mathematics

MDPI

Article

R-LWE-Based Distributed Key Generation and Threshold Decryption

Ferran Alborch *,†,‡,§, **Ramiro Martínez** * and **Paz Morillo**

Department of Mathematics, Campus Nord, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain; paz.morillo@upc.edu

* Correspondence: ferran.alborch@orange.com (F.A.); ramiro.martinez@upc.edu (R.M.)

† Current affiliation: Applied Crypto Group, Orange Labs, 14000 Caen, France.

‡ Current affiliation: LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France.

§ Current affiliation: LIRMM, Université de Montpellier, CNRS, 34095 Montpellier, France.

Abstract: Ever since the appearance of quantum computers, prime factoring and discrete logarithm-based cryptography have been questioned, giving birth to the so-called post-quantum cryptography. The most prominent field in post-quantum cryptography is lattice-based cryptography, protocols that are proved to be as difficult to break as certain hard lattice problems like Learning with Errors (LWE) or Ring Learning with Errors (*R*-LWE). Furthermore, the application of cryptographic techniques to different areas, like electronic voting, has also nourished a great interest in distributed cryptography. In this work, we will give two original threshold protocols based in the lattice problem *R*-LWE: one for key generation and one for decryption. We will prove them both correct and secure under the assumption of hardness of some well-known lattice problems. Finally, we will give a rough implementation of the protocols in C to give some tentative results about their viability, in particular our model generates keys in the order of 10^3 ms and decrypts and encrypts in the order of 10^2 ms.

Keywords: post-quantum cryptography; threshold cryptography; lattices; Ring Learning with Errors (*R*-LWE); *R*-LWE encryption

MSC: 94A60; 68P25

check for updates

Citation: Alborch, F.; Martínez, R.; Morillo, P. *R*-LWE-Based Distributed Key Generation and Threshold Decryption. *Mathematics* **2022**, *10*, 728. https://doi.org/10.3390/math10050728

Academic Editor: Ioana Boureanu

Received: 3 February 2022
Accepted: 22 February 2022
Published: 25 February 2022

1. Introduction

The appearance of the computer in the XXth century caused the explosion of cryptography, the safety of which enabled the huge development of the connected society (for example, recent cryptographic endeavours into facilitating the implementation of the Smart City model [1]). Similarly, the development of quantum computing and specifically Shor's algorithm [2], which renders cryptography based on the discrete logarithm and prime factoring problems effectively useless against a quantum adversary, spawned new types of cryptography.

There are two main types of cryptography that have been developed to overcome the attacks of quantum computers: quantum and post-quantum cryptography. Quantum cryptography relies on quantum algorithms that cannot be broken by quantum adversaries, while post-quantum deals with classical (non-quantum) algorithms that cannot be broken by quantum adversaries. We focus on post-quantum cryptography given that widespread use of moderately powerful quantum computers seems unachievable in the short run. In this realm, the area that has had more recent advancements is lattice-based cryptography, as shown by the fact that in the Status Report on the Second Round of the National Institute of Standards and Technology (NIST) Post-Quantum Cryptography Standardization Process [3] most third-round finalists are lattice-based schemes. Within lattice based schemes, cryptography based in the Learning with Errors (LWE) problem and its variants is especially relevant. We will build our proposals around the Ring Learning with Errors (R-LWE) problem.

There are many applications of post-quantum cryptography, but the one we we focus on is electronic voting. Electronic voting is characterized by a significant lack of trust. The lack of trust in other entities is what initially spawned the concept of cryptography, thus going further in this direction is the next logical step to follow. Therefore, the aim is to "spread" that trust, so that one single corrupt player can no longer break the protocol. Distributed cryptography is this idea of spreading the tasks between several players so that only certain subsets of them can perform the cryptographic protocol. The interest in distributed cryptography, and in particular post-quantic distributed cryptography, can be seen in interesting recent advancements in the area like [4,5].

Adding post-quantum and distributed cryptography finally brings us to our main subject: R-LWE-based distributed key generation and threshold decryption.

1.1. State-of-the-Art

Despite the shown interest and potential usefulness of lattice-based threshold public key encryption cryptography, the amount of existing proposals is scarce, especially of proposals focusing on the R-LWE problem. Most of the existing proposals revolve around the LWE problem (see, for example, [5–8]), which has the potential problem of keys and ciphertexts growing with $O(n^2)$ instead of with $O(n)$ like the R-LWE variant (with n the dimension of the lattice), thus having a higher possibility to need a greater amount of operations and therefore computation time. In the world of threshold encryption based on R-LWE, as far as we know there is only one proposal given in [9], which is based on the homomorphic properties of the presented Fully Homomorphic Encryption scheme. However, this proposal does not come with a distributed key generation protocol (as it relies on a Trusted Third-Party (TTP) for that). Additionally, to the best of our knowledge, none of the mentioned proposals give an implementation to truly analyze computation times, even if implementations of post-quantic protocols is a hot topic, as shown by the Open Quantum Safe project [10]. Furthermore, the Open Quantum Safe project also shows us that despite recent developments, most applications are still on the prototype phase. A summary of the current state of the art can be found in Table 1.

Table 1. State-of-the-art.

Proposal	Lattice Problem	Key Generation	Implementation
[6]	LWE	✓	✗
[7]	LWE	✗	✗
[8]	LWE	✓	✗
[5]	LWE	✗	✗
[9]	R-LWE	✗	✗
Our proposal	R-LWE	✓	✓

1.2. Contributions

In this work, we present original protocols of both distributed key generation and threshold decryption that, as far as we know, are the first R-LWE-based threshold protocols including both decryption and key generation. The protocols are based on the LWE proposal given by Bendlin and Damgård in [6], whose ideas are transported into the R-LWE setting. Furthermore, we prove these protocols both correct and secure, we give a set of parameters for which our protocols have more than 100 bits of security and we give a rough implementation in C of the protocols to analyze their performance.

1.3. Structure

In Section 2 we will give the preliminaries necessary to follow the work in the fields of cryptographic primitives, distributed cryptography, and Ring Learning with Errors. In Section 3, we present the protocols that act as the main contribution of this work, while in Section 4 we will prove their correctness, and in Section 5 we will prove them secure. Section 6 will be dedicated to analyzing the implementation in C of the protocols, and finally in Section 7 we will give our final concluding thoughts and possible future work. For the appendices, in Appendix A we give correctness and security proofs against an adversary acting actively in both protocols, in Appendix B we give the proofs for auxiliary results, and in Appendix C we give the link to a repository containing the relevant code of our implementation.

Most of the introduction has been taken from the introduction in the Master's Thesis [11] by Ferran Alborch.

2. Preliminaries

2.1. Notation

Elements in $\mathbb{R}$, $\mathbb{Z}$, or $\mathbb{Z}_q$ will be indicated as lower case letters $(a, b, \ldots)$, while elements in $\mathbb{R}^n$, $\mathbb{Z}^n$, or $\mathbb{Z}_q^n$ will be indicated as bold lower case letters $(\boldsymbol{a}, \boldsymbol{b}, \ldots)$. We will consider $\mathbb{Z}_q$ with the representatives in $\left[-\frac{q}{2}, \frac{q}{2}\right)$. Let X be a random variable, $X \sim \chi$ means X follows the probability distribution χ, $x \longleftarrow \chi$ means x is sampled from a random variable following the distribution χ, $Y \longleftarrow \chi^n$ means Y is a vector such that every coordinate is independently sampled from the distribution χ and for any set $\mathcal{J}$, $j \overset{\$}{\leftarrow} \mathcal{J}$ is the action of choosing j uniformly at random from $\mathcal{J}$. We will also identify any polynomial of degree $n-1$, $f(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} \in \mathbb{Z}_q[x]$ with the vector $\boldsymbol{f} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}_q^n$. A function g is said to be negligible over n ($g := \mathrm{neg}(n)$) if $\forall k \in \mathbb{Z}_{>0}$, $\exists n_0 \in \mathbb{Z}_{>0}$ such that $\forall n \geq n_0$, $|g(n)| < \frac{1}{n^k}$. Finally, a function f is said to be $O(g(n))$ if there exists $M > 0$ and n_0 such that $|f(n)| \leq M \cdot g(n)$ for all $n > n_0$, while a function f is said to be $\omega(g(n))$ if for all $M > 0$ exists n_0 such that $|f(n)| > M \cdot g(n)$ for all $n > n_0$.

2.2. Cryptographic Primitives

We will start by giving some cryptographic primitives, well-known definitions, protocols, or techniques used in cryptography upon which we will build our encryption scheme and protocols. First, we will formally define what an encryption scheme is.

Definition 1. *An encryption scheme is a tuple $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ such that:*

- *$\mathcal{M}$ is a set called plaintext space.*
- *$\mathcal{C}$ is a set called ciphertext space.*
- *$\mathcal{K}$ is a set called key space. Generally a key generation procedure is also specified to generate $k \in \mathcal{K}$.*
- *$\mathcal{E} = \{\mathrm{E}_k : k \in \mathcal{K}\}$ is a set of functions $\mathrm{E}_k : \mathcal{M} \times \mathcal{R} \to \mathcal{C}$ called encryption functions. $\mathcal{R}$ is a randomness space to account for probabilistic encryption schemes.*
- *$\mathcal{D} = \{\mathrm{D}_k : k \in \mathcal{K}\}$ is a set of functions $\mathrm{D}_k : \mathcal{C} \to \mathcal{M}$ called decryption functions.*

Note that if $\mathcal{D}$ and $\mathcal{E}$ use the same key, then we call it symmetric encryption; otherwise, we call it asymmetric or public key encryption. In public key encryption, $\mathcal{K}$ can be divided into two different sets, $\mathcal{K}_s$, the secret key space, and $\mathcal{K}_p$, the public key space. The public key (known by all entities) is used to encrypt messages and the secret key (known only to the decrypting entity) is used to decrypt them.

Once we have defined a cryptosystem, we need to prove some properties about it to ensure it is useful. The most important of these properties are correctness and security, and from now on we will focus solely on public key encryption.

Definition 2. *Let $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. The encryption scheme is said to be correct if for all $e \in \mathcal{K}_p$ exists some computable $d \in \mathcal{K}_s$ such that, given λ the security parameter, the probability*

$$\Pr[D_d(E_e(m)) \neq m] = neg(\lambda)$$

for all $m \in \mathcal{M}$.

In contrast to the correctness of an encryption scheme, security can be defined in various ways. This is due to the fact that a decryption should always be correct (or always, except with negligible probability) but security depends on the properties of the adversary we want to protect us against (information available, computational power) and what we want to ensure (that the adversary cannot know what message was encrypted or that he cannot distinguish which message has been encrypted from a pool of plaintexts). In our case, we are interested in CPA security.

Attack Game 1 (Attack Games 5.2 and 11.2, [12]). *Let $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Given an adversary $\mathcal{A}$, the Chosen Plaintext Attack (CPA) attack game, has two experiments: Experiment 0 and Experiment 1. For $b \in \{0,1\}$ we define Experiment b as*

- *The challenger chooses $e \overset{\$}{\leftarrow} \mathcal{K}_p$ and sends it to the adversary.*
- *The adversary submits polynomially many queries to the challenger. For $i = 1, 2, \ldots$, $\mathcal{A}$ submits two same-length messages $m_{0_i}, m_{1_i} \in \mathcal{M}$. The challenger computes $c_i = E_e(m_{b_i})$ and sends it to the adversary.*
- *The adversary outputs a bit $\hat{b} \in \{0,1\}$.*

Let W_b be the event in which $\mathcal{A}$ outputs 1 in the event b, then we define $\mathcal{A}$'s CPA advantage *as*

$$CPAAdv[\mathcal{A}, \mathcal{S}] := |\Pr[W_0] - \Pr[W_1]|.$$

Definition 3 (Definitions 5.2 and 11.4, [12]). *An encryption scheme $\mathcal{S}$ is said to be* CPA secure *if for all efficient adversaries $\mathcal{A}$, the value $CPAAdv[\mathcal{A}, \mathcal{S}]$ is negligible.*

Furthermore, security can be achieved against different types of adversaries depending on what capacities they have. We will focus basically on *passive* adversaries (also known as honest but curious), who can see all the information the corrupted players have but cannot make them deviate from the protocol, and *active* adversaries (also known as malicious), who can make the corrupted players deviate arbitrarily from the protocol. Another important distinction to make is whether it is a *static* adversary, who picks the subset of players it will corrupt prior to the start of the protocol; or a *dynamic* adversary, who can change the subset of corrupted players during the execution of the protocol. Our security model will be against static adversaries.

2.3. Distributed Cryptography

The specific branch of cryptography we are interested in this work is distributed cryptography.

Definition 4 ([13]). *A threshold secret sharing scheme of threshold t and u players is a scheme such that given some data D it divides it into u pieces $D_1 \ldots, D_u$ such that:*

- *Knowledge of $t+1$ or more pieces D_i makes D easily computable.*
- *Knowledge of t or less pieces D_i leaves D completely undetermined (i.e., all its possible values are equally likely).*

Definition 5. *We will call a threshold encryption scheme a secret sharing scheme where what we try to recover is a plaintext from a ciphertext.*

One of the first secret sharing schemes and one of the most used up to this day due to its simplicity to compute and understand, is Shamir Secret Sharing. We will use it profusely throughout our work.

Technique 1 ([13]). *Shamir Secret Sharing over a field $\mathbb{F}$ of a secret $s \in \mathbb{F}$ of threshold t works as follows:*

- *Choose t elements $b_i \in \mathbb{F}$ and define the polynomial $f(x) := s + \sum_{i=1}^{t} b_i x^i$ (i.e., choose a random polynomial $f(x) \in \mathbb{F}[x]$ such that $f(0) = s$).*
- *For every player P_j, their share of the secret is $f(i_j)$, with $i_j \in \mathbb{F} \setminus \{0\}$ being different for every player and agreed before-hand.*
- *When $t+1$ players want to recover the secret, they use Lagrange interpolation to find $f(x)$ and then compute $f(0)$.*

The convenience of this secret sharing scheme lays in two main properties: the recovery of the secret is done through Lagrange interpolation (which is easy to compute) and the shares are linear, which means that a linear combination of the shares is a share of the linear combination of secrets. Both of these properties will be used in our work.

Other distributed cryptographic tools we will use are the Pseudo-Random Secret Sharing (PRSS) and the Non-Interactive Verifiable Secret Sharing (NIVSS) techniques. These tools will be crucial in our proposal, as the security of our protocols is based on being able to mask the relevant information with noise in such a way that the adversary cannot retrieve it. To generate this noise we will use these two protocols.

