Lattice-based Threshold Traitor Tracing with Public Traceability

Sébastien Canard[0009-0006-9588-7418], Nathan Papon[0009-0002-6013-3959], Duong Hieu Phan[0000-0003-1136-4064]

Télécom Paris, LTCI, Institut Polytechnique de Paris, Palaiseau, France
{sebastien.canard,nathan.papon,hieu.phan}@telecom-paris.fr

Abstract. Since the introduction of Threshold Traitor Tracing by Boneh, Partap and Rotem at CRYPTO '24, several works have extended the functionalities within the framework or improved the parameters. However, most of the existing solution fall short in providing post quantum security guarantees. The only lattice-based construction, due to Das et al. from EUROCRYPT '26, achieves post-quantum security but is limited to private tracing: a dedicated tracing authority holds a secret tracing key. In a threshold system, where the fundamental goal is to distribute trust, such a single point of failure is undesirable.
In this work, we construct the first threshold traitor tracing scheme that simultaneously achieves post-quantum security and public traceability, where anyone can trace a pirate decoder without any secret tracing key. Our core building block is a *Q-Partite Threshold Public Key Encryption* (QTPKE) scheme, which is known to imply threshold traitor tracing when combined with a robust IPP code: we build QTPKE from plain *Learning With Errors* (LWE) using a key-shifting mechanism on top of Regev's encryption scheme thresholdised via $\{0,1\}$-Linear Secret Sharing. We finally prove the security of our scheme in the standard model under standard lattice assumptions.

Keywords: Threshold Traitor Tracing, Lattice Based Cryptography, Public Traceability

1 Introduction

Decentralising operations among different parties of a system has been a growing desire of recent real world applications of cryptography. Threshold cryptography aims to eliminate the need for trusted authorities by allowing users to dynamically add keys or distribute decryption trust among multiple parties. In a traditional public key encryption scheme, a user in possession of a secret key can fully decrypt any ciphertext encrypted for the associated public key. However, when sending sensitive content, it might be desirable to require the cooperation of more than one user to recover the plaintext.

Hence, in a threshold encryption scheme [20], the secret decryption key is shared among several users. When all N users of the system are required to

cooperate, the scheme is said to be *full threshold*, or *distributed*. This design breaks down if a party is absent for decryption, which can happen if a server goes offline or a user loses their key. Therefore, relaxing the threshold to any t of the N members of the system is more desirable. In such a scheme, any coalition of up to $t-1$ corrupted parties cannot learn any non-trivial information about the encrypted plaintext.

By definition, if a coalition of malicious users grows larger than the threshold t, then the confidentiality of the message gets compromised. Motivated by this limitation, [9] adapted the notions of traitor tracing to such a setting of threshold decryption. Then, if it is still not possible to completely prevent coalitions larger than the threshold from misbehaving, users who decided to actively join the bad coalition can be traced and held accountable.

More precisely, in a traitor tracing scheme, a coalition of malicious users builds an obfuscated device called a *decoder*, which works by receiving a ciphertext as input, and successfully outputs a plaintext with non-negligible probability. Initially introduced in [17], it offers a rigorous framework for identifying the source of a decoder while preventing false accusations of honest users. Original traitor tracing was mainly used to deter piracy of protected media content such as movies, radio streams or music. Threshold traitor tracing has found more recent applications such as electronic voting, encrypted mempools or sealed-bid auctions. While not completely preventing dishonest behaviour of large coalitions, this mechanism deters them from selling their key shares since they could be held accountable, identity revealed or rights revoked.

Since the introduction of threshold traitor tracing in [9], several works have extended this framework. In [7], the authors add both CCA security to the original Bipartite Threshold KEM of [9], and the possibility to encrypt for a specific identity, making it identity-based. While the initial construction in [9] necessitates a specific tracing key, several works [15,11,16] consider public traceability, where anyone can trace traitors. The work in [11] introduces new techniques of tracing and allows to catch up to t traitors, when traditional traitor tracing techniques allow for a single party to be accused. However, their construction relies on a non-black box transformation of a witness encryption scheme, which is proven secure in the Generic Group Model. Therefore, the security of their construction also falls in the Generic Group Model paradigm.

Post quantum Threshold Traitor Tracing. A drawback common to all these schemes is their lack of resistance against a quantum adversary. All the constructions are pairing-based (except in [15], but the transformation to a pairing-based construction with constant size public keys is straightforward following the pattern given in [9]). To date, the only construction from lattice assumptions is the one in [19]. However, since their aim is to achieve optimal share size, they use ramp secret sharing [3]. Indeed, it implies that the threshold for recovering the secret is different from the maximal number of corruptions. Therefore, there is a security gap in which a coalition smaller than the decryption threshold but larger than the security threshold can learn non-trivial information about the shared secret while it is impossible to trace them. As mentioned in [9], it is close

to impossible to trace a coalition that is smaller than the decryption threshold with traitor tracing techniques. Furthermore, their ramp secret sharing scheme is constructed from LDPC erasure codes, for which the required properties are for now only conjectured.

Additionally, their scheme is limited to the private tracing mode, since they extend a well-known primitive in traitor tracing called *Private Linear Broadcast Encryption* (PLBE) to support threshold decryption.
Therefore, it remains an open problem to construct a post-quantum secure threshold traitor tracing scheme that supports public traceability. This is precisely what we do in this paper.

This work. We extend the work of [15] by constructing a lattice-based *Q-Partite Threshold Encryption Scheme*. This building block is known to imply Threshold Traitor Tracing with *public traceability*. Our main technique is to employ a key-shifting mechanism on top of Regev's encryption scheme, thresholdised via $\{0,1\}$-Linear Secret Sharing. The final scheme is proven secure under the standard LWE assumption.

1.1 Related Works

Since its introduction by Boneh, Partap, Rotem in [9], several works have followed, improving their framework. The works in [15,11,16] achieve a functionality called public traceability. [11,16] additionally remove the trusted setup for generating users' private keys although their schemes are proven secure in the Generic Group Model. In [7], the authors extend the *BTKEM* from [9] to achieve CCA security. Starting from an Identity-Based *BTKEM*, their scheme additionally provides Identity-Based Encryption. The previously cited works are all based on pairing assumptions.

The only Threshold Traitor Tracing scheme based on a lattice assumption is [19][1]. In this work, the authors introduce a primitive named Attribute-Based Threshold Encryption (ABTE), which is an Attribute-Based Encryption scheme in which the decryption procedure is thresholdised. Then, they follow the blueprint of [22] for constructing traitor tracing scheme by combining their ABTE with mixed functional encryption. This gives a Private Linear Broadcast Threshold Encryption, the thresholdised version of Private Linear Broadcast Encryption. With this primitive in hand, they can finally construct their Th-TT scheme.

We follow a completely different approach since our traitor tracing scheme is strongly using the traceability property of robust IPP codes [4,21]. Another difference with our work lies in the choice of the secret sharing scheme. In [19], they use a *ramp secret sharing scheme* to achieve optimal share size. In this sharing scheme, there is a threshold for the security of the secret which is different from the threshold for reconstruction. Between these two bounds, a coalition of traitors can learn non-trivial information about the secret. In a Th-TT scheme,

[1] The authors also give a pairing-based construction for their new primitive, but we focus on the lattice construction in this overview.

the coalition of traitors is assumed to be larger than the decryption threshold. Therefore, there is a gap in which a coalition can learn non-trivial information about the secret being shared but can not be traced by the tracing algorithm. We use $\{0,1\}$-Linear Secret Sharing schemes ($\{0,1\} - LSSS$), which allows for exact threshold.

Another line of work focuses on constructing threshold primitives from lattice assumptions. In contrast with signatures, constructing satisfying public key encryption from lattices has been more challenging. One of the first works to achieve this is [5]. They construct the first full-threshold public key encryption from lattice assumptions. The authors provide a simulation-based proof of security of their scheme. In [8], they construct a threshold fully homomorphic encryption scheme. The threshold technique is possible thanks to their introduction of $\{0,1\}$ Linear Secret Sharing Scheme, which allows one to share a secret while keeping the reconstruction coefficients small (namely 0 and 1). While the class of $\{0,1\}$-LSSS scheme is not as large as the entire class of LSSS, it covers at least threshold access structure which is sufficient for practical scenarios. However, to argue the security of the partial decryption shares, the authors rely on a noise flooding technique which requires the modulus q to be superpolynomial. Several subsequent works have then reduced the size of the modulus q, such as [26,14,12,28]. However, they either only support full threshold or small threshold $t << N$. In [28], the authors use a non-trusted server such that ciphertexts are encrypted modulo an exponentially large modulus but the server floods and rounds (hence the name of their technique *double flood and round*) the ciphertext to switch the modulo down to a polynomial one.

More recently, in [18], the authors introduce a new assumption that they call *threshold LWE*. In this assumption, along with the LWE challenge, the adversary is also given a set of Shamir Shares of the LWE secret. Their new assumption, together with the use of subtractive sets over carefully chosen cyclotomic rings [1], allows for the modulus q to be polynomial. They additionally prove that their scheme is secure against a stronger type of adversary, which corrupts only a few parties and can also see some partial decryption of the challenge ciphertext.

In [24], followed by [13], the authors construct a threshold lattice KEM with short ciphertexts, based on NTRU trapdoors. Finally, [2] generalises the lattice splitting trapdoor techniques from [6]. Their technique allows them to build a threshold IBE scheme. However it seems incompatible with the construction of our Q-Partite Threshold Public Key Encryption (QTPKE) since the set of reconstruction needs to be known when sampling pre-images, which is impossible in our case. In fact, we would need to sample pre-images such that any arbitrary set of users satisfying the access policy can recover the secret, i.e., before knowing which party will take part in the final decryption procedure.

1.2 Overview of the QTPKE

Next, we recall at a high level how a QTPKE is designed. In the QTPKE, a number of positions ℓ and an alphabet size Q (which are respectively the length and the alphabet size of the IPP code (see definition sec. 5) used to construct

the Threshold Traitor Tracing scheme) define the number of public keys, secret keys and ciphertext size in the system. For each user $i \in [N]$, their secret key has the following form:

$$\begin{pmatrix} sk_{i,1}^1 \dots & sk_{i,\alpha}^1 \dots & sk_{i,Q}^1 \\ & \vdots & \\ sk_{i,1}^j \dots & sk_{i,\alpha}^j \dots & sk_{i,Q}^j \\ & \vdots & \\ sk_{i,1}^\ell \dots & sk_{i,\alpha}^\ell \dots & sk_{i,Q}^\ell \end{pmatrix}$$

Each row j of this matrix is associated with a public key pk_j, and each pk_j has itself Q sub public keys: $pk_j = (pk_{j,1}, \dots, pk_{j,Q})$.
Ciphertexts in the QTPKE have $Q + 1$ parts: $ct = (j, ct_0, ct_1, \dots, ct_Q)$ where ct_0 is the mask that hides the message and $ct_1, \dots, ct_Q$ are the Q parts of the ciphertext. (ct_0 corresponds to the session key in the threshold KEM version of the scheme). Additionally, a ciphertext is created with respect to a position $j \in [\ell]$. Given a position j and a ciphertext ct encrypted w.r.t j, user i can compute their partial decryption share using any pair $(ct_\alpha, sk_{i,\alpha}^j)$ of their choice. A key requirement in a QTPKE is that the partial decryption must be the same no matter which pair $(ct_\alpha, sk_{i,\alpha}^j)$ was used. Equivalently, when running the combination algorithm, the result should be the same no matter which pair was used by each user to compute their partial decryption. This property is called *all-sided correctness* and is essential for the correctness of the Threshold Traitor Tracing scheme, where users receive only one key per position, depending on their personal codeword. In fact, a QTPKE is only really useful when turned into a Th-TT scheme, since there is a lot of redundant information when users are given more than one key for the same position j.

1.3 From QTPKE to Th-TT

In [15], the authors show how to construct a TH-TT scheme using a robust IPP code and with a black box access to the QTPKE. In the TH-TT, the Encryption, Partial Decryption and Combination algorithms are exactly the same as the QTPKE. The only difference is in the Key Generation, where users receive a subset of the QTPKE. More precisely, a robust IPP code is defined over an alphabet $\Sigma = [Q]$ and codewords have length ℓ. Each user $i \in [N]$ is associated with a codeword $\omega_i = \omega_{i,1} \dots \omega_{i,\ell}$, and receives the set of secret key $(sk_{i,\omega_{i,1}}^1, \dots, sk_{i,\omega_{i,\ell}}^\ell)$. Note that in the Th-TT scheme, users' secret keys are significantly smaller (by a factor Q) than in the QTPKE.
Now, to trace a decoder D, the tracers proceed in two steps:

- Construct a pirate codeword ω^* using the decoder D and the QTPKE
- Use the Tracing procedure of the Robust IPP on ω^* to identify a traitor i that actively participated in crafting the decoder.

To construct the pirate codeword $\omega^* = \omega_1^* \dots \omega_\ell^*$, the tracer repeats for each $j \in [\ell]$ the following steps:

– Start from a normal ciphertext which is defined as $ct = (j, ct_0, ct_1 \ldots, ct_Q) \leftarrow QTPKE.\mathsf{Enc}(pk, j, \mu)$, use decoder D on input ct and count the number of correct guesses from D.
– Then, switch ct_Q to a random element, use decoder D on input the ciphertext $ct = (j, ct_0, ct_1, \ldots, ct_Q)$ and count the number of correct guesses from D.

  ⋮
– Switch ct_1 to a random element, use decoder D on input the cipjhertext $ct = (j, ct_0, ct_1, \ldots, ct_Q)$ and count the number of correct guesses from D.

