





Gladius: LWR based efficient hybrid public key encryption with distributed decryption

Kelong Cong¹, Daniele Cozzo¹, Varun Maram², and Nigel P. Smart^{1,3}

¹ imec-COSIC, KU Leuven, Leuven, Belgium.

² ETH, Zurich, Switzerland.

³ University of Bristol, Bristol, UK.

`kelong.cong@esat.kuleuven.be,`

`daniele.cozzo@kuleuven.be,`

`nigel.smart@kuleuven.be,`

`vmaram@inf.ethz.ch`

Abstract. Standard hybrid encryption schemes based on the KEM-DEM framework are hard to implement efficiently in a distributed manner whilst maintaining the CCA security property of the scheme. This is because the DEM needs to be decrypted under the key encapsulated by the KEM, before the whole ciphertext is declared valid. In this paper we present a new variant of the KEM-DEM framework, closely related to Tag-KEMs, which sidesteps this issue. We then present a post-quantum KEM for this framework based on Learning-with-Rounding, which is designed specifically to have fast distributed decryption. Our combined construction of a hybrid encryption scheme with Learning-with-Rounding based KEM, called Gladius, is closely related to the NIST Round 3 candidate called Saber. Finally, we give a prototype distributed implementation that achieves a decapsulation time of 4.99 seconds for three parties.

1 Introduction

The potential development of quantum computers means that we need to rethink which algorithms are going to be used for public key encryption and signatures; resulting in the subarea called post-quantum cryptography. The early days of post-quantum cryptography looked at how to build basic primitives such as simple public key encryption or signatures. However, now we realise that our existing (pre-quantum) public key algorithms often offer more than what is offered by basic public key primitives. For example one may have group signatures, identity-based encryption, or proofs-of-knowledge of the secret key, etc. In this work, we look at distributed decryption for IND-CCA *hybrid* public key encryption.

Even in the context of pre-quantum cryptography, distributed decryption for hybrid systems is problematic for many schemes, as to maintain security one would need to apply a distributed decryption procedure to the symmetric component, which is rather expensive. This problem, of the difficulty of constructing threshold IND-CCA encryption/encapsulation schemes Π_p , was first pointed out

in [37] and then elaborated upon in [48,49]. The problem being that Π_p would seem to require a publicly checkable CCA test. For historical (i.e. impractical) CCA secure public key encryption schemes such as Naor-Yung [41] and Dolev-Dwork-Naor [25] the check is simply the verification of a zero-knowledge proof, and is thus publicly verifiable.

However, for almost all practical encryption schemes the check is non-public and thus requires often expensive machinery to deploy in a threshold manner. In [48,49] Shoup and Gennaro present two schemes (called TDH1 and TDH2) which are IND-CCA and are based on the discrete logarithm problem, for which an efficient threshold decryption algorithm is possible. Both schemes bear a strong resemblance to Cramer-Shoup encryption [17]. These two constructions are however non-hybrid encryption mechanisms, but can be turned into hybrid threshold schemes using the Tag-KEM framework [1].

Our first contribution is to provide two transforms (one secure in the ROM and one secure in the QROM) which supports distributed decryption for hybrid encryption schemes. Our transform is closely related to the previous REACT [43] transform, the Tag-KEM framework [1], or the second hybrid-variant of the Fujisaki-Okamoto transform [27]. The key take away from our (general) hybrid construction is that the DEM component can be a generic one-time IND-CPA encryption scheme, and the KEM component can be either a *rigid*⁴ deterministic OW-CPA secure public key encryption scheme or (with a minor modification) a *rigid* OW-PCA-secure⁵ probabilistic scheme. In the case of public-key encryption schemes which are not perfectly correct, i.e. they exhibit decryption errors, we require an additional hardness assumption.

As our second contribution, to utilize our hybrid construction in the post-quantum setting we build a *rigid* deterministic encryption scheme which has a relatively efficient distributed decryption procedure based on the standard (or module) Learning-with-Rounding (LWR) problem. Our scheme is competitive (in terms of execution time and parameters) with Saber, the Learning-with-Rounding based submission in the third round of the NIST competition. Indeed the module-LWR version of our scheme has almost exactly the same parameters as Saber⁶, meaning that any run-times for Saber in hybrid encryption mode will be similar to the run-times for our scheme.

Due to the similarity with Saber we name our constructions of a hybrid encryption scheme, which has an efficient distributed decryption operation, based on Learning-with-Rounding, after the Roman sword Gladius; which came in four basic forms: A large one called Gladius-Hispaniensus, a smaller ‘standard’ one called Gladius-Pompeii, and two related ones called Gladius-Mainz and

⁴ A scheme is defined to be rigid if decryption of a ‘ciphertext’, which is not the output of an encryption operation, always returns \perp .

⁵ A scheme is said to be PCA (plain-check attack) secure if it is secure in the presence of an oracle which allows the adversary to check whether a given ciphertext encrypts a given plaintext.

⁶ Although there is an issue of having comparable security for these parameters, due to our reliance on LWE in the key generation phase, see Table 1 for more details.

Gladius–Fulham. In addition, we give in the full version a pre-quantum hybrid scheme based on ElGamal encryption and the gap-Diffie–Hellman assumption, along with a methodology to perform a distributed hybrid decryption.

Of the three lattice based finalists in Round 3 of the NIST competition two of them, Crystals-Kyber [47], and Saber [22], all construct a hybrid encryption scheme by first building an IND-CPA encryption scheme, and then creating an IND-CCA hybrid scheme using the Fujisaki-Okamoto transform [26]. The problem with the Fujisaki-Okamoto design pattern is that the decryption procedure needs to perform a hash to obtain the random coins used for encryption. In the threshold setting this is a problem as one needs to hash both the DEM key k and the DEM value itself (or the message) in the Fujisaki-Okamoto transform to perform the re-encryption; and this must be done *before* one reveals k and m to the decrypting parties. The hash function used for re-encryption also needs to produce the random values used in encryption, which can be a complicated process to perform in a threshold manner for the lattice based schemes; especially if this involves sampling discrete Gaussians or other distributions which are not ‘native’ to whichever underlying methodology one is using to perform the threshold decryption.

The other remaining lattice based finalist in Round 3, NTRU [54], also builds a traditional KEM, with the difference that the KEM does not require re-encryption. However, NTRU builds a traditional KEM, which requires the DEM to be implemented in a threshold manner so as to maintain the CCA security. Thus threshold variants of all the remaining Round 3 lattice based schemes will be problematic if one wishes to maintain CCA security of the threshold variant.

Of the Round-2 lattice-based systems which did not progress to be finalists in Round-3, FrodoKEM [40], Round 5 [28], LAC [38], NewHope [45], and ThreeBears [30], also follow the Fujisaki-Okamoto pattern, bar NTRUprime [10]. NTRUprime differs from the previous ones in that it is based on a *rigid* deterministic base encryption scheme which is then turned into a KEM using [24, Section 6]. However, the underlying rigid deterministic encryption scheme still requires re-encryption to be secure, and as we remarked above this causes problems for thresholdizing the scheme.

1.1 Prior Work and Our Contribution

Threshold Decryption: As stated at the beginning our main goal is to provide an efficient threshold decryption procedure for a post-quantum *hybrid* encryption algorithm. We do this by providing an algorithm which is efficient, within a generic MPC framework, to perform distributed decryption. Thus, on the assumption the algorithm we implement is correct, the security of said algorithm follows from the security of the base MPC framework.

In [8] a *non-hybrid* lattice based encryption scheme is given. But the security of the underlying encryption scheme is only IND-CPA (although an actively secure distributed decryption protocol for the IND-CPA scheme is given). In [13] a generic procedure for obtaining an arbitrary threshold variant of any functional-

ity, however the construction makes use of Fully Homomorphic Encryption and is not practical.

In an earlier work [35] on distributing the decryption for a Round-1 NIST candidate which was based on Ring-LWE, namely LIMA, a distributed decryption operation was given for a basic (non-hybrid) encryption scheme. An outline for the hybrid scheme was given, but the instantiation would not preserve the CCA security guarantees of the hybrid construction, i.e. the method presented was *not* secure.

From a performance perspective the problem with the distributed decryption of LIMA was that it is a scheme based on the Fujisaki-Okamoto transform. As mentioned above the secure evaluation of the hash function and re-encryption operation is costly in the distributed setting. But this is not the only problem with [35], the decryption procedure itself is rather complicated in that it requires rounding of integers, for example. In [35] these two technical complexities meant the protocol (to be fast) was only a 3-party protocol with one dishonest party. The distributed decryption of a single non-hybrid LIMA encryption would take 4.2 seconds, with a similar time for the *insecure* hybrid KEM distributed decapsulation.

Traditionally, in the non-hybrid encryption setting, threshold decryption is preferred using the least amount of interaction, for example see [36,48,49]. Our threshold decryption procedure for our post-quantum hybrid scheme utilizes explicitly generic MPC techniques; thus it definitely does not minimize the level of interaction between the parties needed. An open problem would be to develop a methodology, or scheme, which can utilize the minimal amount of communication possible.

We note that there has been some work on threshold post-quantum signature schemes, e.g. [15,16,23], but the techniques and issues are rather different from those employed and discussed here.

Hybrid Encryption: Hybrid encryption is the standard method to encrypt large message via a public key scheme. The actual message is encrypted via a standard block cipher in a secure AEAD mode, such as AES-GCM. Then the one-time symmetric key for this symmetric encryption scheme is transferred to the recipient using a public key methodology. The traditional method of combining the public key encryption scheme $\Pi_p = (\mathcal{K}_p, \mathcal{E}_p, \mathcal{D}_p)$, with message space \mathcal{M}_p , and symmetric key encryption scheme $\Pi_s = (\mathcal{K}_s, \mathcal{E}_s, \mathcal{D}_s)$ into a hybrid scheme $\Pi_h = (\mathcal{K}_h, \mathcal{E}_h, \mathcal{D}_h)$ is called KEM-DEM [18]. Where $\mathcal{K}_\star, \mathcal{E}_\star$ and \mathcal{D}_\star are the various schemes key-generation, encryption and decryption algorithms respectively.

The KEM-DEM method of [18] requires Π_s to be a (one-time) IND-CCA symmetric cipher⁷ and an IND-CCA KEM scheme Π_p (a KEM is a public key scheme designed to encrypt only symmetric keys). The scheme Π_p encrypts the key k for Π_s , and then Π_s is used to encrypt the message using the key k . In particular the encryption algorithm, outputting (c_1, c_2) for \mathcal{E}_h is along the lines

⁷ One time meaning that the attacker does not get access to an encryption oracle.

of

$$k \leftarrow \mathcal{M}_p, \quad k \leftarrow H(k), \quad c_1 \leftarrow \mathcal{E}_p(\text{pk}, k), \quad c_2 \leftarrow \mathcal{E}_s(k, m).$$

However, there is a problem with this construction when one looks for a distributed variant of the decryption algorithm. Even if the decryption algorithm of the KEM Π_p has an efficient distributed decryption operation one cannot derive an efficient distributed hybrid cipher as the decryption of the scheme Π_s needs to be executed also in a distributed manner. Executing Π_s in a distributed manner for standard symmetric encryption scheme is possible, but very inefficient for long messages.

One obvious way to get around this problem is for the distributed decryption operation for the hybrid cipher Π_h to output k in the clear after the Π_p part has been executed, enabling the decryption using Π_s to be done in the clear. We call such a hybrid scheme ‘leaky’, as the decryption algorithm leaks the underlying symmetric key even if the symmetric component does not decrypt correctly. This intuitively seems attractive, however it breaks the IND-CCA security of the hybrid scheme Π_h via a trivial attack.

The most popular generic transform to turn a public key encryption scheme into a hybrid scheme in the KEM-DEM paradigm is the Fujisaki-Okamoto transform [26,27]. This comes in two forms, either (from [26])

$$k \leftarrow \mathcal{M}_p, \quad k \leftarrow H(k), \quad c_1 \leftarrow \mathcal{E}_p(\text{pk}, k; G(k, m)), \quad c_2 \leftarrow \mathcal{E}_s(k, m),$$

or (from [27])

$$k \leftarrow \mathcal{M}_p, \quad k \leftarrow H(k), \quad c_2 \leftarrow \mathcal{E}_s(k, m), \quad c_1 \leftarrow \mathcal{E}_p(\text{pk}, k; G(k, c_2)),$$

where G is a hash function which produces the random coins needed by the encryption algorithm \mathcal{E}_p . The authors of [26,27] show that this hybrid scheme, assuming some (mild) technical conditions on the encryption algorithm, is IND-CCA if Π_p is OW-CPA and Π_s is IND-CPA. Note, for the first variant one needs to decrypt c_2 before one can verify the c_1 component, as the decryption operation \mathcal{D}_p requires re-encryption to perform the necessary CCA checks. Because of this, the first Fujisaki-Okamoto hybrid construction can never be securely ‘leaky’.

