenge random oracle queries as in $\mathsf{Hyb}_2^{(b)}$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with $\{\mathsf{pk}_i\}_{i \in \mathcal{H}}$ where $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$.
- **Hint queries:** Algorithm $\mathcal{B}$ answers hint queries as in $\mathsf{Hyb}_2^{(b)}$. Notably, it aborts if any pre-challenge hint query includes τ^*_{SS} in the ciphertext.

- **Challenge phase:** When algorithm $\mathcal{A}$ outputs the public keys $\{\mathsf{pk}_i\}_{i\in[N]\setminus\mathcal{H}}$ where $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$, algorithm $\mathcal{B}$ executes the challenge phase exactly as in $\mathsf{Hyb}_2^{(b)}$ except it submits the tuple

$$(C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}}))$$

  to the simulation-sound extractability challenger to get the π^*_{ct} component of the challenge ciphertext. As defined in $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$, the tuple $\mathsf{ct}^*_{\mathsf{rest}} = (\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}}, \mathbf{c}_2^\intercal, c_3)$ contains the other components of the challenge ciphertext.

- **Post-challenge hint queries:** Algorithm $\mathcal{B}$ answers post-challenge hint queries as in $\mathsf{Hyb}_2^{(b)}$. Note that the post-challenge hint queries cannot be on the challenge ciphertext ct^*.

- **Output phase:** Algorithm $\mathcal{B}$ samples a bit $b^* \xleftarrow{\mathrm{R}} \{0,1\}$. If $b^* = 0$, algorithm $\mathcal{B}$ samples $j \xleftarrow{\mathrm{R}} [N]$ and gives $(C_{\mathsf{sk}}, (\mathsf{id}_j, \mathbf{B}, \mathbf{t}_j), \pi_j)$ to the simulation-sound extractability challenger. If $b^* = 1$, algorithm $\mathcal{B}$ samples $j \xleftarrow{\mathrm{R}} [Q]$ (where Q is the number of hint queries made), parses the j^{th} hint query as $((\hat{\pi}_{\mathsf{ct}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_1^\intercal, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3), i)$, sets $\hat{\mathsf{ct}}_{\mathsf{rest}} = (\hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$, and gives the statement-proof pair $(C_{\mathsf{ct}}, (\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}})$ to the simulation-sound extractability challenger.

By construction, algorithm $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_2^{(b)}$ and $\mathsf{Hyb}_3^{(b)}$ for $\mathcal{A}$. Thus, it must be that with probability at least δ, one of the following holds:

- For some $j^* \in [N] \setminus \mathcal{H}$: $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_{j^*}, \mathbf{B}, \mathbf{t}_{j^*}), \pi_{j^*}) = 1$ (since IsValid must be 1) and the extracted vector $\mathbf{y}_{j^*} = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_{j^*}, \mathbf{B}, \mathbf{t}_{j^*}), \pi_{j^*})$ satisfies either $\mathbf{y}_{j^*} \notin \{0,1\}^{4m^5}$ or $\mathbf{B}\mathbf{y}_{j^*} \neq \mathbf{t}_{j^*}$, which means $C_{\mathsf{sk}}((\mathsf{id}_{j^*}, \mathbf{B}, \mathbf{t}_{j^*}), \mathbf{y}_{j^*}) = 0$.

- For some $j^* \in [Q]$: the $(j^*)^{\text{th}}$ hint query $((\hat{\pi}_{\mathsf{ct}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_1^\intercal, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3), i)$ satisfies

$$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}}) = 1,$$

  where $\hat{\mathsf{ct}}_{\mathsf{rest}} = (\hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$ and the extracted pair $(\mathbf{s}, \mathbf{e})$ satisfies either $\|\mathbf{e}\| > \sqrt{m}\cdot\sigma_{\mathsf{LWE}}$ or $\hat{\mathbf{c}}_1^\intercal \neq \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}^\intercal$, which means $C_{\mathsf{ct}}((\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), (\mathbf{s}, \mathbf{e})) = 0$.

We now argue that algorithm $\mathcal{B}$ is admissible. Consider the statements that algorithm $\mathcal{B}$ submits to the simulation-sound extractability challenger:

- To simulate the key-generation queries, algorithm $\mathcal{B}$ submits a tuple $(C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i))$ to the simulation-sound extractability challenger for all indices $i \in \mathcal{H}$. By construction, for the relation C_{sk}, algorithm $\mathcal{B}$ only requests proofs on statements of the form $(\mathsf{id}_{j^*}, \mathbf{B}, \mathbf{t}_{j^*})$ where $j^* \notin \mathcal{H}$. Thus, algorithm $\mathcal{B}$ outputs an admission statement-proof pair in this case.

- To simulate the challenge ciphertext, algorithm $\mathcal{B}$ submits the tuple $(C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}}))$ to the simulation-sound extractability challenger, where $\mathsf{ct}^*_{\mathsf{rest}} = (\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}}, \mathbf{c}_2^\intercal, c_3)$ contains the other components of the challenge ciphertext. Let π^*_{ct} be the proof algorithm $\mathcal{B}$ receives from the challenger and ct^* be the challenge ciphertext. We argue that all of the statement-proof pairs associated with pre-challenge and post-challenge queries are admissible. Let $(\hat{\mathsf{ct}}, i)$ be a query where $\hat{\mathsf{ct}} = (\hat{\pi}_{\mathsf{ct}}, \hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_1^\intercal, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$. Let $\hat{\mathsf{ct}}_{\mathsf{rest}} = (\hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$

  - For the pre-challenge queries, algorithm $\mathcal{B}$ aborts if $\hat{\tau}_{\mathsf{SS}} = \tau^*_{\mathsf{SS}}$. If algorithm $\mathcal{B}$ does not abort, then this means $\hat{\mathsf{ct}}_{\mathsf{rest}} \neq \mathsf{ct}^*_{\mathsf{rest}}$.

– For the post-challenge queries, we have that $\hat{\mathsf{ct}} \neq \mathsf{ct}^*$. This means that either $\hat{\pi}_{\mathsf{ct}} \neq \pi^*_{\mathsf{ct}}$ or $\mathbf{c}_1^\intercal \neq \hat{\mathbf{c}}_1^\intercal$ or $\mathsf{ct}^*_{\mathsf{rest}} \neq \hat{\mathsf{ct}}_{\mathsf{rest}}$ must hold.

In all cases, we conclude that

$$\left((\mathbf{B}, \mathbf{c}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}}), \pi^*_{\mathsf{ct}}\right) \neq \left((\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}}\right),$$

and thus algorithm $\mathcal{B}$ outputs an admissible statement-proof pair in this case.

Thus, algorithm $\mathcal{B}$ wins the simulation-sound extractability game with probability at least $\frac{\delta}{2Q^*}$ (where Q^* is an upperbound for Q and N), which is non-negligible since $Q^* = \mathsf{poly}(\lambda)$. □

Lemma 3.10. *Suppose $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot m^{3/2}\sigma_{\mathsf{LWE}}$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

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

Proof. For every hint query $((\pi_{\mathsf{ct}}, \tau_{\mathsf{SS}}, \tau_{\mathsf{virt}}, \mathbf{c}_1^\intercal, \mathbf{c}_2^\intercal, c_3), i)$, the challenger extracts a pair $(\mathbf{s}, \mathbf{e})$ such that $\mathbf{c}_1^\intercal = \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}^\intercal$ and $\|\mathbf{e}\| \leq \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$. This implies $\mathbf{c}_1^\intercal\mathbf{y}_i + e = \mathbf{s}^\intercal\mathbf{t}_i + \mathbf{e}^\intercal\mathbf{y}_i + e$ in $\mathsf{Hyb}_3^{(b)}$. Thus, the only difference between this and $\mathsf{Hyb}_4^{(b)}$ is the omission of $\mathbf{e}^\intercal\mathbf{y}_i$ and not truncating e. Since $|\mathbf{e}^\intercal\mathbf{y}_i| \leq m^{3/2}\sigma_{\mathsf{LWE}}$ and we assume $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot m^{3/2}\sigma_{\mathsf{LWE}}$, the hint distribution in $\mathsf{Hyb}_4^{(b)}$ is statistically indistinguishable from the one in $\mathsf{Hyb}_3^{(b)}$ by Lemmas 2.4 and 2.6. The claim follows by a hybrid argument over all $\mathsf{poly}(\lambda)$ hint queries. □