During the first step, the input to the decoder is a real ciphertext. The number of correct guesses from D *must be high*. In the end, $ct_1, ct_2, \ldots, ct_Q$ are random hence independent from ct_0. The decoder answers randomly and the number of correct guesses *must be low*. It follows that there must be a step $\alpha \in [Q]$ at which the number of correct guesses must drop significantly (compared to every other step). This means that at iteration $\alpha - 1$ the decoder D was using some secret key $sk^j_{i,\alpha}$ of some traitor i, and at the next iteration when ct_α is switched to a random element this secret key has now become useless to the decoder. Therefore, the tracer can conclude that for the current position j, at least one of the traitor's letters in their personal codeword must be α, hence sets $\omega^*_j \leftarrow \alpha$.

The tracing heavily relies on a property called *Adaptive One-Sided Security* that is detailed in the next section. For the complete and formal definition of Robust IPP codes, decoder D and Th-TT construction, we refer the reader to [15]. We show in section 5 how to combine our QTPKE with robust IPP codes into a fully fledged threshold traitor tracing scheme that satisfies the public traceability requirement.

1.4 Technical Overview

In order to construct our QTPKE, as explained above, we opted to use the now standard $\{0,1\}$-Linear Secret Sharing Scheme from [8]. This scheme has a reconstruction algorithm with small coefficients. While not optimal, it is still relatively efficient when the number of users is not too large. To obtain our QTPKE, we shift each user's secret share by a vector associated with a different public key. This key-shifting mechanism allows us to build our primitive from Regev's thresholdised encryption scheme.

Technique. Let us start from the standard Regev encryption scheme. The public key is a matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{b} = \mathbf{A}^\top \mathbf{x} + \mathbf{e} \in \mathbb{Z}_q^m$, where $\mathbf{x} \leftarrow \mathbb{Z}_q^n$ is the secret key and $\mathbf{e} \leftarrow \mathcal{D}^m$. To encrypt a bit $\mu \in \{0,1\}$, the algorithm samples a short random vector $r \leftarrow \{0,1\}^m$ and creates the ciphertext $(j, \mathbf{ct}_r, ct_0) = (\mathbf{Ar}, \mathbf{b}^\top \mathbf{r} + \mu \lfloor \frac{q}{2} \rceil)$. Decryption works by computing the difference $ct_0 - \mathbf{x}^\top \mathbf{ct}_r$ and then error-correcting it. Correctness holds as long as $|ct_0 - \mathbf{x}^\top \mathbf{ct}_r| \leq q/4$.

A common technique for building threshold encryption from this scheme is to secret share the secret key $\mathbf{x}$ into shares $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Suppose for now that we use an N out of N secret sharing scheme, i.e., $\mathbf{x} = \sum_{[N]} \mathbf{s}_i$. Now, each user

can compute their partial decryption $pd_i = \mathbf{s}_i^\top \mathbf{ct}_r$. By adding all the partial decryption shares, we can decrypt : $ct_0 - \sum_{[N]} pd_i = \mathbf{x}^\top \mathbf{A}\ \mathbf{r} + \mathbf{e}^\top \mathbf{r} + \mu\lfloor \frac{q}{2} \rceil - \sum_{[N]} \mathbf{s}_i^\top \mathbf{A}\ \mathbf{r} = \mathbf{e}^\top \mathbf{r} + \mu\lfloor \frac{q}{2} \rceil$ since $\mathbf{x} = \sum_{[N]} \mathbf{x}_i$. However, as shown in [5], simply forwarding the share pd_i is insecure and can leak information on users' secret key and ciphertext noise. A simple solution to counter these leaks is to add a uniformly large noise to the partial decryption. This technique is referred to as adding *flooding* or *smudging* noise. However, it requires the modulus q to be exponentially large in the security parameter. Now, decryption shares are computed as follows: $pd_i = \mathbf{s}_i^\top \mathbf{ct}_r + \tilde{e}_{flood}$ where $\tilde{e}_{flood} \leftarrow [-B_{flood}, B_{flood}]$.

This first step gives us a simple threshold variant of Regev's encryption scheme. The next step is to extend the public key $(\mathbf{A}, \mathbf{b} = \mathbf{A}^\top \mathbf{x} + \mathbf{e})$ and secret keys $\mathbf{s}_1, \ldots, \mathbf{s}_N$ such that they have the desired form.

First, let us fix an index $j \in [\ell]$. While we will drop its notation for ease of exposition, the following public and secret keys all refer to the keys sampled with respect to position j. The full public and secret keys are then obtained by repeating the next steps for each position $j \in [\ell]$.

We start by sampling Q random vectors $\mathbf{y}_1, \ldots, \mathbf{y}_Q$ together with Q short errors $\mathbf{e}_1, \ldots, \mathbf{e}_Q$ and randomising the rows of A by computing:

$$\mathbf{pk}_1 = \mathbf{A}^\top \mathbf{y}_1 + \mathbf{e}_1, \ldots,\ \mathbf{pk}_Q = \mathbf{A}^\top \mathbf{y}_Q + \mathbf{e}_Q.$$

The public key now has the following structure: $pk = (\mathbf{A}, \mathbf{A}^\top \mathbf{x} + \mathbf{e}, \mathbf{A}^\top \mathbf{y}_1 + \mathbf{e}_1, \ldots, \mathbf{A}^\top \mathbf{y}_Q + \mathbf{e}_Q)$.

Next, for each user $i \in [N]$, we shift their share $\mathbf{s}_i$ by the randomising vector $\mathbf{y}_\alpha$ for each $\alpha \in [Q]$: $sk_i = (\mathbf{sk}_{i,1}, \ldots, \mathbf{sk}_{i,Q})$ where for each $\alpha \in [Q]$:

$$\mathbf{sk}_{i,\alpha} = \mathbf{s}_i - \mathbf{y}_\alpha.$$

where $\mathbf{s}_i$ is the share vector of user i.

Finally, we extend the threshold Regev's ciphertext by adding the rerandomisation of the public keys $\mathbf{pk}_1, \ldots, \mathbf{pk}_Q$ by sampling a short vector $\mathbf{r} \leftarrow \{0,1\}^m$ and computing:

$$\begin{aligned}
\mathbf{ct}_r &= \mathbf{Ar} \\
ct_0 &= \mathbf{x}^\top \mathbf{Ar} + \mathbf{e}^\top \mathbf{r} + \mu\lfloor \frac{q}{2} \rceil \\
ct_1 &= \mathbf{pk}_1^\top \mathbf{r} = \mathbf{y}_1^\top \mathbf{Ar} + \mathbf{e}_1^\top \mathbf{r} \\
&\vdots \\
ct_Q &= \mathbf{pk}_Q^\top \mathbf{r} = \mathbf{y}_Q^\top \mathbf{Ar} + \mathbf{e}_Q^\top \mathbf{r}
\end{aligned}$$

We slightly change the ciphertext structure of QTPKE/QTKEM by adding a component ct_r by setting the ciphertext as $ct = (j, \mathbf{ct}_r, ct_0, ct_1, \ldots, ct_Q)$. This $\mathbf{ct}_r$ component is necessary for the correctness of the scheme.

When user i computes their partial decryption, they can remove the y_α component from their secret key by adding ct_α as follows:

$$pd_i = \mathbf{sk}_{i,\alpha}^\top \mathbf{ct}_r + ct_\alpha + \tilde{e}_i$$

where $\tilde{e}_i \leftarrow [-B_{flood}, B_{flood}]$. The final decryption is the same as the threshold Regev encryption scheme i.e., the combiner error corrects $ct_0 - \sum_{[N]} pd_i$.

Security. There are 4 main security properties this scheme needs to satisfy:

- *All sided-correctness.* Correctness should hold no matter which pair $(ct_\alpha, sk_{i,\alpha})$ was used during the partial decryption.
- *Share simulatability.* The flooding noise added during decryption must prevent any information from the secret key and ciphertext noise to leak.
- *Cpa Security.* An adversary in possession of a smaller than threshold number of secret keys cannot distinguish the encryption of 0 from the encryption of 1 with non-negligible probability.
- *Adaptive one-sided Security.* An adversary cannot distinguish ct_α from a uniformly random element, even given the secret keys of every user $i \in [N]$ *except* the keys related to pk_α (i.e., the adversary receives $\{sk_{i,\alpha}\}_{\alpha[Q]\setminus\{\alpha^*\}}$ for every $i \in [N]$ and some adversarially chosen α^*.

The all sided-correctness follows from the choice of our parameters. One can note that the noise from the partial decryption comes from the ct_α used, and that the inner product $\mathbf{sk}_{i,\alpha}^\top \mathbf{ct}_r$ is not noisy. By the Share simulatability property, the noise from ct_α should be hidden by the flooding term $\tilde{e}_i$. During the final decryption, the noise growth is linear in the reconstruction set size:

$$\sum_T pd_i = \sum_T \mathbf{s}_i^\top \mathbf{A}\mathbf{r} + \sum_T \mathbf{e}_{\alpha_i}^\top \mathbf{r} + \sum_T \tilde{e}_i \tag{1}$$

Hence, the parameters of the scheme can be optimised for the special case of small threshold.

The share simulatability is a property common to most lattice-based threshold PKE/FHE, and our proof follows the general template of other constructions using the same secret sharing scheme as ours [8].

Next, the CPA security proof strategy follows from a standard reduction to LWE: we start by switching the public key $\mathbf{A}^\top\mathbf{x}+\mathbf{e}$ to a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^m$. This is possible because the adversary is not allowed to receive enough secret keys to decrypt. This requires the underlying secret sharing scheme to have a property preventing small coalitions from distinguishing honestly generated shares from uniformly sampled ones. This is a standard property required by almost every secret sharing scheme.

Finally, for the all-sided correctness we also rely on the LWE assumption to switch the public key component $\mathbf{pk}_{\alpha^*}$[2] to a random vector $\mathbf{u} \leftarrow \mathbb{Z}_q^m$ sampled uniformly at random. This is even easier than for the CPA security, since the adversary receives no secret key related to pk_{α^*}. However, since the adversary receives all other secret keys of the system, we need to carefully sample the secret keys $\{sk_{i,\alpha}\}_{\alpha\neq\alpha^*}$ and the challenge ciphertext components $\{ct_\alpha\}_{\alpha<\alpha^*}$ such that correctness of the scheme still holds if the adversary uses these parts to run the partial decryption and final decryption algorithms. We show that these properties are satisfied by our QTPKE in section 4.1.

[2] α^* being the letter chosen by the adversary for the challenge.

2 Preliminaries

We start by giving some preliminaries about linear secret sharing and lattices. We also introduce our new assumption, called LWE with orthogonal hints.

2.1 Linear Secret Sharing

Definition 1 (Monotone Access Structure). *Let $\mathcal{P} = (P_1, \ldots, P_N)$ be a set of parties and $2^{\mathcal{P}}$ its power set. A monotone access structure is a collection of sets $\mathbb{A} \subseteq 2^{\mathcal{P}}$, such that for any $S \in \mathbb{A}$, if $T \supset S$ then $T \in \mathbb{A}$. We say that $\mathbb{A}$ is efficient if membership of $\mathbb{A}$ can be verified in time poly(λ), where $\mathbb{A}$ is viewed as a function of λ.*

We only consider efficient access structures. We identify party P_i with its index $i \in [N]$, viewing each set $S \in \mathbb{A}$ as a subset of $[N]$. For any $S \subset [N]$, and vector $v = (v_1, \ldots, v_N)$, let $v|_S$ denote the vector of shares restricted to v_i for $i \in S$. For the definitions of Linear Secret Sharing and Linear Secret Sharing with Strong $\{0,1\}$-Reconstruction, we follow the notations of [8,14].

Definition 2 (Linear Secret Sharing Scheme). *Let q, L, N be positive integers and $\mathbb{A}$ a monotone access structure. A linear secret sharing scheme LSSS for $\mathbb{A}$ is defined by a randomised algorithm $\mathsf{Share} : \mathbb{Z}_q \to (\mathbb{Z}_q^L)^N$ and a family of deterministic algorithms $\mathsf{Combine}_S : (\mathbb{Z}_q^L)^{|S|} \to \mathbb{Z}_q$, for $S \subseteq [N]$, which satisfy:*
***Privacy**: For any set $S \notin \mathbb{A}$, any $x, x' \in \mathbb{Z}_q$ and $v \in (\mathbb{Z}_q^L)^{|S|}$, it holds that*

$$Pr\left[\mathsf{Share}(x)|_S = v\right] = Pr\left[\mathsf{Share}(x')|_S = v\right].$$

***Reconstruction**: For any set $S \in \mathbb{A}$, any $x \in \mathbb{Z}_q$ and $v = \mathsf{Share}(x)$, the reconstruction algorithm outputs $\mathsf{Combine}_S(v|_S) = x$.*
***Linearity**: For any $\alpha, \beta \in \mathbb{Z}_q$, any set S with $S \in \mathbb{A}$, and any share vectors u, v, it holds that:*

$$\mathsf{Combine}_S(\alpha u|_S + \beta v|_S) = \alpha \mathsf{Combine}_S(u|_S) + \beta \mathsf{Combine}_S(v|_S)$$

When the set of shares $S = [N]$, we simply write $\mathsf{Combine}$ instead of $\mathsf{Combine}_{[N]}$

We also need to define the notion of valid and invalid share sets, as follows.

Definition 3. *Let $x \in \mathbb{Z}_q$, $(v_1, \ldots, v_N) = \mathsf{Share}(x)$ and write $v_i = (v_{i,1}, \ldots, v_{i,L})$. A set of pairs of indices $T \subseteq [N] \times [L]$ is an invalid set of share elements if the corresponding shares $(v_{i,l})_{(i,l)\in T}$ reveal no information about x. Otherwise, we say that T is a valid set of share elements, and any such set can be used to recover x. Additionally, we say that:*

- *$T \subseteq [N] \times [L]$ is a maximal invalid set of share elements if it is invalid but for any pair $(i,l) \in [N] \times [L] \backslash T$, the set $T \cup \{(i,l)\}$ is a valid set of share elements. We denote by τ_{max} the smallest maximal invalid set of share elements.*

– $T \subseteq [N] \times [L]$ *is a minimal valid set of share elements if it is valid, but for any* $T' \subsetneq T$*, the set* T' *is an invalid set of share elements. We denote by* τ_{min} *the largest minimal valid set of share elements.*

As in [14], we also require that the reconstruction function $\mathsf{Combine}_S$ takes a 0/1 combination of its inputs, for any valid set of share elements. This property is given by the following definition.