The second Fujisaki-Okamoto variant has been proved secure in the quantum random-oracle model in [53], where the scheme Π_p is assumed to be ‘well-spread’, perfectly correct and OW-CPA secure. This second Fujisaki-Okamoto variant can be considered as a variant of the Tag-KEM framework of [1]. The Tag-KEM framework gives another hybrid construction, which works (roughly speaking in the simplest instance) in the following manner

$$k \leftarrow \mathcal{K}_s, \quad c_2 \leftarrow \mathcal{E}_s(k, m), \quad c_1 \leftarrow \mathcal{E}_p(\text{pk}, k \| G(c_2))$$

where G is a hash function. This hybrid construction is secure if Π_p is IND-CCA secure and Π_s is one-time IND-CPA secure.

Note in [31] a QROM proof of the non-hybrid encryption version of the Fujisaki-Okamoto transform is given, that this is for the public key scheme given

by $c \leftarrow \mathcal{E}_p(\mathbf{pk}, m; G(m))$. However, unlike in [53], the encryption scheme is not assumed to be perfectly correct.

One of the applications of the Tag-KEM framework mentioned in [1] is that of threshold hybrid public key encryption. Their argument is as follows. Since the one-time-pad is one-time IND-CPA secure, outputting m already leaks k . Thus revealing the value k before applying the decryption of c_2 cannot break security, as that would contradict their main theorem. Thus one can apply threshold decryption to obtain the decryption of c_1 , leak the key k and then decrypt c_2 in the clear as long as Π_s is the one-time-pad encryption scheme. Unfortunately, the authors of [1] require an IND-CCA secure Π_p .

The authors of [1] provide other constructions requiring weaker properties of Π_p , but each one adds its own complications. Indeed if one thinks of the hash function G , in our construction below, applied to c_1, c_2 and k as a MAC function applied to c_1 and c_2 with key k , then their ‘weak KEM+MAC’ construction is identical to ours.

In [7] a construction of CCA secure Tag-based encryption which has threshold decryption is discussed. Their generic methodology uses one-time signatures and a concrete instantiation is given based on the decisional bilinear Diffie–Hellman assumption in pairing groups. Another construction of a threshold tag-KEM in the Random Oracle model based on the RSA problem is given in [32].

The solution we propose is to utilize the following modification to the Cramer-Shoup basic construction. Our main construction, which we call Hybrid_1 , outputs a ciphertext of the form (c_1, c_2, c_3) where, for a hash function G modelled as a random oracle,

$$k \leftarrow \mathcal{M}_p, \quad \mathbf{k} \leftarrow H(k), \quad c_1 \leftarrow \mathcal{E}_p(\mathbf{pk}, k), \quad c_2 \leftarrow \mathcal{E}_s(\mathbf{k}, m), \quad c_3 \leftarrow G(c_1, c_2, k).$$

The distributed decryption algorithm checks the c_3 component and then ‘leaks’ the key k in the clear, enabling \mathbf{k} to be produced and hence m decrypted from the c_2 component. We show that this scheme is IND-CCA secure, even with this form of leaky decryption, if the scheme Π_p is *rigid, deterministic* and OW-CPA, or *rigid, randomized* and OW-PCA secure, and the scheme Π_s is one-time IND-CPA secure. If the scheme Π_p is not perfectly correct then we require the additional hardness assumption that it is hard for the adversary to construct a message/ciphertext pair (m, c) such that $c = \mathcal{E}_p(\mathbf{pk}, m)$, but $\mathcal{D}_p(\mathbf{sk}, c) = \perp$. We also require in this case that the probability of the encryption scheme having collisions, i.e. two messages which encrypt to the same ciphertext, is negligible when this probability is computed over the space of all possible public/private key pairs.

When Π_p is randomized and OW-PCA, one needs to include c_1 into the hash function G so as to avoid attacks related to re-randomization of the output of \mathcal{E}_p . In this latter case, of randomized OW-PCA encryption scheme Π_p , our construction looks most closely related to the REACT transform, from [43], which encrypts via

$$k \leftarrow \mathcal{M}_p, \quad \mathbf{k} \leftarrow H(k), \quad c_1 \leftarrow \mathcal{E}_p(\mathbf{pk}, k), \quad c_2 \leftarrow \mathcal{E}_s(\mathbf{k}, m), \quad c_3 \leftarrow G(k, m, c_1, c_2).$$

The authors of [43] show that REACT is secure assuming Π_p OW-PCA secure and the scheme Π_s is IND-CPA secure. The REACT transform has a similar problem with the standard KEM-DEM construction above in that it requires c_2 to be decrypted before the check is applied, i.e. m is needed as an input to G .

In the case when Π_p is rigid and deterministic one can drop the component c_1 from the input to G . So our hybrid construction simplifies to

$$k \leftarrow \mathcal{M}_p, \quad \mathbf{k} \leftarrow H(k), \quad c_1 \leftarrow \mathcal{E}_p(\mathbf{pk}, k), \quad c_2 \leftarrow \mathcal{E}_s(\mathbf{k}, m), \quad c_3 \leftarrow G(c_2, k).$$

In this case one can think of our construction as precisely the second Fujisaki-Okamoto construction utilizing the OW-CPA, ‘well-spread’ public key encryption scheme with encryption algorithm given by

$$\mathcal{E}'_p(\mathbf{pk}, k; r) = (\mathcal{E}_p(\mathbf{pk}, k), r).$$

Thus our construction in this case would be automatically secure in the QROM if one considers only normal decryption oracle queries (i.e. ones which do not leak the key k); assuming that the techniques used in [31] for dealing with non-perfectly correct schemes could be extended to the proof in [53].

However, this hybrid construction seems hard to prove QROM secure when one requires threshold decryption, unless one picks the DEM operation to be a one-time pad encryption scheme. To obtain a full QROM secure efficient hybrid construction with distributed decryption we present a second hybrid construction which adds a ciphertext component, by hashing k with a second hash function H' which has domain and codomain equal to \mathcal{M}_p , as well as hashing k via another hash function H'' , before passing the result into G ; namely we compute

$$k \leftarrow \mathcal{M}_p, \quad \mathbf{k} \leftarrow H(k), \quad \mu \leftarrow H'(k), \\ c_1 \leftarrow \mathcal{E}_p(\mathbf{pk}, k), \quad c_2 \leftarrow \mathcal{E}_s(\mathbf{k}, m), \quad c_3 \leftarrow G(c_2, \mu), \quad c_4 \leftarrow H''(k).$$

This construction, which we call Hybrid_2 , is proved secure, in the QROM, using the techniques of [51].

Most of our technical difficulties arise from the fact we want both efficient distributed decryption and an efficient DEM operation. If we take an AES-based DEM then the output of the hash function H will be a bit vector in $\{0, 1\}^{|k|}$. But the input k will be ‘native’ to the underlying public key scheme, and thus in general an element of a set such as \mathbb{F}_p^n , for some modulus p . This means H needs to map from one arithmetic domain to another. It is to avoid needing to do this in a secure way during distributed decryption that we ‘leak’ the key k and not the key \mathbf{k} . This problem does not occur with the hash function G as we are free to select the hash function so that it can be evaluated securely. In our QROM construction using the c_4 component we need to evaluate H' and H'' securely before releasing k , but this can be done as efficiently as evaluating G , by selecting the hash functions H' and H'' in an appropriate way.

Learning-with-Rounding: After detailing our main hybrid constructions we go on to discuss how one can instantiate a suitable KEM in the post-quantum setting.

For this we utilize the Learning-With-Rounding (LWR) based deterministic encryption algorithm first presented in [52], and then refined in [4]. This is itself inspired by the trapdoor LWE key generation procedure introduced by Micciancio and Peikert [39]. We present an explicit construction, including suggested parameters sizes, and compare the resulting scheme with current NIST PQ-candidates such as Saber [22]. Our basic construction utilizes the fact that LWR encryption is deterministic in nature.

To prove our main hybrid constructions secure we need to assume, for our most efficient construction, a new hard problem, which we dub the Large-Vector-Problem (LVP) problem. Informally, this problem says that for a given LWE key $(A, A \cdot R_1 + R_2)$ with $A \in \mathbb{Z}_q^{n \times n}$ uniformly randomly chosen and $R_1, R_2 \in \mathbb{Z}_q^{n \times n}$ but with ‘small’ entries, it is hard to find a small vector \mathbf{m} such that $R_1 \cdot \mathbf{m}$ is ‘relatively big’. This is needed to establish our scheme satisfies a property that we call \perp -Aware. The \perp -Aware property captures the difficulty of an adversary \mathcal{A} , given the public key pk , to come up with a plaintext/ciphertext pair (m, c) such that $c = \mathcal{E}(\text{pk}, m)$ but $\mathcal{D}(\text{sk}, c) = \perp$; i.e. a ciphertext which is a valid encryption, but which does not decrypt correctly.

The Gladius Family of Hybrid Ciphers: Combining our LWR-based rigid deterministic OW-PCA encryption scheme with our hybrid constructions we obtain a post-quantum secure hybrid cipher, which supports efficient distributed decryption. We can actually derive many variants depending on the choice of Hybrid_1 or Hybrid_2 , the choice of the DEM, and the choice of using plain LWR or Module-LWR. We focus on four specific variants of this construction; Gladius–Hispaniensis (based on Hybrid_1 and plain LWR), Gladius–Pompeii and Gladius–Mainz (based on Hybrid_1 and Module-LWR), and Gladius–Fulham (based on Hybrid_2 and Module-LWR).

Gladius–Hispaniensis, Gladius–Pompeii and Gladius–Fulham all assume *any* one-time IND-CPA secure DEM. For Gladius–Hispaniensis and Gladius–Pompeii we obtain (expected) security in the QROM when the scheme is considered as a standard hybrid encryption scheme, and security in the ROM when we consider the scheme in the threshold setting (due to the additional leakage required). The expected QROM security, which we denote by QROM^* , comes from the fact that Zhandry’s proof [53], for the second Fujisaki–Okamoto transform, only applies to perfectly correct schemes. We also present a third variant Gladius–Mainz (see the full version) which provides QROM^* security in the threshold setting, but this requires the DEM to be a one-time-pad (OTP), and requires a more expensive distributed decryption algorithm. Our fourth variant, Gladius–Fulham, utilizes the second hybrid transform mentioned above, but can achieve full QROM security (including for non-perfectly correct schemes Π_p) even when one allows the leakage from the decryption oracle required in a distributed decryption operation.

In summary the properties of our four schemes are given by the following table, where \checkmark^* in the QROM column refers to the above QROM^* caveat. We also note in the table how many secure hash function operations need to be executed by the distributed decryption algorithm.

	Hard Problem	Hybrid	DEM	Standard QROM Secure	Threshold QROM Secure	Threshold ROM Secure	No. Secure Hashes
Gladius–Hispaniensis	LWR	1	Generic	✓*	✗	✓	1
Gladius–Pompeii	Module-LWR	1	Generic	✓*	✗	✓	1
Gladius–Mainz	Module-LWR	1	OTP	✓*	✓*	✓	1
Gladius–Fulham	Module-LWR	2	Generic	✓	✓	✓	3

Open Questions: Our work leads to a number of new interesting areas of research. On a theoretical level we leave open the problem of establishing \perp –Aware security for our Gladius construction when q is a power of two (since we focus on q prime for threshold reasons) and also on a theoretical level to prove our conjecture related to the hardness of the LVP problem. On a practical level one could examine other ways of using our Hybrid construction to build efficient threshold post-quantum encryption schemes, or remove the need to use generic MPC; which comes about mainly due to the need to execute a hash function for key derivation and to perform the necessary rounding operations.

