Constant Ciphertext-Rate Non-Committing Encryption from Standard Assumptions

Zvika Brakerski[1], Pedro Branco[2], Nico Döttling[3], Sanjam Garg[4], and Giulio Malavolta[5]

[1] Weizmann Institute of Science
zvika.brakerski@weizmann.ac.il
[2] IT, IST - University of Lisbon
pmbranco@math.tecnico.ulisboa.pt
[3] Helmholtz Center for Information Security (CISPA)
doettling@cispa.saarland
[4] UC Berkeley
sanjamg@berkeley.edu
[5] Max Planck Institute for Security and Privacy
giulio.malavolta@hotmail.it

Abstract. Non-committing encryption (NCE) is a type of public key encryption which comes with the ability to equivocate ciphertexts to encryptions of arbitrary messages, i.e., it allows one to find coins for key generation and encryption which "explain" a given ciphertext as an encryption of any message. NCE is the cornerstone to construct adaptively secure multiparty computation [Canetti et al. STOC'96] and can be seen as the quintessential notion of security for public key encryption to realize ideal communication channels.

A large body of literature investigates what is the best message-to-ciphertext ratio (i.e., the rate) that one can hope to achieve for NCE. In this work we propose a near complete resolution to this question and we show how to construct NCE with constant rate in the plain model from a variety of assumptions, such as the hardness of the learning with errors (LWE), the decisional Diffie-Hellman (DDH), or the quadratic residuosity (QR) problem. Prior to our work, constructing NCE with constant rate required a trusted setup and indistinguishability obfuscation [Canetti et al. ASIACRYPT'17].

1 Introduction

Multiparty computation (MPC) considers the problem of mutually distrustful parties computing a function over their inputs, while revealing no information beyond the output of the function [22,13]. Traditionally, the security of MPC protocols is analyzed considering two different adversarial models: In the *static* settings, the adversary is required to announce the set of parties that he wants to corrupt prior to the execution of the protocol. On the other hand, in the *adaptive* settings, the adversary can corrupt parties at any point in time of the execution, possibly depending on previously exchanged messages. Adaptive

security is widely believed to be the correct notion of security to consider when analyzing the security of cryptographic protocols as we do not have any real-life justification for the static model (except that adaptive security is in general harder to achieve).

Non-Committing Encryption (NCE) was presented in [4] as the cornerstone to construct adaptively-secure MPC, both in the stand-alone model [4] and in the UC settings [5]. Loosely speaking, NCE incarnates the notion of an ideal private channel, which retains the security of its messages, even if it is corrupted at a later point in time. NCE is a public-key encryption (PKE) scheme for which there exists a simulator that is able to create a pair of public key pk and ciphertext ct, indistinguishable from a real pair public key/ciphertext. Given any message M at any later point in time, the simulator can craft random coins that explain the transcript $(\mathsf{pk}, \mathsf{ct})$ for M. A central efficiency measure for PKE is the rate of encryption, i.e., the asymptotic ratio between the size of the message and the size of the ciphertext. While we know how to construct high-rate PKE[6] from numerous hardness assumptions, the situation is less cheerful for NCE. The most efficient schemes from the literature in the plain model have ciphertext-rate poly-logarithmic in the security parameter [23,16], whereas (asymptotically) matching the efficiency of PKE currently requires a trusted setup and indistinguishability obfuscation [6]. Motivated by the current state of affairs, we ask the following question:

Can we build NCE with ciphertext rate $\mathcal{O}(1)$ from standard assumptions?

1.1 Our Results

We present a nearly complete resolution of this question by constructing the first NCE schemes with constant ciphertext-rate from a new abstraction, which we call Packed Encryption with Partial Equivocality (PEPE). Then we show how to instantiate PEPE from several standard problems, such as learning with errors (LWE), decisional Diffie-Hellman (DDH), and quadratic residuosity (QR). Specifically, we prove the following main theorem.

Theorem 1 (Informal). *Assuming the hardness of the {LWE, DDH, QR} problem, there exists a non-committing encryption scheme with ciphertext rate $\mathcal{O}(1)$.*

We note that our PEPE schemes achieve rate 1. The rate of our NCE schemes is a small constant which is mostly determined by an information-theoretic technique in the construction of NCE from PEPE.