Definition 4 (Strong $\{0,1\}-$*Reconstruction* **[8]).** *We say that a LSSS has strong* $\{0,1\}-$*Reconstruction if for any secret* x *and* $(v_1, \ldots, v_N) = \mathsf{Share}(x)$*, for any valid set of share elements* $T \subseteq [N] \times [L]$*, there exists a subset* $T' \subseteq T$ *such that* $\sum_{(i,l)\in T'} v_{i,l} = x$ *where* $v_i = (v_{i,1}, \ldots, v_{i,L})$*.*

Sharing vectors in $\mathbb{Z}_q^n$**.** In our application of $LSSS$, we will always be sharing a vector $\mathbf{p} \in \mathbb{Z}_q^n$ instead of a single scalar in $\mathbb{Z}_q$. We do it component-wise, meaning separately sharing each component of the vector $\mathbf{p}$ using fresh randomness. Each party's share $\mathbf{p}_i$, $i \in [N]$ then lies in $(\mathbb{Z}_q^n)^L$. It is easy to see that (i) the secret $\mathbf{p} \in \mathbb{Z}_q^n$ can be reconstructed as a linear combination of the shares $\mathbf{p}_i$ using the same coefficients as for a single field element and (ii) the privacy property of the scheme is maintained.

2.2 Statistical Distance and Smudging lemma

Throughout this paper, let $\lambda \in \mathbb{N}$ denote the security parameter and $\mathsf{negl}(\lambda)$ be a negligible function. For $q = q(\lambda)$, $\mathbb{Z}_q$ is the set of integer reduced modulo q. We treat vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ as column vectors in $\mathbb{Z}_q^{n\times 1}$. We use the notation $x \leftarrow \mathcal{D}$ to denote that the element x is sampled randomly from the distribution $\mathcal{D}$. Next, we introduce the statistical distance and the smudging lemma.

Definition 5 (Statistical distance). *Let* X, Y *be two distributions defined over a finite domain* D*. The* ***statistical distance*** *between* X *and* Y *is defined as:*

$$\Delta(X,Y) = \frac{1}{2}\sum_{\alpha\in D} |Pr[X=\alpha] - Pr[Y=\alpha]|$$

We say that two distributions X*,* Y*, indexed by an index* $n \in \mathbb{N}$ *are statistically close if their statistical distance is negligible in* n*.*

Lemma 1 (Smudging Lemma [27]). *Let* $B_1 = B_1(\lambda)$*,* $B_2 = B_2(\lambda)$ *be positive integers and let* $e_1 \in [-B_1, B_1]$ *be a fixed integer. Let* $e_2 \leftarrow [-B_2, B_2]$ *be chosen uniformly at random. Then the distribution of* $e_2 + e_1$ *is statistically close to that of* e_2 *as long as* $B_1/B_2 = \mathsf{negl}(\lambda)$*.*

2.3 Lattices preliminaries

As it is usually done, we write $D_{\mathbb{Z},\sigma}$, to denote the discrete Gaussian distribution over $\mathbb{Z}$ with width parameter $\sigma > 0$. We usually represent by boldfaced capital letters matrices such as $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and boldfaced letter vectors such as $\mathbf{x} \in \mathbb{Z}_q^n$, and we use non-boldfaced letters for their components (ex $x_1, \ldots, x_n$ coordinates of $\mathbf{x}$). We also recall some basic useful properties of the discrete Gaussian distribution, as well as the leftover hash lemma [23].

Lemma 2 (Leftover Hash Lemma, adapted). *Let n, m, q be integers such that $m \geq 2n \log q$ and $q > 2$ is prime. Then, for all $\ell = \mathsf{poly}(n)$, the statistical distance between the following distributions is $\mathsf{negl}(\lambda)$:*

$$\left\{(A, AZ) \ : \ A \leftarrow \mathbb{Z}_q^{n \times m}, \ Z \leftarrow \{0,1\}^{m \times \ell}\right\}$$
$$\left\{(A, U) \ : \ A \leftarrow \mathbb{Z}_q^{n \times m}, \ U \leftarrow \mathbb{Z}_q^{n \times \ell}, \right\}$$

Lemma 3 (Gaussian Tail Bound [25], adapted). *Let n, m, q be lattice parameters where $m \geq 2n \log q$. Sample $A \leftarrow \mathbb{Z}_q^{n \times m}$. Then for all $\sigma > \log n$, $t \in \mathbb{Z}_q^n$ in the span of A, we have that*

$$Pr\left[\|u\| > \sqrt{n}\sigma \ : \ u \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m \ s.t. \ Au = t\right] \leq O(2^{-n}).$$

In particular for discrete Gaussian over the integers and any $\lambda \in \mathbb{N}$,

$$Pr\left[|x| > \sqrt{\lambda}\sigma \ : \ x \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}\right] \leq 2^{-\lambda}.$$

Definition 6 (Learning with errors [29].). *Let λ be a security parameter and $n = n(\lambda), \ m = m(\lambda), \ q = q(\lambda), \ \sigma = \sigma(\lambda)$ be lattice parameters. For $b \in \{0,1\}$ and any PPT algorithm $\mathcal{A}$, we define the LWE following experiment:*
$\mathsf{Exp\text{-}LWE}_{\mathcal{A}}^b(1^\lambda)$:

- $A \leftarrow \mathbb{Z}_q^{n \times m}$
- $s \leftarrow \mathbb{Z}_q^n, \ e \ \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$
- *if $b = 0$ then $u^\top = s^\top A + e^\top$*
- *if $b = 1$ then $u \ \leftarrow \mathbb{Z}_q^m$*
- *return $\mathcal{A}(A, u)$*

The (decision) LWE *assumption states that for any PPT $\mathcal{A}$, it holds that:*

$$\mathsf{Adv}_{n,m,q,\sigma,\mathcal{A}}^{\mathsf{LWE}}(1^\lambda) = |Pr\left[\mathsf{Exp\text{-}LWE}^0(1^\lambda) = 1\right] - Pr\left[\mathsf{Exp\text{-}LWE}^1(1^\lambda) = 1\right]| = negl(\lambda)$$

3 Q-Partite Threshold PKE

In this section, we recall the definition of Q-Partite Threshold Public Key Encryption (QTPKE), that has been introduced in [15]

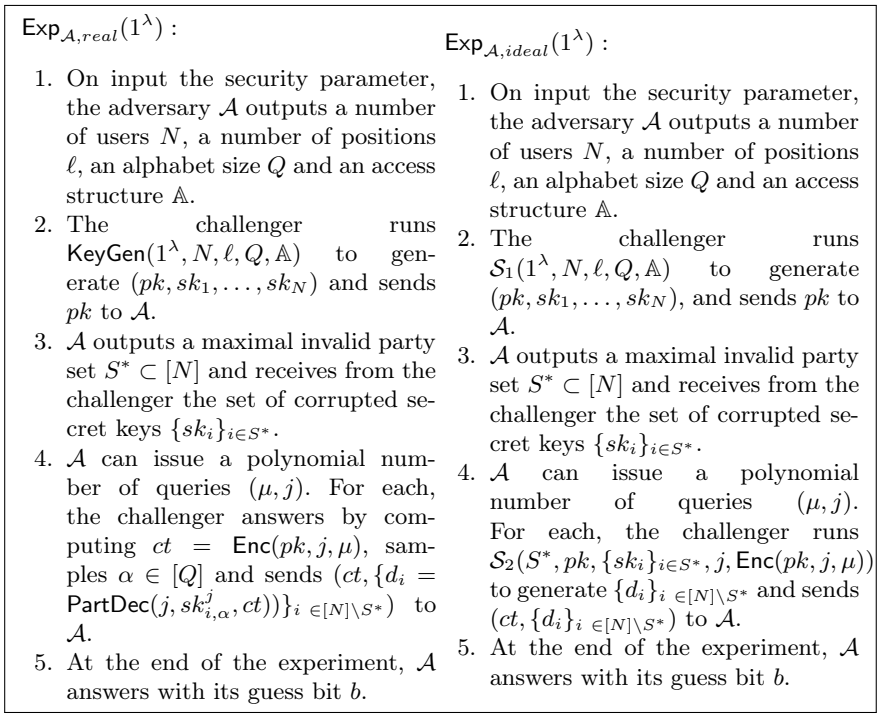

Fig. 1. Share simulatability for QTPKE adversary.

3.1 Main Definition

Definition 7. *Let $\lambda \in \mathbb{N}$ be the security parameter, $N \in \mathbb{N}$ the number of users, $\ell \in \mathbb{N}$ a number of positions, $Q \in \mathbb{N}$ an alphabet size and $\mathbb{A}$ a monotone access structure. A QTPKE $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine})$ is composed of the following algorithms:*

- $\Pi.\mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$*: Takes as input a security parameter λ, the number of users N, a number ℓ of positions, Q an alphabet size and an access structure $\mathbb{A}$. It outputs a public key pk and the set of users secret key $(sk_1, \ldots, sk_N)$, where for each $i \in [N]$, $sk_i = \{sk^j_{i,\alpha}\}_{(j,\alpha)\in[\ell]\times[Q]}$.*
- $\Pi.\mathsf{Enc}(pk, j, \mu)$*: Takes as input a public key pk, a position $j \in [\ell]$ and a message $\mu \in \{0,1\}$, and outputs a ciphertext $ct = (j, ct_0, ct_1, \ldots, ct_Q)$.*
- $\Pi.\mathsf{PartDec}(j, sk^j_{i,\alpha}, ct)$*: Takes as input a position $j \in [\ell]$, a user secret key $sk^j_{i,\alpha}$ with $i \in [N]$ and $\alpha \in [Q]$ and a ciphertext $ct = (j, ct_0, ct_1, \ldots, ct_Q)$. It outputs d_i the decryption share of user i.*
- $\Pi.\mathsf{Combine}(pk, ct, j, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}})$*: Takes as input the public key pk, a ciphertext $ct = (j, ct_0, ct_1, \ldots, ct_Q)$, a position $j \in [\ell]$, a subset of users $\mathcal{J} \subseteq [n]$, and a set of decryption shares $\{d_i\}_{i\in\mathcal{J}}$, and outputs a message $\mu \in \{0,1\}$.*

3.2 Security Properties

We require that $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine})$ satisfies the following properties:

- **All-sided correctness:** We say that a QTPKE Π has all-sided correctness if for a security parameter $\lambda \in \mathbb{N}$, parameters $N = N(\lambda),\ \ell = \ell(\lambda),\ Q = Q(\lambda)$, an access structure $\mathbb{A}$ and a set $\mathcal{J} \in [N],\ s.t\ \mathcal{J} \in \mathbb{A}$, we have that:

$$Pr\left[\mathsf{Combine}(pk, ct, j, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}}) = \mu\right] \geq 1 - negl(\lambda)$$

  where $(pk, sk_1, \ldots, sk_N) = \mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A}),\ \forall j \in [\ell]\ \mu \in \{0,1\},\ ct = \mathsf{Enc}(pk, j, \mu)$ and $\forall i \in \mathcal{J},\ \alpha \in [Q],\ d_i = \mathsf{PartDec}(j, sk^j_{i,\alpha}, ct)$.
- **CPA Security:** We say that a QTPKE Π is CPA secure if all probabilistic polynomial time algorithm $\mathcal{A}$ have at most negligible advantage in the game in figure 2, where $\mathcal{A}$'s advantage is defined as $\mathsf{Adv}^{\mathsf{CPA}}_{\Pi}(\mathcal{A}) = \left|\mathsf{Succ}^{\mathsf{CPA}}(\mathcal{A}) - \frac{1}{2}\right|$ and $\mathsf{Succ}^{\mathsf{CPA}}(\mathcal{A}) = Pr\left[b = b'\right]$ is the probability that the adversary outputs the correct bit in the game in figure 2.
- **Partial Decryption simulatability:** We say that a QTPKE Π has partial decryption simulatability if there exist a pair of probabilistic polynomial time algorithm $(\mathcal{S}_1, \mathcal{S}_2)$ such that all probabilistic polynomial time algorithm $\mathcal{A}$ have negligible advantage in distinguishing the experiments $\mathsf{Exp}_{\mathcal{A},real}(1^\lambda)$ and $\mathsf{Exp}_{\mathcal{A},ideal}(1^\lambda)$[3] defined in figure 1.
- **Adaptive One-Sided Security:** We say that a QTPKE Π is adaptively one-sided secure if all probabilistic polynomial time algorithm $\mathcal{A}$ have at most negligible advantage in the game in figure 3, where $\mathcal{A}$'s advantage is defined as $\mathsf{Adv}^{ad-OSS}_{\Pi}(\mathcal{A}) = \left|\mathsf{Succ}^{ad-OSS}(\mathcal{A}) - \frac{1}{2}\right|$ and $\mathsf{Succ}^{ad-OSS}(\mathcal{A}) = Pr\left[b = b'\right]$ is the probability that the adversary outputs the correct bit in the game in figure 3.

4 Our Post-Quantum QTPKE Construction

In this section, we present our core result, namely the first QTPKE that is post-quantum secure. We then prove its security.

4.1 Description of Our Scheme

Let $\lambda \in \mathbb{N}$ be a security parameter, $N \in \mathbb{N}$ the number of users, $n = n(\lambda, N), m = m(\lambda, N), q = q(\lambda, N), \sigma = \sigma(\lambda, N)$ be lattice parameters. We also let $B_{flood} =$

[3] Note that in the experiment we require $\mathcal{A}$ to output a maximal invalid party set in order to allow $\mathcal{S}_2$ to be able to compute the decryption share of an honest user using only information known to the adversary i.e secret keys of corrupted users. If the set was not maximal, this would not be possible.