Definition 6. *A Pseudo-Random Function (PRF), $\Phi_\cdot(\cdot)$, is a deterministic function that maps two sets (domain and range) on the basis of a key, which when run multiple times with the same input gives the same output but given an arbitrary input the output seems random, i.e., one cannot distinguish the output of a given input from a random oracle. A* random oracle *is an oracle (a theoretical black box) that responds to every query with a (truly) random response chosen uniformly from its output domain. If a query is repeated, it responds the same way every time that query is submitted.*

Technique 2 ([14]). *Pseudo-Random Secret Sharing in $\mathbb{Z}_q$* (PRSS) *allows u players to non-interactively share a common random value x with a threshold of t players ($t < u$) given a pseudo-random function $\Phi_\cdot(\cdot)$ that with input a key and a value μ outputs values in the interval $I = [a, b]$, $a < 0, b > 0$ and whatever group of players of size less or equal than t cannot obtain relevant information on x. The algorithm works as follows:*

- *For each subset H of t players a TTP defines a key $K_H \in \mathbb{Z}_q$ uniformly at random.*
- *Each player P_j is given K_H, $\forall H$ such that $P_j \notin H$.*
- *The pseudo-random number they are sharing is*

$$x := \sum_H \Phi_{K_H}(\mu)$$

  for a value μ. Since there are $\binom{u}{t}$ such subsets H, we know $x \in [\binom{u}{t}a, \binom{u}{t}b]$.
- *To compute x^j a Shamir share of x every player computes*

$$x^j = \sum_{H \not\ni P_j} \Phi_{K_H}(\mu) \cdot f_H(j)$$

  where $f_H(x)$ is the unique degree-t polynomial (in our case $f_H(x) \in \mathbb{Z}_q[x]$ with q prime) such that $f_H(0) = 1$ and $f_H(i) = 0$ for all $P_i \in H$.

Technique 3 ([14]). *Non-Interactive Verifiable Secret Sharing in $\mathbb{Z}_q$ (NIVSS), allows a dealer D to share a secret s with u players with threshold t given a value μ and a pseudo-random function $\phi_\cdot(\cdot)$ that with input a key and μ outputs values in the interval $I = [a, b]$, $a < 0, b > 0$. It works very similarly to PRSS. The algorithm works as follows:*

1. *For each subset H of t players the dealer D chooses a key $K_H \in \mathbb{Z}_q$ uniformly at random.*
2. *The dealer D gives to player P_j all the K_H such that $P_j \notin H$.*
3. *The dealer D reconstructs the pseudo-random value the players share $x = \sum_H \phi_{K_H}(\mu)$, since he has all the keys.*

4. *D broadcasts the value $s - x$, and now all the players have a share of s by adding their shares of x to $s - x$.*

Finally, as we will be using distributed methods, one must ensure that the order in which the different players send information does not compromise the security of the scheme. Broadly speaking, the last player to send information would have an advantage with respect to the first one due to knowing more information when making its decision. To solve this problem, it is standard to use commitment schemes.

Definition 7 (Definition 8.8, [12])**.** *Given a message space $\mathcal{M}$, a commitment scheme is a pair of efficient algorithms $\mathcal{C} = (C, V)$ where C (the commiter) is an algorithm that given $m \in \mathcal{M}$ outputs a commitment c and an opening string o and V (the verifier) is a deterministic protocol that given (m, c, o) outputs* accept *or* reject*, and such that it satisfies the following properties:*

- *Correctness: For all $m \in \mathcal{M}$, if $C(m) = (c, o)$ then*

$$\Pr[V(m, c, o) = \text{'accept'}] = 1.$$

- *Binding: This property is the notion that once a commitment c is generated, it should only commit for one message in $\mathcal{M}$. In particular, for every efficient adversary $\mathcal{A}$ that outputs (c, m_1, o_1, m_2, o_2) we must have that*

$$\Pr\left[\begin{array}{c} m_1 \neq m_2 \text{ and} \\ V(m_1, c, o_1) = V(m_2, c, o_2) = \text{'accept'} \end{array}\right] = neg(\lambda).$$

- *Hiding: This property is the notion that the commitment c alone should not reveal any information about the message m. To properly define this, we use a semantic security attack game (see Attack Game 2.1, [12]) where instead of encrypting the messages we compute its commitment. What we ask is, if W_b denotes the event that the adversary outputs 1 in experiment b, then*

$$|\Pr[W_0] - \Pr[W_1]| = neg(\lambda).$$

2.4. Ring Learning with Errors

The Learning with Errors (LWE) problem was introduced by Regev in 2005 in a previous version of [15] as a generalization of the parity learning problem, and gave both a cryptographic protocol based on it and a reduction of its security to a hard lattice problem (the GAP Shortest Vector Problem (GAPSVP)).

However, cryptosystems based on the LWE problem have several issues. For example, many of them need to encrypt bit by bit and, more importantly, the public keys required are very costly to store as they are usually (big) matrices of elements in $\mathbb{Z}_q$. Coupling these two together we get that a lot of storage space is usually needed to encrypt small amounts of information.

To solve these problems, the Ring Learning with Errors variant was introduced by Lyubasevsky, Peikert and Regev in [16]. It is essentially a particular case of LWE but in polynomial rings over finite rings. The problem is over the polynomial ring $R_q = \mathbb{Z}_q[x]/\langle \boldsymbol{f} \rangle$, where $\boldsymbol{f}$ is a monic polynomial in $\mathbb{Z}_q[x]$.

Given an element $\boldsymbol{a}(x) \in R_q$, one can see the principal ideal generated by $\boldsymbol{a}(x)$

$$\langle \boldsymbol{a}(x) \rangle = \{\boldsymbol{c}(x) \in R_q | \boldsymbol{c}(x) = \boldsymbol{a}(x) \cdot \boldsymbol{b}(x),\ \boldsymbol{b}(x) \in R_q\}$$

as an ideal lattice in $\mathbb{Z}_q^n$, which we will note as $\mathcal{L}(\boldsymbol{a})$. This correspondence is easy to see in the case $\boldsymbol{f}(x) = x^n + 1$ (the particular R_q we will use given its specific properties) due to the fact that the vector of coefficients of the product of polynomials in R_q can be found through the anticyclic matrix as follows:

$$\boldsymbol{a}(x)\cdot\boldsymbol{b}(x) \pmod{x^n+1} \equiv \begin{pmatrix} a_1 & -a_n & -a_{n-1} & \dots & -a_2 \\ a_2 & a_1 & -a_n & \dots & -a_3 \\ a_3 & a_2 & a_1 & \dots & -a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & a_{n-2} & \dots & a_1 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

where $\boldsymbol{a} = (a_1, \dots, a_n)$ and $\boldsymbol{b} = (b_1, \dots, b_n)$ are the coefficients of $\boldsymbol{a}(x)$ and $\boldsymbol{b}(x)$, respectively.

With this out of the way we can finally define the Ring Learning with Errors problem, on which a lot of lattice-based cryptography is based.

Definition 8. *Let χ be a probability distribution over R_q and $\mathbf{s} \in R_q$. Then, the R-LWE distribution $A_{\mathbf{s},\chi}$ is the distribution in $R_q \times R_q$ given by $(\boldsymbol{a},\ \mathbf{b} = \boldsymbol{a}\cdot\mathbf{s} + \boldsymbol{e})$, where $\boldsymbol{a} \in R_q$ is chosen uniformly at random, $\boldsymbol{e} \longleftarrow \chi$ and all the operations to compute $\mathbf{b}$ are made in R_q.*

Definition 9. *The decisional R-LWE problem is to distinguish samples from $A_{\mathbf{s},\chi}$ from the uniform distribution in $R_q \times R_q$ with a probability that is non-negligibly bigger than $\frac{1}{2}$.*

Definition 10. *The search R-LWE problem is to find $\mathbf{s}$ given a polynomial amount of samples from $A_{\mathbf{s},\chi}$ with non-negligible probability.*

Therefore, a sample of the R-LWE distribution is a point of an ideal lattice that has been offset by a margin set by the distribution χ (which is normally taken such that the error is small). Therefore, the search R-LWE problem could be seen as finding a point in the ideal lattice $\mathcal{L}(\boldsymbol{a})$ (remember that a vector $\boldsymbol{a}$ uniquely defines an ideal lattice through its anticyclic matrix) "close" to the sample, and the decision R-LWE could be seen as given an ideal lattice $\mathcal{L}(\boldsymbol{a})$, decide whether the points given are all "close" to $\mathcal{L}(\boldsymbol{a})$ or are uniformly distributed.

When implementing LWE or R-LWE we will use a certain type of probability distribution over $\mathbb{Z}_q$ called discrete Gaussians, given their nice properties and ease in sampling values from them. There are several different definitions of discrete Gaussians but in our implementation we will use the following, given it is much easier to sample.

Definition 11 ([15])**.** *Ψ_σ, $\sigma \in \mathbb{R}^+$ is the distribution in the torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ obtained by sampling a Gaussian random variable X, $X \sim N(0,\sigma)$ centered in 0 and with standard deviation σ and then reducing modulo 1. Therefore,*

$$\Psi_\sigma(r) \;=\; \sum_{k=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{r-k}{\sqrt{2}\sigma}\right)^2},\ \forall r \in [0,1)$$

Note that if $Y \sim \Psi_\sigma$, then $Y = X \pmod 1$, with $X \sim N(0,\sigma)$ where reducing modulo 1 is taking only the decimal part of any real number.

Definition 12 ([15])**.** *The discretization to $\mathbb{Z}_q$, $q \in \mathbb{Z}_{>0}$ of any distribution in $\mathbb{T}$ ($\Psi : \mathbb{T} \to \mathbb{R}^+$), noted as $\overline{\Psi} : \mathbb{Z}_q \to \mathbb{R}^+$ is sampling from Ψ, multiplying by q, and then rounding to the closest integer. Therefore,*

$$\overline{\Psi}(i) := \int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}} \Psi(x)\,dx$$

Note that if $Z \sim \overline{\Psi}_\sigma$ then $Z = \lfloor qY \rceil \pmod q$, with $Y \sim \Psi_\sigma$.

Definition 13. *We define the truncated Discrete Gaussian of parameters σ and κ as the distribution which samples from $\overline{\Psi}_\sigma$ and rejects any sample bigger than κ, when seeing them with representatives in $\left[-\frac{q}{2}, \frac{q}{2}\right)$.*

In particular, we will use $\kappa \in \mathbb{Z}_{>0}$ such that

$$\Pr\left[\left|\overline{\Psi}_\sigma\right| > \kappa\right] \leq 2^{-\lambda}$$

being λ the security parameter.

3. Encryption Scheme and Protocols

Having given all the necessary preliminaries, we can finally present our encryption scheme, threshold decryption protocol and distributed key generation protocol, which we will prove correct in Section 4, secure in Section 5, analyze its implementation in Section 6 and give some last thoughts and possible future work in Section 7.

We will use a version of the LPR encryption scheme presented in [16].

Encryption Scheme 1. *Let $q, n, u \in \mathbb{Z}_{>0}$, where u is the number of players, and χ be a distribution over R_q. The encryption scheme $\mathcal{S} = (\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and key generation we will be using is the following:*

- *$\mathcal{M} = \{0,1\}^n \subseteq \mathbb{Z}_q^n \cong R_q$. We will see every $\boldsymbol{m} \in \mathcal{M}$ as an element in R_q with $\boldsymbol{m}$ being its vector of coefficients.*
- *$\mathcal{C} \subseteq R_q \times R_q$.*
- *This is a public encryption scheme, we have $\mathcal{K}_s \subseteq R_q$ and $\mathcal{K}_p \subseteq R_q \times R_q$.*
  - *For any pair of keys $(\boldsymbol{pk}, \boldsymbol{s}) \in \mathcal{K}_p \times \mathcal{K}_s$ we will have $\boldsymbol{s} \longleftarrow \sum_{i=1}^{u} \chi$ (meaning it is the sum of u samples of χ) and $\boldsymbol{pk} = (\boldsymbol{a_E}, \boldsymbol{b_E}) = (\boldsymbol{a_E}, \boldsymbol{a_E} \cdot \boldsymbol{s} + \boldsymbol{e})$ where $\boldsymbol{a_E} \overset{\$}{\longleftarrow} R_q$ and $\boldsymbol{e} \longleftarrow \sum_{i=1}^{u} \chi$.*
- *$\mathcal{E} = \{\mathrm{E}_{\boldsymbol{pk}} : \boldsymbol{pk} = (\boldsymbol{a_E}, \boldsymbol{b_E}) \in \mathcal{K}_p\}$ such that given a message $\boldsymbol{m} \in \mathcal{M}$:*

$$\begin{aligned} \mathrm{E}_{\boldsymbol{pk}} : \mathcal{M} &\to \mathcal{C} \\ \boldsymbol{m} &\mapsto (\boldsymbol{u}, \boldsymbol{v}) \end{aligned}$$

  where $(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{a_E} \cdot \boldsymbol{r_E} + \boldsymbol{e_u},\ \boldsymbol{b_E} \cdot \boldsymbol{r_E} + \boldsymbol{e_v} + \boldsymbol{m} \cdot \lfloor \frac{q}{2} \rfloor)$ with $\boldsymbol{r_E}, \boldsymbol{e_u}, \boldsymbol{e_v} \longleftarrow \chi$.
- *$\mathcal{D} = \{\mathrm{D}_{\boldsymbol{s}} : \boldsymbol{s} \in \mathcal{K}_s\}$ such that given a ciphertext $(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{C}$:*

$$\begin{aligned} \mathrm{D}_{\boldsymbol{s}} : \mathcal{C} &\to \mathcal{P} \\ (\boldsymbol{u}, \boldsymbol{v}) &\mapsto \boldsymbol{m} \end{aligned}$$

  where we will recover every bit of $\boldsymbol{m}$ by rounding every coefficient of

$$\boldsymbol{v} - \boldsymbol{s} \cdot \boldsymbol{u} = \boldsymbol{e} \cdot \boldsymbol{r_E} + \boldsymbol{e_v} - \boldsymbol{s} \cdot \boldsymbol{e_u} + \boldsymbol{m} \cdot \left\lfloor \frac{q}{2} \right\rfloor$$

  to 0 or $\lfloor \frac{q}{2} \rfloor \pmod q$ and then mapping 0 to 0 and $\lfloor \frac{q}{2} \rfloor$ to 1.

Now, we will define the Threshold Decryption Protocol based on this encryption scheme and a Distributed Key Generation protocol to work together with it. For clarity, we use a TTP to generate the keys in the encryption protocol, however, what we are looking for is a totally distributed scheme, so we also define a Distributed Key Generation Protocol to take the place of the TTP in the threshold decryption protocol.

Protocol 1. *Let χ be a distribution over R_q and $\Phi_{\cdot}(\cdot)$ a pseudo-random function with image in $\mathbb{I}_D^n$, that is a vector of n coordinates with each coordinate being in $\mathbb{I}_D$ an integer interval. Then the Threshold Decryption Protocol works as follows:*

1. *A TTP generates the keys $K_H \in \mathbb{Z}_q$ for every subset H of players of size t and distributes them according to the PRSS technique (Technique 2). It also generates the secret key $\boldsymbol{s} \sim \sum_{i=1}^{u} \chi$ and the public key $(\boldsymbol{a_E}, \boldsymbol{b_E})$ as stated in the Encryption Scheme 1. Then, the TTP sends to*

the players $(\boldsymbol{a_E}, \boldsymbol{b_E})$ and Shamir shares of $\boldsymbol{s}$. We call $\boldsymbol{s}^j$ the Shamir share of $\boldsymbol{s}$ of player P_j, understood as a vector of Shamir shares on the coefficients of $\boldsymbol{s}$.