Lemma 3.11. *Suppose the linear secret sharing scheme* (Share, Reconst, ExplainShares) *satisfies independence of unauthorized shares and explainability (Definition 3.1). Then, for all $b \in \{0,1\}$, $\mathsf{Hyb}_4^{(b)}(\mathcal{A})$ and $\mathsf{Hyb}_5^{(b)}(\mathcal{A})$ are identically distributed.*

Proof. By explainability, the randomness from ExplainShares is identical to the true randomness used in Share. The lemma then follows by independence of unauthorized shares (since the set $[N] \setminus \mathcal{H}$ does not satisfy the policy P). □

Lemma 3.12. *Suppose $n \geq \lambda, m \geq 2n\log q, q > 2$ is prime, and $\sigma_{\mathsf{pp}} > \log m$. Then, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_5^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_6^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The hybrids are indistinguishable by Corollary 2.9. Specifically, by setting $\ell = 2m^2$ and $s = \sigma_{\mathsf{pp}}$ in Corollary 2.9, we have that the following distributions are statistically indistinguishable for any fixed vector $\mathbf{e} \in \mathbb{Z}_q^{4m^5}$:

$$\left\{(\mathbf{W}, \mathbf{R}, \mathbf{p}, \mathbf{e}^\intercal\mathbf{k}_{\mathbf{p}}) : \begin{array}{c}\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5},\ \mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n\end{array}\right\} \text{ and } \left\{(\mathbf{W}, \mathbf{R}, \mathbf{B}\mathbf{k}_{\mathbf{p}}, \mathbf{e}^\intercal\mathbf{k}_{\mathbf{p}}) : \begin{array}{c}\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5}\end{array}\right\},$$

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G}$. Similarly, the following distributions are statistically close for $i \in \mathcal{H}$:

$$\left\{(\mathbf{W}, \mathbf{R}, \mathbf{B}\mathbf{y}_i) : \begin{array}{c}\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{y}_i \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5}\end{array}\right\} \text{ and } \left\{(\mathbf{W}, \mathbf{R}, \mathbf{d}_i) : \begin{array}{c}\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{d}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n\end{array}\right\}.$$

The left and right distributions correspond to $\mathsf{Hyb}_5^{(b)}$ and $\mathsf{Hyb}_6^{(b)}$ respectively. Since $\mathbf{d}_i$ and $\hat{\mathbf{p}}_i$ are independent, the distribution of $\mathbf{d}_i - \hat{\mathbf{p}}_i$ is also uniform random. The lemma follows by a hybrid argument since $|\mathcal{H}| = \mathsf{poly}(\lambda)$. □

Lemma 3.13. *Suppose $n \geq \lambda, m \geq 2n \log q$, and $\sigma_{\mathsf{pp}} \cdot O(m^{25/2} N^3) < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_6^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{7,\kappa}^{(b)}(\mathcal{A}) = 1]| = 1/\kappa + \mathsf{negl}(\lambda).$$

Proof. By construction, in $\mathsf{Hyb}_6^{(b)}$ and $\mathsf{Hyb}_{7,\kappa}^{(b)}$ the challenger sets $\mathbf{R}$ such that $\|\mathbf{R}\| \leq \sqrt{m}\sigma_{\mathsf{pp}}$. The lemma now follows by the explainability property of Theorem 2.13. □

Lemma 3.14. *Suppose $n \geq \lambda, m \geq 2n \log q$, and $\sigma_{\mathsf{pp}} \cdot O(m^{25/2} N^3) < \sigma_{\mathsf{agg}} < 2^{\lambda_{\mathsf{DGS}}}$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

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

Proof. By construction, in $\mathsf{Hyb}_{7,\kappa}^{(b)}$ and $\mathsf{Hyb}_{8,\kappa}^{(b)}$ the challenger sets $\mathbf{R}$ such that $\|\mathbf{R}\| \leq \sqrt{m}\sigma_{\mathsf{pp}}$. The lemma now follows by the sampling distribution property of Theorem 2.13. □

Lemma 3.15. *Suppose $n \geq \lambda, m \geq 2n \log q, q > 2$ is prime, and $\sigma_{\mathsf{pp}} > \log m$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}_{8,\kappa}^{(b)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{9,\kappa}^{(b)}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The hybrids are indistinguishable by Corollary 2.9. Specifically, by setting $\ell = 2m^2$ and $s = \sigma_{\mathsf{pp}}$ in Corollary 2.9, we have that the following distributions are statistically indistinguishable for any fixed vector $\mathbf{e} \in \mathbb{Z}_q^{4m^5}$:

- $\left\{(\mathbf{W}, \mathbf{R}, \mathbf{C}_0, \mathbf{e}^\intercal \mathbf{K}_{\mathbf{C}}) : \mathbf{W} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times 2m^3},\ \mathbf{R} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{pp}}}^{n \times 2m^3}, \mathbf{K}_{\mathbf{C}} \xleftarrow{\mathsf{R}} \{0,1\}^{4m^5 \times m},\ \mathbf{C}_0 \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times m}\right\}$; and
- $\left\{(\mathbf{W}, \mathbf{R}, \mathbf{B}\mathbf{K}_{\mathbf{C}}, \mathbf{e}^\intercal \mathbf{K}_{\mathbf{C}}) : \mathbf{W} \xleftarrow{\mathsf{R}} \mathbb{Z}_q^{n \times 2m^3},\ \mathbf{R} \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{pp}}}^{n \times 2m^3}, \mathbf{K}_{\mathbf{C}} \xleftarrow{\mathsf{R}} \{0,1\}^{4m^5 \times m}\right\}$,

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G}$. The top and bottom distributions correspond to $\mathsf{Hyb}_{8,\kappa}^{(b)}$ and $\mathsf{Hyb}_{9,\kappa}^{(b)}$ respectively since $\mathbf{C}_0 + \sum_{i \in [N]} \mathbf{C}_i$ is still uniform random when $\mathbf{C}_0$ is uniform random (and independent of $\mathbf{C}_i$). We now show that the distribution of $\mathbf{z}_{0,i}$ for $i \in [N]$ is identical in both hybrids. First, in $\mathsf{Hyb}_{9,\kappa}^{(b)}$ we have that for all $i \in [N]$

$$-\mathbf{C}_0 \mathbf{v}_i = -\mathbf{B}\mathbf{K}_{\mathbf{C}} \mathbf{v}_i + \sum_{j \in [N]} \mathbf{C}_j \mathbf{v}_i = -\mathbf{B}\left(\mathbf{K}_{\mathbf{C}} \mathbf{v}_i + \sum_{j \in [N]} \mathbf{z}_{j,i}\right) + \mathbf{t}_i + \hat{\mathbf{p}}_i,$$

which follows by Theorem 2.12. We now consider the two cases:

- If $i \in \mathcal{H}$, then the challenger in both experiment sets $\mathbf{t}_i = \mathbf{d}_i - \hat{\mathbf{p}}_i$. In this case,

$$-\mathbf{C}_0 \mathbf{v}_i = \mathbf{d}_i - \mathbf{B}\left(\mathbf{K}_{\mathbf{C}} \mathbf{v}_i + \sum_{j \in [N]} \mathbf{z}_{j,i}\right).$$

  Thus, the challenger computes $\mathbf{z}_{0,i}$ identically in the two experiments.

- If $i \in [N] \setminus \mathcal{H}$, then in both experiments, $\mathbf{t}_i = \mathbf{B}\mathbf{y}_i$. In this case,

$$-\mathbf{C}_0\mathbf{v}_i = \hat{\mathbf{p}}_i - \mathbf{B}\left(\mathbf{K}_C\mathbf{v}_i - \mathbf{y}_i + \sum_{j\in[N]} \mathbf{z}_{j,i}\right)$$

Once again, we see that the challenger computes $\mathbf{z}_{0,i}$ identically in the two experiments. □

Lemma 3.16. *Suppose $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot O(N\sigma_{\mathsf{pp}} \cdot m^9 \log q \log N)$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}^{(b)}_{9,\kappa}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{10,\kappa}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The hybrids are indistinguishable by Corollary 2.11 given sufficiently large σ_{agg}. First, note that $\|\mathbf{R}\| \le \sqrt{m}\sigma_{\mathsf{pp}}$ in both hybrids. It suffices to bound $\left\|\mathbf{K}_C\mathbf{v}_i + \sum_{j\in[N]} \mathbf{z}_{j,i}\right\|$ for $i \in [N]$ since $\|\mathbf{y}_i\| \le 1$. By Theorem 2.12, we have that

$$\|\mathbf{v}_i\| \le O(\|\mathbf{R}\| \cdot m^4 \log q) = O(\sigma_{\mathsf{pp}} \cdot m^5 \log q)$$
$$\|\mathbf{z}_{j,i}\| \le O(\|\mathbf{R}\| \cdot m^7 \log q \log N) = O(\sigma_{\mathsf{pp}} \cdot m^8 \log q \log N).$$