As a contribution of independent interest, we present a novel ciphertext-compression technique for packed ElGamal encryption schemes which preserves correctness perfectly. As a direct corollary, we obtain a linearly-homomorphic encryption scheme with rate 1 from the DDH assumption.

[6] Rate-1 PKE can be easily constructed using hybrid encryption.

Theorem 2 (Informal). *Assuming the hardness of the DDH problem, there exists a linearly homomorphic encryption scheme with rate* 1*.*

This result generalizes and improves the recent work of Döttling *et al.* [10], where they obtained a rate-1 oblivious transfer from DDH (trivially implied by rate-1 linearly-homomorphic encryption) with inverse polynomial correctness error. Their scheme could be lifted to achieve negligible decryption error at the cost of introducing error-correcting codes, thus losing the additive homomorphism. Among other things, our scheme implies simpler and more direct constructions of rate-1 private information retrieval and rate-1 lossy trapdoor functions (using the same compilers as described in [10]) from the DDH assumption, without error correcting codes.

1.2 Related Work

The study of the rate of NCE has been the subject of a large body of literature. In the following we briefly review prior progress on improving the rate of NCE. We only consider NCE schemes with optimal round complexity, i.e., two-round protocols. The first instantiation of NCE is due to Canetti *et al.* [4] and achieved quadratic ciphertext-rate $\mathcal{O}(\lambda^2)$ under the RSA or the Computational Diffie-Hellman (CDH) assumption. Some three-round protocols were proposed after that [1,8] (both achieving linear rate), but the only improvement in the two-round settings was only made several years later in [7], where an NCE with ciphertext-rate $\mathcal{O}(\lambda)$ was presented, assuming the hardness of factoring Blum integers.

The rate question for NCE has recently received renewed interest: In [17], a scheme based on the ϕ-hiding assumption and achieving polylogarithmic (in the length of the message) ciphertext-rate was presented. This result was improved in a subsequent work [16], where a scheme with polylogarithmic (in the security parameter) ciphertext-rate and based on the LWE assumption with superpolynomial modulus-to-noise ratio was proposed. Finally, a scheme with quasi-optimal (i.e., logarithmic) ciphertext-rate was presented in [23], assuming the hardness of the DDH problem. We also mention the work of Canetti *et al.* [6], which constructs NCE with optimal rate (i.e., $1 - o(1)$) but at the cost of assuming indistinguishability obfuscation ($\mathrm{i}\mathcal{O}$) and a trusted setup. A comparison with our results is presented in Table 1.

1.3 Discussion and Open Problems

We stress that, as done in (most of) prior works improving the rate of NCE (e.g. [17,23]), we do not take the size of the public key into account when measuring the rate of the scheme. This is justified by the fact that (i) the public keys do not depend on the encrypted messages: In some scenarios it might be acceptable to have a more expensive "offline" communication while optimizing for an efficient "online" (i.e. message-dependent) phase. Furthermore, (ii) one can encrypt multiple messages under the same public key. That is, the size of the

	Ciphertext Rate	Hardness Assumption	Setup
[4]	$\mathcal{O}(\lambda^2)$	RSA, CDH	-
[7]	$\mathcal{O}(\lambda)$	Factoring Blum integers	-
[20]	$\mathcal{O}(\lambda)$	DDH, LWE, Factoring Blum integers	-
[17]	$\mathsf{poly}(\log \ell)$	ϕ-hiding	Oblivious sampling of RSA modulus
[16]	$\mathsf{poly}(\log \lambda)$	LWE	-
[6]	$1 - o(1)$	$i\mathcal{O}$	CRS
[23]	$\mathcal{O}(\log \lambda)$	DDH	-
Our result	$\mathcal{O}(1)$	LWE, DDH, QR	-

Table 1. Comparison with previous work. We focus only on constructions which have two rounds of communication. λ denotes the security parameter and ℓ denotes the length of the message to be encrypted.

public key grows linearly with the number of *equivocable* ciphertexts, as opposed to *all* ciphertexts.