2. *Client receives ciphertext $\boldsymbol{c} = (\boldsymbol{u}, \boldsymbol{v})$, and sends all players $\boldsymbol{c}$.*
3. *Each player P_j computes $\tilde{\boldsymbol{e}}^j = \boldsymbol{v} - \boldsymbol{s}^j \cdot \boldsymbol{u}$ that is a Shamir share of $\tilde{\boldsymbol{e}} = \boldsymbol{e} \cdot \boldsymbol{r_E} + \boldsymbol{e_v} - \boldsymbol{s} \cdot \boldsymbol{e_u} + \boldsymbol{m} \cdot \lfloor \frac{q}{2} \rfloor$ with $\boldsymbol{e}, \boldsymbol{r_E}, \boldsymbol{e_v}, \boldsymbol{s}, \boldsymbol{e_u} \leftarrow \chi$.*
4. *Each player P_j computes $\boldsymbol{x}^j$, as in the PRSS protocol but using $\mu = \boldsymbol{u} + \boldsymbol{v}$ (since it changes for every message and it is hard to distinguish from uniformly at random), its Shamir share of $\boldsymbol{x} := \sum_H \Phi_{K_H}(\boldsymbol{u} + \boldsymbol{v})$ and gets $\boldsymbol{x}^j + \tilde{\boldsymbol{e}}^j$ Shamir share of $\boldsymbol{x} + \tilde{\boldsymbol{e}}$.*
5. *Client reconstructs $\boldsymbol{x} + \tilde{\boldsymbol{e}}$ for every allowed subset of $t+1$ players, picks whichever value is repeated more times, then for every coefficient returns 0 if $\boldsymbol{x} + \tilde{\boldsymbol{e}}$ is closer to 0 than to $\lfloor \frac{q}{2} \rfloor$ and returns 1 otherwise, and this is made public.*

See Table 2.

Table 2. Decryption protocol.

Decryption Protocol	
Inputs: $\boldsymbol{s}^j$, K_H s.t. $j \notin H$, $\Phi_\cdot(\cdot)$, $\boldsymbol{f}_H$	
Player $\boldsymbol{P_j}$	
	$\xleftarrow{(\boldsymbol{u},\boldsymbol{v})}$
$\boldsymbol{x}^j = \sum_{H \notin j} \Phi_{K_H}(\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{f}_H(j)$ $\tilde{\boldsymbol{e}}^j = \boldsymbol{v} - \boldsymbol{s}^j \cdot \boldsymbol{u}$	
	$\xrightarrow{\boldsymbol{x}^j + \tilde{\boldsymbol{e}}^j}$

For the key generation protocol, we will assume a commitment scheme is used in the initial steps of interaction, when all the sampling is done and sent.

Protocol 2. *Let χ be a distribution over R_q, $\mu = x + 2x^2 + \ldots + (n-1)x^{n-1} \in R_q$, and $\Phi^{KG}_\cdot(\cdot)$ a pseudo-random function with image in $\mathbb{I}^n_{KG}$, where $\mathbb{I}_{KG}$ is an integer interval. The* Distributed Key Generation Protocol *works as follows:*

1. *For the secret key $\mathbf{s} \in R_q$, each player P_j chooses its contribution $\mathbf{s}_j = (s_{1_j}, \ldots, s_{n_j})$ with $\mathbf{s}_j \sim \chi$. Then, they act as the dealer in a NIVSS (Technique 3) to share every s_{i_j} to all players. All players verify the value broadcast when doing the NIVSS $(s_{i_j} - \sum_H \phi^{KG}_{K^s_{N_{H_j}}}(\mu))$ is in the interval $\binom{u}{t}\mathbb{I}_{KG}$. Now all players have shares of every s_{i_j} and by their linearity also of $s_i = \sum_j s_{i_j}$. Then, $\mathbf{s}$ is the polynomial in R_q with coefficients $(s_1, \ldots, s_n)$.*
2. *For the keys $K_H \in \mathbb{Z}_q$ that will be used for the PRSS in the threshold decryption, for every subset H of t players each player P_j chooses uniformly at random $K_{H_j} \in \mathbb{Z}_q$ their contribution on these keys and shares it with all the players using Shamir secret sharing. Then, the players will have, by adding all the shares received by other players, Shamir shares of $K_H = \sum_j K_{H_j}$. Finally, all players send privately their shares on K_H to all the players in A the complement of H, so they can recover K_H.*
3. *For the contributions to $\boldsymbol{e} \in R_q$ proceed identically to when generating $\mathbf{s}$.*
4. *For $\boldsymbol{a_E} \in R_q$ every player P_j chooses its share $(a_{E,1_j}, \ldots, a_{E,n_j})$ randomly in R^n_q and does a Shamir share of it. Then, all players send to all players their share on all the $(a_{E,1_j}, \ldots, a_{E,n_j})$ so every player can recover (by adding the shares) $\left(\sum_j a_{E,1_j}, \ldots, \sum_j a_{E,n_j}\right)$. The polynomial in R_q with these coefficients will be $\boldsymbol{a_E}$.*
5. *Every player computes locally their Shamir shares on $\boldsymbol{b_E} = \boldsymbol{a_E} \cdot \mathbf{s} + \boldsymbol{e}$ by performing these same operations with the shares they have on $\mathbf{s}$ and $\boldsymbol{e}$.*
6. *Finally, the public key $(\boldsymbol{a_E}, \boldsymbol{b_E})$ is made public.*

See Table 3 for a more detailed look into the steps of interaction needed. Note that we will denote with subindexes the additive contributions and with superindexes the Shamir shares.

Table 3. Key Generation protocol.

Key Generation Protocol	
Inputs: χ, $\Phi^{KG}_{\cdot}(\cdot)$, μ	
Player P_j	
$\boldsymbol{s}_j, \boldsymbol{e}_j \leftarrow \chi$	
$K_{H_j}, K^s_{N_{H_j}}, K^e_{N_{H_j}} \xleftarrow{\$} \mathbb{Z}_q \; \forall \vert H\vert = n - t$	
$\mathring{\boldsymbol{s}}_j = \boldsymbol{s}_j - \sum_H \Phi^{KG}_{K^s_{N_{H_j}}}(\mu)$	
$\mathring{\boldsymbol{e}}_j = \boldsymbol{e}_j - \sum_H \Phi^{KG}_{K^e_{N_{H_j}}}(\mu)$	
$\boldsymbol{a}_{\boldsymbol{E}j} \xleftarrow{\$} R_q$	
$K'_{H_j}, \boldsymbol{a}_{\boldsymbol{E}}{}'_j = \text{Shamir.Shares}(K_{H_j}, \boldsymbol{a}_{\boldsymbol{E}j})$	
$\mathcal{C}_j = \text{Commit}(\mathring{\boldsymbol{s}}_j, \mathring{\boldsymbol{e}}_j, K^s_{N_{H_j}}, K^e_{N_{H_j}}, K'_{H_j}, \boldsymbol{a}_{\boldsymbol{E}}{}'_j)$	
	$\xrightarrow{\mathcal{C}_j}$
	$\xleftarrow{\{\mathcal{C}_k\}_{k=1}^u}$
	$\xrightarrow{\mathring{\boldsymbol{s}}_j, \mathring{\boldsymbol{e}}_j, K^s_{N_{H_j}}, K^e_{N_{H_j}}, K'_{H_j}, \boldsymbol{a}_{\boldsymbol{E}}{}'_j}$
	$\xleftarrow{\left\{\mathring{\boldsymbol{s}}_k, \mathring{\boldsymbol{e}}_k, K^s_{N_{H_k}}, K^e_{N_{H_k}}, K'_{H_k}, \boldsymbol{a}_{\boldsymbol{E}}{}'_k\right\}_{k=1}^u}$
$\left\{\text{Verify}(\mathcal{C}_k)_j = \{\text{'accept' or 'reject'}\}\right\}_{k=1}^u$	
	$\xrightarrow{\left\{\text{Verify}(\mathcal{C}_k)_j\right\}_{k=1}^u}$
	$\xleftarrow{\{\text{Verify}(\mathcal{C}_i)_k\}_{i,k=1}^u}$
if $\text{Verify}(\mathcal{C}_i)_k = \text{'reject'}$ for some i, k abort	
$\left\{\text{Verify.interval}(\mathring{\boldsymbol{s}}_k)_j = \{\text{'accept' or 'reject'}\}\right\}_{k=1}^u$ $\left\{\text{Verify.interval}(\mathring{\boldsymbol{e}}_k)_j = \{\text{'accept' or 'reject'}\}\right\}_{k=1}^u$	
$\boldsymbol{s}^j = \sum_{k=1}^u \mathring{\boldsymbol{s}}_k + \sum_{H \not\ni P_j} \Phi^{KG}_{K^s_{N_{H_j}}}(\mu) \cdot f_H(j)$	
$\boldsymbol{e}^j = \sum_{k=1}^u \mathring{\boldsymbol{e}}_k + \sum_{H \not\ni P_j} \Phi^{KG}_{K^e_{N_{H_j}}}(\mu) \cdot f_H(j)$	
$K^j_H = \sum_{k=1}^u K'_{H_k}$	
$\boldsymbol{a}^j_{\boldsymbol{E}} = \sum_{k=1}^u \boldsymbol{a}_{\boldsymbol{E}}{}'_k$	
	$\xrightarrow{\left\{\text{Verify.interval}(\mathring{\boldsymbol{s}}_k)_j, \text{Verify.interval}(\mathring{\boldsymbol{e}}_k)_j\right\}_{k=1}^u}$
	$\xleftarrow{\{\text{Verify.interval}(\mathring{\boldsymbol{s}}_i)_k, \text{Verify.interval}(\mathring{\boldsymbol{e}}_i)_k\}_{i,k=1}^u}$

if $\text{Verify.interval}(\mathring{\boldsymbol{s}}_i)_k = \text{'reject'}$ for some i, k abort
if $\text{Verify.interval}(\mathring{\boldsymbol{e}}_i)_k = \text{'reject'}$ for some i, k abort	

	$\xrightarrow{\boldsymbol{a}^j_{\boldsymbol{E}}, K^j_H \text{ to } k \text{ s.t. } H \not\ni P_k}$
	$\xleftarrow{\left\{\boldsymbol{a}^k_{\boldsymbol{E}}, K^k_H \text{ s.t. } H \not\ni P_j\right\}_{k=1}^u}$
$K_H = \text{Reconstruct.Shamir}(K^k_H)$ s.t. $H \not\ni P_j$	
$\boldsymbol{a}_{\boldsymbol{E}} = \text{Reconstruct.Shamir}(\boldsymbol{a}^k_{\boldsymbol{E}})$	
$\boldsymbol{b}^j_{\boldsymbol{E}} = \boldsymbol{a}_{\boldsymbol{E}} \cdot \boldsymbol{s}^j + \boldsymbol{e}^j$	
	$\xrightarrow{\boldsymbol{a}_{\boldsymbol{E}}, \boldsymbol{b}^j_{\boldsymbol{E}}}$
	$\xleftarrow{\left\{\boldsymbol{b}^k_{\boldsymbol{E}}\right\}_{k=1}^u}$
$\boldsymbol{b}_{\boldsymbol{E}} = \text{Reconstruct.Shamir}(\boldsymbol{b}^k_{\boldsymbol{E}})$	
	$\xrightarrow{\boldsymbol{b}_{\boldsymbol{E}}}$

4. Correctness

With all the preliminaries on hand and having defined both protocols we can now proceed to prove their correctness. We will give the proof for the case of a passive adversary in the Key Generation phase and an active (or passive) adversary in the Decryption phase, as this will be the case our implementation will use, the reasons for this decision will be explained in Section 6.1. The proof for the case where there is an active adversary during the Key Generation Protocol will be in Appendix A.1.

Theorem 1. *Let $n, q, u \in \mathbb{Z}_{>0}$, $n = 2^{\beta}$ and u being the number of players. Let $R_q = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$, $\Phi^D_{\cdot}(\cdot)$ be a pseudo-random function with image interval $\mathbb{I}_D^n$ where*

$$\mathbb{I}_D = \left[-(2nu\kappa^2 + \kappa) \cdot 2^{\lambda+\beta}, (2nu\kappa^2 + \kappa) \cdot 2^{\lambda+\beta}\right],$$

χ be a n-dimensional distribution obtained by n independent truncated Discrete Gaussian with parameters σ and κ and

$$\left\lfloor \frac{q}{4} \right\rfloor \geq (2nu\kappa^2 + \kappa)\left(\binom{u}{t} 2^{\lambda+\beta} + 1\right).$$

Then Protocol 1 will have correct output against an active static adversary corrupting up to $t < \frac{u}{3}$ players.

Proof. What we want to see first is that $|\boldsymbol{x} + \hat{\boldsymbol{e}}|_i \leq \frac{q}{4}$ $\forall i$, where $\cdot_i$ notes the coefficient i on the polynomial, given the way the decryption works in Encryption Scheme 1.

Let $\hat{\boldsymbol{e}} = \boldsymbol{e} \cdot \boldsymbol{r_E} + \boldsymbol{e_v} - \boldsymbol{s} \cdot \boldsymbol{e_u}$. Since the product in R_q is done through the anticyclic matrix, we know that

$$|\hat{\boldsymbol{e}}|_i \leq |e_i \cdot r_{E_1}| + |e_{i-1} \cdot r_{E_2}| + \ldots + |e_{i+2} \cdot r_{E_{n-1}}| + |e_{i+1} \cdot r_{E_n}| + |e_{v_i}| + |s_i \cdot e_{u_1}| + \ldots + |s_{i+1} \cdot e_{u_n}|$$

and, therefore, as $\boldsymbol{r_E}, \boldsymbol{e_v}, \boldsymbol{e_u} \leftarrow \chi$ and $\boldsymbol{s}, \boldsymbol{e} \leftarrow \sum_u \chi$, where every coefficient of χ is truncated by κ, we get that

$$|\hat{\boldsymbol{e}}|_i \leq 2nu\kappa^2 + \kappa$$

Furthermore, given that there are $\binom{u}{t}$ keys K_H, we know that

$$\boldsymbol{x}_i \in \binom{u}{t} \mathbb{I}_D.$$

Adding both results, we then get that

$$\begin{aligned} |\boldsymbol{x} + \hat{\boldsymbol{e}}|_i &\leq \binom{u}{t}(2nu\kappa^2 + \kappa) \cdot 2^{\lambda+\beta} + (2nu\kappa^2 + \kappa) \\ &= (2nu\kappa^2 + \kappa)\left(\binom{u}{t} 2^{\lambda+\beta} + 1\right) \\ &\leq \left\lfloor \frac{q}{4} \right\rfloor \end{aligned}$$

as we wanted to see.