2 Preliminaries

Learning-with-Errors and Learning-with-Rounding: We let σ denote a standard deviation, and we let \mathcal{D}_σ denote a distribution which ‘looks like’ a discrete Gaussian distribution with standard deviation σ . In practice this can be generated by the NewHope methodology [3], namely if we have $\sigma = \sqrt{(B+1)/2}$ then we sample from \mathcal{D}_σ by generating $2 \cdot B + 2$ random bits (b_i, b'_i) for $i = 0, \dots, B$, and then generating a sample by computing $\sum_{i=0}^B (b_i - b'_i)$.

Given a secret vector $\mathbf{s} \in \mathbb{Z}_q^d$ (for some integer q), then a Learning-with-Errors (LWE) sample is a pair $(A, A \cdot \mathbf{s} + \mathbf{e})$ where $A \in \mathbb{Z}_q^{m \times d}$ is chosen uniformly at random and $\mathbf{e} \leftarrow \mathcal{D}_\sigma$. The decision LWE problem is to distinguish LWE samples from uniformly random samples (A, \mathbf{u}) , for $\mathbf{u} \leftarrow \mathbb{Z}_q^m$, we denote this problem by $\text{LWE}_{q,(m,d),\sigma}$. The search LWE problem is to recover the secret vector \mathbf{s} from a set of LWE samples. For suitable choices of the parameters both these problems are known to be equivalent and assumed to be hard. Suitable parameters to ensure hardness given known attack algorithms can be found using Albrecht’s *LWE-estimator* tool⁸.

For integers p and q we define the following map

$$[x]_p : \begin{cases} \mathbb{Q} & \longrightarrow & \mathbb{Z}_p \\ x & \longmapsto & [x \cdot p/q] \pmod{p} \end{cases}$$

where $[\cdot]$ is the round to nearest integer function, with rounding towards zero in the case of values of the form $i/2$ for i an odd integer. If the input value $x \in (-q/2, \dots, q/2] \subset \mathbb{Z}$ then the final reduction modulo p is only required (if p does not divide q) when the rounding ends up being outside the interval $(-p/2, \dots, p/2]$, which happens with probability about $1/p$, resulting in needing a single addition of p to accomplish the reduction modulo p .

⁸ <https://bitbucket.org/malb/lwe-estimator/src/master/>

Given a secret vector $\mathbf{s} \in \mathbb{Z}_q^d$, then a Learning-with-Rounding (LWR) sample is a pair $(A, \lfloor A \cdot \mathbf{s} \rfloor_p)$ where $A \in \mathbb{Z}_q^{m \times d}$ is chosen uniformly at random. The decision LWR problem is to distinguish LWR samples from uniformly random samples $(A, \lfloor \mathbf{u} \rfloor_p)$, for $\mathbf{u} \leftarrow \mathbb{Z}_q^d$, we denote this problem by $\text{LWR}_{q,p,(m,d)}$. The search problem is similarly defined as the problem of recovering \mathbf{s} from a number of LWR samples.

Relation between (Module-) LWE and (Module-) LWR: In the full version we extend these notions to the case of Learning-with-Rounding over modules. The search (Module-) LWE and (Module-) LWR problems are linked theoretically by the following theorem [12, Theorem 1 and 2]⁹.

Theorem 2.1. *Let p, q, n, d, m and B be integers such that $q > 2 \cdot p \cdot B$. For every algorithm Learn there is an algorithm Learn' such that*

$$\begin{aligned} \Pr_{A, \mathbf{s}, \mathbf{e}} [\text{Learn}'(A, A \cdot \mathbf{s} + \mathbf{e}) = \mathbf{s}] &\geq \Pr_{A, \mathbf{s}, \mathbf{e}} [\text{Learn}(A, \lfloor A \cdot \mathbf{s} + \mathbf{e} \rfloor_p) = \mathbf{s}] \\ &\geq \frac{\Pr_{A, \mathbf{s}} [\text{Learn}(A, \lfloor A \cdot \mathbf{s} \rfloor_p) = \mathbf{s}]^2}{(1 + 2 \cdot p \cdot B/q)^{n \cdot m \cdot d}} \end{aligned}$$

where $A \leftarrow \mathcal{R}_q^{m \times d}$, the noise \mathbf{e} is independent over all m coordinates, B -bounded and balanced in each coordinate, and $\mathbf{s} = (s_i) \in \mathcal{R}_q^d$ is chosen from any distribution such that $s_i \in \mathcal{R}_q^*$ for some i .

Note, the first inequality is not from [12] but it is immediate. To apply this result, we would take $B = \mathfrak{c} \cdot \sigma$, for some suitable constant \mathfrak{c} .

The fact that the square of the LWR advantage is bounded by the LWE advantage implies that one will need larger parameters to bound the LWR advantage by a given value, than to bound the LWE advantage by the same value. Thus using this theoretical reduction will result in very large parameters indeed. To avoid the problem with the above reduction submissions to the NIST Post-Quantum cryptography competition based on LWR, such as Saber [21,22], estimate their parameters by using the best attack scenario. In other words the security is estimated using Albrecht's LWE-estimator directly, or by assuming the above theorem is an exact inequality between the various one-way advantages.

This approach is examined in detail in [2], where to utilize Albrecht's tool the authors need to translate the LWR parameters into LWE parameters. In [2] this is done by setting the LWE standard deviation to be

$$\sigma = \sqrt{\frac{(q/p)^2 - 1}{12}}.$$

⁹ The result in [12] is only given for normal and Ring LWE/LWR, but extending the result to the module variants is immediate.

Asymmetric and Symmetric Encryption: An asymmetric encryption scheme is a triple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, all of which are probabilistic polynomial time (PPT) algorithms. We let \mathcal{M} denote the plaintext space of Π , \mathcal{C} the ciphertext space and \mathcal{R} the space of random coins of Π . The key generation algorithm \mathcal{K} takes as input 1^t , where t is a security parameter and outputs a public/private key pair $(\mathbf{pk}, \mathbf{sk})$. A symmetric encryption scheme is one in which $\mathbf{pk} = \mathbf{sk}$. The standard security definitions are given in the full version.

The algorithm $\mathcal{E}(\mathbf{pk}, m; r)$ takes a message $m \leftarrow \mathcal{M}$, a public key \mathbf{pk} and random coins $r \leftarrow \mathcal{R}$ and returns a ciphertext c . The decryption algorithm $\mathcal{D}(\mathbf{sk}, c)$ recovers the message m or returns the special symbol \perp . For correctness we require

$$\Pr \left[\mathcal{D}(\mathbf{sk}, c) = m : (\mathbf{pk}, \mathbf{sk}) \leftarrow \mathcal{K}(1^t), m \leftarrow \mathcal{M}, r \leftarrow \mathcal{R}, c \leftarrow \mathcal{E}(\mathbf{pk}, m; r) \right] = 1 - \delta,$$

where δ is an exponentially small probability of decryption failure. If $\delta = 0$ we say the scheme is perfectly correct. A public key scheme will be called *deterministic* if \mathcal{R} contains only the empty string (or equivalently one element), otherwise it will be called *randomized*.

A scheme which is not perfectly correct can exhibit two forms of decryption failures; either two messages could map under encryption to the same ciphertext or a valid ciphertext could decrypt to \perp . For the first case we say an encryption scheme is δ_c -Collision Free if

$$\Pr_{(\mathbf{sk}, \mathbf{pk}) \leftarrow \mathcal{K}(1^t)} \left[\exists m_1, m_2 \in \mathcal{M}, \exists r_1, r_2 \in \mathcal{R} : m_1 \neq m_2, \mathcal{E}(\mathbf{pk}, m_1; r_1) = \mathcal{E}(\mathbf{pk}, m_2; r_2) \right] = \delta_c.$$

A perfectly correct encryption scheme is 0-Collision Free.

For the second case of decryption failure we consider the following game, which we call \perp -Aware. The adversary \mathcal{A} is given the public key \mathbf{pk} and is required to come up with a plaintext/ciphertext pair (m, c) such that $c = \mathcal{E}(\mathbf{pk}, m)$ but $\mathcal{D}(\mathbf{sk}, c) = \perp$. We define

$$\text{Adv}_{\Pi, \mathcal{A}}^{\perp\text{-Aware}}(t) = \Pr \left[(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathcal{K}(1^t), (m, c) \leftarrow \mathcal{A}(\mathbf{pk}) : c = \mathcal{E}(\mathbf{pk}, m), \mathcal{D}(\mathbf{sk}, c) = \perp \right]$$

and say that Π is \perp -Aware if $\text{Adv}_{\Pi, \mathcal{A}}^{\perp\text{-Aware}}(t)$ is a negligible function of t for all PPT \mathcal{A} . Note, if Π is perfectly correct then $\text{Adv}_{\Pi, \mathcal{A}}^{\perp\text{-Aware}}(t) = 0$.

An asymmetric encryption scheme is said to be *rigid*, see [11] (where the definition is given just for deterministic schemes, but the generalization to probabilistic schemes is immediate) if

$$\Pr \left[(\mathbf{pk}, \mathbf{sk}) \leftarrow \mathcal{K}(1^t), c \leftarrow \mathcal{C} \setminus \mathcal{C}^\perp, \exists r \in \mathcal{R} : \mathcal{E}(\mathbf{pk}, \mathcal{D}(\mathbf{sk}, c); r) = c \right] = 1,$$

where $\mathcal{C}^\perp \subset \mathcal{C}$ is the set of all ciphertexts $c \in \mathcal{C}$ for which $\mathcal{D}(\mathbf{sk}, c) = \perp$. The effect of rigidity is that unless c is the output of $\mathcal{E}(\mathbf{pk}, m; r)$ for some m and r , then decryption will always return \perp . ElGamal is an example of a perfectly correct, rigid probabilistic scheme as every ciphertext pair $(c_1 = g^r, c_2 = m \cdot h^r)$ corresponds to the encryption of some message.

If we let $\|X\|$ be the infinity norm on the probability space X of a finite set S , then the min-entropy of X is $-\log \|X\|$. A randomized asymmetric encryption scheme is said to be γ -spread if

$$-\log \max_{y \in \{0,1\}^*} \Pr \left[r \leftarrow \mathcal{R} : y = \mathcal{E}(\mathbf{pk}, m; r) \right] \geq \gamma$$

for all $(\mathbf{pk}, \mathbf{sk})$ output by \mathcal{K} and all $m \in \mathcal{M}$. A scheme is said to be *well-spread* if $\gamma = \omega(\log t)$. This basically means that the probability of a specific ciphertext occurring is negligibly small.

Note, if the set \mathcal{R} is suitably large then we can turn a deterministic scheme Π_p into a randomized well-spread scheme Π'_p by setting $\mathcal{E}'_p(\mathbf{pk}, k; r) = (\mathcal{E}_p(\mathbf{pk}, k), r)$. It is from this observation, the QROM^{*} security in the standard hybrid (non-leaky) encryption model for our construction based on deterministic public key encryption, mentioned in the introduction, follows.

Encryption With Distributed Decryption: Given a set $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ of parties, we consider access structures A consisting of a monotonically increasing set of subsets of $2^{\mathcal{P}}$. A set S is said to be qualified if $S \in A$, and unqualified otherwise. Given an encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ we say that the scheme admits a distributed decryption functionality for an access structure A , if there are two n -party protocols $\Pi_{\mathcal{K}}$ and $\Pi_{\mathcal{D}}$. The protocol $\Pi_{\mathcal{K}}$ produces some data \mathbf{sk}_i for each party, called the secret key shares. The protocol $\Pi_{\mathcal{D}}$ on input of an agreed ciphertext c from all parties in $S \in A$, and the value \mathbf{sk}_i from all parties in S , will output the value $m = \mathcal{D}(\mathbf{sk}, c)$.

The distributed decryption protocols are said to be secure (in the IND-ATK sense) if an unqualified set of adversarial parties cannot, while interacting with a qualified set of parties, break the IND-ATK security of the underlying encryption scheme. This security definition can be made more formal by saying that the distributed decryption protocol should act like an ideal decryption functionality. See [48,49] for a specific instantiation.

We shall assume an actively secure MPC protocol for the access structure A , and will then construct an algorithm which implements the algorithm \mathcal{D} within the MPC protocol. Thus it automatically becomes a distributed protocol $\Pi_{\mathcal{D}}$ for the decryption functionality, and its security is inherited from the underlying MPC protocol. The challenging part is to develop the encryption scheme and the specific instantiation of \mathcal{D} to enable the underlying MPC system to provide an efficient distributed implementation.

By using a generic MPC functionality, as opposed to a specific protocol, we restrict ourselves to the threshold case where *all* parties have to be involved in the computation; but where security is maintained against an adversary controlling

a given threshold. This is in contrast to the models proposed in [48,49] which allow for a subset of the key-share holding parties to participate.