Since $\|\mathbf{K}_C\| \le 1$, we have

$$\left\|\mathbf{K}_C\mathbf{v}_i + \sum_{j\in[N]} \mathbf{z}_{j,i}\right\| \le O(N\sigma_{\mathsf{pp}} \cdot m^8 \log q \log N).$$

Since we assume that $\sigma_{\mathsf{agg}} > \lambda^{\omega(1)} \cdot m \left\|\mathbf{K}_C\mathbf{v}_i + \sum_{j\in[N]} \mathbf{z}_{j,i}\right\|$, the claim holds by a hybrid argument over all $N = \mathsf{poly}(\lambda)$ indices. □

Lemma 3.17. *Suppose $n \ge \lambda, m \ge 2n\log q, q > 2$ is prime, and $\sigma_{\mathsf{agg}} > \log m$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}^{(b)}_{10,\kappa}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{11,\kappa}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. The hybrids are indistinguishable by Corollary 2.10. Specifically, by setting $\ell = 2m^2$ and $s = \sigma_{\mathsf{pp}}$ in Corollary 2.10, we have that the following distributions are statistically indistinguishable for $i \in \mathcal{H}$:

$$\left\{(\mathbf{W}, \mathbf{R}, \hat{\mathbf{z}}_{0,i}, \mathbf{d}_i) : \begin{array}{c} \mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{d}_i \xleftarrow{R} \mathbb{Z}_q^n,\ \hat{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}(\mathbf{d}_i) \end{array}\right\} \text{and} \left\{(\mathbf{W}, \mathbf{R}, \hat{\mathbf{z}}_{0,i}, \mathbf{B}\hat{\mathbf{z}}_{0,i}) : \begin{array}{c} \mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \hat{\mathbf{z}}_{0,i} \leftarrow D^{4m^5}_{\mathbb{Z},\sigma_{\mathsf{agg}}} \end{array}\right\}.$$

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^T \otimes \mathbf{G}$. Similarly, for $i \in [N] \setminus \mathcal{H}$ we have that the following distributions are statistically indistinguishable:

$$\left\{(\mathbf{W}, \mathbf{R}, \hat{\mathbf{z}}_{0,i}, \hat{\mathbf{p}}_i) : \begin{array}{c} \mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \hat{\mathbf{p}}_i \xleftarrow{R} \mathbb{Z}_q^n,\ \hat{\mathbf{z}}_{0,i} \leftarrow \mathbf{B}^{-1}_{\sigma_{\mathsf{agg}}}(\hat{\mathbf{p}}_i) \end{array}\right\} \text{and} \left\{(\mathbf{W}, \mathbf{R}, \hat{\mathbf{z}}_{0,i}, \mathbf{B}\hat{\mathbf{z}}_i) : \begin{array}{c} \mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \hat{\mathbf{z}}_{0,i} \leftarrow D^{4m^5}_{\mathbb{Z},\sigma_{\mathsf{agg}}} \end{array}\right\}.$$

The left and right distributions correspond to $\mathsf{Hyb}^{(b)}_{10,\kappa}$ and $\mathsf{Hyb}^{(b)}_{11,\kappa}$ respectively. The lemma follows by a hybrid argument since $N = \mathsf{poly}(\lambda)$. □

Lemma 3.18. *Suppose $n \geq \lambda, m \geq 2n \log q, q > 2$ is prime, and $\sigma_{\mathsf{pp}}, \sigma_{\mathsf{LWE}} > \log m$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

$$|\Pr[\mathsf{Hyb}^{(b)}_{11,\kappa}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{12,\kappa}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda).$$

Proof. These two experiments only differ if in the setup phase the challenger samples $\mathbf{R} \leftarrow D^{m \times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}}$ such that $\|\mathbf{R}\| > \sqrt{m}\sigma_{\mathsf{pp}}$ or if in the challenge phase, the challenger samples $\mathbf{e} \leftarrow D^{4m^5}_{\mathbb{Z},\sigma_{\mathsf{LWE}}}$ such that $\|\mathbf{e}\| > \sqrt{m}\sigma_{\mathsf{LWE}}$. By Lemma 2.4, both events happens with negligible probability. □

Lemma 3.19. *Suppose the $(2m^2, \sigma_{\mathsf{pp}})$-decomposed LWE assumption with lattice parameters $(n, m, q, \sigma_{\mathsf{LWE}})$ holds. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

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

Proof. Suppose $|\Pr[\mathsf{Hyb}^{(b)}_{12,\kappa}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}^{(b)}_{13,\kappa}(\mathcal{A}) = 1]| = \delta$ for some non-negligible δ. We use $\mathcal{A}$ to construct an efficient adversary $\mathcal{B}$ that breaks $(2m^2, \sigma_{\mathsf{pp}})$-decomposed LWE:

- **Setup phase:** At the beginning of the game, algorithm $\mathcal{B}$ gets the challenge $(\mathbf{W}, \mathbf{R}, \mathbf{c}^\intercal)$. Algorithm $\mathcal{B}$ starts running $\mathcal{A}(1^\lambda)$. Algorithm $\mathcal{B}$ sets $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathsf{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times 4m^5}$ and $\mathsf{pp}_{\mathsf{com}} = (\mathbf{W}, \mathbf{R}, \mathbf{B})$. Algorithm $\mathcal{B}$ samples $(\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{td}_{\mathsf{NIZK}}) \leftarrow \mathsf{NIZK.TrapSetup}(1^\lambda)$ along with $\mathbf{k}_{\mathbf{p}} \xleftarrow{R} \{0,1\}^{4m^5}$, $\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}} \xleftarrow{R} \{0,1\}^\lambda$. It sets $\mathbf{p} = \mathbf{B}\mathbf{k}_{\mathbf{p}}$ and gives $\mathsf{pp} = (\mathsf{crs}_{\mathsf{NIZK}}, \mathsf{pp}_{\mathsf{com}}, \mathbf{p})$ to $\mathcal{A}$.
- **Queries to H_{SS}:** Whenever algorithm $\mathcal{A}$ queries H_{SS} on a tuple $(\tau_{\mathsf{SS}}, P', i) \in \{0,1\}^*$, algorithm $\mathcal{B}$ replies with a string $\gamma_{\mathsf{SS}} \xleftarrow{R} \{0,1\}$. However, if a tuple $(\tau^*_{\mathsf{SS}}, \cdot, \cdot)$ is queried before the challenge phase, algorithm $\mathcal{B}$ aborts and outputs 0. If the adversary queries $(\tau^*_{\mathsf{SS}}, P, i)$ for $i \in [\rho_{\mathsf{SS}}]$ after the challenge phase, the challenger replies with the i^{th} bit of γ^*_{SS}.
- **Queries to H_{virt}:** Whenever algorithm $\mathcal{A}$ queries H_{virt} on a pair $(\tau_{\mathsf{virt}}, i) \in \{0,1\}^*$, algorithm $\mathcal{B}$ replies with a string $\gamma_{\mathsf{virt}} \xleftarrow{R} \{0,1\}$. However, if a tuple $(\tau^*_{\mathsf{virt}}, \cdot)$ is queried before the challenge phase, algorithm $\mathcal{B}$ aborts and outputs 0. If the adversary queries $(\tau^*_{\mathsf{virt}}, i)$ for $i \in [\rho]$ after the challenge phase, the challenger replies with the i^{th} bit of γ^*_{virt}.
- **Key-generation queries:** When algorithm $\mathcal{A}$ outputs a policy P, a set of indices $(\mathsf{id}_1, \ldots, \mathsf{id}_N)$, and a set of honest slots $\mathcal{H} \subseteq [N]$, algorithm $\mathcal{B}$ outputs 0 if $P([N] \setminus \mathcal{H}) = 1$. If algorithm $\mathcal{B}$ does not abort, it samples $\hat{\mathbf{z}}_{0,i} \leftarrow D^{4m^5}_{\mathbb{Z},\sigma_{\mathsf{agg}}}$ for all $i \in [N]$. For $i \in [N] \setminus \mathcal{H}$, algorithm $\mathcal{B}$ sets $\hat{\mathbf{p}}_i = \mathbf{B}\hat{\mathbf{z}}_{0,i}$, computes shares $\{\hat{\mathbf{p}}_i\}_{i \in \mathcal{H}} \leftarrow \mathsf{Complete}(P, \mathbf{p}, \{\hat{\mathbf{p}}_i\}_{i \in [N] \setminus \mathcal{H}})$, and computes $\gamma^*_{\mathsf{SS}} \leftarrow \mathsf{ExplainShares}(P, \mathbf{p}, (\hat{\mathbf{p}}_1, \ldots, \hat{\mathbf{p}}_N))$. For $i \in \mathcal{H}$, algorithm $\mathcal{B}$ sets $\mathbf{d}_i = \mathbf{B}\hat{\mathbf{z}}_{0,i}$ and $\mathbf{t}_i = \mathbf{d}_i - \hat{\mathbf{p}}_i$. Then, algorithm $\mathcal{B}$ computes a simulated proof $\pi_i \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i))$ and replies to $\mathcal{A}$ with $\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)$ for all $i \in \mathcal{H}$.
- **Hint queries:** When algorithm $\mathcal{A}$ makes a hint query $(\hat{\mathsf{ct}}, i)$ for $i \in \mathcal{H}$, the algorithm $\mathcal{B}$ parses $\hat{\mathsf{ct}} = (\hat{\pi}_{\mathsf{ct}}, \hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_1^\intercal, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$, sets $\mathsf{ct}_{\mathsf{rest}} = (\hat{\tau}_{\mathsf{SS}}, \hat{\tau}_{\mathsf{virt}}, \hat{\mathbf{c}}_2^\intercal, \hat{c}_3)$, checks that