Finally, we just need to see that when the client reconstructs, there is indeed a majority of correct results, and this derives directly from having at most t corrupt players, therefore there will be a majority of subsets of $t + 1$ players where they are all honest, and thus output the correct decryption. □

For the case where we combine both protocols, which means that we replace the TTP in Protocol 1 with Protocol 2, the outputs of this protocol and the TTP are equally generated for the case of a passive adversary in the Key Generation phase, as it cannot make any player deviate from the protocol. Therefore, we can directly apply Theorem 1. Furthermore, the same theorem and proof is valid against a passive adversary corrupting up to $t = u - 1$ players, only noting that we will have all of the decryptions correct as a passive adversary cannot make any player deviate from the protocol.

5. Security

We will divide the proofs of security for the protocols into several theorems to ease the proofs. First, we will prove the CPA security of Encryption Scheme 1 as a one-player scheme, and then we will prove that no information is leaked when distributing the protocols. Finally, we will add everything to prove the CPA security of both protocols used together.

5.1. Security of Encryption Scheme

We will split the proof of security of Encryption Scheme 1 in three distinct parts: reducing the security of the encryption scheme to the decisional R-LWE problem, reducing the R-LWE problem with the $\overline{\Psi}^n$ distribution to the R-LWE problem with truncated discrete Gaussian, and finally reducing the decisional R-LWE problem to the Discrete Gaussian Sampling over K (K-DGS) with K the field such that R is its ring of integers, a well-known lattice problem assumed to be hard to solve. We will make this splitting because the first reduction will be for any distribution χ, while the second reduction will be specifically for the distribution $\overline{\Psi}^n$. The first reduction follows the ideas from the reduction of Regev's encryption scheme to LWE given in [15]. For the detailed proof see Appendix B.

Theorem 2. *Given χ a distribution over R_q, there exists a reduction to the semantic security of the Encryption Scheme 1 from the decisional R-LWE problem with distribution χ.*

Note that the reduction is to the semantic security of the scheme and not the CPA security. However, it is well-known that in public key encryption both notions are equivalent (see, for example, Theorem 11.1 in [12]).

Second, we want to be able to ensure that if we know how to solve an instance of the decision R-LWE problem with a truncated discrete Gaussian we can solve an instance of the decision R-LWE problem with the $\overline{\Psi}^n$ distribution. This is clearly so given an instance of the decision R-LWE problem with the $\overline{\Psi}^n$ distribution one can see it as an instance with the truncated discrete Gaussian distribution except for a negligible amount of times. Therefore, the advantage of the adversary solving both instances will differ at most a negligible amount, as we needed.

Finally, we need to see that our R-LWE instance is as hard to solve as a lattice problem, in our case as hard to solve as K-DGS, where K is the field such that R is its ring of integers, in other words, $R = \mathcal{O}_K$ (for more detail in the definition of K-DGS refer to Definition 2.10 and Section 2.3.3 in [17]). This job has already been done in [17], though to do so properly we need to give some clarifications about different ways to define the R-LWE distribution.

Let K be a number field with R its ring of integers. Let $R^\vee$ be the fractional co-differential ideal of K ($R^\vee = \{x \in K \mid \mathrm{Tr}(xR) \subset \mathbb{Z}\}$), and let $\mathbb{T}^R = K_\mathbb{R}/R^\vee$. Let $q \geq 2$ be an integer modulus. Let us unpack this. First, in our specific case of K being a cyclotomic field with $n = 2^k$ for some k, we have $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$, so in turn it can be seen that $R^\vee$ is isomorphic to R. Second, $K_\mathbb{R} = K \otimes_\mathbb{Q} \mathbb{R}$ which is isomorphic to $\mathbb{R}^n$, so looking it component by component $\mathbb{T}^R$ could be seen as isomorphic to $\mathbb{T}^n$ with $\mathbb{T} = \mathbb{R}/\mathbb{Z}$. With this out of the way we can see their definition.

Definition 14 (Definition 2.14, [17]). *For $s \in R_q^\vee$ and an error distribution ψ over $K_\mathbb{R}$, the R-LWE distribution $A_{s,\psi}$ over $R_q \times \mathbb{T}^R$ is sampled by independently choosing $a \overset{\$}{\leftarrow} R_q$ and an error term $e \leftarrow \psi$, and outputting $(a, b = (a \cdot s)/q + e \mod R^\vee)$.*

Now our postulate is that this definition taking as Ψ an n-dimensional spherical continuous Gaussian with parameter ξ (which is a distribution used in [17]) and then raising it to R_q again, is a more general definition to our Definition 8 using $\overline{\Psi}_{\frac{\xi}{q}}$, in the sense that if we can solve an instance of the R-LWE problem defined with the distribution in Definition 8 we can solve an instance of the R-LWE problem with the distribution in Definition 14. It can be seen as one, as a spherical Gaussian in $\mathbb{R}^n$ can be seen as the product of n independent Gaussians over $\mathbb{R}$ with the same standard deviation. Then, in essence what we are doing in Definition 14 is multiply a times s, then divide the result by q (which we can as we are seeing the elements in $K_{\mathbb{R}}$ which is a field) and adding the error distribution. Then, we reduce it modulo $R^\vee$ thus landing in $\mathbb{T}^R$. Now, if we look it component by component we have in essence computed $a \cdot s/q$ and then added to each component a sample of $\Psi_{\frac{\xi}{q}}$, so when raising it again to $R_q^\vee$ (by multiplying by q and rounding) we get that $q(a \cdot s/q) = a \cdot s \in R_q^\vee$ and to every component we have added an independent sample taken from $\overline{\Psi}_{\frac{\xi}{q}}$. Therefore, if ρ_ξ^n is the spherical Gaussian with parameter ξ, given an adversary who solves R-LWE$_{\overline{\Psi}_{\frac{\xi}{q}}}$ it is easy to give an adversary who solves R-LWE$_{\rho_\xi^n}$.

Therefore, we can apply the result from [17], for which we need to give two quick lattice definitions.

Definition 15 ([17])**.** *The minimum distance of a lattice $\mathcal{L}$ is the length of the shortest non-zero lattice vector*

$$\lambda_1(\mathcal{L}) := \min_{0 \neq \boldsymbol{v} \in \mathcal{L}} \|\boldsymbol{v}\|$$

The dual lattice of a given lattice $\mathcal{L}$ is defined as

$$\mathcal{L}^* := \{\boldsymbol{x} \in \mathbb{Z}^n | \langle \mathcal{L}, \boldsymbol{x} \rangle \in \mathbb{Z}\}$$

Definition 16 (Definition 3.1, [18])**.** *For a lattice $\mathcal{L}$, and a positive real $\epsilon > 0$, we define its* smoothing parameter *$\eta_\epsilon(\mathcal{L})$ to be the smallest s such that $\rho_{\frac{1}{s}}(\mathcal{L}^* \setminus \{\mathbf{0}\}) \leq \epsilon$, where $\rho_r(\mathcal{L}) := \sum_{\boldsymbol{x} \in \mathcal{L}} \rho_r(\boldsymbol{x})$ for a lattice $\mathcal{L}$, and $\rho_r(\boldsymbol{x}) := \exp(-\pi \|\boldsymbol{x}/r\|^2)$ for some element $\boldsymbol{x} \in \mathbb{R}^n$.*

Finally, we can give the following result.

Lemma 1 (Corollary 7.3, [17])**.** *There is a polynomial-time quantum reduction from K-DGS with function γ to the (average-case, decision) problem of solving R-LWE$_{\rho_\xi^n}$ using l samples with $\xi = \alpha \left(\frac{nl}{\log(nl)} \right)^{\frac{1}{4}}$, $\alpha > 0$ and*

$$\gamma(\mathcal{I}) = \max\left\{ \eta(\mathcal{I}) \cdot \frac{\sqrt{2}}{\alpha} \cdot \omega\left(\sqrt{\log(n)}\right), \ \frac{\sqrt{2n}}{\lambda_1(\mathcal{I}^*)} \right\}$$

as long as $\alpha q \geq \omega\left(\sqrt{\log(n)}\right)$, where $\mathcal{I}$ is an ideal lattice.

In conclusion, we have seen that breaking the security of Encryption Scheme 1 is at least as hard as solving the decision R-LWE problem with a truncated discrete Gaussian, which is at least as hard as solving the decision R-LWE problem with the $\overline{\Psi}^n$ distribution, which in turn is at least as hard as solving the K-DGS problem.

5.2. Non-Leakage of Information

In this section, we need to see that the adversary does not gain any extra information by interacting with the distributed protocol. We will start first with the Protocol 1, seeing that an adversary $\mathcal{A}$ cannot distinguish between interacting with the protocol or with random inputs. Furthermore, we will also give the adversary the ability to choose its shares of the secret key and the PRSS keys, as it makes the game easier and it only serves to see that the protocol's security is even stronger than what is usually required.

To appropriately do so we will need the following auxiliary lemmas about statistical distance, the proofs of which will be in Appendix B.

Lemma 2. *Let Y be a probability distribution over $\mathbb{Z}$ such that $|Y|$ is bounded by κ and X be a discrete uniform distribution in the integer interval $[-a, a]$ with $a \geq \kappa \cdot 2^{\lambda}$. Then, $\Delta(X, \tilde{X}) \leq 2^{-\lambda}$, where $\tilde{X} = X + Y$.*

Lemma 3. *Let X, Y be two probability distributions over a countable support $\mathcal{N}$ such that $\Delta(X, Y) \leq 2^{-\lambda}$, and $n \in \mathbb{Z}_{>0}$ with $n = 2^{\beta}$ for some $\beta \in \mathbb{R}_{>0}$. Then $\Delta(X^n, Y^n) \leq 2^{-\lambda+\beta}$.*

With these auxiliary lemmas we can go ahead and prove the adversary cannot distinguish between interacting with the protocol and random values.

Theorem 3. *Assume that $\Phi_{\cdot}(\cdot)$ is a secure pseudo-random function modeled as a random oracle, that the keys K_H have been securely generated and distributed, that the secret key s has been securely generated and shared and that the parameters follow the conditions of Theorem 1. Then, the Decryption Protocol (Protocol 1) is secure against a passive and static adversary, corrupting up to $t = u - 1$ players.*

Proof. We want to construct an Attack Game in which the adversary cannot distinguish between the protocol executed correctly or with random values to show that the distribution does not leak anything about the secret key $\boldsymbol{s}$ nor the error $\boldsymbol{e}$.

Let C denote the set of corrupted players and B the set of honest players. The Attack Game works as follows. Assume that the challenger knows the secret key $\boldsymbol{s}$ and the K_H such that $C \supseteq H$ (the keys that the adversary does not know) which have been securely generated. Assume that the challenger sends to the adversary $\mathcal{A}$ the ciphertext $(\boldsymbol{u}, \boldsymbol{v})$ and then $\mathcal{A}$ submits $(\boldsymbol{s}'_C, K_{H_C}, \boldsymbol{d}'_C)$ as the challenge, where $\boldsymbol{s}'_C$ are the shares on the secret key of the corrupted players, K_{H_C} are the keys K_H such that $C \not\supseteq H$ (the keys $\mathcal{A}$ knows) chosen by $\mathcal{A}$, and $\boldsymbol{d}'_C$ are the shares on the decryption of the corrupted players. Then, the challenger generates consistent shares on $\boldsymbol{s}$ for the players not in C.

Once all these preliminaries are done, the challenger chooses $b \xleftarrow{\$} \{0, 1\}$ and proceeds as following:

- If $b = 0$: The challenger uses the decryption protocol to compute the shares of the decryption d'_B for the honest players. It computes the decrypted message $\boldsymbol{m}$ and outputs $(\boldsymbol{d}'_B, \boldsymbol{m})$.
- If $b = 1$: The challenger computes for every H such that $C \supseteq H$ some element $\boldsymbol{r_H} \in \mathbb{I}_D^n$ uniformly at random and we denote as $\boldsymbol{y}$ the polynomial in R_q with vector of coefficients $\sum_{C \not\supseteq H} \Phi_{K_H}(\boldsymbol{u} + \boldsymbol{v}) + \sum_{C \supseteq H} \boldsymbol{r_H}$. Then the challenger generates d'_B consistent shares of $\boldsymbol{y} + \boldsymbol{m}\lfloor \frac{q}{2} \rfloor$ (the challenger knows $\boldsymbol{m}$ as it can be computed using the protocol, given that everything needed is known) and outputs $(\boldsymbol{d}'_B, \boldsymbol{m})$.

Finally $\mathcal{A}$ outputs $\tilde{b} \in \{0, 1\}$, meaning whether it thinks it has interacted with the protocol or with a simulation, and the Game concludes.

It is clear that $\boldsymbol{m}$ will be correct in both cases given the proof of Theorem 1, and furthermore, $\boldsymbol{y} + \boldsymbol{m}\lfloor \frac{q}{2} \rfloor$ will be an effective "decryption" of $\boldsymbol{m}$ in the sense that every coefficient will be closer to 0 if $m_i = 0$ and closer to $\lfloor \frac{q}{2} \rfloor$ if $m_i = 1$, because

$$|\boldsymbol{y}|_i \leq \binom{u}{t} \left| \frac{\mathbb{I}_D}{2} \right| \leq \frac{q}{4}.$$

Therefore we only need to see that $\boldsymbol{d}'_B$ are indistinguishable whether they are computed with $b = 0$ or with $b = 1$. Let us see it. First of all, $\boldsymbol{y}$ and $\boldsymbol{x}$ are computationally indistinguishable to the adversary given the properties of pseudo-randomness of $\Phi_{\cdot}(\cdot)$. We now want to see that the way $\boldsymbol{y}$ and $\boldsymbol{y} + \boldsymbol{e} \cdot \boldsymbol{r_E} + \boldsymbol{e_v} - \boldsymbol{s} \cdot \boldsymbol{e_u} = \boldsymbol{y} + \hat{\boldsymbol{e}}$ are distributed are at a negligible statistical distance. It is clear that $\boldsymbol{y}$ is distributed in the interval $\binom{u}{t}\mathbb{I}_D^n$ (with $\binom{u}{t}$ values distributed uniformly in $\mathbb{I}_D$) and as we have seen in the proof of Theorem 1 $\hat{\boldsymbol{e}}$ is in the interval $[-2nu\kappa^2 + \kappa, 2nu\kappa^2 + \kappa]^n$. Therefore, as the distribution of every coefficient is identical and independent we have that by Lemma 2

$$\Delta(\boldsymbol{y}_i, \{\boldsymbol{y} + \hat{\boldsymbol{e}}\}_i) \leq 2^{-\lambda-\beta}$$

and by Lemma 3

$$\Delta(\boldsymbol{y}, \boldsymbol{y} + \hat{\boldsymbol{e}}) \leq 2^{-\lambda}$$

so the distribution of $\boldsymbol{y}$ and $\boldsymbol{y} + \hat{\boldsymbol{e}}$ are at a negligible statistical distance. Therefore, we get that $\boldsymbol{y} + \boldsymbol{m}\lfloor \frac{q}{2} \rfloor$ and $\boldsymbol{x} + \hat{\boldsymbol{e}} + \boldsymbol{m}\lfloor \frac{q}{2} \rfloor$ are computationally indistinguishable.