KEM-DEM Philosophy: A central tenet when using public key encryption in practice, is that one never encrypts a large message with a public key algorithm. Instead one encrypts the actual message with a fast symmetric key algorithm, such as AES-GCM, and then the symmetric key is transferred to the recipient using a public key scheme. Thus producing a *hybrid* encryption scheme. In this way the symmetric key is only used once in the symmetric cipher, and thus we do not need a fully secure AEAD scheme but the weaker notion of a DEM, and the public key scheme is only needed to transport a single random key (and not a message) leading to the simpler public key construction of a KEM. See [18] for an extensive discussion, with the standard definitions and proofs.

$\mathcal{K}_h(1^t)$: $(\text{pk}, \text{sk}) \leftarrow \mathcal{K}_p(1^t)$ Return (pk, sk)	$\mathcal{E}_h(\text{pk}, m)$: $k \leftarrow \mathcal{M}_p$ $r \leftarrow \mathcal{R}_p, r' \leftarrow \mathcal{R}_s$ $k \leftarrow H(k)$ $c_1 \leftarrow \mathcal{E}_p(\text{pk}, k; r)$ $c_2 \leftarrow \mathcal{E}_s(k, m; r')$ Return (c_1, c_2)	$\mathcal{D}_h(\text{sk}, (c_1, c_2))$: $k \leftarrow \mathcal{D}_p(\text{sk}, c_1)$ If $k = \perp$ then return \perp $k \leftarrow H(k)$ $m \leftarrow \mathcal{D}_s(k, c_2)$ Return m
---	---	---

Fig. 1. The Standard KEM-DEM Construction

We let $\Pi_p = (\mathcal{K}_p, \mathcal{E}_p, \mathcal{D}_p)$ denote an IND-CCA public key encryption scheme with message space \mathcal{M}_p , ciphertext space \mathcal{C}_p , and space of random coins \mathcal{R}_p , and let $\Pi_s = (\mathcal{K}_s, \mathcal{E}_s, \mathcal{D}_s)$ denote an IND-CCA symmetric key encryption scheme (which recall for us is always one-time and hence a DEM) with message space $\mathcal{M}_s = \{0, 1\}^*$. From these two components one can construct a KEM-DEM encryption scheme for arbitrary long messages as follows: We first define a hash function $H : \mathcal{M}_p \rightarrow \mathcal{K}_s$ where by abuse of notation by \mathcal{K}_s we mean the key space of Π_s . We can then define a hybrid encryption scheme $\Pi_h = (\mathcal{K}_h, \mathcal{E}_h, \mathcal{D}_h)$ in Figure 1.

Naive Threshold KEM-DEM: The goal of our work is to produce threshold public key encryption for long messages; namely we would want to share the decryption key amongst a set of entities so that a given subset needs to come together to decrypt. Clearly we would not want the extra expense of the threshold decryption to impact when encrypting very large messages. Thus we would want to use something akin to the KEM-DEM philosophy, with the main message being encrypted and decrypted via a fast cipher such as AES-GCM.

Finding KEM-like constructions which admit distributed decryption protocols is relatively easy. However, whilst it is possible to execute AES in a threshold manner, see e.g. [33,44], the performance for long messages is prohibitive. Thus

distributed DEMs are much harder to obtain. For this reason we would like to apply the decryption of the large message ‘in the clear’, but this implies that the decryption algorithm will need to ‘leak’ the decryption key k of the DEM component. In particular this key will leak irrespective of whether the DEM decrypts correctly or not; since the decrypting parties need to obtain the DEM key before it is known whether the key is valid for the DEM.

$$\begin{aligned}
 &\mathcal{D}_h(\mathbf{sk}, (c_1, c_2)): \\
 &\quad k \leftarrow \mathcal{D}_p(\mathbf{sk}, c_1) \\
 &\quad \text{If } k = \perp \text{ then return } (\perp, \perp) \\
 &\quad k \leftarrow H(k). \\
 &\quad m \leftarrow \mathcal{D}_s(k, c_2) \\
 &\quad \text{Return } (k, m)
 \end{aligned}$$

Fig. 2. Decryption Funtionality for Standard Distributed KEM-DEM

Our decryption algorithm functionality, and thus the functionality of any decryption oracle given to an adversary, would therefore be of the form in Figure 2. This provides an immediate attack in the standard IND-CCA model on the hybrid construction. An adversary takes the target ciphertext (c_1^*, c_2^*) , submits (c_1^*, c_2) to the decryption oracle for a random value c_2 . With high probability, they will receive (k, \perp) . Then, they can use k to obtain k and thus decrypt c_2^* , and so win the security game. It is to avoid this attack that we modify the KEM-DEM framework in the next section.

Generic Multi-Party Computation: Our methodology uses a *generic* actively-secure-with-abort MPC functionality defined via Linear Secret Sharing (LSS) over a finite field \mathbb{F}_q . Note, we could utilize in the case when q is not a prime in our main Gladius construction a ring \mathbb{Z}_q , but for the purposes of this paper we restrict to q being a prime in our used MPC methodology; see the full version. for a discussion of the issues when q is a power of two. This means that inputs of the parties remain private throughout the execution of the protocol, and when a set of adversaries deviate from the protocol, honest parties will catch this with overwhelming probability and then abort the protocol. This should be compared to passively secure protocols which offer a much weaker guarantee that security is only preserved if all parties follow the precise protocol steps correctly. We present in Figure 3 the base MPC functionality. Despite using a generic underlying protocol, our protocol ends up being surprisingly efficient. This is because we carefully designed Gladius to be both efficient in a distributed and a non-distributed manner.

To ease notation we denote a variable $x \in \mathbb{F}_q$ stored within the MPC functionality via $\langle x \rangle$, and write addition and multiplication of shares as $\langle x \rangle + \langle y \rangle$ and $\langle x \rangle \cdot \langle y \rangle$. We extend the notation to vectors and matrices in the obvious way via

Operations for Secure Computation, \mathcal{F}_{MPC} .

The functionality runs with $\mathcal{P} = \{\mathcal{P}^1, \dots, \mathcal{P}^n\}$ and an ideal adversary \mathcal{A} , that statically corrupts a set A of parties. Given a set I of valid identifiers, all values are stored in the form (varid, x) , where $\text{varid} \in I$.

Initialize: On input (init, p) from all parties, the functionality stores (domain, p) ,

Input: On input $(\text{input}, \mathcal{P}^i, \text{varid}, x)$ from \mathcal{P}^i and $(\text{input}, \mathcal{P}^i, \text{varid}, ?)$ from all other parties, with varid a fresh identifier, the functionality stores (varid, x) .

Add: On command $(\text{add}, \text{varid}_1, \text{varid}_2, \text{varid}_3)$ from all parties (if $\text{varid}_1, \text{varid}_2$ are present in memory and varid_3 is not), the functionality retrieves (varid_1, x) , (varid_2, y) and stores $(\text{varid}_3, x + y)$.

Multiply: On input $(\text{multiply}, \text{varid}_1, \text{varid}_2, \text{varid}_3)$ from all parties (if $\text{varid}_1, \text{varid}_2$ are present in memory and varid_3 is not), the functionality retrieves (varid_1, x) , (varid_2, y) and stores $(\text{varid}_3, x \cdot y)$.

Output: On input $(\text{output}, \text{varid}, i)$ from all honest parties (if varid is present in memory), the functionality retrieves (varid, y) and outputs it to the environment. The functionality waits for an input from the environment. If this input is **Deliver** then y is output to all players if $i = 0$, or y is output to player i if $i \neq 0$. If the adversarial input is not equal to **Deliver** then all players abort.

Figure 3. Operations for Secure Computation, \mathcal{F}_{MPC} .

$\langle \mathbf{x} \rangle$ and $\langle A \rangle$. If $\langle \mathbf{x} \rangle$ is a shared vector we let $\langle x_i \rangle$ denote the shared entries, and if $\langle A \rangle$ is a shared matrix we let $\langle A^{(i,j)} \rangle$ denote the shared entries; with a similar notation for vectors and matrices of non-shared values.

The cost model for LSS-based MPC protocols is such that addition of such shared entities is ‘for free’, whereas multiplication consumes resources (typically communication). Many MPC protocols in this setting, such as [9,20,34,50], work in an offline/online manner. In this setting the multiplication not only consumes communication resources in the online phase, but also consumes some correlated randomness (so-called Beaver triples) from the offline phase. However, an advantage of these offline/online models is that one can prepare other forms of correlated randomness in the offline phase; such as shares of random bits $\langle b \rangle$ with an unknown $b \in \{0, 1\}$. In our algorithms below we will write this as $\langle b \rangle \leftarrow \text{Bits}()$. If we sample a shared random element in \mathbb{F}_q , we will denote this by $\langle x \rangle \leftarrow \mathbb{F}_q$. To open an element we will write $x \leftarrow \text{Output}(\langle x \rangle)$ when it is output to all players.

MPC Friendly Hash Functions. Rescue: Our LWR-based construction of a hybrid cipher with efficient distributed decryption will make use of an MPC-friendly hash function, such as those in [5,29]. These hash function constructions are sponge-based, and there are two types; those suitable for MPC over characteristic two fields (StarkAD and Vision) and those suitable for MPC over large prime fields (Poseidon and Rescue). In this paper, we concentrate on the Rescue design from [5], which seems more suited to our application.

Rescue has a state of $t = r + c$ finite field elements in \mathbb{F}_q , for a prime q . The initial state of the sponge is defined to be the vector of t zero elements. A message is first mapped into $n = d \cdot r$ elements in \mathbb{F}_q , m_0, m_1, \dots, m_{n-1} . The elements are absorbed into the sponge in d absorption phases, where r elements are absorbed in each phase. At each phase a permutation $f : \mathbb{F}_q^t \rightarrow \mathbb{F}_q^t$ is applied resulting in a state s_0, \dots, s_{t-1} . At the end the absorption the r values s_c, \dots, s_{t-1} are output from the state. This process can then be repeated, with more data absorbed and then squeezed out. Thus we are defining a map $H : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^r$.

Each primitive call f in the Rescue sponge is performed by executing a round function rnds times. The round function is parametrized by a (small prime) value α , an MDS matrix $M \in \mathbb{F}_q^{t \times t}$ and two step constants $\mathbf{k}_i, \mathbf{k}'_i \in \mathbb{F}_q^t$. The value α is chosen to be the smallest prime such that $\gcd(q - 1, \alpha) = 1$. The round function applies exponentiation by $1/\alpha$, followed by application of the MDS matrix, followed by addition of the round constant \mathbf{k}_i , followed by exponentiation by α , followed by a further application of the MDS matrix, followed by addition of the round constant \mathbf{k}'_i . See [14] for a discussion of implementing Rescue in an MPC system, albeit for a large prime characteristic q of more than 256-bits. In our application q will be in the region of 21-bits.

3 Generic Hybrid Constructions

We let $\Pi_p = (\mathcal{K}_p, \mathcal{E}_p, \mathcal{D}_p)$ denote a OW-CPA secure, rigid, deterministic (resp. a OW-PCA secure, rigid and randomized) public key encryption scheme with message space \mathcal{M}_p and ciphertext space \mathcal{C}_p which is OW-CPA secure. We let $\Pi_s = (\mathcal{K}_s, \mathcal{E}_s, \mathcal{D}_s)$ denote a (one-time) IND-CPA symmetric key encryption scheme with message space $\mathcal{M}_s = \{0, 1\}^*$ and ciphertext space $\mathcal{C}_s \subset \{0, 1\}^*$. Again by abuse of notation we let \mathcal{K}_s also denote the key space of Π_s .

3.1 Hybrid₁ Construction

For this construction we define two hash functions

$$H : \mathcal{M}_p \rightarrow \mathcal{K}_s,$$

$$G : \begin{cases} \{0, 1\}^* \times \mathcal{M}_p \rightarrow \{0, 1\}^{|G|} & \text{If } \Pi_p \text{ is deterministic} \\ \mathcal{C}_p \times \{0, 1\}^* \times \mathcal{M}_p \rightarrow \{0, 1\}^{|G|} & \text{If } \Pi_p \text{ is randomized} \end{cases}$$

Note, G is defined to take elements in \mathcal{M}_p as the last entry for efficiency reasons (see below). We can then define our first hybrid encryption scheme $\Pi_h = (\mathcal{K}_h, \mathcal{E}_h, \mathcal{D}_h)$ in Figure 4. Notice how the decryption function ‘leaks’ the key k which is encrypted by the deterministic function even when the decryption function \mathcal{D}_s fails. This will allow us, in our threshold decryption operation, to also leak this key before the algorithm \mathcal{D}_s is called, enabling \mathcal{D}_s to be applied in the clear. The only question though is whether leaking this key is secure. The attack described from the last section does not apply, as the invalid ciphertext is already rejected by the testing for the correct value of G , which does not leak k if the test fails. In what follows we call this check the G -check.