  $$\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}}) = 1,$$

  and replies with $\perp$ if the check fails. For pre-challenge queries, algorithm $\mathcal{B}$ aborts the experiment and outputs 0 if $\hat{\tau}_{\mathsf{SS}} = \tau^*_{\mathsf{SS}}$. Otherwise, algorithm $\mathcal{B}$ computes

  $$(\mathbf{s}, \mathbf{e}) = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \hat{\mathbf{c}}_1^\intercal, \sqrt{m} \cdot \sigma_{\mathsf{LWE}}, \hat{\mathsf{ct}}_{\mathsf{rest}}), \hat{\pi}_{\mathsf{ct}}).$$

  If $\|\mathbf{e}\| > \sqrt{m} \cdot \sigma_{\mathsf{LWE}}$ or $\hat{\mathbf{c}}_1^\intercal \neq \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}^\intercal$, algorithm $\mathcal{B}$ outputs 0. Otherwise, algorithm $\mathcal{B}$ samples $e \leftarrow D_{\mathbb{Z},\sigma_{\mathsf{agg}}}$ and replies with $\mathbf{s}^\intercal\mathbf{t}_i + e$.

- **Challenge phase:** When Algorithm $\mathcal{A}$ outputs the public keys $\{\mathsf{pk}_i = (\mathbf{t}_i, \pi_i)\}_{i\in[N]\setminus\mathcal{H}}$ to complete the challenge tuple, algorithm $\mathcal{B}$ outputs 0 if $\mathsf{IsValid}(\mathsf{pp}, \mathsf{id}_i, \mathsf{pk}_i) = 0$ for any $i \in [N] \setminus \mathcal{H}$. If algorithm $\mathcal{B}$ does not abort, it computes $\mathbf{y}_i = \mathsf{NIZK.Extract}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{sk}}, (\mathsf{id}_i, \mathbf{B}, \mathbf{t}_i), \pi_i)$ for each $i \in [N] \setminus \mathcal{H}$. If $\mathbf{y}_i \notin \{0,1\}^{4m^5}$ or $\mathbf{B}\mathbf{y}_i \neq \mathbf{t}_i$ for any $i \in [N] \setminus \mathcal{H}$, algorithm $\mathcal{B}$ outputs 0. Algorithm $\mathcal{B}$ then samples $\mathbf{K}_{\mathbf{C}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5\times m}$ and for each $i \in [N]$ computes

$$\mathbf{C}_i = \mathsf{Com}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^\intercal \otimes (\mathbf{t}_i + \hat{\mathbf{p}}_i)) \quad \text{and} \quad \mathbf{v}_i = \mathsf{Ver}(\mathsf{pp}_{\mathsf{com}}, N, i).$$

  Algorithm $\mathcal{B}$ also computes $\mathbf{z}_{i,j} = \mathsf{Open}(\mathsf{pp}_{\mathsf{com}}, \mathbf{u}_i^\intercal \otimes (\mathbf{t}_i + \hat{\mathbf{p}}_i), j)$ for $i, j \in [N]$. Algorithm $\mathcal{B}$ constructs the re-randomization components as

$$\begin{aligned}
\mathbf{C}_0 &= \mathbf{B}\mathbf{K}_{\mathbf{C}} - \textstyle\sum_{i\in[N]} \mathbf{C}_i \\
\mathbf{z}_{0,i} &= \hat{\mathbf{z}}_{0,i} - \mathbf{K}_{\mathbf{C}}\mathbf{v}_i - \textstyle\sum_{j\in[N]} \mathbf{z}_{j,i} \;\forall i \in \mathcal{H} \\
\mathbf{z}_{0,i} &= \hat{\mathbf{z}}_{0,i} - \mathbf{K}_{\mathbf{C}}\mathbf{v}_i + \mathbf{y}_i - \textstyle\sum_{j\in[N]} \mathbf{z}_{j,i} \;\forall i \in [N] \setminus \mathcal{H} \\
\gamma^*_{\mathsf{virt}} &\leftarrow \mathsf{Explain}(\mathsf{pp}_{\mathsf{com}}, 1^{\lambda_{\mathrm{DGS}}}, 1^\kappa, (\mathbf{C}_0, \mathbf{z}_{0,1}, \ldots, \mathbf{z}_{0,N}), \sigma_{\mathsf{agg}}).
\end{aligned}$$

  Algorithm $\mathcal{B}$ computes the ciphertext validity proof by setting $\mathsf{ct}^*_{\mathsf{rest}} = (\tau^*_{\mathsf{SS}}, \tau^*_{\mathsf{virt}}, \mathbf{c}^\intercal\mathbf{K}_{\mathbf{C}}, \mathbf{c}^\intercal\mathbf{k}_{\mathbf{p}})$ and computing

$$\pi^*_{\mathsf{ct}} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}_{\mathsf{NIZK}}, C_{\mathsf{ct}}, (\mathbf{B}, \mathbf{c}^\intercal, \sqrt{m}\cdot\sigma_{\mathsf{LWE}}, \mathsf{ct}^*_{\mathsf{rest}})).$$

  Finally, algorithm $\mathcal{B}$ constructs the challenge ciphertext

$$\mathsf{ct}^* = (\pi^*_{\mathsf{ct}}, \gamma^*_{\mathsf{SS}}, \gamma^*_{\mathsf{virt}}, \mathbf{c}^\intercal, \mathbf{c}^\intercal\mathbf{K}_{\mathbf{C}}, \mathbf{c}^\intercal\mathbf{k}_{\mathbf{p}})$$

  and gives ct^* to $\mathcal{A}$.

- **Post-challenge hint queries:** Algorithm $\mathcal{B}$ handles post-challenge hint queries exactly the same way as the pre-challenge queries, except it no longer aborts with output 0 if $\tau_{\mathsf{SS}} = \tau^*_{\mathsf{SS}}$. However, it does abort with output 0 if algorithm $\mathcal{A}$ makes a hint query on the challenge ciphertext ct^*.