1. The adversary outputs a number of parties N a length ℓ, an alphabet size Q, and an access structure $\mathbb{A}$ given a security parameter λ.
2. The challenger runs $\mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$ to generate $(pk, sk_1, \ldots, sk_N)$ and sends pk to the adversary.
3. The adversary can corrupt some subsets of parties by outputting a set S.
4. If $S \in \mathbb{A}$, the challenger stops the game. Otherwise, the challenger samples a position $j^* \leftarrow [\ell]$, a bit $b \leftarrow \{0,1\}$, computes a challenge ciphertext $ct_b \leftarrow \mathsf{Enc}(pk, j^*, b)$ and sends ct_b to the adversary.
5. Finally, $\mathcal{A}$ returns its guess $b' \in \{0,1\}$. It wins this game if $b = b'$.

Fig. 2. CPA security for QTKEM and adversary $\mathcal{A}$

1. The adversary, given a security parameter λ, outputs a number of parties N, a number of positions ℓ, an integer Q and an access structure $\mathbb{A}$ and sends them to the challenger.
2. The challenger runs the $\mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$ and outputs $(pk, \{(sk^j_{i,1}, \ldots, sk^j_{i,Q})_{i\in[N]}\}_{j\in[\ell]})$ it sends pk to the adversary.
3. The adversary outputs a position j^*, a letter α^* and sends them to the challenger.
4. The challenger samples $b \in \{0,1\}$, $ct^{(0)} \leftarrow \mathsf{Enc}(pk, j^*, 0)$ and $ct^{(1)} \leftarrow \mathbb{Z}_q^m \times (\mathbb{Z}_q)^{Q+1}$ and generates the challenge ciphertext $ct^* = (j^*, \mathbf{ct}_r^{(0)} ct_1^{(0)}, \ldots, ct_{\alpha^*-1}^{(0)}, ct_{\alpha^*}^{(b)}, ct_{\alpha^*+1}^{(1)}, \ldots, ct_Q^{(1)})$. Then, the challenger sends to the adversary $ct^*, \{(sk^j_{i,1}, \ldots, sk^j_{i,Q})_{i\in[N]}\}_{j\in[\ell]\setminus\{j^*\}}$ as well as $(sk^{j^*}_{i,1}, \ldots, sk^{j^*}_{i,\alpha^*-1}, sk^{j^*}_{i,\alpha^*+1}, sk^{j^*}_{i,Q})_{i\in[N]}$
5. Finally, the adversary outputs its guess bit $b' \in \{0,1\}$. The adversary wins this game if $b = b'$.

Fig. 3. $\mathsf{ad}-\mathsf{OSS}$ for QTPKE and adversary $\mathcal{A}$

$B_{flood}(\lambda, N)$ be a norm bound. Let $\mathbb{A}$ be an access structure, $\ell = \ell(\lambda, N), Q = Q(\lambda, N)$ be the number of positions and letters of the QTPKE. We omit the dependence of the ciphertext on $j \in [\ell]$ but it is assumed that it is public information attached to each ciphertext.
We construct a QTPKE scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine})$ as follows:

- $\Pi.\mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$: On input the security parameter λ, number of users N, number of positions ℓ, number of letters Q and access structure $\mathbb{A}$, the key generation algorithm proceeds as follows:
  - Sample $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{x}_j \leftarrow \mathbb{Z}_q^n$ for each position $j \in [\ell]$.
  - For each position $j \in [\ell]$, generate the shares of $\mathbf{x}_j$ for the access control $\mathbb{A}$ as $(\mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,N}) = \mathsf{SS.Share}(\mathbf{x}_j, \mathbb{A})$, where $\mathbf{s}_{j,i} = (\mathbf{s}_{j,i,1}, \ldots, \mathbf{s}_{j,i,L})$. Additionally, sample vectors $\mathbf{y}_{j,1}, \ldots, \mathbf{y}_{j,Q} \leftarrow \mathbb{Z}_q^n$.
  - For each $j \in [\ell]$, set the public key pk_j as $pk_j = (\mathbf{A}_j, \mathbf{A}_j^\top \mathbf{x}_j + \mathbf{e}_0, \mathbf{A}_j^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}_j^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$.
  - For each user $i \in [N]$, set sk_i as: $sk_i = \{\mathbf{sk}^j_{i,\alpha,l}\}_{(j,\alpha,l)\in[\ell]\times[Q]\times[L]}$ where $\forall (j,\alpha,l) \in [\ell]\times[Q]\times[L]$, $\mathbf{sk}^j_{i,\alpha,l} = \mathbf{s}_{j,i,l} - \mathbf{y}_{j,\alpha} \in \mathbb{Z}_q^n$.

Output $pk = (pk_1, \ldots, pk_\ell)$ and $sk_1, \ldots, sk_N$.

- $\Pi.\mathsf{Enc}(pk, j, \mu)$: On input the public key pk, a position $j \in [\ell]$ and a message $\mu \in \{0,1\}$, the encryption algorithm samples $\mathbf{r} \leftarrow \{0,1\}^m$ and computes:
  - $\mathbf{ct}_r = \mathbf{A}_j \mathbf{r}$
  - $ct_0 = \left(\mathbf{x}_j^\top \mathbf{A}_j + \mathbf{e}_0^\top\right) \mathbf{r} + \mu \lfloor \frac{q}{2} \rceil$
  - $\forall \alpha \in [Q],\ ct_\alpha = (\mathbf{y}_{j,\alpha}^\top \mathbf{A}_j + \mathbf{e}_\alpha^\top)\mathbf{r}$

  It outputs $ct = (j, \mathbf{ct}_r, ct_0, ct_1, \ldots, ct_Q)$.
- $\Pi.\mathsf{PartDec}(j, \mathbf{sk}_{i,\alpha}^j, ct)$: On input a position $j \in [\ell]$, a secret key $\mathbf{sk}_{i,\alpha}^j$ and a ciphertext ct parsed as $ct = (j, \mathbf{ct}_r, ct_0, ct_1, \ldots, ct_Q)$, the algorithm samples $\tilde{e}_i \leftarrow [-B_{flood}, B_{flood}]$, and computes for $l \in [L]$:

$$d_{i,l} = ct_\alpha + {\mathbf{sk}_{i,\alpha,l}^j}^\top \mathbf{ct}_r + \tilde{e}_{i,l}.$$

  It outputs $d_i = (d_{i,1}, \ldots, d_{i,L})$.
- $\Pi.\mathsf{Combine}(pk, ct, j, \mathcal{J}, \{d_i\}_{i \in \mathcal{J}})$: On input the public key $pk = (pk_1, \ldots, pk_\ell)$, the ciphertext $ct = (j, \mathbf{ct}_r, ct_0, ct_1, \ldots, ct_Q)$, a position $j \in [\ell]$, a set of parties $\mathcal{J} \subseteq [N]$ s.t. $\mathcal{J} \in \mathbb{A}$ and their set of decryption shares $\{d_i\}_{i \in \mathcal{J}}$, the combine algorithm computes

$$z = ct_0 - \mathsf{SS.Combine}(\{d_i\}_{i \in \mathcal{J}}) \tag{2}$$

  and outputs $\lfloor z \rceil$ where $\lfloor z \rceil$ equals 0 if $-q/4 \leq z < q/4$ and 1 otherwise.

4.2 Correctness

Theorem 1 (Correctness). *Suppose the modulus q is prime such that $q \geq 4(1+NL)\sqrt{nm}\sigma + 4NLB_{flood}$, $m \geq 2n \log q$, then Π has all-sided correctness.*

Proof. Let $(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$. Parse $pk = (pk_1, \ldots, pk_\ell)$ where for each $j \in [\ell]$, $pk_j = (\mathbf{A}_j, \mathbf{A}_j^\top \mathbf{x}_j + \mathbf{e}_0, \mathbf{A}_j^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}_j^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$, and for each $i \in [N]$, $sk_i = \{\mathbf{sk}_{i,\alpha,l}^j\}_{(j,\alpha,l) \in [\ell] \times [Q] \times [L]}$.
Let us fix some position $j \in [\ell]$ and plaintext $\mu \in \{0,1\}$. Let $\mathcal{J} \subseteq [N]$ be a valid set i.e $\mathcal{J} \in \mathbb{A}$. Let $ct = (j, \mathbf{ct}_r, ct_0, \ldots, ct_Q)$ be the output of the encryption algorithm $\Pi.\mathsf{Enc}(pk, j, \mu)$ with $\mathbf{ct}_r = \mathbf{A}_j \mathbf{r}$, $ct_0 = \mathbf{x}_j^\top \mathbf{A}_j \mathbf{r} + \mathbf{e}_0^\top \mathbf{r} + \mu \lfloor \frac{q}{2} \rceil$ and $ct_\alpha = \mathbf{y}_{j,\alpha}^\top \mathbf{A}_j \mathbf{r} + \mathbf{e}_\alpha^\top \mathbf{r}$, with $\mathbf{r} \leftarrow \{0,1\}^m$.
For each $i \in \mathcal{J}$, consider the output of the partial decryption algorithm $d_i = \Pi.\mathsf{PartDec}(j, \mathbf{sk}_{i,\alpha}^j, ct)$. Let $\tilde{e}_{i,l} \leftarrow [-B_{flood}, B_{flood}]$ be the flooding error terms sampled during partial decryption. We have that:

$$\begin{aligned} d_{i,l} &= ct_\alpha + {\mathbf{sk}_{i,\alpha,l}^j}^\top \mathbf{ct}_r + \tilde{e}_{i,l} \\ &= \mathbf{y}_{j,\alpha}^\top \mathbf{A}_j \mathbf{r} + \mathbf{e}_\alpha^\top \mathbf{r} + (\mathbf{s}_{j,i,l} - \mathbf{y}_{j,\alpha})^\top \mathbf{A}_j \mathbf{r} + \tilde{e}_{i,l} \\ &= \mathbf{s}_{j,i,l}^\top \mathbf{A}_j \mathbf{r} + \mathbf{e}_\alpha^\top \mathbf{r} + \tilde{e}_{i,l} \end{aligned}$$

Now, consider the output of $\Pi.\mathsf{Combine}(pk, ct, j, \mathcal{J}, \{d_i\}_{i \in \mathcal{J}})$. Let T be a minimal valid set of share elements $\{\mathbf{s}_{j,i,l}\}_{(i,l) \in T}$. By the strong $\{0,1\}$-reconstruction

property of the LSS and the validity of T, we have that $\sum_{(i,l)\in T} \mathbf{s}_{j,i,l} = \mathbf{x}_j$. It follows that:

$$\begin{aligned}\mathsf{Combine}(\{d_i\}_{i\in\mathcal{J}}) &= \sum_{(i,l)\in T} (\mathbf{s}_{j,i,l}^\top \mathbf{A}_j \mathbf{r} + \ \mathbf{e}_\alpha^\top \mathbf{r} + \tilde{e}_{i,l}) \\ &= \sum_{(i,l)\in T} \mathbf{s}_{j,i,l}^\top \mathbf{A}_j \mathbf{r} + \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} + \sum_{(i,l)\in T} \tilde{e}_{i,l}. \\ &= \mathbf{x}_j^\top \mathbf{A}_j \mathbf{r} + \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} + \sum_{(i,l)\in T} \tilde{e}_{i,l}\end{aligned}$$

Since $z = ct_0 - \mathsf{SS.Combine}(\{d_i\}_{i\in\mathcal{J}}$, we get that:

$$z = \mathbf{x}_j^\top \mathbf{A}_j \mathbf{r} + \mathbf{e}_0^\top \mathbf{r} + \mu\lfloor\frac{q}{2}\rceil - \mathbf{x}_j^\top \mathbf{A}_j \mathbf{r} - \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} - \sum_{(i,l)\in T} \tilde{e}_{i,l}$$

$$z = \mu\lfloor\frac{q}{2}\rceil + \mathbf{e}_0^\top \mathbf{r} - \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} - \sum_{(i,l)\in T} \tilde{e}_{i,l}.$$

It now suffices to show that $|\mathbf{e}_0^\top \mathbf{r} - \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} - \sum_{(i,l)\in T} \tilde{e}_{i,l}| \leq q/4$. We show that this inequality holds with overwhelming probability.

- Since Enc samples $\mathbf{r} \leftarrow \{0,1\}^m$ and KeyGen samples $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_Q \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, by Lemma 3 with overwhelming probability we have that $|\mathbf{e}_\alpha^\top \mathbf{r}| \leq \sqrt{mn}\sigma$ for each $\alpha \in [0, Q]$.
- By definition a valid set of shares elements, $|T| \leq NL$.

It follows that:

$$|\mathbf{e}_0^\top \mathbf{r} - \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} - \sum_{(i,l)\in T} \tilde{e}_{i,l}| \leq (1 + NL)\sqrt{nm}\sigma + NLB_{flood}$$

Since we have supposed that $q \geq 4(1 + NL)\sqrt{nm}\sigma + 4NLB_{flood}$, it follows that $|\mathbf{e}_0^\top \mathbf{r} - \sum_{(i,l)\in T} \mathbf{e}_{i,\alpha}^\top \mathbf{r} - \sum_{(i,l)\in T} \tilde{e}_{i,l}| \leq q/4$ which concludes the correctness.

4.3 CPA security

We now prove the security of our construction, starting from the CPA security, as defined in Section 2. In the proof, the adversary can not corrupt a valid set of users. Therefore, after arguing that the shares of $\mathbf{x}_j$ are uniformly distributed, the proof follows the template of Regev's PKE. After using the LWE assumption to switch the public key component $\mathbf{A}_j^\top \mathbf{x}_j + \mathbf{e}_0$ to a random vector, we use the standard leftover hash lemma to switch the ciphertext components $\mathbf{ct}_r$ (resp ct_0) to a random vector (resp scalar).