$\mathcal{K}_h(1^t)$: $(pk, sk) \leftarrow \mathcal{K}_p(1^t)$ Return (pk, sk)	$\mathcal{E}_h(pk, m)$: $k \leftarrow \mathcal{M}_p$ $k \leftarrow H(k)$ $r \leftarrow \mathcal{R}_s$ $c_1 \leftarrow \mathcal{E}_p(pk, k)$ $c_2 \leftarrow \mathcal{E}_s(k, m; r)$ $c_3 \leftarrow G(c_2, k)$ (resp. $c_3 \leftarrow G(c_1, c_2, k)$) Return (c_1, c_2, c_3)	$\mathcal{D}_h(sk, (c_1, c_2, c_3))$: $k \leftarrow \mathcal{D}_p(sk, c_1)$ If $k = \perp$ then return (\perp, \perp) . $t \leftarrow G(c_2, k)$ (resp. $t \leftarrow G(c_1, c_2, k)$) If $t \neq c_3$ then return (\perp, \perp) . $k \leftarrow H(k)$. $m \leftarrow \mathcal{D}_s(k, c_2)$ Return (k, m) .
---	---	--

Fig. 4. Hybrid₁ Construction

As remarked in the introduction the variant of the hybrid construction which utilizes a deterministic Π_p can be seen as a special form of the second Fujisaki-Okamoto hybrid construction; assuming the space \mathcal{M}_p is exponentially large to ensure the resulting ‘randomized’ public key scheme is well-spread. Thus, the above hybrid construction is secure not only in the ROM, but also in the QROM, when we do not leak the secret key k during the decryption process and when Π_p is perfectly correct.

In the standard random oracle model, the following theorem (proved in the full version) shows that first hybrid construction is secure in a model in which the key k does leak during decryption as above, and where we combine it with a generic one-time IND-CPA DEM.

Theorem 3.1. *If H and G are modelled as random oracles then if \mathcal{A} is an IND-CCA adversary against Π_h then there is an OW-CPA adversary (resp. OW-PCA) adversary \mathcal{B} against the deterministic (resp. randomized) rigid public key scheme Π_p , which is δ_c -Collision Free, a (one-time) IND-CPA adversary \mathcal{C} against Π_s , and a \perp -Aware adversary \mathcal{D} against Π_p such that*

$$\begin{aligned} \text{Adv}_{\Pi_h, \mathcal{A}}^{\text{ind-cca}}(t) &\leq \text{Adv}_{\Pi_p, \mathcal{B}}^{\text{ow-cpa}}(t) + \text{Adv}_{\Pi_s, \mathcal{C}}^{\text{ind-cpa}}(t) + q_d \cdot \text{Adv}_{\Pi_p, \mathcal{D}}^{\perp\text{-Aware}}(t) \\ &\quad + \frac{1}{|\mathcal{M}_p|} + \frac{2 \cdot q_d + q_G^2}{2^{|G|}} + \delta_c \end{aligned}$$

where q_d (resp. q_G) is an upper bound on the number of decryption oracle (resp. G -oracle) queries and the decryption oracle queries made to the hybrid scheme leak the key k as above.

3.2 Hybrid₂ Construction

Our second hybrid construction focuses solely on the case of Π_p being a rigid deterministic OW-CPA public key encryption scheme, we show that the generic hybrid transform, given in Figure 5, which uses the four hash functions,

$$H : \mathcal{M}_p \longrightarrow \mathcal{K}_s,$$

$$\begin{aligned}
H', H'' &: \mathcal{M}_p \longrightarrow \mathcal{M}_p \\
G &: \{0, 1\}^* \times \mathcal{M}_p \longrightarrow \{0, 1\}^{|G|}
\end{aligned}$$

is secure in the QROM. Namely we have the following theorem (proved in the full version)

Theorem 3.2. *If G , H , H' and H'' are modelled as quantum random oracles then if \mathcal{A} is an IND-CCA adversary against Π_h then there is a (one-time) IND-CPA adversary \mathcal{B} against Π_s , a \perp -Aware adversary \mathcal{C} against the deterministic rigid public key scheme Π_p – which is δ_c -Collision Free and δ being the probability of its decryption failure for a uniformly random message – and OW-CPA adversaries \mathcal{D} and \mathcal{E} against Π_p such that*

$$\begin{aligned}
\text{Adv}_{\Pi_h, \mathcal{A}}^{\text{ind-cca}}(t) &\leq \text{Adv}_{\Pi_s, \mathcal{B}}^{\text{ind-cpa}}(t) + \delta \\
&+ 4q_1 \sqrt{\frac{q_3}{\sqrt{|\mathcal{M}_p|}} + \frac{q_d}{2^{|G|}} + \delta' + \text{Adv}_{\Pi_p, \mathcal{D}}^{\text{ow-cpa}}(t) + 2q_2 \sqrt{\delta' + \text{Adv}_{\Pi_p, \mathcal{E}}^{\text{ow-cpa}}(t)}}
\end{aligned}$$

for $q_1 = q_H + q_{H'} + 2q_d$, $q_2 = q_{H''} + q_d$, $q_3 = 2(q_G + q_d + 1)$ and

$$\delta' = \delta_c + q_d \cdot \text{Adv}_{\Pi_p, \mathcal{C}}^{\perp\text{-Aware}}(t) + \frac{1}{|\mathcal{M}_p|},$$

where q_d , q_G , q_H , $q_{H'}$ and $q_{H''}$ are respective upper bounds on the number of decryption oracle, G -oracle, H -oracle, H' -oracle and H'' -oracle queries and the decryption oracle queries made to the hybrid scheme leak the key k as above.

$\mathcal{K}_h(1^t):$ $(\text{pk}, \text{sk}) \leftarrow \mathcal{K}_p(1^t)$ Return (pk, sk)	$\mathcal{E}_h(\text{pk}, m):$ $k \leftarrow \mathcal{M}_p$ $k \leftarrow H(k)$ $\mu \leftarrow H'(k)$ $r \leftarrow \mathcal{R}_s$ $c_1 \leftarrow \mathcal{E}_p(\text{pk}, k)$ $c_2 \leftarrow \mathcal{E}_s(k, m; r)$ $c_3 \leftarrow G(c_2, \mu)$ $c_4 \leftarrow H''(k)$ Return (c_1, c_2, c_3, c_4)	$\mathcal{D}_h(\text{sk}, (c_1, c_2, c_3, c_4)):$ $k \leftarrow \mathcal{D}_p(\text{sk}, c_1)$ If $k = \perp$ then return (\perp, \perp) . $t \leftarrow H''(k)$ If $t \neq c_4$ then return (\perp, \perp) . $\mu \leftarrow H'(k)$ $t' \leftarrow G(c_2, \mu)$ If $t' \neq c_3$ then return (\perp, \perp) . $k \leftarrow H(k)$. $m \leftarrow \mathcal{D}_s(k, c_2)$ Return (k, m) .
--	--	---

Fig. 5. Hybrid₂ Construction

3.3 Threshold Variant

Assuming there are protocols $\Pi_{\mathcal{K}_p}$ and $\Pi_{\mathcal{D}_p}$ which implement the base public key encryption scheme in a threshold manner a threshold variant of our above

constructions are therefore immediate. We simply apply the threshold decryption operation to c_1^* , keeping the result in a shared form. The parties then securely evaluate G (or G, H' and H'' in our second hybrid construction). Our distributed decryption operation for our Hybrid_1 construction Π_h would then consist of the following steps, with a similar methodology for Hybrid_2 (which would also require a secure evaluation of H' and H'')

1. Absorb c_2 (resp. c_1 and c_2) into G in the clear.
2. Apply $\Pi_{\mathcal{D}_p}$ to obtain a distributed decryption operation, keeping the result k in shared form.
3. Securely absorb these shares of k into the sponge G .
4. Securely evaluate the squeezing of G to obtain t in the clear.
5. Reject the ciphertext if $c_3 \neq t$.
6. Open k to all players.
7. Compute $k = H(k)$ in the clear
8. Compute $m = \mathcal{D}_s(k, c_2)$ in the clear and output it.

We notice that if we use a sponge-like function for G , such as `Rescue` [5] (see the full version) or `SHA-3`, then in the clear we can insert the first arguments for G (c_1 and c_2) during a distributed decryption, as they are public. Thus we only need to execute a secure distributed version of G for the final absorption of k , and then the squeezing phase to obtain c_3 .

4 The Large Vector Problem (LVP)

We also need to give a new hardness assumption, which we call LVP. This is needed in order to establish the \perp -Aware property of our encryption scheme; namely that it is hard for an adversary \mathcal{A} , given the public key pk , to come up with a plaintext/ciphertext pair (m, c) such that $c = \mathcal{E}(\text{pk}, m)$ but $\mathcal{D}(\text{sk}, c) = \perp$; i.e. a ciphertext which is a valid encryption, but which does not decrypt correctly.

Consider the following experiment. The challenger constructs a matrix $A_1 \in \mathbb{Z}_q^{n \times n}$ uniformly at random, and then selects $R_1, R_2 \in \mathbb{Z}_q^{n \times n}$ with entries selected from the distribution \mathcal{D}_σ . The challenger constructs $A_2 = A_1 \cdot R_1 + R_2$ and gives the pair (A_1, A_2) to the adversary \mathcal{A} . The adversary's goal is to come up with a vector $\mathbf{m} \in [-1/2, \dots, 1/2]^n$ such that

$$\|R_1 \cdot \mathbf{m}\|_\infty \geq \mathbf{c} \cdot \sigma \cdot \sqrt{n}/2$$

for some constant \mathbf{c} . We define the advantage of an adversary \mathcal{A} against this hard problem as

$$\text{Adv}_{\mathcal{A}}^{\text{LVP}}(n, \mathbf{c}, \sigma) = \Pr \left[A_1 \leftarrow \mathbb{Z}_q^{n \times n}, R_1, R_2 \leftarrow \mathcal{D}_\sigma^{n \times n}, A_2 = A_1 \cdot R_1 + R_2, \right. \\ \left. \mathbf{m} \leftarrow \mathcal{A}(A_1, A_2) : \|R_1 \cdot \mathbf{m}\|_\infty \geq \mathbf{c} \cdot \sigma \cdot \sqrt{n}/2 \right].$$

We note that (see the full version for details) the probability that there are *no solutions at all* to the above problem (when we sample over all secret keys

R_1 and R_2) is $1 - n \cdot \text{erfc}(\mathbf{c})$. Thus the probability that there are **ANY** solutions to this problem is already very small if \mathbf{c} is large enough. Thus for randomly chosen R_1 and \mathbf{c} large enough, the adversary already has an impossible task (i.e. information theoretically impossible) in solving LVP. If we set $\mathbf{c} = 9.3$ (resp. 13.2) this would give us a bound on the advantage of (approximately) 2^{-128} (resp. 2^{-256}).

We note that if one can solve the search-LWE problem for the pair (A_1, A_2) then finding such a \mathbf{m} is potentially trivial (if such a \mathbf{m} exists). In the ‘unlucky’ event that there is a solution, since R_1 is hidden (due to search-LWE being hard), the adversary is left with outputting a small vector and ‘hoping’ it works.