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

Algorithm $\mathcal{B}$ efficiently and perfectly simulates the setup phase, the public keys, the hints, and the random oracle queries exactly as in $\mathsf{Hyb}^{(b)}_{12,\kappa}$ or $\mathsf{Hyb}^{(b)}_{13,\kappa}$. If $\mathbf{c}^\intercal = \mathbf{s}^\intercal\mathbf{B} + \mathbf{e}^\intercal$ where $\mathbf{s} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D^{4m^5}_{\mathbb{Z},\sigma_{\mathsf{LWE}}}$, then algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}^{(b)}_{12,\kappa}(\mathcal{A})$. If $\mathbf{c}^\intercal \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{4m^5}$, then algorithm $\mathcal{B}$ simulates $\mathsf{Hyb}^{(b)}_{13,\kappa}(\mathcal{A})$. Thus, algorithm $\mathcal{B}$ breaks $(2m^2, \sigma_{\mathsf{pp}})$-decomposed LWE with advantage δ. □

Lemma 3.20. *Suppose $n \geq \lambda, m \geq 2n\log q, q > 2$ is prime, and $\sigma_{\mathsf{pp}} > \log m$. Then, for all polynomials $\kappa = \kappa(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $b \in \{0,1\}$ and all $\lambda \in \mathbb{N}$,*

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

Proof. The hybrids are indistinguishable by Corollary 2.9 and Lemma 2.3. Specifically, by setting $\ell = 2m^2$ and $s = \sigma_{\mathsf{pp}}$ in Corollary 2.9, we have that the following distributions are statistically indistinguishable for any $\mathbf{c}_1 \in \mathbb{Z}_q^{4m^5}$:

$$\left\{(\mathbf{W}, \mathbf{R}, \mathbf{B}\mathbf{k}_{\mathbf{p}}, \mathbf{c}_1^\intercal\mathbf{k}_{\mathbf{p}}) : \begin{array}{c}\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5}\end{array}\right\} \text{ and } \left\{(\mathbf{W}, \mathbf{R}, \mathbf{p}, \mathbf{c}_1^\intercal\mathbf{k}_{\mathbf{p}}) : \begin{array}{c}\mathbf{W} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n\times 2m^3},\ \mathbf{R} \leftarrow D^{n\times 2m^3}_{\mathbb{Z},\sigma_{\mathsf{pp}}} \\ \mathbf{k}_{\mathbf{p}} \xleftarrow{\mathrm{R}} \{0,1\}^{4m^5},\ \mathbf{p} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n\end{array}\right\},$$

where $\mathbf{B} = \mathbf{W}(\mathbf{I}_{2m^2} \otimes \mathbf{R}) + \mathrm{vec}(\mathbf{I}_{2m^2})^\intercal \otimes \mathbf{G}$. By Lemma 2.3, the following distributions are statistically close:

$$\left\{(\mathbf{c}_1^\intercal, \mathbf{c}_1^\intercal \mathbf{k}_\mathbf{p}) : \mathbf{c}_1 \xleftarrow{R} \mathbb{Z}_q^{4m^5},\ \mathbf{k}_\mathbf{p} \xleftarrow{R} \{0,1\}^{4m^5}\right\} \text{ and } \left\{(\mathbf{c}_1^\intercal, c_3) : \mathbf{c}_1 \xleftarrow{R} \mathbb{Z}_q^{4m^5},\ c_3 \xleftarrow{R} \mathbb{Z}_q\right\}.$$

The left and right distributions correspond to $\mathsf{Hyb}_{13,\kappa}^{(b)}$ and $\mathsf{Hyb}_{14,\kappa}^{(b)}$ respectively. The lemma follows by appealing to Corollary 2.9 to switch the first pair of distributions, and then appealing to Lemma 2.3 to switch the second pair of distributions. □

Completing the proof. To complete the proof of Theorem 3.6, first notice that the challenger's behavior in $\mathsf{Hyb}_{14,\kappa}^{(b)}(\mathcal{A})$ is independent of $b \in \{0,1\}$, so $\Pr[\mathsf{Hyb}_{14,\kappa}^{(0)}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_{14,\kappa}^{(1)}(\mathcal{A}) = 1]$. Now, recall the assumption that algorithm $\mathcal{A}$ wins the semi-policy-and-query-selective security game with advantage ε. This implies $|\Pr[\mathsf{Hyb}_0^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0^{(1)}(\mathcal{A}) = 1]| = \varepsilon(\lambda)$. Since ε is non-negligible, there exists some polynomial κ' such that for infinitely many $\lambda \in \mathbb{N}$, $\varepsilon(\lambda) \geq 1/\kappa'(\lambda)$. Let $\kappa(\lambda) = 3\kappa'(\lambda)$. By Lemmas 3.7 to 3.20, we have for all $\lambda \in \mathbb{N}$ (and recalling that for $i \leq 6$, $\mathsf{Hyb}_{i,\kappa}^{(b)}(\mathcal{A}) \equiv \mathsf{Hyb}_i^{(b)}(\mathcal{A})$),

$$\begin{aligned}
|\Pr[\mathsf{Hyb}_0^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0^{(1)}(\mathcal{A}) = 1]| &\leq \sum_{i=0}^{13} |\Pr[\mathsf{Hyb}_{i,\kappa}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i+1,\kappa}^{(0)}(\mathcal{A}) = 1]| \\
&\quad + |\Pr[\mathsf{Hyb}_{14}^{(0)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{14}^{(1)}(\mathcal{A}) = 1]| \\
&\quad + \sum_{i=0}^{13} |\Pr[\mathsf{Hyb}_{i+1,\kappa}^{(1)}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{i,\kappa}^{(1)}(\mathcal{A}) = 1]| \\
&\leq 2/\kappa(\lambda) + \delta(\lambda),
\end{aligned}$$

where $\delta(\lambda) = \mathsf{negl}(\lambda)$ is a negligible function. Thus, for infinitely many $\lambda \in \mathbb{N}$, $2/\kappa(\lambda) + \delta(\lambda) \geq 1/\kappa'(\lambda) = 3/\kappa(\lambda)$. Hence $\delta(\lambda) \geq 1/\kappa(\lambda)$ for infinitely many λ, contradicting the fact that δ is negligible, which proves the theorem. □

Parameter instantiation. We provide an example of lattice parameters for Construction 3.3 that satisfy the conditions in Theorems 3.4 to 3.6. Let λ be the security parameter, $N \leq 2^\lambda$ be the number of slots, and $\varepsilon \in (0,1)$ be a constant. We set

$$\begin{aligned}
n &= (\lambda \log N)^{1/\varepsilon} \cdot \mathrm{poly}(\log \lambda, \log N) \\
m &= 3n \log q \\
\sigma_{\mathsf{LWE}} &= \mathrm{poly}(n) \\
\sigma_{\mathsf{pp}} &= \log m \\
\sigma_{\mathsf{agg}} &= 2^\lambda \cdot O(m^{14} N^3) \cdot \mathrm{poly}(n) \\
q &= 2^\lambda \cdot \mathrm{poly}(n, N) \\
\lambda_{\mathsf{DGS}} &= O(\lambda).
\end{aligned}$$

Note that $q < 2^{n^\varepsilon}$, so we require a sub-exponential modulus-to-noise ratio. The instantiation above can be summarized as follows:

Corollary 3.21 (Distributed Monotone-Policy Encryption)**.** *Let λ be a security parameter. Then, under the ℓ-decomposed LWE assumption with $\ell = \mathrm{poly}(\lambda)$ and a sub-exponential modulus-to-noise ratio, there*

exists a semi-policy-and-query-selective distributed monotone-policy encryption scheme in the static model for any policy family (on at most $N \leq 2^\lambda$ inputs) with an explainable $\{0,1\}$-linear secret sharing scheme (Definition 3.1) in the random oracle model. The size of the public parameters, the size of the user public/private keys, and the size of the ciphertext are all $\mathsf{poly}(\lambda)$.

Corollary 3.22 (Distributed Monotone-Policy Encryption for DNF Formulas)**.** *Let λ be a security parameter. Then, under the ℓ-decomposed LWE assumption with $\ell = \mathsf{poly}(\lambda)$ and a sub-exponential modulus-to-noise ratio, there exists a semi-policy-and-query-selective distributed monotone-policy encryption scheme in the static model for policies that can be computed by DNF formulas (with up to $N = 2^\lambda$ variables). The size of the public parameters and the size of the ciphertext are* $\mathsf{poly}(\lambda)$ *and the size of each user's public/private key is $K \cdot \mathsf{poly}(\lambda)$, where K is a bound on the number of times each variable can appear in the DNF computing a policy.*

Acknowledgments

We thank Valerio Cini, Pratish Datta, Giulio Malavolta, and Hoeteck Wee for helpful discussions. David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Sloan Fellowship, a Microsoft Research Faculty Fellowship, a Google Research Scholar Award, an Amazon Research Award, and a gift from the Stellar Development Foundation.

References

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

[ABI+23] Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. In *STOC*, 2023.

[ADM+24] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. Signature-based witness encryption with compact ciphertext. In *ASIACRYPT*, 2024.

[AGY26] Abtin Afshar, Rishab Goyal, and Saikumar Yadugiri. Distributed monotone-policy encryption with silent setup from lattices. *IACR Cryptol. ePrint Arch.*, 2026. Available at https://eprint.iacr.org/2026/372.pdf.