Theorem 2 (CPA security). *Let λ be a security parameter, $N = N(\lambda)$, $\ell = \ell(\lambda)$, $Q = Q(\lambda)$. Suppose that $n \geq \lambda$, $m \geq 2n \log q$, $\sigma \geq \frac{2\sqrt{n}}{q}$. Under the LWE assumption with parameters (n, m, q, σ), Π is CPA secure.*

Proof. We prove the CPA security by a sequence of hybrids.

- $\mathsf{H}_1^{(b)}$: This is the initial CPA security game. At the beginning, $\mathcal{A}$ sends $N, \ell, Q, \mathbb{A}$ to the challenger $\mathcal{C}$. The challenger samples $(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$, with $pk = (pk_1, \ldots, pk_\ell)$ where for $j \in [\ell]$, $pk_j = (\mathbf{A}_j, \mathbf{A}_j^\top \mathbf{x}_j + \mathbf{e}_0, \mathbf{A}_j^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}_j^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$. Then, the challenger secret shares x_j into $(\mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,N}) = \mathsf{SS.Share}(\mathbf{x}_j, \mathbb{A})$ where for each $i \in [N], j \in [\ell], \mathbf{s}_{j,i} = (\mathbf{s}_{j,i,1}, \ldots, \mathbf{s}_{j,i,L})$ then it sets for each $i \in [N]$, $sk_i = \{\mathbf{sk}^j_{i,\alpha,l}\}_{(j,\alpha,l)\in[\ell]\times[Q]\times[L]}$ with $\mathbf{sk}^j_{i,\alpha,l} = \mathbf{s}_{j,i,l} - \mathbf{y}_{i,\alpha}$. Then the challenger sends pk to $\mathcal{A}$, which declares a set of parties $S \subseteq [N]$. If $S \in \mathbb{A}$, the challenger stops the game.
  Next, the challenger samples a bit $b \in \{0,1\}$ and a position $j^* \in [\ell]$, computes $ct_b = (j, \mathbf{ct}_r, ct_0, \ldots, ct_Q) \leftarrow \mathsf{Enc}(pk, j^*, b)$. More precisely, it samples $\mathbf{r} \leftarrow \{0,1\}^m$ and computes $(j, \mathbf{ct}_r, ct_0, \ldots, ct_Q)$ as:

$$\begin{aligned} \mathbf{ct}_r &= \mathbf{A}_{j^*}\mathbf{r}, \\ ct_0 &= \mathbf{x}_{j^*}^\top \mathbf{A}_{j^*}\mathbf{r} + \mathbf{e}_0^\top \mathbf{r} + b\lfloor \frac{q}{2} \rceil, \\ \forall \alpha \in [Q],\ ct_\alpha &= \mathbf{y}_{j^*,\alpha}^\top \mathbf{A}\mathbf{r} + \mathbf{e}_\alpha^\top \mathbf{r}. \end{aligned}$$

  Finally, it sends $(ct_b, j^*, \{sk_i\}_{i\in S})$ to the adversary $\mathcal{A}$. At the end of the experiment, the adversary outputs a bit $b' \in \{0,1\}$.
- $\mathsf{H}_2^{(b)}$: Same as $\mathsf{H}_1^{(b)}$ except the challenger now does a secret sharing of $\mathbf{0}$ instead of $\mathbf{x}_j$, that is for each $j \in [\ell]$, $(s_{j,1}, \ldots, s_{j,N}) = \mathsf{SS.Share}(\mathbf{0}, \mathbb{A})$.
- $\mathsf{H}_3^{(b)}$: Same as $\mathsf{H}_2^{(b)}$ except now the challenger samples uniformly at random a vector $\mathbf{u} \leftarrow \mathbb{Z}_q^m$ instead of setting the second component of the secret key to $\mathbf{A}_{j^*}^\top \mathbf{x}_{j^*} + \mathbf{e}_0$ in the second component of pk_{j^*}.
- $\mathsf{H}_4^{(b)}$: Same as $\mathsf{H}_3^{(b)}$ except the challenger samples $\mathbf{ct}_r$ uniformly at random as $\mathbf{ct}_r \leftarrow \mathbb{Z}_q^m$.
- $\mathsf{H}_5^{(b)}$: Same as $\mathsf{H}_4^{(b)}$ except the challenger samples uniformly at random ct_0 as $ct_0 \leftarrow \mathbb{Z}_q$.

We write $\mathsf{H}_k^{(b)}(\mathcal{A})$ to denote the output distribution of experiment $\mathsf{H}_k^{(b)}$ with adversary $\mathcal{A}$. We argue in the following lemmas that the hybrid experiments are two-by-two indistinguishable.

Lemma 4. *Suppose $n \geq \lambda$ and that* SS *is a Linear Secret Sharing Scheme according to Def 2. Then, $\mathsf{H}_1^{(b)} \approx \mathsf{H}_2^{(b)}$.*

Proof. The adversary $\mathcal{A}$ is only allowed to query invalid sets $S \notin \mathbb{A}$. The lemma follows by the **privacy** property of the secret sharing scheme SS.

Lemma 5. *Suppose that $n \geq \lambda$, $m \geq 2n \log q$ and $\sigma \geq \frac{2\sqrt{n}}{q}$. Then, under the* LWE *assumption w.r.t parameters (n, m, q, σ), $\mathsf{H}_2^{(b)} \approx \mathsf{H}_3^{(b)}$.*

Proof. Suppose there exists a bit $b \in \{0,1\}$ and an adversary $\mathcal{A}$ that distinguishes $\mathsf{H}_2^{(b)}$ from $\mathsf{H}_3^{(b)}$ with non-negligible advantage. We build a simulator $\mathcal{B}$ that breaks the LWE assumption with non-negligible probability.

- $\mathcal{B}$ starts by requesting the public parameters $N, \ell, Q, \mathbb{A}$. Then it receives the challenge $(A, \mathbf{u})$ with $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^m$.
- Next, $\mathcal{B}$ samples the challenge position $j^* \leftarrow [\ell]$ and samples the public keys as follows:
  - For positions $j \neq j^*$, $\mathcal{B}$ samples the keys honestly. That is, $\mathcal{B}$ samples $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{x}_j, \mathbf{y}_{j,1}, \ldots, \mathbf{y}_{j,Q} \leftarrow \mathbb{Z}_q^n,\ \mathbf{e}_0, \ldots, \mathbf{e}_Q \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, and sets the public key pk_j as $pk_j = (\mathbf{A}_j, A_j^\top \mathbf{x}_j + \mathbf{e}_0, \mathbf{A}_j^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}_j^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$.
  - For position j^*, $\mathcal{B}$ samples $\mathbf{y}_{j,1}, \ldots, \mathbf{y}_{j,Q} \leftarrow \mathbb{Z}_q^n,\ \mathbf{e}_1, \ldots, \mathbf{e}_Q \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$ and sets the public key pk_{j^*} as $pk_{j^*} = (\mathbf{A}, \mathbf{u}, \mathbf{A}^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$.
  - $\mathcal{B}$ sets pk as $pk = (pk_1, \ldots, pk_\ell)$ and sends it to $\mathcal{A}$.
- Next, $\mathcal{B}$ receives from $\mathcal{A}$ the set $S \subseteq [N]$ of corrupted users. If $S \in \mathbb{A}$, $\mathcal{B}$ stops the game. Otherwise, $\mathcal{B}$ sets the secret keys as follows:
  - For positions $j \neq j^*$, $\mathcal{B}$ samples the secret keys honestly. That is, $\mathcal{B}$ computes $(\mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,N}) \leftarrow \mathsf{SS.Share}(\mathbf{x}_j, \mathbb{A})$, and sets $\mathbf{sk}_{i,\alpha,l}^j = \mathbf{s}_{j,i,l} - \mathbf{y}_{j,\alpha}$ for each $(i, \alpha, l) \in [N] \times [Q] \times [N]$.
  - For position j^*, $\mathcal{B}$ samples $\mathbf{sk}_{i,\alpha,l}^j \leftarrow \mathbb{Z}_q^n$.
- Finally, $\mathcal{B}$ samples $\mathbf{r} \leftarrow \{0,1\}^m$, and sets the challenge ciphertext as:

$$\begin{aligned} \mathbf{ct}_r &= \mathbf{A}\mathbf{r}, \\ ct_0 &= \mathbf{u}^\top \mathbf{r} + b\lfloor \frac{q}{2} \rceil, \\ \forall \alpha \in [Q],\ ct_\alpha &= \mathbf{y}_{j^*,\alpha}^\top \mathbf{A}\ \mathbf{r} + \mathbf{e}_\alpha^\top \mathbf{r}. \end{aligned}$$

- Finally, $\mathcal{B}$ sends $(ct_b, j^*, \{sk_i\}_{i \in S})$ to $\mathcal{A}$ [4] and forwards the bit b' output by $\mathcal{A}$.

We now observe the distribution of the challenge ciphertext:

- If $\mathbf{u} = \mathbf{A}^\top \mathbf{s} + \mathbf{e}$ where $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, the public key second component of pk_{j^*} is distributed as $\mathbf{A}^\top \mathbf{s} + \mathbf{e}$ and the ct_0 is distributed as $ct_0 = \mathbf{s}^\top \mathbf{A}\mathbf{r} + \mathbf{e}^\top \mathbf{r} + b\lfloor \frac{q}{2} \rceil$ which is the distribution of the public key in $\mathsf{H}_2^{(b)}$.
- Conversely, if $\mathbf{u} \leftarrow \mathbb{Z}_q^m$ then the public key and challenge ciphertext are distributed as in $\mathsf{H}_3^{(b)}$.

Lemma 6. *Suppose $m \geq 2n \log q$. Then $\mathsf{H}_4^{(b)} \approx \mathsf{H}_3^{(b)}$.*

Proof. This follows from lemma 2 applied to the following distributions:

$$\begin{aligned} &\left\{(\mathbf{A}, \mathbf{A}\mathbf{r})\ :\ \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m},\ r \leftarrow \{0,1\}^m\right\} \\ &\left\{(\mathbf{A}, \mathbf{z})\ :\ \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m},\ \mathbf{z} \leftarrow \mathbb{Z}_q^n\right\} \end{aligned}$$

[4] Note that since the $s_{j,i}$ are a sharing of $\mathbf{0}$, they are indistinguishable from uniformly sampled element from $\mathbb{Z}_q^{n \times L}$. Hence, the distribution of the secret keys are themselves indistinguishable from random elements sampled in $\mathbb{Z}_q^n$

Lemma 7. *Suppose $m \geq 2n \log q$. Then, $\mathsf{H}_4^{(b)} \approx \mathsf{H}_5^{(b)}$.*

Proof. This follows by applying lemma 2 to the distributions

$$\Big\{(\mathbf{u}, \mathbf{u}^\top \mathbf{r} + b\lfloor \frac{q}{2} \rceil) \quad \mathbf{u} \leftarrow \mathbb{Z}_q^m, \mathbf{r} \leftarrow \{0,1\}^m, b \leftarrow \{0,1\}\Big\}$$
$$\big\{(\mathbf{u}, z) \ : \ \mathbf{u} \leftarrow \mathbb{Z}_q^m, z \leftarrow \mathbb{Z}_q\big\}$$

4.4 Adaptive One-Sided Security

Next, we prove that our construction satisfies the adaptive one-sided security property, as defined in section 3. This proof slightly differs from the one in [15]. In their proof, the challenger is given a triple (X, Y, Z). The Y components can be seen as the randomness used to set the ciphertext. In comparison, the challenger here is only given a matrix A and a vector u. We use the LWE assumption to argue that the public key component $\mathbf{A}_j^\top \mathbf{y}_{j^*,\alpha^*}$ is indistinguishable from random. Since the adversary is not given any secret key related to the challenge position j^* and letter α^*, we can then argue that the ciphertext component ct_{α^*} can be sampled uniformly at random, since the associated public key component is now also random.

Theorem 3 (Adaptive One-sided security). *Let $\lambda \in \mathbb{N}$ be a security parameter, $N = N(\lambda)$, $\ell = \ell(\lambda)$, $Q = Q(\lambda)$. Suppose that $n \geq \lambda$, $m \geq n \log q$. Under the LWE assumption w.r.t parameters (n, m, q, σ), Π has $\mathsf{ad-OSS}$.*

Proof. We prove that the following a reduction between two experiments followed by a sequence of hybrids are two-by-two indistinguishable:

- $\mathsf{H}_1^{(b)}$: This is the initial $\mathsf{ad-OSS}$ security game. At the beginning, $\mathcal{A}$ sends $N, \ell, Q, \mathbb{A}$ to the challenger $\mathcal{C}$. The challenger samples $(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$ with $pk = (pk_1, \ldots, pk_\ell)$ where for $j \in [\ell]$, $pk_j = (\mathbf{A}_j, \mathbf{A}_j^\top \mathbf{x}_j + \mathbf{e}_0, \mathbf{A}_j^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}_j^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$, and for each $i \in [N]$, $sk_i = \{\mathbf{sk}_{i,\alpha,l}^j\}_{(j,\alpha,l) \in [\ell] \times [Q] \times [L]}$ with $\mathbf{sk}_{i,\alpha,l}^j = \mathbf{s}_{j,i,l} - \mathbf{y}_{i,\alpha}$. Then the challenger sends pk to $\mathcal{A}$. The adversary answers with a position and letter pair $(j^*, \alpha^*) \in [\ell] \times [Q]$. The challenger samples $b \leftarrow \{0,1\}$ together with $ct^{(0)} \leftarrow \mathsf{Enc}(pk, j^*, 0)$ and $ct^{(1)} \leftarrow \mathbb{Z}_q^m \times \mathbb{Z}_q \times \mathbb{Z}_q^Q$. Next, the challenger sets the challenge ciphertext as $ct^* = (j, \mathbf{ct}_r^{(0)}, ct_1^{(0)}, \ldots, ct_{\alpha^*-1}^{(0)}, ct_{\alpha^*}^{(b)}, ct_{\alpha^*+1}^{(1)}, \ldots, ct_Q^{(1)})$. Finally, the challenger sends $ct^*, \{\mathbf{sk}_{i,\alpha,l}^j\}_{(j,\alpha,l) \in [\ell] \setminus \{j^*\} \times [Q] \times [L]}$ and $\{\mathbf{sk}_{i,\alpha,l}^{j^*}\}_{(\alpha,l) \in [Q] \setminus \{\alpha^*\} \times [L]}$ to $\mathcal{A}$.
- $\mathsf{H}_2^{(b)}$: Same as $\mathsf{H}_1^{(b)}$ except the challenger samples a pair $(j^*, \alpha^*) \in [\ell] \times [Q]$ at the start of the game. If it is different from the pair output by $\mathcal{A}$ then the challenger stops the game.
- $\mathsf{H}_3^{(b)}$: Same as $\mathsf{H}_2^{(b)}$ except that the challenger parses the public key $pk_{j^*} = (\mathbf{A}_{j^*}, \mathbf{pk}_{j^*,1}, \ldots, \mathbf{pk}_{j^*,Q})$, samples $\mathbf{pk}_{j^*,\alpha^*}$ uniformly at random as $\mathbf{pk}_{j^*,\alpha^*} = \mathbf{u} \leftarrow \mathbb{Z}_q^m$.
- $\mathsf{H}_4^{(b)}$: Same as $\mathsf{H}_3^{(b)}$ except the challenger now samples ct_{α^*} uniformly at random as $ct_{\alpha^*} \leftarrow \mathbb{Z}_q$.