We would like to use a smaller constant than $\mathbf{c} = 9.3$ (resp. 13.2). Assuming a solution exists and sampling over all keys, the only plausible attack (due to R_1 being hidden by LWE) is for the adversary to select a message at random and hope it solves the problem. Suppose the adversary selects a message with entries in the range $[-v/2 + u, \dots, u + v/2]$ for $u \in [0, 1/2)$ and $v < 1 - 2 \cdot u$. The n random variables given by the entries of $R_1 \cdot \mathbf{m}$ will still have mean zero (as the entries of R_1 are pulled from a symmetric distribution of mean zero), but they will have variance given by $V = n \cdot \sigma^2 \cdot (u^2 + v^2/12)$. Thus with probability $\text{erfc}(\mathbf{c}')$ the adversary will obtain a value of size greater than $\mathbf{c}'\sqrt{V}$. To win the game (assuming a solution exists) thus requires

$$\mathbf{c}' \geq \frac{\mathbf{c}}{2 \cdot \sqrt{u^2 + v^2/12}}.$$

The right hand side of this last equation is minimized when $u = 1/2, v = 0$ and thus we have $\mathbf{c}' > \mathbf{c}$. But this assumes a solution exists, thus our final probability for the attack to work is given by $n \cdot \text{erfc}(\mathbf{c})^2$. If this was the best possible attack then this would mean we would have $\text{Adv}_{\mathcal{A}}^{\text{LVP}}(n, \mathbf{c}, \sigma) \leq n \cdot \text{erfc}(\mathbf{c})^2$. Indeed, we conjecture that the hardness of this problem is indeed given by $\text{Adv}_{\mathcal{A}}^{\text{LVP}}(n, \mathbf{c}, \sigma) \leq n \cdot \text{erfc}(\mathbf{c})^2$, and assuming this allows us to obtain smaller parameters for our Gladius scheme.

Conjecture 4.1 (LVP Hardness Conjecture). We have $\text{Adv}_{\mathcal{A}}^{\text{LVP}}(n, \mathbf{c}, \sigma) \leq n \cdot \text{erfc}(\mathbf{c})^2$.

5 Gladius–Hispaniensis: Plain LWR Based Encryption

According to Wikipedia the *Gladius–Hispaniensis* was the earliest and heaviest of the different types of Gladii that we know about; it is thus fitting we reserve this name for our encryption scheme based on standard LWR. The scheme is defined in Figure 6 and is parametrized by values $t, p, q, n, \ell, \sigma, \epsilon$. We define the message space \mathcal{M} to be the set \mathbb{Z}_t^n . From these parameters we define $\mu \in \mathbb{Z}$ and $\psi \in (-1/2, 1/2]$ via

$$\frac{p \cdot \ell}{q} = \left\lfloor \frac{p \cdot \ell}{q} \right\rfloor + \psi = \mu + \psi. \tag{1}$$

The Gladius–Hispaniensis Deterministic Encryption Scheme Π_p .

Key Generation: \mathcal{K}_p .

1. $R_1, R_2 \leftarrow \mathcal{D}_\sigma^{n \times n}$, i.e. two $n \times n$ matrices with coefficients sampled from \mathcal{D}_σ .
2. $A_1 \leftarrow \mathbb{Z}_q^{n \times n}$.
3. $A_2 \leftarrow A_1 \cdot R_1 + R_2 + G$, where G is the gadget matrix $\ell \cdot I_n$.
4. $\mathbf{pk} \leftarrow (A_1, A_2)$.
5. $\mathbf{sk} \leftarrow (\mathbf{pk}, R_1)$.
6. Return $(\mathbf{pk}, \mathbf{sk})$.

Encryption: $\mathcal{E}_p(\mathbf{pk}, \mathbf{m})$.

1. $\mathbf{c}_1 \leftarrow \lfloor \mathbf{m}^\top \cdot A_1 \rfloor_p$.
2. $\mathbf{c}_2 \leftarrow \lfloor \mathbf{m}^\top \cdot A_2 \rfloor_p$.
3. Return $(\mathbf{c}_1, \mathbf{c}_2)$.

Decryption: $\mathcal{D}_p(\mathbf{sk}, (\mathbf{c}_1, \mathbf{c}_2))$.

1. $\mathbf{w}^\top \leftarrow \mathbf{c}_2 - \mathbf{c}_1 \cdot R_1 \pmod{q}$
2. $\mathbf{e}^\top \leftarrow \mathbf{w}^\top \pmod{p}$.
3. $\mathbf{v}^\top \leftarrow \mathbf{e}^\top \pmod{\mu}$.
4. $\mathbf{m}^\top \leftarrow (\mathbf{e}^\top - \mathbf{v}^\top) / \mu$.
5. $(\mathbf{c}'_1, \mathbf{c}'_2) \leftarrow \mathcal{E}_p(\mathbf{pk}, \mathbf{m})$.
6. If $\mathbf{c}_1 \neq \mathbf{c}'_1$ or $\mathbf{c}_2 \neq \mathbf{c}'_2$ return \perp .
7. Return \mathbf{m}^\top .

Figure 6. The Gladius–Hispaniensis Deterministic Encryption Scheme Π_p .

Note when μ and p are powers of two, say $\mu = 2^\nu$ and $p = 2^\pi$, and $t = 2$ then lines 3 and 4 of the decryption procedure in Figure 6 becomes $m_i \leftarrow w_i^{(\nu)} \oplus w_i^{(\nu+1)}$, where $\mathbf{m} = (m_i)$ and $\mathbf{w} = (w_i)$ and $w_i^{(j)}$ is the j -th bit of w_i . This is again a useful simplification in our distributed decryption procedure, thus we will assume that μ and p are powers of two.

See the full version, where we discuss the criteria which need to be satisfied to ensure correctness of decryption, and security of this construction. We found the parameters in Table 1 using this analysis. Note we are only able to establish our \perp -Aware property (assuming the LVP-problem is hard) when q is a prime, an interesting open question would be to establish this for the parameter sets where q is a power-of-two.

The above describes solely the KEM-like component Π_p of our hybrid construction from Section 3. The DEM-like component Π_s can be any (one-time) IND-CPA cipher; for example a one-time pad or AES in CTR-mode. The remaining item to define is the associated hash function G (and in the case of using Hybrid₂ the hash functions H' and H''). Recall G takes the ciphertext c_2 output from the DEM, and the key k which the KEM encapsulates, and produces the hash result $G(c_2, k)$. Here we focus solely on the case of prime q variants of Gladius.

In our construction, to aid distributed decryption, we construct G as in Figure 7, assuming we take the message modulus $t = 2$ in our above construction. Minor tweaks are needed in the case when $t \neq 2$. The construction makes use of

	n	t	q	p	ℓ	σ	μ	LWE	LWR Security		\perp -Aware Security	
								Security	Theoretical	Best-Attack	ϵ'	Adv^{-1}
prime q	971	2	$2^{21} - 9$	2^9	2^{19}	$\sqrt{1/2}$	128	$2^{128.3}$	$2^{61.25}$	$2^{465.7}$	5.673	2^{89}
	1024	2	$2^{21} - 9$	2^9	2^{19}	$\sqrt{1/2}$	128	$2^{135.7}$	$2^{64.78}$	$2^{492.7}$	5.523	2^{84}
	1982	2	$2^{23} - 15$	2^{10}	2^{21}	$\sqrt{1/2}$	256	$2^{256.6}$	$2^{125.3}$	$2^{465.7}$	8.036	2^{183}
	2048	2	$2^{23} - 15$	2^{10}	2^{21}	$\sqrt{1/2}$	256	$2^{266.0}$	$2^{129.9}$	$2^{975.5}$	7.906	2^{176}
	4096	2	$2^{26} - 5$	2^{11}	2^{21}	$\sqrt{1/2}$	512	$2^{519.0}$	$2^{256.2}$	2^{2034}	11.247	2^{361}
prime q	4096	2	$2^{25} - 39$	2^{11}	2^{23}	$\sqrt{1/2}$	512	$2^{537.0}$	$2^{263.8}$	2^{1951}	N/A	2^∞
	8192	2	$2^{27} - 39$	2^{12}	2^{25}	$\sqrt{1/2}$	1024	$2^{1098.0}$	$2^{542.4}$	2^{3918}	N/A	2^∞
$q = 2^k$	710	2	2^{14}	2^{10}	2^{11}	$\sqrt{1/2}$	128	$2^{128.9}$	$2^{550.1}$	$2^{187.6}$	-	-
	1024	2	2^{14}	2^{10}	2^{12}	$\sqrt{1/2}$	256	$2^{188.4}$	$2^{792.1}$	$2^{274.8}$	-	-
	1437	2	2^{15}	2^{11}	2^{12}	$\sqrt{1/2}$	256	$2^{256.6}$	$2^{1115.}$	$2^{366.1}$	-	-
	2048	2	2^{15}	2^{11}	2^{12}	$\sqrt{1/2}$	256	$2^{376.6}$	$2^{1584.}$	$2^{535.3}$	-	-

Table 1. Gladius–Hispaniensus parameters (based on plain LWR), and the associated LWE, LWR and \perp -Aware security. For the first five parameter sets with q prime we establish \perp -Aware assuming Conjecture 4.1, for the second two \perp -Aware security is established unconditionally since B_V is always less than $\mu/2$. For the $q = 2^k$ parameters we cannot establish \perp -Aware security

Rescue with rate r satisfying $r \geq 2 \cdot \kappa / \lfloor \log_2 q \rfloor$, as well as SHA-3. The combined hash function can clearly be treated as a random oracle if one assumes SHA-3 and Rescue are themselves random oracles. In the final distributed decryption variant only lines 5 and 6 need to be performed in a secure manner (which are based on Rescue, which is an MPC-friendly hash function). Thus irrespective of how long the initial message is which is being encrypted, the number of applications of Rescue which need to be performed securely is given by $\lceil w/r \rceil + 1$. If we take parameters $\kappa = 128$, $n = 1024$ and $q = 2^{21} - 9$ then we have $r = 13$, $w = 52$ and the number of secure rounds of Rescue is five in order to absorb the key k and produce the output $G(c_2, k)$. For the case of Hybrid₂ we select H' and H'' based on Rescue as well.

For q a power-of-two a different methodology will be required. We know of no MPC-friendly hash function defined over rings of the form \mathbb{Z}_{2^k} . Thus for the case of power-of-two values of q it would seem one would need to use a standard sponge-based hash function (such as SHA-3), which would not be as amenable to threshold implementation via a generic MPC methodology.

6 Distributed Decryption of Gladius

In this section we present how to perform distributed decryption of the hybrid cipher obtained from our generic construction composed with Gladius. For ease of implementation we select parameters for which q is prime, $p = 2^\pi$ and $\mu = 2^\nu$ are powers of two, and the message space modulus is $t = 2$. Although this section focuses on the simpler standard LWR variant (Gladius–Hispaniensus) and not on the Ring-LWR variants (Gladius–Pompeii, Gladius–Mainz and Gladius–Fulham, see the full version), the procedure is virtually identical in all cases.

We use an MPC system defined for the q prime case for our experiments, as this is the only case for which we have both a full proof of security and a suitable MPC-friendly hash function (Rescue). Selecting q prime also means we

The Hash Function $G(c_2, k)$.

On input of c_1, c_2 and $k \in \{0, 1\}^n$.

1. Apply the SHA-3 hash function to c_2 to obtain a $2 \cdot \kappa$ -bit string \mathbf{s} .
2. Parse \mathbf{s} into r bit-strings (s_1, \dots, s_r) each of length $\lfloor \log_2 q \rfloor$. This is possible due to the choice of r .
3. Treat each s_i as an element of \mathbb{F}_q and absorb the set (s_1, \dots, s_r) into a fresh **Rescue** state. This requires one application of the **Rescue** absorption phase. Note, this is done in the clear during threshold decryption as c_1 and c_2 are public.
4. Take the bit string $k \in \{0, 1\}^n$ and parse again into bit-strings of length $\lfloor \log_2 q \rfloor$. This will produce $w = \lceil n / \lfloor \log_2 q \rfloor \rceil$ bit-strings k_1, \dots, k_w , each of which we think of as elements in \mathbb{F}_q .
5. The w finite field elements k_1, \dots, k_w are absorbed into the **Rescue** state, this will require $\lceil w/r \rceil$ executions of the **Rescue** function. Since during distributed decryption k is not known at this stage, this needs to be carried out securely.
6. Finally the output is obtained by squeezing out r output field elements from **Rescue** using a single application of the **Rescue** function.

Figure 7. The Hash Function $G(c_2, k)$.

can utilize an existing library such as SCALE-MAMBA [6], for not only the underlying MPC system, but also many of the necessary sub-routines which our distributed decryption method requires. In the full version we discuss changes to the algorithms which would be needed if future work could establish a secure variant in the case when q is a power of two (including a suitable MPC-friendly hash function for this case).

We first present our distributed Key Generation protocol Π_{KeyGen} . Since the key generation method is based on Learning-with-Errors, with the error distribution coming from the NewHope distribution with $\sigma = 1/\sqrt{2}$, we can utilize the simple method described in [35,46]. This is described in Figure 8.