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

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Zeroizing attacks against evasive and circular evasive lwe. In *TCC*, 2025.

[ANP23] Benny Applebaum, Oded Nir, and Benny Pinkas. How to recover a secret with $O(n)$ additions. In *CRYPTO*, 2023.

[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.

[BCF+25] Jan Bormet, Arka Rai Choudhuri, Sebastian Faust, Sanjam Garg, Hussien Othman, Guru-Vamsi Policharla, Ziyan Qu, and Mingyuan Wang. BEAST-MEV: batched threshold encryption with silent setup for MEV prevention. *IACR Cryptol. ePrint Arch.*, 2025. Available at https://eprint.iacr.org/2025/1419.pdf.

[BDE+18] Jonathan Bootle, Claire Delaplace, Thomas Espitau, Pierre-Alain Fouque, and Mehdi Tibouchi. LWE without modular reduction and improved side-channel attacks against BLISS. In *ASIACRYPT*, 2018.

[BDJ+25] Pedro Branco, Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, Spencer Peters, and Vinod Vaikuntanathan. Pseudorandom obfuscation and applications. In *CRYPTO*, 2025.

[Bei96] Amos Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, Technion - Israel Institute of Technology, Israel, 1996.

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

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, 2018.

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In *ASIACRYPT*, 2024.

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

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

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

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

[DDP+25] Sourav Das, Pratish Datta, Aditi Partap, Swagata Sasmal, and Mark Zhandry. Optimal threshold traitor tracing. *IACR Cryptol. ePrint Arch.*, 2025. Available at https://eprint.iacr.org/2025/2154.pdf.

[Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO*, 1987.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.

[DJM+25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive LWE. In *CRYPTO*, 2025.

[DJWW25] Lalita Devadas, Abhishek Jain, Brent Waters, and David J. Wu. Succinct witness encryption for batch languages and applications. In *ASIACRYPT*, 2025.

[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.

[Fra89] Yair Frankel. A practical protocol for large group oriented networks. In *EUROCRYPT*, 1989.

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

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

[GJKR99] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EUROCRYPT*, 1999.

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

[GLWW23] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Realizing flexible broadcast encryption: How to broadcast to a public-key directory. In *ACM CCS*, 2023.

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

[GWWW26] Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. Threshold batched identity-based encryption from pairings in the plain model. In *EUROCRYPT*, 2026.

[HHY25] Tzu-Hsiang Huang, Wei-Hsiang Hung, and Shota Yamada. A note on obfuscation-based attacks on private-coin evasive LWE. *IACR Cryptol. ePrint Arch.*, 2025. Available at https://eprint.iacr.org/2025/421.pdf.

[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.

[HJL25] Yao-Ching Hsieh, Aayush Jain, and Huijia Lin. Lattice-based post-quantum iO from circular security with random opening assumption (part II: zeroizing attacks against private-coin evasive LWE assumptions). In *CRYPTO*, 2025.

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

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, 2010.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, 2011.

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

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, 2016.

[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.

[RSY21] Leonid Reyzin, Adam Smith, and Sophia Yakoubov. Turning HATE into LOVE: Compact homomorphic ad hoc threshold encryption for scalable MPC. In *CSCML*, 2021.

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

[SDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *STOC*, 1994.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11), 1979.

[TCZ+20] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *IEEE S&P*, 2020.

[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.

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

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

[WQZDF10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption. In *ACM CCS*, 2010.

[WW25] Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In *CRYPTO*, 2025.

[WW26] Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. In *EUROCRYPT*, 2026.

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

A Additional Lattice Properties

In this section, we give the formal proofs of Corollaries 2.9 to 2.11. We start by recalling a few standard definitions. Our presentation is adapted from [CHW25, Appendix B].

Standard lattice definitions. For $m \in \mathbb{N}$, a lattice $\Lambda \subset \mathbb{R}^m$ is a discrete additive subgroup of $\mathbb{R}^m$. For a lattice $\Lambda \subset \mathbb{R}^m$, the dual lattice is defined to be $\Lambda^* = \{\mathbf{w} \in \mathbb{R}^m \mid \forall \mathbf{x} \in \Lambda : \mathbf{w}^\intercal \mathbf{x} \in \mathbb{Z}\}$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, the q-ary lattice $\Lambda^\perp(\mathbf{A})$ is

$$\Lambda^\perp(\mathbf{A}) := \left\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\right\} \subseteq \mathbb{Z}^m$$

For a Gaussian width parameter $\sigma > 0$, we write $\rho_\sigma : \mathbb{R}^m \to \mathbb{R}$ to denote the Gaussian function $\rho_\sigma(\mathbf{x}) := \exp(-\pi\|\mathbf{x}\|_2^2/\sigma^2)$. For a lattice $\Lambda \subset \mathbb{R}^m$, we write $\rho_\sigma(\Lambda) := \sum_{\mathbf{x}\in\Lambda} \rho_\sigma(\mathbf{x})$. For a lattice $\Lambda \subset \mathbb{R}^m$ and a positive real number $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest real value $\sigma > 0$ such that $\rho_{1/\sigma}(\Lambda^*) \le 1 + \varepsilon$ [MR04]. It is easy to see then that for any $\varepsilon > 0$ and any choice of matrices $\mathbf{M}, \mathbf{M}'$,

$$\eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \ge \eta_\varepsilon(\Lambda^\perp([\mathbf{M} \mid \mathbf{A} \mid \mathbf{M}'])).$$

We now state some additional properties on the smoothing parameter:

Lemma A.1 (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 A.2 (Smoothing Parameter Bound [GPV08, Lemma 5.3, adapted])**.** *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 $\eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \le \log m$.*

Remark A.3 (Restating Lemmas 2.5 and 2.7 Using the Smoothing Parameter)**.** We can restate Lemma 2.5 (adapted from [GPV08]) and Lemma 2.7 (adapted from [CHW25]) to hold for *all* matrices $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, provided that the width parameter σ in the respective lemma statements satisfies $\sigma \ge \eta_\varepsilon(\Lambda^\perp(\mathbf{A}))$ for some negligible $\varepsilon(m) = \mathsf{negl}(m)$. The statements in Lemmas 2.5 and 2.7 then follow from Lemma A.2.

We now give the proofs of Corollaries 2.9 to 2.11.

Proof of Corollary 2.9. Let $\mathbf{W} = [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_\ell]$ where each $\mathbf{W}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} = [\mathbf{R}_1 \mid \cdots \mid \mathbf{R}_\ell]$ where each $\mathbf{R}_i \in \mathbb{Z}_q^{m\times m}$. Then we have $\mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) = [\mathbf{W}_1\mathbf{R}_1 \mid \mathbf{W}_1\mathbf{R}_2 \mid \cdots \mid \mathbf{W}_\ell\mathbf{R}_\ell]$. Let $\mathbf{K}_i \in \mathbb{Z}_q^{m\times k}$ be the i^{th} vertical block of $\mathbf{K}$ and $\mathbf{e}_i \in \mathbb{Z}_q^m$ be the i^{th} vertical block of $\mathbf{e}$ for $i \in [\ell^2]$. We now define a sequence of hybrid distributions:

- $\mathcal{D}_0$: $(\mathbf{W}, \mathbf{R}, \mathbf{BK}, \mathbf{e}^\intercal\mathbf{K})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m\times \ell m}, \mathbf{K} \xleftarrow{R} \{0,1\}^{\ell^2 m\times k}$.
- $\mathcal{D}_1$: $(\mathbf{W}, \mathbf{R}, \mathbf{BK}, \mathbf{e}^\intercal\mathbf{K})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}, \mathbf{R}_2 \leftarrow (\mathbf{W}_1)_s^{-1}(\mathbf{A}), [\mathbf{R}_1 \mid \mathbf{R}_3 \mid \cdots \mid \mathbf{R}_\ell] \leftarrow D_{\mathbb{Z},s}^{m\times(\ell-1)m}$, and $\mathbf{K} \xleftarrow{R} \{0,1\}^{\ell^2 m\times k}$. Note that $\mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) = [\mathbf{W}_1\mathbf{R}_1 \mid \mathbf{A} \mid \cdots \mid \mathbf{W}_\ell\mathbf{R}_\ell]$ here.
- $\mathcal{D}_2$: $(\mathbf{W}, \mathbf{R}, \mathbf{U}, \mathbf{e}^\intercal\mathbf{K})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}, \mathbf{R}_2 \leftarrow (\mathbf{W}_1)_s^{-1}(\mathbf{A}), [\mathbf{R}_1 \mid \mathbf{R}_3 \mid \cdots \mid \mathbf{R}_\ell] \leftarrow D_{\mathbb{Z},s}^{m\times(\ell-1)m}, \mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n\times k}$, and $\mathbf{K} \xleftarrow{R} \{0,1\}^{\ell^2 m\times k}$.
- $\mathcal{D}_3$: $(\mathbf{W}, \mathbf{R}, \mathbf{U}, \mathbf{e}^\intercal\mathbf{K})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m\times \ell m}, \mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n\times k}, \mathbf{K} \xleftarrow{R} \{0,1\}^{\ell^2 m\times k}$.