We argue in the following lemmas that each hybrid experiment are two-by-two indistinguishable.

Lemma 8. *If there exist an adversary an adversary $\mathcal{A}$ against the* $\mathsf{ad}-\mathsf{OSS}$ *experiments that wins with probability ε. Then, there exists an adversary $\mathcal{B}$ against* $\mathsf{H}_2^{(b)}$ *such that* $\mathsf{Adv}_{H_2}(\mathcal{B}) \geq \frac{\varepsilon}{\ell Q}$.

Proof. Let $\mathcal{A}$ be an adversary in experiment $\mathsf{H}_1^{(b)}$ with advantage ε. We construct a reduction algorithm $\mathcal{B}$ that wins $\mathsf{H}_2^{(b)}$. $\mathcal{B}$ samples uniformaly at random a pair $(\tilde{j}, \tilde{\alpha}) \leftarrow [\ell] \times [Q]$. Next, $\mathcal{B}$ samples the keys using the KeyGen algorithm exactly as in $\mathsf{H}_1^{(b)}$, except for the keys $sk^{\tilde{j}_{i,\tilde{\alpha}}}$ for all $i \in [N]$. Note that the generation of the public keys is independant of the choice on (j^*, α^*) so the distribution are the same. $\mathcal{B}$ sends the public key pk to $\mathcal{A}$. Next, $\mathcal{A}$ outputs a pair (j^*, α^*). If $(j^*, \alpha^*) \neq (\tilde{j}, \tilde{\alpha})$, $\mathcal{B}$ aborts the game and outputs a bit $b' \leftarrow \{0,1\}$ uniformly at random. Otherwise, $\mathcal{B}$ follows the rest of the game honestly and outputs whatever $\mathcal{A}$ outputs. Since guessing the correct pair (j^*, α^*) happens with probability $\frac{1}{\ell Q}$, and that the advantage of $\mathcal{A}$ in winning $\mathsf{H}_1^{(b)}$ is ε, we conclude that the advantage of $\mathcal{B}$ in winning $\mathsf{H}_2^{(b)}$ is exactly $\frac{\varepsilon}{\ell Q}$.

Lemma 9. *Suppose that $n \geq \lambda$ and that* LWE *is hard with parameters (n, m, q, σ). Then,* $\mathsf{H}_2^{(b)} \approx \mathsf{H}_3^{(b)}$.

Proof. Suppose that there exists an adversary $\mathcal{A}$ that distinguishes $\mathsf{H}_2^{(b)}$ from $\mathsf{H}_3^{(b)}$ with non-negligible probability. We build a simulator $\mathcal{B}$ that breaks the LWE assumption with non-negligible probability.

- $\mathcal{B}$ starts by requesting the public parameters $N, \ell, Q, \mathbb{A}$. Then, it receives the challenge (A, u) with $A \in \mathbb{Z}_q^{n \times m}$ and $u \in \mathbb{Z}_q^m$.
- Next, $\mathcal{B}$ samples a pair $(j^*, \alpha^*) \leftarrow [\ell] \times [Q]$ and samples the keys as follows:
  - For positions $j \neq j^*$, $\mathcal{B}$ samples the keys honestly. That is, it samples $\mathbf{A}_j \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{x}_j, \mathbf{y}_{j,1}, \ldots, \mathbf{y}_{j,Q} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0, \ldots, \mathbf{e}_Q \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, and sets the public key pk_j as $pk_j = (\mathbf{A}_j, \mathbf{A}_j^\top \mathbf{x}_j + \mathbf{e}_0, \mathbf{A}_j^\top \mathbf{y}_{j,1} + \mathbf{e}_1, \ldots, \mathbf{A}_j^\top \mathbf{y}_{j,Q} + \mathbf{e}_Q)$. Then $\mathcal{B}$ computes $(\mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,N}) \leftarrow \mathsf{SS.Share}(\mathbf{x}_j, \mathbb{A})$ and sets $\mathbf{sk}_{i,\alpha,l}^j = \mathbf{s}_{j,i,l} - \mathbf{y}_{j,\alpha}$ for each $(i, \alpha, l) \in [N] \times [Q] \times [L]$.
  - For position j^*, $\mathcal{B}$ sets $\mathbf{A}_{j^*} = \mathbf{A}$, samples $\mathbf{x}_j, \mathbf{y}_{j,1}, \ldots, \mathbf{y}_{j,\alpha^*-1} \leftarrow \mathbb{Z}_q^n$, $, \mathbf{y}_{j,\alpha^*+1}, \ldots, \mathbf{y}_{j,Q} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e}_0, \ldots, \mathbf{e}_{\alpha^*-1}, \mathbf{e}_{\alpha^*+1}, \ldots, \mathbf{e}_Q \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$ and sets pk_{j^*} as $pk_{j^*} = (\mathbf{A}_{j^*}, \mathbf{A}_{j^*}^\top \mathbf{x}_{j^*} + \mathbf{e}_0, \mathbf{A}_{j^*}^\top \mathbf{y}_{j^*,1} + \mathbf{e}_1, \ldots, \mathbf{A}_{j^*}^\top \mathbf{y}_{j^*,\alpha^*-1} + \mathbf{e}_{\alpha^*-1}, \mathbf{u}, \mathbf{A}_{j^*}^\top \mathbf{y}_{j^*,\alpha^*+1} + \mathbf{e}_{\alpha^*+1}, \ldots, \mathbf{A}_{j^*}^\top \mathbf{y}_{j^*,Q} + \mathbf{e}_Q)$.
    Finally, $\mathcal{B}$ has to compute $(\mathbf{s}_{j^*,1}, \ldots, \mathbf{s}_{j^*,N}) \leftarrow \mathsf{SS.Share}(\mathbf{x}_{j^*}, \mathbb{A})$ and sets $\mathbf{sk}_{i,\alpha,l}^{j^*} = \mathbf{s}_{j^*,i,l} - \mathbf{y}_{j^*,\alpha}$ for each $(i, \alpha, l) \in [N] \times [Q] \backslash \{\alpha^*\} \times [L]$.
  - $\mathcal{B}$ sets the public key pk as $pk = (pk_1, \ldots, pk_\ell)$ and sends it to $\mathcal{A}$
- $\mathcal{B}$ request a pair $(j_\mathcal{A}, \alpha_\mathcal{A})$ from $\mathcal{A}$. If $(j_\mathcal{A}, \alpha_\mathcal{A}) \neq (j^*, \alpha^*)$, then $\mathcal{B}$ stops the game.

– Next, $\mathcal{B}$ samples $r \in \{0,1\}^m$ and sets the challenge ciphertext ct_b as follows:

$$\begin{aligned} \mathbf{ct}_r &= \mathbf{A}_{j^*}\mathbf{r} \\ ct_0 &= \mathbf{x}_{j^*}^\top \mathbf{A}_{j^*}\mathbf{r} + \mathbf{e}_0^\top \mathbf{r} \\ \forall \alpha \in [Q]\backslash\{\alpha^*\},\ ct_\alpha &= \mathbf{y}_{j^*,\alpha}^\top \mathbf{A}_{j^*}\mathbf{r} + \mathbf{e}_\alpha^\top \mathbf{r} \\ ct_{\alpha^*} &= \mathbf{u}^\top \mathbf{r} \end{aligned}$$

– $\mathcal{B}$ sends $ct, \{\mathbf{sk}_{i,\alpha,l}^j\}_{(j,\alpha,l)\in[\ell]\backslash\{j^*\}\times[Q]\times[L]}$ and $\{\mathbf{sk}_{i,\alpha,l}^{j^*}\}_{(\alpha,l)\in[Q]\backslash\{\alpha^*\}\times[L]}$.
– When $\mathcal{A}$ answers with its guess b', $\mathcal{B}$ forwards it.

We now observe the distribution of the public key and challenge ciphertext:

– If $\mathbf{u} = \mathbf{A}^\top\mathbf{y} + \mathbf{e}$, where $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}^m$, then by parsing pk_{j^*} as $pk_{j^*} = (\mathbf{A}_{j^*}, \mathbf{A}_{j^*}^\top\mathbf{x}_{j^*} + \mathbf{e}_0, \mathbf{pk}_{j^*,1}, \ldots, \mathbf{pk}_{j^*,Q})$ then we have that $\mathbf{pk}_{j^*,\alpha^*}$ is distributed as $\mathbf{pk}_{j^*,\alpha^*} = \mathbf{A}_{j^*}^\top\mathbf{y} + \mathbf{e}$ which is the distribution of $\mathbf{pk}_{j^*,\alpha^*}$ in $\mathsf{H}_2^{(b)}$ and the ciphertext component ct_{α^*} is distributed as $ct_{\alpha^*} = \mathbf{y}^\top\mathbf{A}_{j^*}\mathbf{r} + \mathbf{e}^\top\mathbf{r}$ which is the distribution of the ct_{α^*} in $\mathsf{H}_2^{(b)}$.
– Conversely, if $\mathbf{u} \leftarrow \mathbb{Z}_q^m$ the $\mathbf{pk}_{j^*,\alpha^*}$ and ct_{α^*} are distributed as in $\mathsf{H}_3^{(b)}$.

Lemma 10. *Suppose $m \geq 2\log q$. Then $\mathsf{H}_3^{(b)} \approx \mathsf{H}_4^{(b)}$.*

Proof. We apply the leftover hash lemma to the following two distributions:

$$\begin{aligned} &\left\{(\mathbf{u}, \mathbf{u}^\top\mathbf{r}) \ :\ \mathbf{u} \leftarrow \mathbb{Z}_q^m, \mathbf{r} \leftarrow \{0,1\}^m\right\} \\ &\left\{(\mathbf{u}, \mathbf{z}) \ :\ \mathbf{u} \leftarrow \mathbb{Z}_q^m, \mathbf{z} \leftarrow \mathbb{Z}_q\right\} \end{aligned}$$

4.5 Share Simulatability

Theorem 4. *Suppose that $\frac{m\sqrt{m}\sigma}{B_{flood}} = \mathsf{negl}(\lambda)$ and that the secret sharing scheme SS satisfies the **privacy**, **reconstruction** and **linearity** properties from Def 2. Then, Π has share simulation security.*

Proof. We prove the following lemma by a sequence of hybrids.

– $\mathsf{H}_1^{(b)}$: This is the real share decryption simulatability experiment $\mathsf{Exp}_{\mathcal{A},real}(1^\lambda)$. On input the number of users N, the number of positions ℓ, an alphabet size Q and an access structure $\mathbb{A}$, the challenger runs the key generation algorithm to obtain $(pk, sk_1, \ldots, sk_N) \leftarrow \mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$, and sends pk to $\mathcal{A}$. Then, it receives a maximal invalid party set $S^* \subseteq [N]$. The challenger answers by sending the set of corrupted secret keys $\{sk_i\}_{i\in S^*}$. Next, $\mathcal{A}$ can query the partial decryption oracle a polynomial number of times. On input (μ, j), the challenger computes $ct = \Pi.\mathsf{Enc}(pk, j, \mu)$ and the decryption shares of honest users $d_i = \Pi.\mathsf{PartDec}(pk, ct, \mathbf{sk}_{i,\alpha}^j)$ for each $i \in [N]\backslash S^*$ (with a possibly different $\alpha \in [Q]$ sampled uniformly at random for each i). It then sends $(ct, \{d_i\}_{i\in[N]\backslash S^*})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs its guess $b \in \{0,1\}$.