Finally, our work still leaves open the question about the *true* rate of NCE: Is (round-optimal) NCE with (asymptotic) rate 1 possible from standard assumptions and in the plain model, or is a small constant rate, as achieved in this work, the best we can hope for?

2 Technical Overview

Before delving in the presentation of our scheme, we briefly recall the NCE scheme of [16], which is based on LWE with superpolynomial modulus-to-noise ratio. Let $M \in \{0,1\}^\ell$ be a (long) message we want to encrypt. The public key of this scheme is essentially a *packed Regev* key, that is it consists of a matrix $\mathbf{A}$ and vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$. The matrix $\mathbf{A} \in \mathbb{Z}_q^{k \times n}$ is chosen uniformly random (where k, n are two polynomials in the security parameter λ) whereas the vectors $\mathbf{v}_i$ are chosen in two different modes. Let $I_R \subseteq [\ell]$ be a set of indices of size $\ell/8$ chosen at random by the key generator. We think of this set as part of the secret key.

- For all $i \in I_R$ the component public key $\mathbf{v}_i$ is computed by $\mathbf{v}_i = \mathbf{s}_i\mathbf{A} + \mathbf{e}_i$ where $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^k$ is the corresponding component secret key and $\mathbf{e}_i \leftarrow\!\!\$\, \chi^n$ is a noise term, chosen from an appropriate LWE error distribution χ.
- For all $i \notin I_R$ the component keys $\mathbf{v}_i \leftarrow\!\!\$\, \mathbb{Z}_q^n$ are chosen uniformly random.

To encrypt a message M, it is first encoded into a binary string $\mathbf{y} \in \{0,1\}^\ell$ using a suitable error-correcting code (ECC), the choice of which is rather delicate[7]. The encrypter then chooses a random subset $I_S \subseteq [\ell]$, also of size $\ell/8$. For

[7] We need a code ECC which can efficiently decode from a $1/2 - \delta$ fraction of *random* errors.

all indices $i \in I_S$, we replace the i-th component of the string $\mathbf{y}$ by uniformly random bits.

The (modified) string $\mathbf{y}$ is then encrypted using a noisy version of the packed Regev scheme [21] in its gaussian variant. More precisely, one first samples a vector $\mathbf{r}$ from a suitable discrete gaussian over $\mathbb{Z}^n$ and computes

$$\mathbf{c}_1 = \mathbf{A}\mathbf{r}^T$$
$$\forall i \in [\ell] : w_i = \mathbf{v}_i\mathbf{r}^T + e_i^* + \mathbf{y}_i \cdot q/2,$$

where the *masking noise terms* e_i^* are chosen from an appropriate discrete gaussian. To decrypt a ciphertext $(\mathbf{c}_1, w_1, \ldots, w_\ell)$ one proceeds as follows. For all indices $i \in I_R$, the decrypter is in possession of a component secret key $\mathbf{s}_i$ which allows him to recover $\mathbf{y}_i$ by computing $w_i - \mathbf{s}_i\mathbf{c}_1 \approx \mathbf{y}_i \cdot q/2$ and rounding. All components with indices outside of I_R are effectively erased from the view of the receiver. However, by the above choice of parameters the receiver will be able to recover the message M with high probability using the efficient decoder of ECC. This establishes correctness of the scheme.

We will briefly discuss how we can equivocate messages if the system is set up in simulation mode. Instead of running honest key generation, the simulator chooses a set $I_b \subseteq [\ell]$ of size $\ell/4$. We call I_b the set of *bad* indices. Now, the simulator chooses the matrix $\mathbf{A}$ jointly with the $\mathbf{v}_i$ for $i \in I_b$ via a lattice trapdoor sampler. I.e., the simulator generates a matrix $\mathbf{B} \in \mathbb{Z}_q^{(k+\ell/4)\times n}$ with a lattice trapdoor $\mathsf{td}_{\mathbf{B}}$, then sets $\mathbf{A}$ to be the first k rows of $\mathbf{B}$ and uses the remaining $\ell/4$ rows for the vectors $\mathbf{v}_i$ with indices $i \in I_b$. The remaining $\mathbf{v}_i$ with indices in the good set $I_g = [\ell] \setminus I_b$ will be chosen as LWE samples, i.e. for these components the simulator will know a corresponding secret key $\mathbf{s}_i$.