Protocol for Distributed Key Generation Π_{KeyGen} .

1. For $i, j \in [1, \dots, n]$
 - $\langle b \rangle, \langle b' \rangle, \langle c \rangle, \langle c' \rangle \leftarrow \text{Bits}()$.
 - $\langle R_1^{(i,j)} \rangle \leftarrow \langle b \rangle - \langle b' \rangle$.
 - $\langle R_2^{(i,j)} \rangle \leftarrow \langle c \rangle - \langle c' \rangle$.
 - $A_1^{(i,j)} \leftarrow \mathbb{F}_q$.
2. $\langle A_2 \rangle \leftarrow A_1 \cdot \langle R_1 \rangle + \langle R_2 \rangle + G$.
3. $A_2 \leftarrow \text{Output}(\langle A_2 \rangle)$.
4. $\text{pk} \leftarrow (A_1, A_2)$.
5. $\text{sk} \leftarrow (A_1, A_2, \langle R_1 \rangle)$.

Figure 8. Protocol for Distributed Key Generation Π_{KeyGen} .

The distributed decryption procedure itself is more complex. It makes use of the following protocols from other works, e.g. [19,42]. In each of these protocols we *can* run the protocol with clear entries. For example $\text{BitDecomp}(a)$ will form the bit decomposition of an integer a , but here we also need to specify how many bits we require. Since a may not necessarily be reduced in the range $(-q/2, \dots, q/2)$. Thus we would write $\text{BitDecomp}(a, t)$ to obtain t bits.

- $\langle \mathbf{a} \rangle \leftarrow \text{BitDecomp}(\langle a \rangle)$: Given a secret shared value $\langle a \rangle$ with $a \in \mathbb{F}_q$ this procedure produces a vector of shared bits $\langle \mathbf{a} \rangle = (\langle a_0 \rangle, \dots, \langle a_{\lceil \log_2 q \rceil} \rangle)$ such that $a = \sum_i a_i \cdot 2^i$. Note this means a is in the non-centred interval $[0, \dots, q)$. The method we use is from [42], which is itself built upon the work in [19].
- $\langle \mathbf{c} \rangle \leftarrow \text{BitAdd}(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle)$: Given shared bits $\langle \mathbf{a} \rangle$ and $\langle \mathbf{b} \rangle$ this executes a binary adder to produce the vector of shared bits $\langle \mathbf{c} \rangle$ such that $\sum_i c_i \cdot 2^i = \sum_i (a_i + b_i) \cdot 2^i$. This algorithm is also presented in [19]. Note this returns one bit more than the maximum of the lengths of $\langle \mathbf{a} \rangle$ and $\langle \mathbf{b} \rangle$.
- $\langle \mathbf{c} \rangle \leftarrow \text{BitNeg}(\langle \mathbf{a} \rangle)$: This performs the two-complement negative of the bit vector $\langle \mathbf{a} \rangle$. It flips the bits of $\langle \mathbf{a} \rangle$ to produce $\langle \bar{\mathbf{a}} \rangle$, and then executes the function $\text{BitAdd}(\langle \bar{\mathbf{a}} \rangle, \mathbf{1})$, where $\mathbf{1} = \text{BitDecomp}(1, \lceil \bar{\mathbf{a}} \rceil)$ is the bit-vector of the correct length representing the integer one.
- $\langle c \rangle \leftarrow \text{BitLT}(\langle \mathbf{a} \rangle, \langle \mathbf{b} \rangle)$: This computes the single bit output $\langle c \rangle$ of the comparison $\sum_i a_i \cdot 2^i < \sum_i b_i \cdot 2^i$. Again we use the method from [19].

When running $\text{BitDecomp}(\langle a \rangle)$ on a secret shared value the run time is not deterministic, it needs to loop to produce a shared value which is uniformly distributed modulo q . It does this by rejection sampling; where the probability of rejecting a sample is given by

$$\frac{2^{\lceil \log_2 q \rceil} - q}{2^{\lceil \log_2 q \rceil}}.$$

This is another reason to select q to be close to a power-of-two, as well as to ensure μ is a power of two.

In Figure 10 we divide our distributed decryption procedure into four phases: KEM Decapsulate, KEM Validity Check, the Hash-Check (for the checking of the DEM component) and finally the Message Extraction. As we select μ and p to be powers of two the first stage is relatively straightforward given we can implement $\text{BitDecomp}(\langle a \rangle)$. There is a minor complication due to the need to map the bit-decomposition into the centred interval but this is easily dealt with using the sub-routine in Figure 9. The third stage complexity depends on the choice of the underlying hash function G ; our choice of G from Section 5 using SHA-3 and Rescue combined was to ensure this step is as efficient as possible. Due to our hybrid design the final step can be performed in the clear; which is not possible for other hybrid schemes.

Thus, the main complexity of the decryption procedure is the second stage, namely the KEM Validity Check, as for this we need to re-encrypt the message

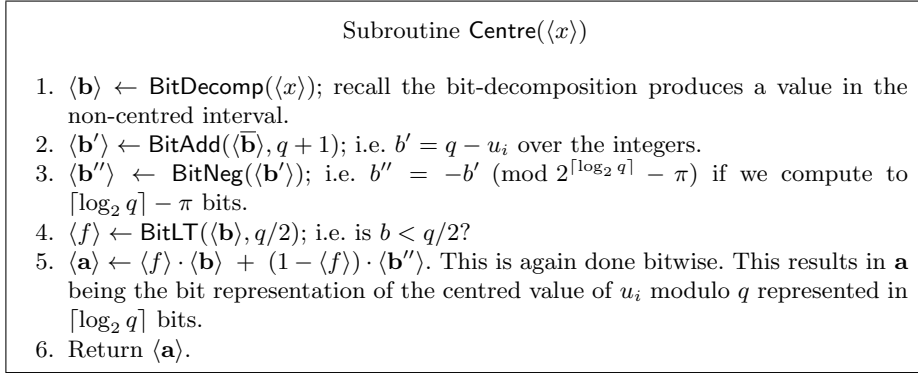


Figure 9. Subroutine Centre($\langle x \rangle$)

and check the result is equal to the KEM ciphertext component. We need to verify equations of the following form

$$c = \lfloor \langle x \rangle \rfloor_p = \left\lfloor \frac{p}{q} \cdot \langle x \rangle \right\rfloor \pmod{p}$$

where c is publicly given, but the value $\langle x \rangle$ cannot be opened to the parties. We write the equation, over the integers, as $c = \frac{p}{q} \cdot \langle x \rangle + \epsilon + p \cdot v$, where $\epsilon \in (-1/2, 1/2]$, $v \in \{0, 1\}$ and we think of the shared value $\langle x \rangle$ being in the centred representation modulo q . The value v is equal to one only if the reduction modulo p in the LWR equation needs to move the rounded value $-p/2$ to $p/2$. This happens when

$$x \leq \frac{q}{p} \left(\frac{1}{2} - \frac{p}{2} \right) = \frac{q \cdot (1 - p)}{2 \cdot p}.$$

This means we simply need to compute the bit representation $\langle \mathbf{s} \rangle$ of the value $|q \cdot c - p \cdot \langle x \rangle - p \cdot q \cdot \langle v \rangle|$ over the integers and then check the result is less than $q/2$. The last check can be performed using the $\text{BitLT}(\langle \mathbf{s} \rangle, q/2)$ algorithm mentioned above.

But to compute the bit representation of $\langle \mathbf{s} \rangle$ we need the bit representation of the modulo q centred value $\langle x \rangle$. However, the BitDecomp routine only produces the bit-decomposition in the non-centred interval of a value modulo q . We could use the method from the first stage and apply the Centre sub-routine. However, this is inefficient as on its own it requires two calls to BitAdd (one explicitly to BitAdd and one implicitly in the call to BitNeg). The procedure BitAdd is our most expensive subroutine so we want to minimize the number of calls to this.

Thus instead we proceed as follows: If we think of the value $\langle x \rangle$ as the reduction in the centred interval, and $\langle u \rangle$ as the value in the non-centred interval then we have $x = u - b \cdot q$, where b is the bit given by $b = 1 - (u \leq q/2)$. We write $\langle \mathbf{u} \rangle$ for the corresponding shared bit decomposition of u . We can then re-write the equation for determining v above in terms of u , as opposed to x , as $v = b \cdot \left(u \leq \frac{q \cdot (p+1)}{2 \cdot p} \right)$. We note that $v = 0$ when $b = 0$, which is important in what follows.

Protocol for Distributed Decryption Π_{Dec} .

Input: A ciphertext $c_1 = (\mathbf{c}_1, \mathbf{c}_2), c_2, c_3$, the public key (A_1, A_2) and the secret key in shared form $\langle R_1 \rangle$.

KEM Decapsulation:

1. $\langle \mathbf{x} \rangle \leftarrow \mathbf{c}_2 - \mathbf{c}_1 \cdot \langle R_1 \rangle$.
2. For $i \in [1, \dots, n]$
 - $\langle \mathbf{w} \rangle \leftarrow \text{Centre}(\langle x_i \rangle)$.
 - $\langle k_i \rangle \leftarrow \langle w_i^{(\nu)} \rangle \oplus \langle w_i^{(\nu+1)} \rangle = \langle w_i^{(\nu)} \rangle + \langle w_i^{(\nu+1)} \rangle - 2 \cdot \langle w_i^{(\nu)} \rangle \cdot \langle w_i^{(\nu+1)} \rangle$.

KEM Validity Check:

1. $\langle \mathbf{y} \rangle \leftarrow \langle \mathbf{k} \rangle \cdot (A_1 \| A_2)$.
2. $\langle z \rangle \leftarrow 1$.
3. For $i \in [1, \dots, 2 \cdot n]$
 - $\langle \mathbf{u} \rangle \leftarrow \text{BitDecomp}(\langle y_i \rangle)$.
 - $\langle b \rangle \leftarrow 1 - \text{BitLT}(\langle \mathbf{u} \rangle, q/2)$.
 - $\langle v \rangle \leftarrow \langle b \rangle \cdot \text{BitLT}(\langle \mathbf{u} \rangle, q \cdot (p+1)/(2 \cdot p))$. This computes the adjustment bit for dealing with the wrap around modulo p . Note, this can only apply when $a < 0$.
 - $\langle \mathbf{u}' \rangle \leftarrow \langle \mathbf{u} \rangle \ll \pi$; i.e. shift left by π bits, where $p = 2^\pi$. Hence $u' = p \cdot u$ over the integers, represented in $\lceil \log_2 q \rceil + \pi$ bits.
 - $\langle \mathbf{w} \rangle \leftarrow \text{BitAdd}(\langle \mathbf{u}' \rangle, 2^{\lceil \log_2 q \rceil + \pi} - c_i \cdot q)$. Here $c_i = \mathbf{c}_1^{(i)}$ if $i \leq n$ and $\mathbf{c}_2^{(i-n)}$ otherwise. This produces $w = p \cdot u - c_i \cdot q$ over the integers with $\lceil \log_2 q \rceil + \pi$ bits.
 - $\langle \mathbf{f} \rangle \leftarrow \text{BitAdd}(\langle \mathbf{w} \rangle, (\langle b \rangle - \langle v \rangle) \cdot (-p \cdot q))$. This applies the adjustment when $b = 1$ and $v = 0$. We now have $f = p \cdot u_i - (b - v) \cdot p \cdot q - c_i \cdot q$ over the integers with $\lceil \log_2 q \rceil + \pi$ bits.
 - $\langle \mathbf{f}' \rangle \leftarrow \text{BitNeg}(\langle \mathbf{f} \rangle)$, hence $f' = -f$ over the integers.
 - $\langle g \rangle \leftarrow \langle f_{\pi + \lceil \log_2 q \rceil - 1} \rangle$; i.e. the sign bit of f .
 - $\langle \mathbf{s} \rangle \leftarrow \langle g \rangle \cdot \langle \mathbf{f}' \rangle + (1 - \langle g \rangle) \cdot \langle \mathbf{f} \rangle$. Again a bitwise operation computing $s = |f|$ as an integer.
 - $\langle j \rangle \leftarrow \text{BitLT}(\langle \mathbf{s} \rangle, q/2)$; is one if this coefficient is OK.
 - $\langle z \rangle \leftarrow \langle z \rangle \cdot \langle j \rangle$; is one if the ciphertext is OK up to this point.
4. $z \leftarrow \text{Output}(\langle z \rangle)$
5. If $z \neq 1$ then return \perp .