– $\mathsf{H}_2^{(b)}$: Same as $\mathsf{H}_1^{(b)}$ except the challenger simulates the partial decryptions for each of $\mathcal{A}$'s queries. Given $S^* \subseteq [N]$ the set of corrupted parties, let $S_L = \{(i,l)\}_{(i,l)\in S\times[L]}$ the associated set of shares. The challenger commits to a maximal invalid set of share elements T such that $S_L \subseteq T$. Secret keys of user i can be parsed as $sk_i = \left(\mathbf{sk}_{i,\alpha,l}^j\right)_{(j,\alpha,l)\in[\ell]\times[Q]\times[L]}$. The partial decryption shares of honest users are computed as follows:
  - For $(i,l) \in T\backslash S_L$, the share is computed normally. That is, the challenger samples $\alpha \leftarrow [Q]$, $\tilde{e}_{i,l} \leftarrow [-B_{flood}, B_{flood}]$ and sets $d_{i,l} = \mathbf{sk}_{i,\alpha,l}^{j\top}\mathbf{ct}_r + ct_\alpha + \tilde{e}_{i,l} = \mathbf{s}_{j,i}^\top \mathbf{A}_j\mathbf{r} + \mathbf{e}_0^\top\mathbf{r} + \tilde{e}_{i,l}$.
  - For $(i,l) \in [N]\backslash T$, let $T_{i,l} \subset T \cup \{(i,l)\}$ be a minimal valid set of share elements. This set exists since T is a maximal invalid set of share elements. Then the partial decryption share is set as $d_{i,l} = ct_0 - \mu\lfloor\frac{q}{2}\rceil - \sum_{(j',i')\in T_{i,l}\backslash\{(i,l)\}} \mathbf{s}_{j',i'}^\top \mathbf{A}_j\mathbf{r} + \tilde{e}_{i,l}$, for $\tilde{e}_{i,l} \leftarrow [-B_{flood}, B_{flood}]$.
– $\mathsf{H}_3^{(b)}$: Same as $\mathsf{H}_2^{(b)}$ except that for each $j \in [\ell]$, the challenger secret shares the $\mathbf{0}$ vector instead of $\mathbf{x}_j \in \mathbb{Z}_q^n$.
  In this hybrid, the partial decryption share d_i of all honest users only depend on the secret key shares of corrupted users from an invalid set. They do not leak information on the shared $\mathbf{x}_j$.

We now prove that each hybrid is two-by-two indistinguishable.

Lemma 11. *Suppose that* $\frac{m\sqrt{m}\sigma}{B_{flood}} = \mathsf{negl}(\lambda)$. *Then,* $\mathsf{H}_1^{(b)} \approx \mathsf{H}_2^{(b)}$

Proof. The only difference in $\mathcal{A}$'s view between $\mathsf{H}_1^{(b)}$ and $\mathsf{H}_2^{(b)}$ is how the partial decryption of honest parties outside the maximal invalid set of shares are computed. In $\mathsf{H}_1^{(b)}$, we have for some $\alpha \in [Q]$, $d_{i,l} = ct_\alpha + \mathbf{sk}_{i,\alpha,l}^{j\top}\mathbf{ct}_r + \tilde{e}_{i,l} = \mathbf{s}_{j,i,l}^\top\mathbf{A}_j\mathbf{r} + \mathbf{e}_\alpha^\top\mathbf{r} + \tilde{e}_{i,l}$, with $\tilde{e}_{i,l} \leftarrow [-B_{flood}, B_{flood}]$ and setting $d_i = (d_{i,1}, \ldots, d_{i,L})$. In $\mathsf{H}_2^{(b)}$, the challenger commits to a maximal invalid set of shares T and shares $d_{i,l}$ are computed as in $\mathsf{H}_1^{(b)}$ for $i \in T$. The only difference is when $(i,l) \in ([N]\times[L])\backslash T$. For such pairs, the challenger computes a minimal valid share set $T_{i,l} \subset T \cup \{(i,l)\}$ and sets $d_{i,l} = ct_0 - \mu\lfloor\frac{q}{2}\rceil - \sum_{(j',i')\in T_{i,l}\backslash\{(i,l)\}} \mathbf{s}_{j',i'}^\top\mathbf{A}_j\mathbf{r} + \tilde{e}_{i,l}$, for $\tilde{e}_{i,l} \leftarrow [-B_{flood}, B_{flood}]$. By correctness of the secret sharing scheme, we have that $\mathbf{x}_j = \sum_{(i',j')\in T_{i,l}} \mathbf{s}_{j,i,l}$. It follows that:

$$d_{i,l} = \mathbf{s}_{j,i,l}^\top\mathbf{A}_j\mathbf{r} + \mathbf{e}_0^\top\mathbf{r} + \tilde{e}_{i,l}$$

By assumption, $|\mathbf{e}_0^\top\mathbf{r}| \leq m\sqrt{m}\sigma$ w.o.p. Therefore, by the choice of B_{flood}, the two distributions of $d_{i,l}$ in $\mathsf{H}_1^{(b)}$ and $\mathsf{H}_2^{(b)}$ are statistically indistinguishable by lemma 1.

Lemma 12. *If* SS *is a secret sharing scheme that satisfies privacy as defined in def 2, then* $\mathsf{H}_2^{(b)} \approx \mathsf{H}_3^{(b)}$.

Proof. This follows directly from the privacy of the secret sharing scheme SS. In $\mathsf{H}_2^{(b)}$, the vector $\mathbf{x}_j \in \mathbb{Z}_q^n$ is secret shared into $\mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,N} \leftarrow \mathsf{SS.Share}(\mathbf{x}_j, \mathbb{A})$. In

$\mathsf{H}_3^{(b)}$ the $\mathbf{0}$ vector is secret shared into $\mathbf{s}_{j,1}, \ldots, \mathbf{s}_{j,N} \leftarrow \mathsf{SS.Share}(\mathbf{0}, \mathbb{A})$. Since $\mathcal{A}$ can only choose a (maximal) invalid set S^*, by the privacy property of SS, the two distributions are indistinguishable.

□

4.6 Instantiation.

Let λ be a security parameter, N be the number of users, ℓ a length parameter, Q be the alphabet size and $\mathbb{A}$ be a monotone access structure that can be represented by a $\{0,1\}$-LSS. We set the rest of the parameters of the system to satisfy theorem

- We set the lattice dimension $n = \lambda$ and $m = O(n \log q)$.
- We set the noise parameter $\sigma = poly(n)$ such that LWE is hard with parameters (n, m, q, σ).
- For codes typically used in traitor tracing such as robust IPP codes, we require $\ell = Q = \frac{d \log N}{\log d \log N}$ with $d = f^2/(1 - \theta - f\theta)$, where f is the number of traitors the code can support and θ is the proportion of erasures.
- We set the flooding noise bound $B_{flood} = m\sqrt{m}\sigma 2^{\lambda}$.
- We set the modulus q to be a prime such that $q \geq 4(1 + NL)\sqrt{nm}\sigma + 4NLB_{flood} = O(2^{\lambda} NL\sqrt{nm}\sigma)$
- The ciphertext ct has size $|ct| = O\left((m + Q)\log q\right)$, which adds only an additional vector of size $m \log q$ compared to the ciphertext size in [15].

5 A Threshold Traitor Tracing Scheme

As shown in [15], threshold traitor tracing is known to be implied by QTPKE combined with robust Q-ary IPP codes. In this section, we give a formal definition of such codes and recall the construction of a Threshold Traitor Tracing scheme that supports public traceability. This section is adapted from [15] in the case of an encryption scheme. We also adapt the inputs of the algorithms to support access structures supported by $\{0,1\}$-LSSS which include thresholds. For the full details of the proofs, we refer the reader to [15].

5.1 Robust IPP Codes

Let $\mathcal{C} = \{\omega_1, \ldots, \omega_N\} \subset \Sigma^{\ell}$ be a Q-ary code of size N and length ℓ. Let Δ be the minimum Hamming distance over alphabet $\Sigma = \{1, \ldots, Q\}$. We assume that $\omega_i = \omega_{i,1} \ldots \omega_{i,\ell} \in \Sigma^{\ell}$. Given a positive integer t, a subset of codewords $X = \{\omega_1, \ldots, \omega_t\} \subset \mathcal{C}$ is called a coalition of size t. Let $X_i = \{\omega_{1,i}, \ldots, \omega_{t,i}\}$ be the set of the i-th coordinate of the coalition X. If the cardinality of X_i is equal to 1, the coordinate i is called *undetectable*, else it is called *detectable*. The set of detectable coordinates for the coalition X is denoted by $D(X)$. The set of descendants of X, denoted $\mathsf{desc}(X)$, is defined by $\mathsf{desc}(X) = \{x = (x_1, \ldots, x_{\ell}) \in \Sigma^{\ell} \mid x_j \in X_j,\ 1 \leq j \leq \ell\}$. In the coalition X, we call codewords the parents

of the set $\mathsf{desc}(X)$. We define a t-descendant of the code $\mathcal{C}$ the set $\mathsf{desc}_t(\mathcal{C}) = \bigcup_{X\subset\mathcal{C},|X|\leq t} \mathsf{desc}(X)$. The $\mathsf{desc}_t(\mathcal{C})$ consists of all ℓ-tuples that could be generated by some coalition of size at most t. Codes with identifiable parent property (IPP) are then given by the following definition.

Definition 8. *Given a code $\mathcal{C} \subseteq \Sigma^\ell$, with parameters (ℓ, N, Q). Let $t \geq 2$ be an integer. The code $\mathcal{C}$ is called a t-IPP code if for all $x \in \mathsf{desc_t}(\mathcal{C})$, it holds that:*

$$\bigcap_{X\subset\mathcal{C},|X|\leq t,\ s.t.\ x\in\mathsf{desc}(X)} X \neq \emptyset.$$

In a t-IPP code, we can always identify a parent codeword of any descendant $x \in \mathsf{desc}_t(\mathcal{C})$. The *marking assumption*, introduced in [10] then states that any coalition of users can only create a codeword that lies in the set of descendants of the coalition.

We say that a code $\mathcal{C}$ is a *robust* IPP code if identification of the parents of a codeword x created by a coalition X is possible, even when x breaks the marking assumption in at most $\varepsilon\ell$ positions. We consider two types of mutations: erasure when $x_i = *$ and replacement when $x_i \in \Sigma\backslash X_i$. We say a word is ε-*noisy* when at most a fraction ε of its letter are $*$ mutations. Let $X \subset \mathcal{C},\ |X| \leq t$ be a coalition. Assume $\mathsf{desc}(X)$ is not empty. Let $x \in \mathsf{desc}(X)$ such that x follows the marking assumption except in $\varepsilon\ell$ positions. Call i a coordinate of x a *mutation* if $x_i \notin X_i$. We denote by $\mathsf{desc}(X)_\varepsilon$ the set of all vectors x formed from the vectors in the coalition X such that $x_i \in X_i$ for $\ell(1-\varepsilon)$ coordinates i and x_i is a mutation in at most $\varepsilon\ell$ coordinates. Codes with robust identifiable parent property (RobustIPP) are formally defined as follows.

Definition 9. *Given a code $\mathcal{C} \subseteq \Sigma^\ell$ with parameters (ℓ, N, Q), let $t \geq 2$, be an integer. The code $\mathcal{C}$ is called a (t, ε)-IPP (robust t-IPP) if it holds that*

$$\bigcap_{X\subset\mathcal{C},|X|\leq t, x\in\mathsf{desc}(X)_\varepsilon} X \neq \emptyset$$

Informally speaking, the code $\mathcal{C}$ guarantees exact identification of at least one member of the pirate coalition of size at most t for any collusion attack with at most $\varepsilon\ell$ mutations. A robust IPP code is said to have the *traceability property* if for any $x \in \mathsf{desc}(X)$, the codeword $c \in \mathcal{C}$ closest to x by the Hamming distance is always one of the parents of x, i.e $c \in \bigcap_{X\subset\mathcal{C},|X|\leq t, x\in\mathsf{desc}(X)_\varepsilon} X$. This implies that a pirate can be identified by finding any vector $c \in \mathcal{C}$ such that its distance to x is the shortest. We shall use robust IPP with traceability property.

Requirement on Δ. From [21], the minimal distance Δ must verify:

$$\Delta > \ell\left(1 - \frac{1}{t^2} + \varepsilon\left(\frac{1}{t} - \frac{1}{t^2}\right)\right) = \ell\left(1 + \frac{\varepsilon}{t} - \frac{1-\varepsilon}{t^2}\right)$$

5.2 Threshold Traitor Tracing

In this subsection, we recall the formal definition of a Threshold Traitor Tracing Encryption scheme.

Definition 10. *A Threshold Traitor Tracing KEM is a tuple of five polynomial time algorithms* $\mathsf{TT} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine}, \mathsf{Trace})$ *defined as follows:*

- $\mathsf{KeyGen}(1^\lambda, N, \mathbb{A})$*: Takes as input the security parameter* λ*, the number of users* N *and an access structure* $\mathbb{A}$*. The algorithm outputs the public key* pk*,and the users secret keys* $sk_1, \ldots, sk_N$*.*
- $\mathsf{Enc}(pk, \mu)$*: Takes as input the public key* pk *and a plaintext* μ*. The algorithm outputs a ciphertext* ct*.*
- $\mathsf{PartDec}(sk_i, ct)$*: Takes as input the user secret key* sk_i *and a ciphertext* ct*. The algorithm outputs users* i*'s decryption share* d_i*.*
- $\mathsf{Combine}(ct, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}})$*: Takes as input a ciphertext* ct*, a set of users* $\mathcal{J} \subseteq [N]$ *and its associated set of shares* $\{d_i\}_{i\in\mathcal{J}}$*. The algorithm outputs the plaintext* μ *or a rejection symbol* $\perp$*.*
- $\mathsf{Trace}(pk, 1^{1/\varepsilon}, D)$*: Takes as input the public key* pk*, an error bound* ε*, and a decoder* D*. The algorithm outputs a subset* $\mathcal{J} \subseteq [N]$*.*

A Threshold Traitor Tracing is required to satisfy the definitions of correctness and CPA *security of standard threshold public key encryption schemes.*

Originally, the decoder D was modeled as a device that decrypts successfully with non negligible probability. This model was found to be flawed in [22] and fixed in the same work. We follow their new definition, in line with modern works on traitor tracing.