We show that each pair of adjacent distributions are statistically indistinguishable:

- By Lemma 2.5 on the tuple $(\mathbf{R}_1, \mathbf{W}_1\mathbf{R}_1)$, $\Delta(\mathcal{D}_0, \mathcal{D}_1) = \mathsf{negl}(n)$. The reverse direction of Lemma 2.5 similarly shows that $\Delta(\mathcal{D}_2, \mathcal{D}_3) = \mathsf{negl}(n)$.
- By Lemma 2.3, $\Delta(\mathcal{D}_1, \mathcal{D}_2) = \mathsf{negl}(n)$. Given adversary $\mathcal{A}$ that distinguishes $\mathcal{D}_1$ and $\mathcal{D}_2$ with non-negligible advantage ε, we construct adversary $\mathcal{B}$ to contradict Lemma 2.3. When given $(\mathbf{A}, \mathbf{U}', \mathbf{e}_2^\intercal\mathbf{K}_2)$ from the Lemma 2.3 challenger, algorithm $\mathcal{B}$ samples $\mathbf{K}_i \xleftarrow{R} \mathbb{Z}_q^{m\times k}$ for $i \in [\ell^2] \setminus \{2\}$, $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}$, $\mathbf{R}_1 \leftarrow (\mathbf{W}_1)_s^{-1}(\mathbf{A})$, and $[\mathbf{R}_1 \mid \mathbf{R}_3 \mid \cdots \mid \mathbf{R}_\ell] \leftarrow D_{\mathbb{Z},s}^{m\times(\ell-1)m}$. Algorithm $\mathcal{B}$ sets $\mathbf{B} = [\mathbf{W}_1\mathbf{R}_1 \mid \mathbf{A} \mid \cdots \mid \mathbf{W}_\ell\mathbf{R}_\ell] + \mathrm{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G}$ and outputs the tuple $(\mathbf{W}, \mathbf{R}, \mathbf{BK}, [\mathbf{e}_1^\intercal\mathbf{K}_1 \mid \cdots \mid \mathbf{e}_{\ell^2}^\intercal\mathbf{K}_{\ell^2}])$ by using $\mathbf{U}'$ to simulate the second block of $\mathbf{B}$ multiplied by the second block of $\mathbf{K}$. If $\mathbf{U}'$ is $\mathbf{AK}_2$, then $\mathcal{B}$ simulates $\mathcal{D}_1$. If $\mathbf{U}'$ is uniform, then $\mathcal{B}$ simulates $\mathcal{D}_2$. Thus, algorithm $\mathcal{B}$ distinguishes the distributions in Lemma 2.3 with advantage ε, a contradiction.

By a hybrid argument, $\Delta(\mathcal{D}_0, \mathcal{D}_3) = \mathsf{negl}(n)$ as desired. □

Proof of Corollary 2.10. Let $\mathbf{W} = [\mathbf{W}_1 \mid \cdots \mid \mathbf{W}_\ell]$ where each $\mathbf{W}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} = [\mathbf{R}_1 \mid \cdots \mid \mathbf{R}_\ell]$ where each $\mathbf{R}_i \in \mathbb{Z}_q^{m\times m}$. Then we have $\mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) = [\mathbf{W}_1\mathbf{R}_1 \mid \mathbf{W}_1\mathbf{R}_2 \mid \cdots \mid \mathbf{W}_\ell\mathbf{R}_\ell]$. We now define a sequence hybrid distributions:

- $\mathcal{D}_0$: $(\mathbf{W}, \mathbf{R}, \mathbf{x}, \mathbf{Bx})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m\times \ell m}, \mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}^{\ell^2 m}$.
- $\mathcal{D}_1$: $(\mathbf{W}, \mathbf{R}, \mathbf{x}, \mathbf{Bx})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}, \mathbf{R}_2 \leftarrow (\mathbf{W}_1)_s^{-1}(\mathbf{A}), [\mathbf{R}_1 \mid \mathbf{R}_3 \mid \cdots \mid \mathbf{R}_\ell] \leftarrow D_{\mathbb{Z},s}^{m\times(\ell-1)m}$, and $\mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}^{\ell^2 m}$. Note that $\mathbf{W}(\mathbf{I}_\ell \otimes \mathbf{R}) = [\mathbf{W}_1\mathbf{R}_1 \mid \mathbf{A} \mid \cdots \mid \mathbf{W}_\ell\mathbf{R}_\ell]$ here.
- $\mathcal{D}_2$: $(\mathbf{W}, \mathbf{R}, \mathbf{x}, \mathbf{y})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}, \mathbf{R}_2 \leftarrow (\mathbf{W}_1)_s^{-1}(\mathbf{A}), [\mathbf{R}_1 \mid \mathbf{R}_3 \mid \cdots \mid \mathbf{R}_\ell] \leftarrow D_{\mathbb{Z},s}^{m\times(\ell-1)m}, \mathbf{y} \xleftarrow{R} \mathbb{Z}_q^n$, and $\mathbf{x} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{y})$.
- $\mathcal{D}_3$: $(\mathbf{W}, \mathbf{R}, \mathbf{x}, \mathbf{y})$ where $\mathbf{W} \xleftarrow{R} \mathbb{Z}_q^{n\times \ell m}, \mathbf{R} \leftarrow D_{\mathbb{Z},s}^{m\times \ell m}, \mathbf{y} \xleftarrow{R} \mathbb{Z}_q^n, \mathbf{x} \leftarrow \mathbf{B}_\sigma^{-1}(\mathbf{y})$.

We show that each pair of adjacent distributions are statistically indistinguishable:

- By Lemma 2.5 on the tuple $(\mathbf{R}_1, \mathbf{W}_1\mathbf{R}_1)$, $\Delta(\mathcal{D}_0, \mathcal{D}_1) = \mathsf{negl}(n)$. The reverse direction of Lemma 2.5 similarly shows that $\Delta(\mathcal{D}_2, \mathcal{D}_3) = \mathsf{negl}(n)$.

- Since $\mathbf{B} = [\mathbf{W}_1\mathbf{R}_1 \mid \mathbf{A} \mid \cdots \mid \mathbf{W}_\ell\mathbf{R}_\ell] + \mathrm{vec}(\mathbf{I}_\ell)^\intercal \otimes \mathbf{G}$ where $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}$ in $\mathcal{D}_1$ and $\mathcal{D}_2$, with all but $\mathsf{negl}(n)$ probability we have $\log m \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{B}))$ since the columns of $\mathbf{A}$ are contained in the columns of $\mathbf{B}$. This immediately implies that $\Delta(\mathcal{D}_1, \mathcal{D}_2) = \mathsf{negl}(n)$.

By a hybrid argument, $\Delta(\mathcal{D}_0, \mathcal{D}_3) = \mathsf{negl}(n)$ as desired. □

Proof of Corollary 2.11. The proof is analogous to the proof of Corollary 2.10, except $\Delta(\mathcal{D}_1, \mathcal{D}_2) = \mathsf{negl}(\lambda)$ by Lemma 2.7 instead of Lemma 2.5. In particular, we once again appeal to the fact that $\log m \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{A})) \geq \eta_\varepsilon(\Lambda^\perp(\mathbf{B}))$ since the columns of $\mathbf{A}$ are contained in the columns of $\mathbf{B}$. □

B Explainable $\{0,1\}$-Linear Secret Sharing for DNFs

In this section, we show how to obtain the explainable linear secret sharing scheme for DNF formulas that satisfies Definition B.3. We start by recalling some preliminaries on linear secret sharing:

Definition B.1 (Access Structure [Bei96])**.** An access structure on n parties is a set $\mathbb{A} \subseteq 2^{[n]} \setminus \varnothing$ of non-empty subsets of S. We refer to the sets in $\mathbb{A}$ as authorized sets and the sets outside $\mathbb{A}$ as unauthorized sets. An access structure is *monotone* if for all sets $T_1, T_2 \in 2^T$, if $T_1 \in \mathbb{A}$ and $T_1 \subseteq T_2$, then $T_2 \in \mathbb{A}$.