Finally, adding it all together we get that the output $(\boldsymbol{d}'_B, \boldsymbol{m})$ is computationally indistinguishable whether it has been computed with $b = 0$ or with $b = 1$, so

$$\left| \Pr\left[\tilde{b} = b\right] - \frac{1}{2} \right| = \text{neg}(\lambda)$$

as we wanted to see. □

After Theorem 3, we have only seen that Protocol 1 is secure when the keys are securely generated and against a passive adversary corrupting $t \leq u - 1$ players, but it is standard to see that the same protocol is secure against an active adversary corrupting $t < \frac{u}{3}$ players if instead of the client reconstructs $\boldsymbol{m}$ using the shares of all subsets of $t + 1$ players, as that will give a majority of correct outputs.

The reason behind this is that we have already seen that no information is leaked, so we only need to see that the adversary cannot abort the protocol or cause an incorrect output. In case of an active adversary (who can cause players to deviate arbitrarily from the protocol), what is needed is that if all combinations of $t + 1$ players are decrypting the message, there needs to be a majority of combinations of $t + 1$ players with no corrupt players. This gives us that $t < \frac{u}{3}$ is enough.

Now, we need to see that Protocol 2 leaks no information against an adversary corrupting up to $t = u - 1$ players. To do so we will once again see that the adversary cannot distinguish between interacting with the protocol or a simulation where the challenger sets before-hand the values of the keys.

Theorem 4. *Assuming that the image interval of the pseudo-random function $\Phi_{\cdot}^{KG}(\cdot)$ is $\mathbb{I}_{KG}^n$ where*

$$\mathbb{I}_{KG} = \left[-\kappa \cdot 2^{\lambda+\beta}, \kappa \cdot 2^{\lambda+\beta}\right],$$

that $\mathcal{C}$ is a commitment scheme such that it has a trapdoor and the parameters follow the conditions on Theorem of correctness, then the Key Generation Protocol (Protocol 2) is secure against a passive and static adversary, corrupting up to $t = u - 1$ players.

Proof. We want to construct an Attack Game in which the adversary cannot distinguish between the protocol executed correctly and a simulation where the challenger sets the values of $\boldsymbol{s}, \boldsymbol{e}, \boldsymbol{a_E}$ and K_H for all H before-hand.

Let C denote the set of corrupt players and B the set of honest players. The Attack Game works as follows. Assume that whenever a corrupt player needs to sample a uniform distribution it sends a query to the challenger for a random value from a random oracle. Let $\mathcal{C}_C = \text{Commit}(\mathring{\boldsymbol{s}}_C, \mathring{\boldsymbol{e}}_C, K^s_{N_{H_C}}, K^e_{N_{H_C}}, K'_{H_C}, \boldsymbol{a_E}'_C)$ the challenge output by $\mathcal{A}$, the first step of the interaction in Protocol 2 as we can see in Table 3. Then, the challenger chooses $b \overset{\$}{\leftarrow} \{0,1\}$ and proceeds as follows:

- If $b = 0$: The challenger and the adversary follow Protocol 2 to generate $\boldsymbol{a_E}, \boldsymbol{b_E}$ and the shares $\boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B$ and outputs $(\boldsymbol{a_E}, \boldsymbol{b_E}, \boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B)$.
- If $b = 1$: The challenger samples $\boldsymbol{s}, \boldsymbol{e} \sim \sum_u \chi$, $\boldsymbol{a_E} \overset{\$}{\leftarrow} R_q$ and every $K_H \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes $\boldsymbol{b_E} = \boldsymbol{a_E} \cdot \boldsymbol{s} + \boldsymbol{e}$. Then he uses the trapdoor in the commitment scheme to recover $(\mathring{\boldsymbol{s}}_C, \mathring{\boldsymbol{e}}_C, K^s_{N_{H_C}}, K^e_{N_{H_C}}, K'_{H_C}, \boldsymbol{a_E}'_C)$, and proceeds as follows. We will divide the explanation depending on what he is simulating to ease comprehension, but everything will be done simultaneously, following the flow of information seen in Table 3.
  - For the "generation" of $\boldsymbol{s}$, the challenger will use the keys $K^s_{N_{H_C}}$ (of which he knows all of them given that they were generated through queries to the random oracle through the challenger) to recover $\boldsymbol{s}_C$, the contribution of the corrupt players to $\boldsymbol{s}$. With this information, the challenger can compute $\boldsymbol{s}_B$ the contribution of the honest players to $\boldsymbol{s}$ such that $\boldsymbol{s} = \boldsymbol{s}_C + \boldsymbol{s}_B$. With these values computed the challenger follows with the protocol.
  - For the "generation" of $\boldsymbol{e}$ the challenger proceeds identically as with generating $\boldsymbol{s}$.
  - For the "generation" of K_H, the challenger samples random values in $\mathbb{Z}_q$ for K'_{H_B} (the first step) and commits them. It then will receive K^C_H from the adversary (the shares of K_H pertaining to the corrupt players) and will compute consistent Shamir shares K^B_H so that the players share K_H. Then, as in the protocol, the challenger sends the shares K^B_H to all players not in H.
  - For the "generation" of $\boldsymbol{a_E}$, the challenger samples random values in R_q for $\boldsymbol{a_E}_B$ (the first step) and commits them. It then will receive $\boldsymbol{a}^C_{\boldsymbol{E}}$ (the shares of $\boldsymbol{a_E}$ pertaining to the corrupt players) and will compute consistent Shamir shares $\boldsymbol{a}^B_{\boldsymbol{E}}$ so that the players share $\boldsymbol{a_E}$. Then, as in the protocol, the challenger sends the shares $\boldsymbol{a}^B_{\boldsymbol{E}}$ to all players.
  - For the "generation" of $\boldsymbol{b_E}$ the challenger outputs $\boldsymbol{b_E}$ at the end of the protocol.

  Then, the challenger outputs $(\boldsymbol{a_E}, \boldsymbol{b_E}, \boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B)$.

Finally, $\mathcal{A}$ outputs $\tilde{b} \in \{0,1\}$, meaning whether it thinks it has interacted with the protocol or with the simulation, and the Game concludes.

It is clear that the flow of information is the same in both cases and that the values will be both correct and what the challenger sampled beforehand, so we just need to see that the adversary cannot distinguish between the values received when $b = 0$ from the ones received when $b = 1$. For $\boldsymbol{s}$ (and $\boldsymbol{e}$) it is clear that they are indistinguishable, as we used the trapdoor in the commitment scheme to set the values necessary before any messages were sent from the adversary to the challenger. Furthermore, we know that no information was leaked in the NIVSS as because of Lemmas 2 and 3 we know that no information was leaked as in the proof of Theorem 3.

For K_H (and in turn $\boldsymbol{a_E}$ as they are analogous), we need to see that the adversary cannot distinguish from K'_{H_B} generated by the protocol or them being random in $\mathbb{Z}_q$. To see this we will use the security of Shamir secret sharing, as the adversary can only control up to t players. Therefore, the value shared is completely undetermined by the shares of the corrupt players, so both cases ($b = 0$ and $b = 1$) are indistinguishable to the adversary.

Finally, by adding everything up, we get that $(\boldsymbol{a_E}, \boldsymbol{b_E}, \boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B)$ are indistinguishable whether we have $b = 0$ or $b = 1$, so

$$\left| \Pr\left[\tilde{b} = b\right] - \frac{1}{2} \right| = \operatorname{neg}(\lambda)$$

as we wanted to see. □

As in Section 4, we have also proven the equivalent to this last theorem for an active adversary, however we will not use the result for the implementation, for reasons we will state in Section 6.1. The proof can be found in Appendix A.2.

Having proved the security of each protocol individually, we only need to see that using both protocols together still gives us an encryption scheme which is semantically secure.

Theorem 5. *Assume the conditions in Theorems 1 and 4 are fulfilled. Then, if K-DGS$_\gamma$ is hard, then encryption under keys generated by Protocol 2 and decryption following Protocol 1 is semantically secure against a static and passive adversary corrupting up to $t = u - 1$ players acting through the Key Generation phase and the same adversary being active corrupting up to $t < \frac{u}{3}$ players in the Decryption phase.*

Proof. First, using the result in Theorem 4, we can see that the adversary cannot distinguish between executing both protocols, or replacing the key generation with keys generated by the challenger. Then, using Theorem 1, we can see that the adversary cannot distinguish between taking part in the decryption or having the challenger decrypt all by itself. Therefore, we get that the adversary cannot distinguish between the semantic security game when both distributed protocols are used from the basic semantic security game of Encryption Scheme 1. This means, using what we have seen in Section 5.1 that breaking semantic security when both protocols are being used is as hard as breaking semantic security of the encryption scheme, so using the reduction to K-DGS$_\gamma$ and that we assume this problem to be hard, we have that our protocols are semantically secure, as we wanted to see. □

6. Implementation

The first step for the implementation is finding good parameters that guarantee the security of the particular instance of the R-LWE problem. To verify it we will use the bounds on ξ in Lemma 1 and the LWE hardness estimator given by Albrecht et al. in [19]. We use the LWE estimator because, as far as we know, no major attacks are known to exploit the particular properties of R-LWE, so the estimated hardness for LWE translates as estimated hardness for R-LWE.

6.1. Choosing Parameters

We set the security parameter $\lambda = 100$. We need to find the following parameters: n, q, κ and ξ which will then allow us to compute $\mathbb{I}_D$ and $\mathbb{I}_{KG}$. We will first leave everything in function of n and q, and we will then use the concrete hardness of an instance of the R-LWE problem to fix n and q.

Let $n = 2^\beta$ and $q \in \mathbb{Z}_{>0}$. Using the conditions on $\mathbb{I}_D$ on Theorem 1 we get that

$$\kappa = \left\lfloor \frac{-1 + \sqrt{1 + \frac{2nuq}{\binom{u}{t}2^{\lambda+\beta}+1}}}{4nu} \right\rfloor$$

To find ξ, we will use the following lemma, the proof of which is in Appendix B.

Lemma 4. *Let $\overline{\Psi}_{\frac{\sigma}{q}}$ be a discrete Gaussian. Then, $\forall c > 0$:*

$$\Pr\left[\left|\overline{\Psi}_{\frac{\sigma}{q}}\right| > c\right] \leq \sqrt{\frac{2}{\pi}} \frac{e^{-\left(\frac{\lceil c \rceil - \frac{1}{2}}{\sqrt{2}\sigma}\right)^2}}{\lceil c \rceil - \frac{1}{2}}.$$

Using the bound on Definition 13 and Lemma 4 we can get the following bound:

$$\begin{aligned}\Pr\left[\left|\overline{\Psi}_{\frac{\xi}{q}}\right| > \kappa\right] &= \Pr\left[\left|\overline{\Psi}_{\frac{\xi}{q}}\right| > \kappa + \frac{1}{2}\right] \\ &\leq \sqrt{\frac{2}{\pi}}\frac{e^{-\left(\frac{\kappa+\frac{1}{2}}{\sqrt{2}\xi}\right)^2}}{\kappa+\frac{1}{2}} \\ &\leq 2^{-\lambda}\end{aligned}$$

which when isolating the ξ gives us the following bound:

$$\xi \leq \sqrt{\frac{\left(\kappa+\frac{1}{2}\right)^2}{-2\log\left(\sqrt{\frac{\pi}{2}}2^{-\lambda}\left(\kappa+\frac{1}{2}\right)\right)}}.$$

From here we will take the equality, as with a fixed q the larger the standard deviation the greater the hardness of that specific instance of the decision R-LWE problem.

Now, we can find n and q using the LWE hardness estimator, which given n, q, α outputs the concrete hardness of that specific instance. We will set n as a power of 2, as it allows us to use more efficient multiplication algorithms and q as a prime near a power of 2. Using this, we implemented a Python algorithm to find these parameters. The code can be found in the repository in Appendix C. This has yielded the following results as parameters for $u = 7$ and $t = 2$ and more than 100 bits of security, as can be seen in Table 4.

Table 4. Parameters for secure implementation.

n	=	4096
q	=	713,623,846,352,979,940,529,142,984,724,747,568,191,373,381
κ	=	168
ξ	=	14.897861091181875
$\mathbb{I}_D$	=	8,403,614,205,785,368,527,542,540,898,258,331,059,093,504
$\mathbb{I}_{KG}$	=	872,305,872,233,851,041,593,123,383,308,976,128
Bits of Security	=	121

In this code, there is also the computing of parameters for the case of an active adversary in Key Generation phase using the conditions on Theorem A1, meaning that to have 100 bits of security against this type of adversary we need to bump up to $n = 8192$. This, as we will see with the results in Section 6.3, hurts the viability of the protocols, that is why we take our main proposal as secure against an adversary who acts passively in the Key Generation phase and actively in the Decryption phase.

6.2. Implementation Particulars

There are several implementation decisions we have taken and need to be discussed. First, we have not coded a truly interactive protocol between u different players, but rather a simulation where one processor computes all the steps simulating the interaction, in the sense that the protocols are divided by steps between communication phases where all computing can be done without the need to interact with other entities. Then, the program computes how much time every step costs for every player and picks the maximum as the "official" time for that step. As we only want to analyze roughly how viable our protocols are, this approximate works for us. This approximation also means that the execution of the simulation lasts considerably longer than the "real time" for the execution, thus limiting us with the amount of players we can reasonably use.

Second, to have the most compact possible form of Shamir Secret Sharing we have used Shamir over the field of $\mathbb{Z}_q$ instead of embed it in $\mathbb{Q}$. This is the main reason why we have taken q prime, as none of the reductions require it.

Third, regarding the implementation of the PRF, we have used the main result in [20], which says that Hash-based Message Authentication Code (HMAC) is a PRF under the condition that the underlying compression function is a PRF. To ensure this condition is satisfied we have used the HMAC based around SHA-3.

Finally, regarding the Commitment Scheme we have used for the Key Generation protocol, we have used the hash of the message we want to send concatenated with a random string. We have used SHA-2 because, as far as we know, it is secure enough. However, should the need arise, it could be swapped for a more secure alternative. Furthermore, we only needed to use a commitment scheme after the first round where every player sets their values. This is so because once all the values have been set and all the shares sent, the contributions of the adversary are no longer needed, as the honest players can already generate a majority of correct values. And given that no other value needs to be set in a way the adversary cannot exploit (as the adversary becomes irrelevant), a commitment scheme in any further communication step seems unnecessary. However, this commitment phases could be added with no major change to the protocol nor the proof of security or correctness, only a slightly slower execution.