Definition 11. *Let* $\varepsilon = \varepsilon(\lambda)$ *be a function of the security parameter and* negl *a negligible function of the security parameter* $\lambda \in \mathbb{N}$*. A threshold traitor tracing PKE* $\mathsf{TT} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine}, \mathsf{Trace})$ *is* ε*-secure if for every probabilistic polynomial time algorithm* $\mathcal{A}$ *and for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$Pr\left[\mathsf{GoodTrace}\right] \geq Pr\left[\mathsf{GoodDec}\right] - \mathsf{negl}(\lambda) \quad \textit{and} \quad Pr\left[\mathsf{BadTrace}\right] \leq \mathsf{negl}(\lambda)$$

The events GoodTrace*,* GoodDec *and* BadTrace *are defined as follows:*

- *Event* GoodDec *occurs when:*

$$Pr\left[D(ct_b) = b \;:\; \begin{array}{c} ct_b \leftarrow \mathsf{Enc}(pk, b) \\ b \leftarrow \{0,1\} \end{array}\right] \geq \frac{1}{2} + \varepsilon.$$

- *Event* GoodTrace *occurs when* $\mathcal{J}' \neq \emptyset$ *and* $\mathcal{J}' \subseteq \mathcal{J}$*, i.e.* Trace *identified users only in the alliance.*
- *Event* BadTrace *occurs when* $\mathcal{J}' \neq \emptyset$ *and* $\mathcal{J}' \nsubseteq \mathcal{J}$*, i.e.* Trace *accused a user that is not in the coalition of traitors.*

Where $D, \mathcal{J}, \mathcal{J}'$ *are output of the trace experiment in figure 4.*

1. The adversary outputs a number of users N together with an access structure $\mathbb{A}$ and sends them to the challenger.
2. The challenger runs the KeyGen algorithm to obtain the public key pk and the set of users secret keys $\{sk_1, \ldots, sk_N\}$.
3. The adversary can request corruption queries on user $i \in [N]$. i is then added to the set of corrupted users $\mathcal{J}$. The challenger answers by sending i's secret key sk_i.
4. After some time, the adversary outputs a decoder D and sends it to the challenger.
5. The challenger obtains $\mathcal{J}'$ by running the Trace algorithm using decoder D.

Fig. 4. Trace experiment $\mathsf{TraceExperiment}$ for a threshold traitor tracing encryption scheme and adversary $\mathcal{A}$

We also define the following advantage:

$$\mathsf{Adv}^{TT}_{\mathsf{TT}}(\mathcal{A}) = \max\{Pr\left[\mathsf{GoodDec}\right] - Pr\left[\mathsf{GoodTrace}\right], Pr\left[\mathsf{BadTrace}\right]\}$$

In this section, we use the Q-Partite TPKE alongside a robust IPP code to construct a Threshold Traitor Tracing scheme. Thanks to the code properties, our scheme benefits from the desired public traceability property. Note that since the coalition of traitors $\mathcal{J}$ we consider is of size f such that $\mathcal{J} \in \mathbb{A}$, it is large enough to decrypt on its own. Thus, we can use classical tools such as linear tracing used for robust IPP codes. Hence, our tracing strategy is inspired from classical traitor tracing such as [21].

5.3 Construction

We now give a construction of a Threshold Traitor Tracing Scheme, combining a $QTPKE$ with a robust IPP code. This construction follows directly from [15]. Let N be the maximal number of users, $\mathbb{A}$ the access structure and ε the avantage of a pirate distinguisher. Each party is identified by an integer from $\{1, \ldots, N\}$. Let $\mathcal{C}$ be an IPP code with parameters (ℓ, N, Q), able to identify coalitions of size up to f. Let Δ be its Hamming distance and $0 < \theta = \frac{1/2-\varepsilon}{1/2-2/\sqrt{\lambda}} < \frac{1}{f+1}$ satisfying the condition:

$$\Delta > \ell\left(1 + \frac{\theta}{f} - \frac{1-\theta}{f^2}\right)$$

Next, let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine})$ be a Q-Partite Threshold PKE. Finally, given a security parameter λ, we assume the size of the traitors set is large enough to verify the access structure. We define a threshold traitor tracing scheme TT composed of the tuple of probabilistic polynomial time algorithms $\mathsf{TT} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Combine}, \mathsf{Trace})$ as follows:

$\mathsf{TT.KeyGen}(1^\lambda, 1^{1/\varepsilon}, N, \mathbb{A})$: Takes as input the security parameter λ, the number of users N, the access structure $\mathbb{A}$ and the decoder advantage ε. By calling the

$\Pi.\mathsf{KeyGen}(1^\lambda, N, \ell, Q, \mathbb{A})$, we obtain the public key pk and the set of secret keys $\{(sk^j_{i,1}, \ldots, sk^j_{i,Q})_{i\in[N]}\}_{j\in[\ell]}$. Finally, for each user i, set: $sk_i = (sk^1_{i,\omega_{i,1}}, \ldots, sk^\ell_{i,\omega_{i,\ell}})$ where $\omega_{i,j}$ is the value at position j of codeword ω_i.

$\mathsf{TT.Enc}(pk, \mu)$: Takes as input the public key pk. Samples a position $j \in [\ell]$. The ciphertext is $ct = (j, \mathbf{ct}_r, ct_0, ct_1, \ldots, ct_Q))$ and ct was obtained using the encryption algorithm $\Pi.Enc(pk, j, \mu)$.

$\mathsf{TT.PartDec}(sk_i, ct)$: Takes as input the public key pk, the position j, a user secret key sk_i, and a ciphertext $ct = (j, \mathbf{ct}_r, ct_0, ct_1, \ldots, ct_Q)$. By calling the decryption algorithm $\Pi.\mathsf{PartDec}(pk, sk^j_{i,\alpha}, ct)$, the user gets their decryption share d_i.

$\mathsf{TT.Combine}(ct, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}})$: Takes as input the public parameters pk, the public key combiner pkc, a ciphertext $c = (j, ct_0, \mathbf{ct}_r, ct_1, \ldots, ct_Q)$, a set of valid users $\mathcal{J} \in \mathbb{A}$ with the associated set of decryption shares d_i. By calling the algorithm $\Pi.\mathsf{Combine}(ct, \mathcal{J}, \{d_i\}_{i\in\mathcal{J}})$, the user recovers plaintext μ.

$\mathsf{TT.Trace}(pk, D, 1^{1/\varepsilon})$: Takes as input the public key pk, a decoder D and an error bound ε. We run the decoder for every position $j \in [\ell]$ to form new letters of the pirate codeword ω^*. After all the positions have been tested, we use the Trace algorithm from $\mathcal{C}$ to accuse a party. We consider the following tracing procedure:

for $j = 1$ to ℓ **do**
  $count \leftarrow 0$
  for $\alpha = 0$ to $Q + 1$ **do**
    loop $W \leftarrow 8\lambda^{3/2}(N/\varepsilon)^2$ **times**
      $(ct^{(0)} = (j, \mathbf{ct}^{(0)}_r, ct^{(0)}_0, c^{(0)}_1, \ldots, c^{(0)}_Q)) \leftarrow \Pi.Enc(pk, j, 0)$
      $(ct^{(1)} = (j, \mathbf{ct}^{(1)}_r, ct^{(1)}_0, c^{(1)}_1, \ldots, c^{(1)}_Q)) \leftarrow \Pi.Enc(pk, j, 1)$
      $b \leftarrow \{0, 1\}$
      $ct^* \leftarrow (j, \mathbf{ct}^{(b)}_r, ct^{(b)}_0, ct^{(b)}_1, \ldots, c^{(b)}_{\alpha-1}, c^{(1-b)}_\alpha, \ldots, c^{(1-b)}_Q)$
      Call D on input ct^*
      if $D(ct^*) = b$ **then**
        $count \leftarrow count + 1$
      end if
    end loop
    Let $p_{j,\alpha} \leftarrow count/W$ be the fraction of time where D outputs b correctly.
  end for
  If there is a letter $\alpha^* \in [1, Q]$ such that $|p_{j,\alpha^*+1} - p_{j,\alpha^*}| \geq \dfrac{\varepsilon}{Q}$ then letter α^* is accused and $\omega_j \leftarrow \alpha^*$.
  Else, ω_j is assigned $*$.
end for

From the pirate word $\omega = \omega_1 \dots \omega_\ell$ found after the loop finished, call tracing procedure in $\mathcal{C}$ on input ω. The Trace returns a traitor.

Theorem 5. *For every probabilistic polynomial time algorithm $\mathcal{A}$ against the ϵ-security of the scheme TT, there exists a probabilistic polynomial time algorithm $\mathcal{B}$ against the One Sided Security of the Q-Partite Threshold PKE Π such that for all $\lambda \in \mathbb{N}$, the following holds:*

$$\mathsf{Adv}^{TT}_{\mathsf{TT}}(\mathcal{A}) \leq 4\ell e^{-4\lambda\frac{N^2}{Q\varepsilon^2}} + \frac{2\ell Q^3}{Q-f}\mathsf{Adv}^{ad-OSS}(\mathcal{B})$$

where $\ell = \ell(\lambda),\ Q = Q(\lambda)$ are the codewords length and alphabet size as defined in the scheme and f is the size of the coalition.

For the full tracing security proof, we refer the reader to [15]. The correctness and CPA of the scheme follows directly from the $QTPKE$'s correctness and security.

Acknowledgments. We would like to thank anonymous reviewers for their helpful discussions and valuable comments. We would also like to thank Ho Nguyen Pham and Jinwei Zhang for their helpful comments on an earlier version of this work. This work was supported by the France 2030 ANR Projects ANR-22-PECY-003 SecureCompute and the Banque Publique d'Investissement under the VisioConfiance project.

References

1. Albrecht, M.R., Lai, R.W.F.: Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 519–548. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_18
2. Albrecht, M.R., Lai, R.W.F., Lapiha, O., Woo, I.K.Y.: Partial lattice trapdoors: How to split lattice trapdoors, literally. In: ASIACRYPT 2025, Part III. pp. 265–296. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5099-9_9
3. Applebaum, B., Nir, O., Pinkas, B.: How to recover a secret with $o(n)$ additions. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 236–262. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_8
4. Barg, A., Blakley, G.R., Kabatiansky, G., Tavernier, C.: Robust parent-identifying codes. In: 2010 IEEE Information Theory Workshop. pp. 1–4 (2010). https://doi.org/10.1109/CIG.2010.5592920
5. Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 201–218. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_13
6. Bendlin, R., Krehbiel, S., Peikert, C.: How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In: Jacobson, Jr., M.J., Locasto, M.E., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 218–236. Springer, Berlin, Heidelberg (Jun 2013). https://doi.org/10.1007/978-3-642-38980-1_14

7. Bhattacharyya, R., Bormet, J., Faust, S., Mukherjee, P., Othman, H.: CCA-secure traceable threshold (ID-based) encryption and application. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 2324–2338. ACM Press (Oct 2025). https://doi.org/10.1145/3719027.3744876
8. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 565–596. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_19
9. Boneh, D., Partap, A., Rotem, L.: Accountability for misbehavior in threshold decryption via threshold traitor tracing. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 317–351. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68394-7_11
10. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data (extended abstract). In: Coppersmith, D. (ed.) CRYPTO'95. LNCS, vol. 963, pp. 452–465. Springer, Berlin, Heidelberg (Aug 1995). https://doi.org/10.1007/3-540-44750-4_36
11. Bormet, J., Hofmann, J., Othman, H.: Traceable threshold encryption without a trusted dealer. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025. pp. 471–505. Springer Nature Singapore, Singapore (2026)
12. Boudgoust, K., Costache, A.: Improved Rényi arguments for lattice-based threshold encryption. Cryptology ePrint Archive, Report 2025/735 (2025), https://eprint.iacr.org/2025/735
13. Boudgoust, K., Lapiha, O., del Pino, R., Prest, T.: IND-CCA lattice threshold KEM under 30 KiB. Cryptology ePrint Archive, Paper 2026/021 (2026), https://eprint.iacr.org/2026/021
14. Boudgoust, K., Scholl, P.: Simple threshold (fully homomorphic) encryption from LWE with polynomial modulus. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part I. LNCS, vol. 14438, pp. 371–404. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8721-4_12
15. Canard, S., Papon, N., Phan, D.H.: Public traceability in threshold decryption. CiC **2**(2), 21 (2025). https://doi.org/10.62056/akjb0lmol
16. Chakrabarti, A., Maitra, M., Mondal, A., Sehgal, K.: Silent threshold traitor tracing & enhancing mempool privacy. In: Huang, C.Y., Chen, J.C., Shieh, S.P., Lie, D., Cortier, V. (eds.) ACM CCS 2025. pp. 1769–1783. ACM Press (Oct 2025). https://doi.org/10.1145/3719027.3765099
17. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 257–270. Springer, Berlin, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48658-5_25
18. Cini, V., Lai, R.W.F., Woo, I.K.Y.: Pilvi: Lattice threshold PKE with small decryption shares and improved security. In: ASIACRYPT 2025, Part VI. pp. 539–569. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5119-4_17
19. Das, S., Datta, P., Partap, A., Sasmal, S., Zhandry, M.: Optimal threshold traitor tracing. Cryptology ePrint Archive, Paper 2025/2154 (2025), https://eprint.iacr.org/2025/2154
20. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_28

21. Do, X.T., Phan, D.H., Yung, M.: A concise bounded anonymous broadcast yielding combinatorial trace-and-revoke schemes. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 2020, Part II. LNCS, vol. 12147, pp. 145–164. Springer, Cham (Oct 2020). https://doi.org/10.1007/978-3-030-57878-7_8
22. Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) 50th ACM STOC. pp. 660–670. ACM Press (Jun 2018). https://doi.org/10.1145/3188745.3188844
23. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In: 21st ACM STOC. pp. 12–24. ACM Press (May 1989). https://doi.org/10.1145/73007.73009
24. Lapiha, O., Prest, T.: A lattice-based IND-CCA threshold KEM from the BCHK+ transform. In: ASIACRYPT 2025, Part III. pp. 461–494. LNCS, Springer, Singapore (Dec 2025). https://doi.org/10.1007/978-981-95-5099-9_15
25. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Berlin, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_41
26. Micciancio, D., Suhl, A.: Simulation-secure threshold PKE from LWE with polynomial modulus. CiC **1**(4), 2 (2024). https://doi.org/10.62056/a0zogy4e-
27. Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_26
28. Passelègue, A., Stehlé, D.: Low communication threshold fully homomorphic encryption. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part I. LNCS, vol. 15484, pp. 297–329. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0875-1_10
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). https://doi.org/10.1145/1060590.1060603