Hash Check:

1. $\langle t \rangle \leftarrow G(c_2, \langle \mathbf{k} \rangle)$.
2. $t \leftarrow \text{Output}(\langle t \rangle)$.
3. If $t \neq c_3$ then return \perp .

Message Extaction:

1. $k \leftarrow \text{Output}(\langle \mathbf{k} \rangle)$.
2. $\mathbf{k} \leftarrow H(k)$.
3. $m \leftarrow \mathcal{D}_s(\mathbf{k}, c_2)$
4. If $m = \perp$ then return \perp .
5. Return m .

Figure 10. Protocol for Distributed Decryption Π_{Dec} .

We then rewrite the equation for $\langle \mathbf{s} \rangle$ as

$$\left| p \cdot \langle \mathbf{u} \rangle - p \cdot q \cdot (\langle b \rangle - \langle v \rangle) - c_i \cdot q \right|$$

The bit representation of $p \cdot \langle \mathbf{u} \rangle$ can be determined by simply shifting bits, as p is a power-of-two. The bit representation of $-p \cdot q \cdot (\langle b \rangle - \langle v \rangle)$ can be determined by bit-wise multiplications as $b-v \in \{0, 1\}$ by construction. From these observations we can produce a method for Stage 2 which requires three calls to `BitAdd`, as opposed to the naive method which would go through `Centre` which would require four calls to `BitAdd`.

Security Discussion and Implementation: As remarked previously the security of our implementation follows from the security of the underlying MPC protocol. By using SCALE-MAMBA [6] we can obtain active security, and the above sub-procedures are all provided as built in functions. In addition, the large local only operations in KEM Decapsulation (line 1) and KEM Validity Check (line 1) can be carried out efficiently in C++ using the SCALE `LOCAL_FUNCTION` operation. This enables one to perform complex local only operations, i.e. complex linear functions, natively in C++ as opposed to needing them to be implemented with the MPC system (which adds a lot of overhead).

We implemented our distributed decryption procedure in the case of Shamir sharing within SCALE-MAMBA. This is because the Shamir implementation module allows the MPC sub-system to be instantiated over any finite field \mathbb{F}_q . In using a full threshold access structure one would need (with SCALE-MAMBA as currently implemented) to select a prime q which is FHE friendly; so as to enable the SHE scheme at the basis of SPDZ [20] to be instantiated. None of the q values in the various parameter sets for Gladius are FHE friendly; not even the Gladius-Pompeii variants which have $q - 1$ divisible by a large power of two. In our experiments, each party ran on a machine with a Intel(R) Core(TM) i9-9900 CPU at 3.10GHz and 128 GB of memory. The machines were connected in a local network using a 10 gigabit switch.

For three parties, tolerating a threshold of one dishonest party, we obtained a run time for the first three phases of 1.19, 3.62, and 0.18 seconds respectively; for our parameter set of $q = 2^{21} - 9$ and $n = 1024$ in the plain LWR setting. Making a total decapsulation time of 4.99 seconds in 136491 rounds of communication. Whilst this might at first sight seem slower than the 4.20 seconds reported for LIMA in [35] the results are incomparable. Recall, the method in [35] to perform distributed decapsulation is insecure, as indeed would be any distributed decapsulation of any algorithm making use of the traditional KEM-DEM construction.

In our second experiment, we used the parameter set of $q = 2^{23} - 15$ and $n = 2048$, which has a better \perp `-Aware` security of 2^{176} . We obtained a run time for three phases of 7.16, 19.1 and 0.99 seconds, respectively; which amounts to a total of 27.3 seconds and 274157 rounds of communication.

Acknowledgments

We would like to thank Alexandra Boldyreva for clarifying some issues with the PRIV definition of security for deterministic encryption, Frederik Vercauteren for clarifying some issues in relation to Learning-with-Rounding, Andrej Bogdanov for clarifying issues related to the theoretical reductions between LWE and LWR, and Ward Beullens on comments on an earlier draft.

This work was supported in part by CyberSecurity Research Flanders with reference number VR20192203, by ERC Advanced Grant ERC-2015-AdG-IMPACT, by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. FA8750-19-C-0502, and by the FWO under an Odysseus project GOH9718N. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the ERC, DARPA, the US Government or the FWO. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

1. Abe, M., Gennaro, R., Kurosawa, K.: Tag-KEM/DEM: A new framework for hybrid encryption. *Journal of Cryptology* 21(1), 97–130 (Jan 2008)
2. Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the LWE, NTRU schemes! In: Catalano, D., De Prisco, R. (eds.) *SCN 18*. LNCS, vol. 11035, pp. 351–367. Springer, Heidelberg (Sep 2018)
3. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: Holz, T., Savage, S. (eds.) *USENIX Security 2016*. pp. 327–343. USENIX Association (Aug 2016)
4. Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited - new reduction, properties and applications. In: Canetti, R., Garay, J.A. (eds.) *CRYPTO 2013, Part I*. LNCS, vol. 8042, pp. 57–74. Springer, Heidelberg (Aug 2013)
5. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of symmetric-key primitives for advanced cryptographic protocols. *Cryptology ePrint Archive, Report 2019/426* (2019), <https://eprint.iacr.org/2019/426>
6. Aly, A., Cong, K., Keller, M., Orsini, E., Rotaru, D., Scherer, O., Scholl, P., Smart, N.P., Tanguy, T., Wood, T.: *SCALE and MAMBA v1.9: Documentation* (2020), <https://homes.esat.kuleuven.be/~nsmart/SCALE/Documentation.pdf>
7. Arita, S., Tsurudome, K.: Construction of threshold public-key encryptions through tag-based encryptions. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) *ACNS 09*. LNCS, vol. 5536, pp. 186–200. Springer, Heidelberg (Jun 2009)
8. Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) *TCC 2010*. LNCS, vol. 5978, pp. 201–218. Springer, Heidelberg (Feb 2010)
9. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Paterson, K.G. (ed.) *EUROCRYPT 2011*. LNCS, vol. 6632, pp. 169–188. Springer, Heidelberg (May 2011)

10. Bernstein, D.J., Chuengsatiansup, C., Lange, T., van Vredendaal, C.: NTRU Prime. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
11. Bernstein, D.J., Persichetti, E.: Towards KEM unification. Cryptology ePrint Archive, Report 2018/526 (2018), <https://eprint.iacr.org/2018/526>
12. Bogdanov, A., Guo, S., Masny, D., Richelson, S., Rosen, A.: On the hardness of learning with rounding over small modulus. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 209–224. Springer, Heidelberg (Jan 2016)
13. Boneh, D., Gennaro, R., Goldfeder, S., Jain, A., Kim, S., Rasmussen, P.M.R., Sahai, A.: Threshold cryptosystems from threshold fully homomorphic encryption. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 565–596. Springer, Heidelberg (Aug 2018)
14. Bonte, C., Smart, N.P., Tanguy, T.: Thresholdizing HashEdDSA: MPC to the Rescue. Cryptology ePrint Archive, Report 2020/214 (2019), <https://eprint.iacr.org/2020/214>
15. Cozzo, D., Smart, N.P.: Sharing the LUOV: Threshold post-quantum signatures. In: Albrecht, M. (ed.) 17th IMA International Conference on Cryptography and Coding. LNCS, vol. 11929, pp. 128–153. Springer, Heidelberg (Dec 2019)
16. Cozzo, D., Smart, N.P.: Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 169–186. Springer, Heidelberg (2020)
17. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO’98. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (Aug 1998)
18. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing* 33(1), 167–226 (2003)
19. Damgård, I., Fitzi, M., Kiltz, E., Nielsen, J.B., Toft, T.: Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 285–304. Springer, Heidelberg (Mar 2006)
20. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (Aug 2012)
21. D’Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18. LNCS, vol. 10831, pp. 282–305. Springer, Heidelberg (May 2018)
22. D’Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: SABER. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
23. De Feo, L., Meyer, M.: Threshold schemes from isogeny assumptions. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 187–212. Springer, Heidelberg (May 2020)
24. Dent, A.W.: A designer’s guide to KEMs. In: Paterson, K.G. (ed.) 9th IMA International Conference on Cryptography and Coding. LNCS, vol. 2898, pp. 133–151. Springer, Heidelberg (Dec 2003)

25. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: 23rd ACM STOC. pp. 542–552. ACM Press (May 1991)
26. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO’99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999)
27. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology* 26(1), 80–101 (Jan 2013)
28. Garcia-Morchon, O., Zhang, Z., Bhattacharya, S., Rietman, R., Tolhuizen, L., Torre-Arce, J.L., Baan, H., Saarinen, M.J.O., Fluhrer, S., Laarhoven, T., Player, R.: Round5. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
29. Grassi, L., Kales, D., Khovratovich, D., Roy, A., Rechberger, C., Schafneggger, M.: Starkad and Poseidon: New hash functions for zero knowledge proof systems. Cryptology ePrint Archive, Report 2019/458 (2019), <https://eprint.iacr.org/2019/458>
30. Hamburg, M.: Three Bears. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
31. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017)
32. Ishihara, T., Aono, H., Hongo, S., Shikata, J.: Construction of threshold (hybrid) encryption in the random oracle model: How to construct secure threshold tag-KEM from weakly secure threshold KEM. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 07. LNCS, vol. 4586, pp. 259–273. Springer, Heidelberg (Jul 2007)
33. Keller, M., Orsini, E., Rotaru, D., Scholl, P., Soria-Vazquez, E., Vivek, S.: Faster secure multi-party computation of AES and DES using lookup tables. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17. LNCS, vol. 10355, pp. 229–249. Springer, Heidelberg (Jul 2017)
34. Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 830–842. ACM Press (Oct 2016)
35. Kraitsberg, M., Lindell, Y., Osheter, V., Smart, N.P., Talibi Alaoui, Y.: Adding distributed decryption and key generation to a ring-LWE based CCA encryption scheme. In: Jang-Jaccard, J., Guo, F. (eds.) ACISP 19. LNCS, vol. 11547, pp. 192–210. Springer, Heidelberg (Jul 2019)
36. Libert, B., Yung, M.: Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 75–93. Springer, Heidelberg (Mar 2012)
37. Lim, C.H., Lee, P.J.: Another method for attaining security against adaptively chosen ciphertext attacks. In: Stinson, D.R. (ed.) CRYPTO’93. LNCS, vol. 773, pp. 420–434. Springer, Heidelberg (Aug 1994)
38. Lu, X., Liu, Y., Jia, D., Xue, H., He, J., Zhang, Z., Liu, Z., Yang, H., Li, B., Wang, K.: LAC. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
39. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (Apr 2012)

40. Naehrig, M., Alkim, E., Bos, J., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
41. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press (May 1990)
42. Nishide, T., Ohta, K.: Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 343–360. Springer, Heidelberg (Apr 2007)
43. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (Apr 2001)
44. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (Dec 2009)
45. Poppelmann, T., Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Schwabe, P., Stebila, D., Albrecht, M.R., Orsini, E., Osheter, V., Paterson, K.G., Peer, G., Smart, N.P.: NewHope. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
46. Rotaru, D., Smart, N.P., Tanguy, T., Vercauteren, F., Wood, T.: Actively secure setup for SPDZ. Cryptology ePrint Archive, Report 2019/1300 (2019), <https://eprint.iacr.org/2019/1300>
47. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>
48. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT’98. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (May / Jun 1998)
49. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology* 15(2), 75–96 (Mar 2002)
50. Smart, N.P., Wood, T.: Error detection in monotone span programs with application to communication-efficient multi-party computation. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 210–229. Springer, Heidelberg (Mar 2019)
51. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Hirt, M., Smith, A.D. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 192–216. Springer, Heidelberg (Oct / Nov 2016)
52. Xie, X., Xue, R., Zhang, R.: Deterministic public key encryption and identity-based encryption from lattices in the auxiliary-input setting. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 1–18. Springer, Heidelberg (Sep 2012)
53. Zhandry, M.: How to record quantum queries, and applications to quantum indistinguishability. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 239–268. Springer, Heidelberg (Aug 2019)
54. Zhang, Z., Chen, C., Hoffstein, J., Whyte, W., Schanck, J.M., Hulsing, A., Rijneveld, J., Schwabe, P., Danba, O.: NTRUEncrypt. Tech. rep., National Institute of Standards and Technology (2019), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>