6.3. Results of the Simulation

In this final section, we will discuss the results we have obtained from the execution of the code for the simulation of both protocols. You can find the code in the repository linked in Appendix C. The specifications of the system where we have executed the programs are found in Table 5. Furthermore, we have used the following C libraries: FLINT (Fast Library for Number Theory) to ease computations in R_q, which in turn uses the GMP and MPFR libraries to deal with multiple precision numbers, and OpenSSL library for cryptography-related functions like Hashes or HMACs. It is also worth mentioning that any result we obtain from the execution of the simulation has been found by averaging the times of 10,000 executions of the code, so as to better portray the results, getting rid of outliers.

Table 5. Specifications of the system.

Operating System	Ubuntu 18.04.5 LTS
CPU	Intel® Core™ i5-8500
Memory	15.4 GiB
Word Size	64 bits
CPU Clock Speed	3.00 GHz

From what we have seen to this point there are two main dependencies: growth of time in respect to the threshold t and growth of time in respect to the dimension of the lattice. This is so because the threshold defines the minimum number of players needed (and vice versa, given a number of players we can get the maximum threshold it allows) depending if we are protecting ourselves against an active or a passive adversary. As we have seen in Section 6.1 given the adversary model, given an n we can find the rest of parameters that make the protocol secure (taking into account that there is a minimum n for which this analysis works).

In regards to the dependency on t, analyzing the protocols theoretically lets us see that when performing either the PRSS or the NIVSS there are $\binom{u}{t}$ different keys K_H, meaning that the number of additions grows asymptotically with the value $\binom{u}{t}$. This means that the dependency should be approximately exponential in the active case where $u = 3t + 1$ and approximately linear in the passive case where $u = t + 1$. When obtaining results from the simulation, we have gathered results for the values of t most frequently used in real-life applications like electronic voting, which means $t < 3$ against active adversaries and $t < 7$

for passive adversaries. This decision is mainly due to how we have implemented the simulation, as instead of having the several players' protocols being executed at the same time, we have them executed consecutively and then take the maximum time spent as the overall time. In the case of the Key Generation, as there are various steps of interaction, this process is applied to every one of the steps. Therefore, due to time constraints, we were limited in how many players we could simulate in the protocols while running 10,000 executions of the codes. Having said all that, the results we obtained for the dependency on t followed our predictions. In the active case they behaved greater than linearly in the three points we had, and in the passive case it behaved approximately linearly. A more in-depth analysis cannot be made unless more extensive data is gathered.

In regards to the dependency on n, given that there is multiplication of polynomials in both protocols, which is implemented using the Karatsuba algorithm that scales by the order of $n^{\log_2(3)} > n^{1.5}$, the time grows asymptotically with this value. The results obtained for the dependency on n, which can be seen in Figure 1, show the expected results only in the decryption phase against a passive adversary. For the other cases, we see linear, or practically linear, behavior for the range of values of n we are interested in for real life applications. This is due to the fact that the protocols need to perform a much higher number of additions than products, and this difference ends up being high enough for the linear growth of the addition to offset the growth of the product at these values of n.

Active adversary $t = 2, u = 7$

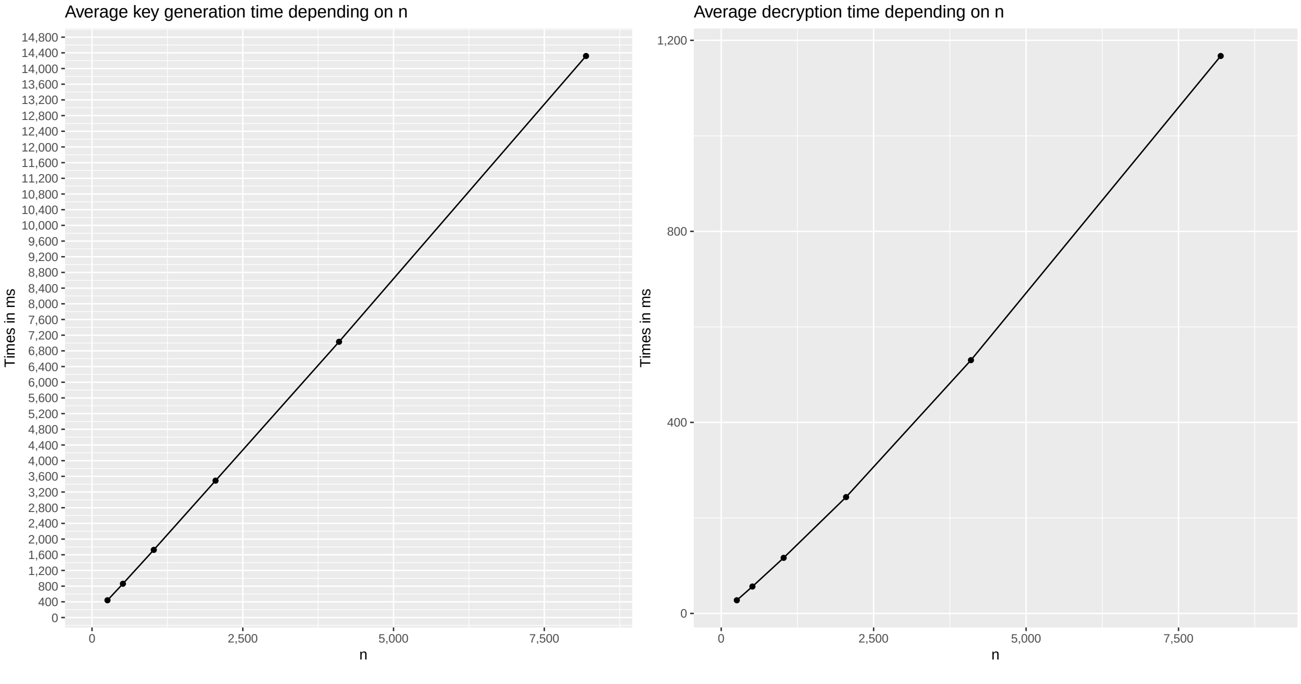

Figure 1. *Cont.*

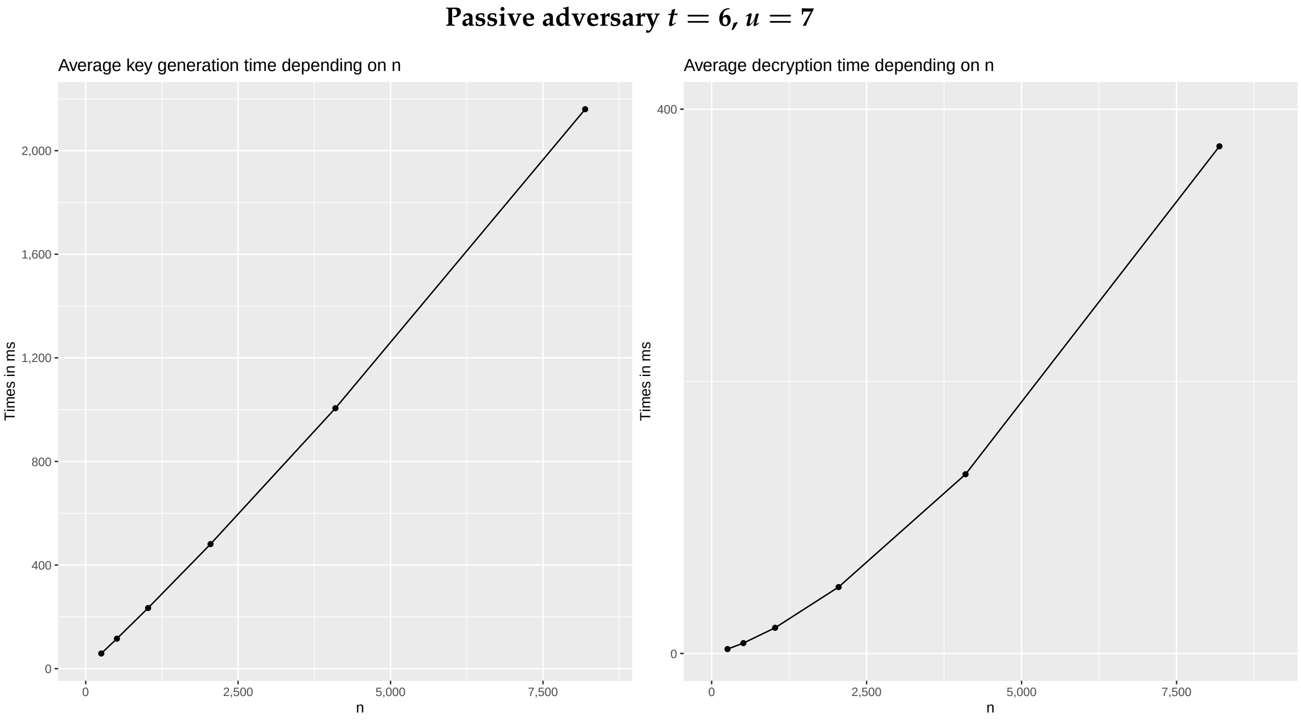

Figure 1. Times of the simulation for $n = 256, 512, 1024, 2048, 4096, 8192$.

Finally, we want to discuss the viability of the protocols. As we can see in Table 6, the Key Generation times are significantly slower than the Decryption time, between 4 and 7 times slower. This, however, does not pose a big problem, as by design in most implementations one round of Key Generation will be used to decrypt many messages; therefore, we can focus our main analysis in the decryption times. In that front, the 530.36 ms per decryption in the active case translates to approximately 7000 messages per hour, while the 131.73 ms per message in the passive case translates to approximately 27,000 messages per hour. As we can see it will be half these votes per hour with $n = 8192$, which will be needed against an active adversary as we have said in Section 6.1.

Table 6. Time comparison between active and passive adversary.

n	Key Generation		Decryption		Encryption
	Active	Passive	Active	Passive	
4096	7031.34 ms	1005.63 ms	530.36 ms	131.73 ms	191.79 ms
8192	14320.01 ms	2160.05 ms	1167.24 ms	372.75 ms	539.71 ms

7. Conclusions and Future Work

For our concluding considerations, we would like to summarize the limitations that this work has and how it could be improved in future works, as well as to give some insight to some possible other uses outside the ones outlined above.

Construction-wise, our proposal has two main limiting factors: First, the fact that we compute the drowning noises through both the PRSS technique and NIVSS technique means that computation time will increase asymptotically exponentially with the number of players in an active adversary setting, which is not desirable. Should other techniques be used for non-interactively sharing the noise value, this could be avoided. Second, and more inherent to the structure of our protocols, is the fact that using noise drowning by itself causes our implementation to use very high dimensions for the lattices. This is due to the fact that to guarantee statistical indistinguishability we need the noise to be exponentially bigger than the secret while the LPR encryption scheme requires the noise to be smaller than

$\frac{q}{4}$, meaning that the parameter ξ is proportionately very small compared to the modulus q. This then has an influence in the dimension of the lattice needed to ensure that the concrete instance of the R-LWE problem has the desired amount of bits of security. This limitation could only be avoided by using a different way to ensure security than noise drowning, which is a pivotal element in our proposal.

Implementation-wise, we would like to reiterate that our proposed codes are a proof of concept of approximate viability, and in no case an attempt to give a computationally secure and efficient program. As so, the codes should be carefully analyzed to ensure computational security (for example, some of the random samplings are not truly indistinguishable from uniformly at random) and that no other implementation based attacks (like side-channel attacks) can be performed. Furthermore, for any type of important practical application, the codes should be optimized in efficiency.

Finally, we would also want to state the fact that even if the protocols presented were thought of with the idea of using them together, their security proofs are independent, and therefore they can be used separately. This means that in a case where a TTP can be trusted with the key generation, only the decryption protocol may be used, and then make the process completely interactiveless. Additionally, while the key generation protocol is more particularly tailored to the needs of the encryption scheme, it would not be too hard to extract the main principles for generating the different types of elements, therefore being easily generalizable with a similar security proof. This opens the gateway to a whole new set of possibilities for applications.

Author Contributions: Conceptualization, R.M. and P.M.; methodology, R.M. and P.M.; software, F.A. and R.M.; validation, F.A.; formal analysis, F.A., R.M. and P.M.; investigation, F.A., R.M. and P.M.; resources, F.A., R.M. and P.M.; writing—original draft preparation, F.A.; writing—review and editing, F.A., R.M. and P.M.; supervision, P.M.; project administration, F.A., R.M. and P.M.; funding acquisition, P.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially funded by the European Union PROMETHEUS project (Horizon 2020 Research and Innovation Program, grant 780701) and the Spanish Ministry of Economy and Competitiveness, through Project MTM2016-77213-R.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CPA	Chosen Plaintext Attack
GAPSVP	GAP Shortest Vector Problem
HMAC	Hash-based Message Authentication Code
K-DGS	Discrete Gaussian Sampling over K
LWE	Learning with Errors
NIST	National Institute of Standards and Technology
NIVSS	Non-Interactive Verifiable Secret Sharing
R-LWE	Ring Learning with Errors
PRF	Pseudo-Random Function
PRSS	Pseudo-Random Secret Sharing
TTP	Trusted Third-Party

Appendix A. Correctness and Security against Active Adversaries

Appendix A.1. Correctness

We will prove correctness of the Decryption protocol against an active (or passive) adversary when the keys are generated by the Key Generation protocol against an active adversary.

Theorem A1. *Let $n \in \mathbb{Z}_{>0}$ be the number of coefficients in R_q, $u \in \mathbb{Z}_{>0}$ be the number of players, $\Phi_{\cdot}^{D}(\cdot)$ be a pseudo-random function with image interval $\mathbb{I}_D^n$, χ be a distribution in R_q where every coefficient is a truncated Discrete Gaussian with parameters σ and κ,*

$$\mathbb{I}_D = \left[-\left(4n\frac{u}{3}\kappa^2\left(2^{\lambda+\beta}+1\right)+\kappa\right)\cdot 2^{\lambda+\beta}, \left(4n\frac{u}{3}\kappa^2\left(2^{\lambda+\beta}+1\right)+\kappa\right)\cdot 2^{\lambda+\beta}\right],$$

$$\mathbb{I}_{KG} = \left[-\kappa\cdot 2^{\lambda+\beta}, \kappa\cdot 2^{\lambda+\beta}\right].$$

and

$$\left\lfloor\frac{q}{4}\right\rfloor \geq \left(4n\frac{u}{3}\kappa^2\left(2^{\lambda+\beta}+1\right)+\kappa\right)\left(\binom{u}{t}2^{\lambda+\beta}+1\right)$$

Then, Protocol 1 will have correct output except with probability $2^{-\lambda-\beta}$ against an active adversary corrupting up to $t < \frac{u}{3}$ players.

Proof. What we want to see first, as before, is that $|\boldsymbol{x}+\hat{\boldsymbol{e}}|_i \leq \frac{q}{4}\ \forall i$, where $\cdot_i$ notes the coefficient i on the polynomial, given the way the decryption works in Encryption Scheme 1.