Definition B.2 (Linear Secret Sharing Scheme [Bei96])**.** Let $[n]$ be a set of parties and $q \in \mathbb{N}$ be a modulus. A linear secret sharing scheme for monotone policies over $\mathbb{Z}_q$ is a pair $(\mathbf{M}, \rho)$, where $\mathbf{M} \in \mathbb{Z}_q^{\ell\times d}$ is a "share-generating" matrix and $\rho\colon [\ell] \to [n]$ is a "row-labeling" function with the following properties:

- **Share generation:** To share a message $\mu \in \mathbb{Z}_q$, sample $v_2, \ldots, v_d \xleftarrow{R} \mathbb{Z}_q$ and define the vector $\mathbf{v} = [\mu, v_2, \ldots, v_d]^\intercal$. Then, $\mathbf{u} = \mathbf{M}\mathbf{v}$ is the vector of shares where $u_i \in \mathbb{Z}_q$ belongs to party $\rho(i)$.
- **Share reconstruction:** Let $T \subseteq [n]$ be a set of parties. For a set of parties $T \subseteq [n]$, let $\mathbf{M}_T \in \mathbb{Z}_q^{k\times d}$ be the sub-matrix of $\mathbf{M}$ containing the rows $\mathbf{m}_i^\intercal$ of $\mathbf{M}$ where $\rho(i) \in T$. Then, the following properties hold:
  - If T is an authorized set, then there exists a vector $\mathbf{v}_T \in \mathbb{Z}_q^k$ such that $\mathbf{v}_T^\intercal\mathbf{M}_T = \mathbf{e}_1^\intercal$, where $\mathbf{e}_1$ is the first basis vector. Equivalently, the vector $\mathbf{e}_1^\intercal$ is in the row-span of $\mathbf{M}_T$.
  - When T is unauthorized, there exists a vector $\mathbf{v}^* \in \mathbb{Z}^d$ where the first component $v_1^* = 1$ such that $\mathbf{M}_T\mathbf{v}^* = \mathbf{0}$ (i.e., the vector $\mathbf{v}^*$ is orthogonal to the rows of $\mathbf{M}$ associated with the attributes in T).

In this work, we require linear secret sharing schemes satisfying two additional properties. These are the same properties used in [DKW21] to construct multi-authority attribute-based encryption for DNFs from the plain LWE assumption. We define these properties below:

Definition B.3 (Properties of Linear Secret Sharing Scheme)**.** Let $[n]$ be a set of parties and $q \in \mathbb{N}$ be a prime. We define the following properties on a linear secret sharing scheme $(\mathbf{M}, \rho)$ over $\mathbb{Z}_q$:

- $\{0,1\}$**-reconstruction:** The linear secret sharing scheme supports $\{0,1\}$-reconstruction if for every authorized set $T \subseteq [n]$, there exists a vector $\mathbf{v}_T \in \{0,1\}^k$ such that $\mathbf{v}_T^\intercal\mathbf{M}_T = \mathbf{e}_1^\intercal$. In other words, the reconstruction coefficients are binary-valued.
- **Injective:** The row labeling function $\rho\colon [\ell] \to [n]$ is injective and $\ell = n$. By permuting the rows of $\mathbf{M}$, we can take $\rho(i) = i$ without loss of generality. In this case, the policy can be described by the matrix $\mathbf{M}$ alone.

- **Linear independence for unauthorized sets.** For every *unauthorized* set $S \subseteq [n]$, the rows of the matrix $\mathbf{M}_S$ are linearly independent.

Fact B.4 (Linear Secret Sharing Scheme for DNFs [LW11, DKW21])**.** Let $\mathcal{P}$ be the class of policies that can be computed by a DNF formula where each input variable is used exactly once. Then, there exists a linear secret sharing scheme satisfying the additional properties in Definition B.3 for $\mathcal{P}$.

Lemma B.5 (Explainable $\{0,1\}$-Linear Secret Sharing)**.** *Let q be a prime and take any linear secret sharing scheme for monotone policies over $\mathbb{Z}_q$ that satisfies Definition B.3. Then, the same scheme implies an explainable $\{0,1\}$-linear secret sharing scheme with message space $\mathbb{Z}_q$ and share space $\mathbb{Z}_q$.*

Proof. We construct a $\{0,1\}$-explainable linear secret sharing scheme as follows:

- $\mathsf{Share}(P, \mu; (v_2, \ldots, v_d))$: On input a policy P with associated share-generation matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times d}$, a message $\mu \in \mathbb{Z}_q$, and randomness $v_2, \ldots, v_d \in \mathbb{Z}_q$, the share-generation defines the vector $\mathbf{v} = [\mu, v_2, \ldots, v_d]^\intercal$. It computes $\mathbf{s} = \mathbf{M}\mathbf{v}$ and outputs the shares $(s_1, \ldots, s_n)$.
- $\mathsf{Reconst}(P, T)$: On input the policy P with associated share-generation matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times d}$ and a set $T \subseteq [n]$, let $\mathbf{v}_T \in \mathbb{Z}_q^n$ be the vector such that $\mathbf{v}_T^\intercal \mathbf{M}_T = \mathbf{e}_1^\intercal$. Let $T' \subseteq T$ be the subset of indices $i \in T$ where the associated coefficient in $\mathbf{v}_T$ has value 1.
- $\mathsf{ExplainShares}(P, \mu, (s_1, \ldots, s_n))$: On input the policy P with associated share-generation matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times d}$, the message μ and the shares $\mathbf{s} = [s_1, \ldots, s_n]^\intercal$, sample $v_2, \ldots, v_d \xleftarrow{R} \mathbb{Z}_q$ such that $\mathbf{s} = \mathbf{M}\mathbf{v}$ where $\mathbf{v} = [\mu, v_2, \ldots, v_d]$. Output the randomness $(v_2, \ldots, v_d)$.

We now check the properties in Definition B.3. First, note that correctness follows immediately from the fact that the linear secret sharing scheme supports $\{0,1\}$-reconstruction. We now show that unauthorized shares are linearly independent and that the scheme is explainable:

- The independence of unauthorized shares property follows via the linear independence property. To see this, take any policy P with associated share-generation matrix $\mathbf{M} \in \mathbb{Z}_q^{n \times d}$. Take any unauthorized set $T \subseteq [n]$, and let $\mathbf{v}^*$ be the vector where $\mathbf{M}_T \mathbf{v}^* = \mathbf{0}$ (and $v_1^* = 1$). First, observe that the following distributions are identical:
  - Sample $v_2, \ldots, v_d \xleftarrow{R} \mathbb{Z}_q$ and output $[\mu, v_2, \ldots, v_d]$.
  - Sample $\hat{\mathbf{v}} \xleftarrow{R} \mathbb{Z}_q^d$ and output $\hat{\mathbf{v}} + (\mu - \hat{v}_1) \cdot \mathbf{v}^*$.

  Consider the distribution of the unauthorized shares of μ. The shares are distributed as
  $$\mathbf{M}_T(\hat{\mathbf{v}} + (\mu - \hat{v}_1) \cdot \mathbf{v}^*) = \mathbf{M}_T \hat{\mathbf{v}}.$$
  Since $\hat{\mathbf{v}} \xleftarrow{R} \mathbb{Z}_q^d$ and the rows of $\mathbf{M}_T$ are linearly independent and $\hat{\mathbf{v}}$ is uniform, this means the marginal distribution of the shares associated with parties in T are also uniform. Finally, we can define the Complete algorithm to sample $\mathbf{v} \xleftarrow{R} \mathbb{Z}_q^d$ where $v_1 = \mu$ conditioned on $\mathbf{M}_T \mathbf{v}$ matching the given shares. The overall shares are then derived from $\mathbf{u} = \mathbf{M}\mathbf{v}$ in the same way as in the Share algorithm. By construction, this is identical to the real distribution.
- The explainability property holds by construction (the $\mathsf{ExplainShares}$ algorithm samples $v_2, \ldots, v_d \xleftarrow{R} \mathbb{Z}_q$ conditioned on the value of the output shares). □

Remark B.6 (Sharing Longer Messages)**.** We can extend Lemma B.5 to support sharing vectors $\boldsymbol{\mu} \in \mathbb{Z}_q^t$ with share space $\mathbb{Z}_q^t$ by secret sharing each component individually (with independent randomness). This preserves all of the required properties.