Let $\hat{\boldsymbol{e}} = \boldsymbol{e}\cdot\boldsymbol{r_E}+\boldsymbol{e_v}-\boldsymbol{s}\cdot\boldsymbol{e_u}$. As the product in R_q is done through the anticyclic matrix, we know that

$$|\hat{\boldsymbol{e}}|_i \leq |e_i\cdot r_{E_1}| + |e_{i-1}\cdot r_{E_2}| + \ldots + |e_{i+2}\cdot r_{E_{n-1}}| + |e_{i+1}\cdot r_{E_n}| + |e_{v_i}| + |s_i\cdot e_{u_1}| + \ldots + |s_{i+1}\cdot e_{u_n}|.$$

Now, we still have $\boldsymbol{r_E}, \boldsymbol{e_v}, \boldsymbol{e_u} \sim \chi$ but for t of the contributions we can only assure that they are in $2\cdot\mathbb{I}_{KG}$, so we get that

$$\begin{aligned}|\hat{\boldsymbol{e}}|_i &\leq 2n\left(\frac{u}{3}2\kappa\cdot 2^{\lambda+\beta}+2\frac{u}{3}\kappa\right)\kappa+\kappa\\ &= 4n\frac{u}{3}\kappa^2(2^{\lambda+\beta}+1)+\kappa\end{aligned}$$

Furthermore, given that there are $\binom{u}{t}$ keys K_H, we know that

$$\boldsymbol{x}_i \in \binom{u}{t}\mathbb{I}_D.$$

Adding both results we then get that

$$\begin{aligned}|\boldsymbol{x}+\hat{\boldsymbol{e}}|_i &\leq \binom{u}{t}\left(4n\frac{u}{3}\kappa^2\left(2^{\lambda+\beta}+1\right)+\kappa\right)\cdot 2^{\lambda+\beta}+\left(4n\frac{u}{3}\kappa^2\left(2^{\lambda+\beta}+1\right)+\kappa\right)\\ &= \left(4n\frac{u}{3}\kappa^2\left(2^{\lambda+\beta}+1\right)+\kappa\right)\left(\binom{u}{t}2^{\lambda+\beta}+1\right)\\ &\leq \left\lfloor\frac{q}{4}\right\rfloor\end{aligned}$$

as we wanted to see.

Finally, we just need to see that when the client reconstructs, there is indeed a majority of correct results, and this derives directly from having at most t corrupt players; therefore, there will be a majority of subsets of $t+1$ players where they are all honest, and thus output the correct decryption. □

As in Section 4, the same proof works against a passive adversary corrupting up to $t = u - 1$ players.

However, in the case of dealing with an active adversary in the key generation phase, to have a truly correct scheme we need to see that the protocol cannot be halted by any actions performed by a malicious adversary. However, in the protocol, whenever a verification fails the protocol halts. To deal with these we implement the following dispute resolution policy, where a dispute is raised whenever a player receives a value that fails verification stating there which other player sent the values.

The policy works as follows, once the protocol is halted the players look at the disputes that have risen, and then "eliminate" all players involved in them, in the sense that a new execution of the key generation protocol will start without both players involved in every dispute, and in case a player is involved in more than one dispute only the first one will be analyzed. This policy ensures that the protocol will produce a correct output with at most t halts since no dispute can be risen between honest players, and given that the security of the scheme is based only on assuming that there is at least one contribution on $\boldsymbol{s}$ and $\boldsymbol{e}$ following the distribution, the output generates no problems. Finally, note that since in every dispute there is at most one honest player, the ratio of corrupt players will never go above $\frac{u}{3}$.

Appendix A.2. Security

To be able to prove security when the Key Generation is against an active adversary, we will only need to reword the Theorem as follows.

Theorem A2. *Assuming that $\Phi_{\cdot}(\cdot)$ is a secure pseudo-random function modeled as a random oracle, that the keys K_H have been securely generated and distributed, that the secret key s has been securely generated and shared, and that the parameters follow the conditions of Theorem A1, the Decryption Protocol (Protocol 1) is secure against an active and static adversary, corrupting up to $t < \frac{u}{3}$ players.*

The proof is analogous to the proof of Theorem 1.

We will now prove that the Key Generation leaks no information when acting against an active adversary corrupting up to $t < \frac{u}{3}$ players. As before, we will prove that the adversary cannot distinguish between interacting with the protocol or a simulation where the challenger sets before-hand the values of the keys.

Theorem A3. *Assuming that the image interval of the pseudo-random function $\Phi_{\cdot}^{KG}(\cdot)$ is $\mathbb{I}_{KG}^{n}$ where*

$$\mathbb{I}_{KG} = \left[-\kappa \cdot 2^{\lambda+\beta}, \kappa \cdot 2^{\lambda+\beta} \right],$$

that $\mathcal{C}$ is a commitment scheme such that it has a trapdoor and the parameters follow the conditions on Theorem of correctness, then the Key Generation Protocol (Protocol 2) is secure against an active and static adversary, corrupting up to $t < \frac{u}{3}$ players.

Proof. We want to construct an Attack Game in which the adversary cannot distinguish between the protocol executed correctly and a simulation where the challenger sets the values of $\boldsymbol{s}, \boldsymbol{e}, \boldsymbol{a_E}$ and K_H for all H beforehand.

Let C denote the set of corrupt players and B the set of honest players. The Attack Game works as follows. Assume that whenever a corrupt player needs to sample a uniform distribution it sends a query to the challenger for a random value from a random oracle. Let $\mathcal{C}_C = \text{Commit}(\mathring{\boldsymbol{s}}_C, \mathring{\boldsymbol{e}}_C, K^s_{N_{H_C}}, K^e_{N_{H_C}}, K'_{H_C}, \boldsymbol{a_E}'_C)$ the challenge output by $\mathcal{A}$, the first step of the interaction in Protocol 2 as we can see in Table 3. Then, the challenger chooses $b \overset{\$}{\leftarrow} \{0,1\}$ and proceeds as follows:

- If $b = 0$: The challenger and the adversary follow Protocol 2 to generate $\boldsymbol{a_E}, \boldsymbol{b_E}$ and the shares $\boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B$ and outputs $(\boldsymbol{a_E}, \boldsymbol{b_E}, \boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B)$.
- If $b = 1$: The challenger samples $\boldsymbol{s}, \boldsymbol{e} \sim \sum_u \chi$, $\boldsymbol{a_E} \xleftarrow{\$} R_q$ and every $K_H \xleftarrow{\$} \mathbb{Z}_q$ and computes $\boldsymbol{b_E} = \boldsymbol{a_E} \cdot \boldsymbol{s} + \boldsymbol{e}$. Then he uses the trapdoor in the commitment scheme to recover $(\mathring{\boldsymbol{s}}_C, \mathring{\boldsymbol{e}}_C, K^s_{N_{H_C}}, K^e_{N_{H_C}}, K'_{H_C}, \boldsymbol{a_E}'_C)$, and proceeds as follows. We will divide the explanation depending on what he is simulating to ease comprehension, but everything will be done simultaneously, following the flow of information seen in Table 3.
  - For the "generation" of $\boldsymbol{s}$, the challenger will use the keys $K^s_{N_{H_C}}$ (of which he knows all of them as he/she controls more than t players), to recover $\boldsymbol{s}_C$, the contribution of the corrupt players to $\boldsymbol{s}$. With this information, the challenger can compute $\boldsymbol{s}_B$ the contribution of the honest players to $\boldsymbol{s}$ such that $\boldsymbol{s} = \boldsymbol{s}_C + \boldsymbol{s}_B$. With these values computed, the challenger proceeds with the protocol.
  - For the "generation" of $\boldsymbol{e}$ the challenger proceeds identically as with generating $\boldsymbol{s}'$.
  - For the "generation" of K_H, the challenger recovers K_{H_C} (as it controls more than t players) the contribution of the corrupt players to K_H. With this information, the challenger can compute K_{H_B} the contribution of the honest players such that $K_H = K_{H_C} + K_{H_B}$ for all H. With these values computed, the challenger proceeds with the protocol.
  - For the "generation" of $\boldsymbol{a_E}$, the challenger recovers $\boldsymbol{a}_{EC}$ (as it controls more than t players) the contribution of the corrupt players to $\boldsymbol{a_E}$. With this information, the challenger can compute $\boldsymbol{a}_{EB}$ the contribution of the honest players such that $\boldsymbol{a_E} = \boldsymbol{a}_{EC} + \boldsymbol{a}_{EB}$. With these values computed, the challenger proceeds with the protocol.
  - For the "generation" of $\boldsymbol{b_E}$ the challenger outputs $\boldsymbol{b_E}$ at the end of the protocol.

  Then, the challenger outputs $(\boldsymbol{a_E}, \boldsymbol{b_E}, \boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B)$.

Finally, $\mathcal{A}$ outputs $\tilde{b} \in \{0, 1\}$, meaning whether it thinks it has interacted with the protocol or with a simulation, and the Game concludes.

It is clear that the flow of information is the same in both cases and that the values will be correct and what the challenger has sampled beforehand, so we just need to see that the adversary cannot distinguish between the values received when $b = 0$ from the ones received when $b = 1$. We can see they are indistinguishable as the challenger uses the trapdoor in the commitment scheme to get the values necessary before any message were sent from the challenger to the adversary. Moreover, once again, we know that no information was leaked in the NIVSS by the same reasoning from the proof of Theorem 4.

Therefore, $(\boldsymbol{a_E}, \boldsymbol{b_E}, \boldsymbol{s}'_B, \boldsymbol{e}'_B, K^B_H, \boldsymbol{a_E}^B)$ are indistinguishable to the adversary whether they have been computed with $b = 0$ or $b = 1$, so

$$\left| \Pr\left[\tilde{b} = b\right] - \frac{1}{2} \right| = \text{neg}(\lambda)$$

as we wanted to see. □

Having proved the security of each protocol individually, we only need to see that using both protocols together still gives us an encryption scheme which is semantically secure.

Theorem A4. *Assume the conditions in Theorems A1 and A3 are fulfilled. Then, if K-DGS$_\gamma$ is hard, then encryption under keys generated by Protocol 2 and decryption following Protocol 1 is semantically secure against a static and passive adversary corrupting up to $t = u - 1$ players acting through the Key Generation phase and the same adversary being active corrupting up to $t < \frac{u}{3}$ players in the Decryption phase.*

Proof. The proof is analogous to the proof of the Main Theorem but changing Theorems 1 and 4 for Theorems A1 and A3, respectively. □

Appendix B. Proofs of Auxiliary Theorems and Lemmas

Appendix B.1. Proof of Theorem 2

Theorem A5. *Given χ a distribution over R_q, there exists a reduction to the semantic security of the Encryption Scheme 1 from the decisional R-LWE$_\chi$ problem.*

Proof. What we want to see is that given an efficient adversary $\mathcal{A}$ who has non-negligible semantic security advantage, we can construct an efficient adversary $\mathcal{B}$ with access to $\mathcal{A}$ who given an instance of the decisional R-LWE problem, it can solve it with probability non-negligibly bigger than $\frac{1}{2}$.

Let $(\overline{\boldsymbol{a}}_i, \overline{\boldsymbol{b}}_i) \in R_q \times R_q$ be an instance of the decisional R-LWE problem. What we need $\mathcal{B}$ to do is to output whether a polynomial amount of instances are samples of the distribution $A_{\boldsymbol{s},\chi}$ or of the uniform distribution over $R_q \times R_q$, in other words, we want to know whether $\overline{\boldsymbol{b}}_i = \overline{\boldsymbol{a}}_i \cdot \overline{\boldsymbol{s}} + \overline{\boldsymbol{e}}$ for some $\overline{\boldsymbol{s}} \in R_q$ and $\overline{\boldsymbol{e}} \longleftarrow \chi$.

Note that any adversary $\mathcal{A}$ who breaks semantic security may be of one of two types. Either $\mathcal{A}$ has non-negligible semantic security advantage against the encryption scheme when $(\boldsymbol{a_E}, \boldsymbol{b_E})$ are generated independently uniformly at random (instead of having $\boldsymbol{b_E} = \boldsymbol{a_E} \cdot \boldsymbol{s} + \boldsymbol{e}$) or it does not. We will construct two different adversaries for these cases.

Assume first that $\mathcal{A}$ has a negligible semantic security advantage against the encryption scheme when $(\boldsymbol{a_E}, \boldsymbol{b_E})$ are generated independently uniformly at random. Let $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ be an instance of the R-LWE$_\chi$ problem, then we define the following attack game.

Attack Game A1. *The attack game goes as follows:*

- *Set the public key to $(\overline{\boldsymbol{a}}_1, \overline{\boldsymbol{b}}_1)$ and send it to $\mathcal{A}$.*
- *Receive $\boldsymbol{m}_{01}, \boldsymbol{m}_{11}$ from the adversary, and choose $b_1 \overset{\$}{\leftarrow} \{0,1\}$.*
- *Compute $\boldsymbol{u}_1 = \overline{\boldsymbol{a}}_1 \cdot \boldsymbol{r_E} + \boldsymbol{e_u}$ and $\boldsymbol{v}_1 = \overline{\boldsymbol{b}}_1 \cdot \boldsymbol{r_E} + \boldsymbol{e_v} + \boldsymbol{m}_{b_1 1} \lfloor \frac{q}{2} \rfloor$ with $\boldsymbol{r_E}, \boldsymbol{e_u}, \boldsymbol{e_v} \longleftarrow \chi$, and send $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ to $\mathcal{A}$.*
- *Receive $\hat{b}_1$ from the adversary.*

Then, $\mathcal{B}$ will work as follows. When given the instances, it picks $(\overline{\boldsymbol{a}}_1, \overline{\boldsymbol{b}}_1)$ and performs the Attack Game 1 with $\mathcal{A}$ a polynomial amount of times. Then, it computes the advantage

$$\mathrm{SSAdv}_1^*[\mathcal{A}, \mathcal{S}] = \left| \frac{\text{Number of queries where } \hat{b}_1 = b_1}{\text{Total number of queries}} - \frac{1}{2} \right|.$$

We know from how we have defined the adversary $\mathcal{A}$, as $\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]$ is non-negligible, if the instances follow the distribution $A_{\boldsymbol{s},\chi}$ then $\mathrm{SSAdv}_1^*[\mathcal{A}, \mathcal{S}]$ will be non-negligible and if the instances are uniform over $R_q \times R_q$ then $\mathrm{SSAdv}_1^*[\mathcal{A}, \mathcal{S}]$ will be negligible. This means that with non-negligible probability $\mathcal{B}$ can solve the decisional R-LWE$_\chi$ problem as we wanted.

Assume now that $\mathcal{A}$ has a non-negligible semantic security advantage against the encryption scheme when $(\boldsymbol{a_E}, \boldsymbol{b_E})$ are generated independently uniformly at random. Let, once again, $(\boldsymbol{a}_1, \boldsymbol{b}_1)$ and $(\boldsymbol{a}_2, \boldsymbol{b}_2)$ be two instances of the R-LWE$_\chi$ problem, then we define the following attack game.

Attack Game A2. *The attack game goes as follows:*

- *Set the public key to $(\overline{\boldsymbol{a}}_1, \overline{\boldsymbol{a}}_2)$ and send it to the adversary.*
- *Receive $\boldsymbol{m}_{02}, \boldsymbol{m}_{12}$ from the adversary, and choose $b_2 \overset{\$}{\leftarrow} \{0,1\}$.*
- *$\boldsymbol{u}_2 = \overline{\boldsymbol{b}}_1$ and $\boldsymbol{v}_2 = \overline{\boldsymbol{b}}_2 + \boldsymbol{m}_{b_2 2} \lfloor \frac{q}{2} \rfloor$ and send $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ to $\mathcal{A}$.*
- *Receive $\hat{b}_2$ from the adversary.*

Then, $\mathcal{B}$ will work as follows. When given the instances, it picks two of them $(\overline{\boldsymbol{a}}_1, \overline{\boldsymbol{b}}_1)$ and $(\overline{\boldsymbol{a}}_2, \overline{\boldsymbol{b}}_2)$, and performs the Attack Game 2 with $\mathcal{A}$ a polynomial amount of times. Then, it computes the advantage

$$\mathrm{SSAdv}_2^*[\mathcal{A}, \mathcal{S}] = \left| \frac{\text{Number of queries where } \hat{b}_2 = b_2}{\text{Total number of queries}} - \frac{1}{2} \right|.$$

We know from how we have defined the adversary $\mathcal{A}$, since $\mathrm{SSAdv}[\mathcal{A}, \mathcal{S}]$ is non-negligible, if the instances follow the distribution $A_{s,\chi}$ then $\mathrm{SSAdv}_2^*[\mathcal{A}, \mathcal{S}]$ will be non-negligible and if the instances are uniform over $R_q \times R_q$ then $\mathrm{SSAdv}_2^*[\mathcal{A}, \mathcal{S}]$ will be negligible, since if $\overline{\boldsymbol{b}}_i$ are uniformly at random then $\boldsymbol{v}_2$ is independent from the public key and $\boldsymbol{m}_{b_2 2}$. This means that with non-negligible probability $\mathcal{B}$ can solve the decisional R-LWE$_\chi$ problem as we wanted, since it is possible to distinguish, with non-negligible probability, a negligible event from a non-negligible event. □

Appendix B.2. Proofs of Lemmas 2 and 3

Lemma A1. *Let Y be a probability distribution over $\mathbb{Z}$ such that $|Y|$ is bounded by κ and X be a discrete uniform distribution in the integer interval $[-a, a]$ with $a \geq \kappa \cdot 2^\lambda$. Then, $\Delta(X, \tilde{X}) \leq 2^{-\lambda}$, where $\tilde{X} = X + Y$.*

Proof. The first thing to notice is that for any $z \in \mathbb{Z}$ such that $|z| > \kappa + a$, we will clearly have $\tilde{X}(z) = X(z) = 0$, as the support of $\tilde{X}$ will only go from $-\kappa - a$ to $\kappa + a$. Furthermore, for $n \in [-a + \kappa, a - \kappa]$, we can do the following analysis:

$$\begin{aligned} \tilde{X}(n) &= \sum_{m=-\kappa}^{\kappa} Y(m) X(n-m) \\ &= X(n) \sum_{m=-\kappa}^{\kappa} Y(m) \\ &= X(n) \\ &= \frac{1}{2a+1} \end{aligned}$$

using that $n - m$ will always fall in the support of X (thus $X(n-m)$ is never zero), that X is uniform and that Y only takes values in $[-\kappa, \kappa]$.

Now taking everything together we get from the definition of statistical distance:

$$\begin{aligned} \Delta(\tilde{X}, X) &= \frac{1}{2} \sum_{n \in \mathbb{Z}} |\tilde{X}(n) - X(n)| \\ &= \frac{1}{2} \sum_{n \in [-a-\kappa, -a+\kappa-1] \cup [a-\kappa+1, a+\kappa]} |\tilde{X}(n) - X(n)| \\ &\leq \frac{1}{2} \sum_{n \in [-a-\kappa, -a+\kappa-1] \cup [a-\kappa+1, a+\kappa]} \max_m \{X(m)\} \\ &= \frac{1}{2} \sum_{n \in [-a-\kappa, -a+\kappa-1] \cup [a-\kappa+1, a+\kappa]} \frac{1}{2a+1} \\ &= \frac{2 \cdot 2\kappa}{2 \cdot (2a+1)} \\ &\leq \frac{\kappa}{\kappa \cdot 2^\lambda} \\ &= 2^{-\lambda}. \end{aligned}$$

□

Lemma A2. *Let X, Y be two probability distributions over a countable support $\mathcal{N}$ such that $\Delta(X, Y) \leq 2^{-\lambda}$, and $n \in \mathbb{Z}_{>0}$ with $n = 2^{\beta}$ for some $\beta \in \mathbb{R}_{>0}$. Then $\Delta(X^n, Y^n) \leq 2^{-\lambda+\beta}$.*

Proof. We define the n-dimensional distributions

$$\hat{X}_i = (\underbrace{Y, \ldots, Y}_{i}, X, \ldots, X)$$

where we have $X^n = \hat{X}_0$ and $Y^n = \hat{X}_n$. It is also clear that

$$\Delta(\hat{X}_i, \hat{X}_{i+1}) = \Delta(X, Y).$$

Finally, because of the triangle inequality for distances

$$\begin{aligned}\Delta(X^n, Y^n) &= \Delta(\hat{X}_0, \hat{X}_n) \\ &\leq \sum_{i=0}^{n-1} \Delta(\hat{X}_i, \hat{X}_{i+1}) \\ &= n\Delta(X, Y) \\ &\leq 2^{-\lambda+\beta}\end{aligned}$$

□

Appendix B.3. Proof of Lemma 4

To prove the lemma we will use another distribution called rounded discrete Gaussian.

Definition A1. *The* rounded Gaussian distribution *over $\mathbb{Z}$ with parameter $\sigma > 0$ is defined by the probability function*

$$\Omega_\sigma(z) = \int_{z-\frac{1}{2}}^{z+\frac{1}{2}} \rho_\sigma(x)dx$$

for $z \in \mathbb{Z}$ with

$$\rho_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}.$$

The distribution we are interested in is $\hat{\Omega}_\sigma$, its standard reduction modulo q as it is defined as

$$\hat{\Omega}_\sigma(i) = \sum_{k \in \mathbb{Z}} \Omega(i + kq).$$

It is clear once again from the definition that if $Y \sim \Omega_\sigma$, then $Y = \lfloor X \rceil$ with $X = N(0, \sigma)$, hence the name rounded Gaussian.

Now, we can see the relation between $\hat{\Omega}$ and $\overline{\Psi}$.

Lemma A3. *For any $\sigma \in \mathbb{R}$, we have that $\hat{\Omega}_\sigma$ is indeed a random variable and in fact we have that $\hat{\Omega}_\sigma = \overline{\Psi}_{\frac{\sigma}{q}}$.*

Proof. First, we need to see that $\hat{\Omega}_\sigma$ is a random variable. Indeed,

$$\begin{aligned}\sum_{i=0}^{q-1}\hat{\Omega}_\sigma(i) &= \sum_{i=0}^{q-1}\sum_{k\in\mathbb{Z}}\Omega_\sigma(i+kq)\\ &= \sum_{k\in\mathbb{Z}}\sum_{i=0}^{q-1}\Omega_\sigma(i+kq)\\ &= \sum_{\bar{k}\in\mathbb{Z}}\Omega_\sigma(\bar{k})\\ &= \sum_{\bar{k}\in\mathbb{Z}}\int_{\bar{k}-\frac{1}{2}}^{\bar{k}-\frac{1}{2}}\rho_\sigma(x)dx\\ &= \int_{-\infty}^{+\infty}\rho_\sigma(x)dx\\ &= 1.\end{aligned}$$

Now, we can see that $\hat{\Omega}_\sigma(i) = \overline{\Psi}_{\frac{\sigma}{q}}(i)$ for all $i \in \mathbb{Z}_q$, and therefore $\hat{\Omega}_\sigma = \overline{\Psi}_{\frac{\sigma}{q}}$ as random variables.

$$\begin{aligned}\hat{\Omega}_\sigma(i) &= \sum_{k\in\mathbb{Z}}\Omega_\sigma(i+kq)\\ &= \sum_{k\in\mathbb{Z}}\int_{i+kq-\frac{1}{2}}^{i+kq+\frac{1}{2}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\left(\frac{x}{\sqrt{2}\sigma}\right)^2}dx\\ &= \sum_{k\in\mathbb{Z}}\int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}}\frac{1}{\sqrt{2\pi}\sigma}e^{-\left(\frac{q(y+k)}{\sqrt{2}\sigma}\right)^2}q\cdot\, dy\\ &= \int_{\frac{i-\frac{1}{2}}{q}}^{\frac{i+\frac{1}{2}}{q}}\sum_{k\in\mathbb{Z}}\frac{1}{\sqrt{2\pi}\frac{\sigma}{q}}e^{-\left(\frac{y+k}{\sqrt{2}\frac{\sigma}{q}}\right)^2}dy\\ &= \overline{\Psi}_{\frac{\sigma}{q}}(i)\end{aligned}$$

where we have used the change of variables $y = \frac{x-kq}{q}$ and the dominated convergence theorem. □

Therefore, if we know a bound for $\hat{\Omega}$, we know a bound for $\overline{\Psi}$. Given this result, we can now bound the distributions using the fact that Ω is a rounded Gaussian and Mill's inequality:

$$\Pr[|N(0,\sigma)| > t] = 2\int_t^{+\infty}\rho_\sigma(x)dx \leq \sqrt{\frac{2}{\pi}}\frac{e^{-\left(\frac{t}{\sqrt{2}\sigma}\right)^2}}{t}.$$

Lemma A4. *For all $c, \sigma > 0$, then*

$$\Pr\left[\left|\overline{\Psi}_{\frac{\sigma}{q}}\right| > c\right] \leq \sqrt{\frac{2}{\pi}}\frac{e^{-\left(\frac{\lceil c\rceil-\frac{1}{2}}{\sqrt{2}\sigma}\right)^2}}{\lceil c\rceil-\frac{1}{2}}.$$

Proof. Let $c, \sigma > 0$, then

$$\begin{aligned}
\Pr\left[\left|\overline{\Psi}_{\frac{\sigma}{q}}\right| > c\right] &= \Pr\left(\left|\hat{\Omega}_\sigma\right| > c\right) \\
&\leq \Pr\left(\left|\Omega_\sigma\right| > c\right) \\
&= 2 \sum_{j=\lceil c \rceil}^{+\infty} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \rho_\sigma(x) dx \\
&= 2 \int_{\lceil c \rceil - \frac{1}{2}}^{+\infty} \rho_\sigma(x) dx \\
&\leq \sqrt{\frac{2}{\pi}} \frac{e^{-\left(\frac{\lceil c \rceil - \frac{1}{2}}{\sqrt{2}\sigma}\right)^2}}{\lceil c \rceil - \frac{1}{2}}
\end{aligned}$$

where we have used Lemma A3, thus seeing what we wanted. □

Appendix C. Link to Repository

All relevant codes for the implementation can be found in the following GitHub repository, last update made on 20 December 2021: https://github.com/FerranAlborch/Implementation-RLWE-based-distributed-key-generation-and-threshold-decryption.

References

1. Saračević, M.; Adamović, S.; Maček, N.; Elhoseny, M.; Sarhan, S. Cryptographic keys exchange model for smart city applications. *IET Intell. Transp. Syst.* **2020**, *14*, 1456–1464. [CrossRef]
2. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
3. Alagic, G.; Alperin-Sheriff, J.; Apon, D.; Cooper, D.; Dang, Q.; Kelsey, J.; Liu, Y.K.; Miller, C.; Moody, D.; Peralta, R.; et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*; US Department of Commerce, NIST: Washington, DC, USA, 2020.
4. De Feo, L.; Meyer, M. Threshold schemes from isogeny assumptions. In Proceedings of the IACR International Conference on Public-Key Cryptography, Edinburgh, UK, 4–7 May 2020; pp. 187–212.
5. Devevey, J.; Libert, B.; Nguyen, K.; Peters, T.; Yung, M. Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In Proceedings of the IACR International Conference on Public-Key Cryptography, Virtual Event, 10–13 May 2021; pp. 659–690.
6. Bendlin, R.; Damgård, I. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Proceedings of the Theory of Cryptography Conference, Zurich, Switzerland, 9–11 February 2010; pp. 201–218.
7. Singh, K.; Rangan, C.P.; Banerjee, A. Lattice-based identity-based resplittable threshold public key encryption scheme. *Int. J. Comput. Math.* **2016**, *93*, 289–307. [CrossRef]
8. Boneh, D.; Gennaro, R.; Goldfeder, S.; Jain, A.; Kim, S.; Rasmussen, P.M.; Sahai, A. Threshold cryptosystems from threshold fully homomorphic encryption. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2018; pp. 565–596.
9. Zhang, X.; Xu, C.; Jin, C.; Xie, R.; Zhao, J. Efficient fully homomorphic encryption from *RLWE* Ext. A Threshold Encryption Scheme. *Future Gener. Comput. Syst.* **2014**, *36*, 180–186. [CrossRef]
10. OQS Development Team. Open Quantum Safe (OQS). Available online: https://openquantumsafe.org/ (accessed on 20 January 2022).
11. Alborch Escobar, F. *RLWE*-Based Distributed Key Generation and Threshold Decryption. Master's Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 2021.
12. Boneh, D.; Shoup, V. A graduate Course in Applied Cryptography (2020). Draft Version 0.5 2020. Available online: https://toc.cryptobook.us/book.pdf (accessed on 10 December 2021).
13. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
14. Cramer, R.; Damgård, I.; Ishai, Y. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Proceedings of the Theory of Cryptography Conference, Cambridge, MA, USA, 10–12 February 2005; pp. 342–362.
15. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM (JACM)* **2009**, *56*, 1–40. [CrossRef]
16. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. *J. ACM (JACM)* **2013**, *60*, 1–35. [CrossRef]
17. Peikert, C.; Regev, O.; Stephens-Davidowitz, N. Pseudorandomness of *Ring-LWE* for any ring and modulus. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, Montreal, ON, Canada, 19–23 June 2017; pp. 461–473.

18. Micciancio, D.; Regev, O. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.* **2007**, *37*, 267–302. [CrossRef]
19. Albrecht, M.R.; Player, R.; Scott, S. On the concrete hardness of learning with errors. *J. Math. Cryptol.* **2015**, *9*, 169–203. [CrossRef]
20. Bellare, M. New proofs for *NMAC* and *HMAC*: Security without collision-resistance. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2006; pp. 602–619.