To simulate a ciphertext, the simulator chooses a uniformly random bit string $\mathbf{y}'$ and encrypts it as before via noisy Regev encryption. We will briefly sketch the main ideas of how ciphertexts are equivoked. Given a message M, the simulator needs to compute random coins r_G which explain the public key pk and r_E which explain the ciphertext ct. Now, M is encoded into a binary string $\mathbf{y} \in \{0,1\}^\ell$ via ECC. Now, note since the string $\mathbf{y}'$ was chosen at random, it will agree with $\mathbf{y}$ in approximately 50% of the indices. For the remaining 50% of indices on which $\mathbf{y}'$ and $\mathbf{y}$ disagree, the simulator has two strategies at its disposal.

- For all indices $i \in I_b$ it will be able to resample the gaussian $\mathbf{r}$ via a gaussian sampler that uses the lattice trapdoor $\mathsf{td}_{\mathbf{B}}$. This effectively allows the simulator to *reprogram* all ciphertext components w_i with index $i \in I_b$ as encryptions of $\mathbf{y}_i$ (instead of $\mathbf{y}'_i$). This resampling procedure is the reason the masking noise terms e_i^* are needed. The resampling procedure creates small artifacts in the ciphertext components with indices $i \in I_g$, and the masking noise terms are used to statistically drown these artifacts.
- For the remaining indices, it will claim they were in the set I_S by choosing this set appropriately.

A good deal of care has to be taken when opening the sets I_R and I_S in order to ensure that they have *the right statistics*. In order to ensure this, the simulator

will make use of the fact that any component key in the set I_g can be claimed to be either from the set I_R or $[\ell] \setminus I_R$.

2.1 Packed Encryption with Equivocality

Our first contribution is an abstraction of the above framework into a generic construction of NCE using a novel primitive that we call Packed Encryption with Partial Equivocality (PEPE).[8] A PEPE is a cryptographic primitive that allows one to encrypt a message $M \in \{0,1\}^\ell$ into a ciphertext ct, using random coins r_E. Later, we can find random coins r'_E such that the encryption of $M' \neq M$ is exactly ct, conditioned on the fact that M' and M differ only on some predefined positions. More precisely, a PEPE consists of the following algorithms.

- **Key Generation:** Given a subset $I \subset [\ell]$ and a bit b, it outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(b, I; r_G)$ on either the real mode (if $b = 0$) or on the ideal mode (if $b = 1$), created using random coins r_G. Public keys created in different modes should be indistinguishable. A pair of keys created in the ideal mode will allow for equivocation of some of the positions of an encrypted message.
- **Encryption:** Given a message $M \in \{0,1\}^\ell$ and a public key pk, it outputs a ciphertext $\mathsf{ct} \leftarrow \mathsf{E}(\mathsf{pk}, M; r_E)$ encrypted using random coins r_E.
- **Decryption:** Given a secret key sk corresponding to the subset I, it outputs M_i for $i \in I$.

Additionally, a PEPE scheme is equipped with the algorithms $\mathsf{EquivPK}$ and $\mathsf{EquivCT}$ defined as follows.

- **Equivocation of public key randomness:** Given a subset $I' \subset I$ and the pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(b, I; r_G)$, this algorithm outputs r'_G such that $(\mathsf{pk}, \mathsf{sk}') = \mathsf{KG}(0, I'; r'_G)$.
- **Equivocation of ciphertext randomness.** Given a message M' (that differs from M only in the indexes not in I) and random coins r_E, this algorithm outputs random coins r'_E such that $\mathsf{E}(\mathsf{pk}, M; r_E) = \mathsf{E}(\mathsf{pk}, M'; r'_E)$.

As security requirement, the random coins outputted by the algorithms described above should be indistinguishable from real random coins.

NCE from PEPE. Our construction of NCE from PEPE closely follows the outline [16] as explained above, where we replace packed Regev encryption with a PEPE scheme. In the real mode, we also setup the PEPE scheme in real mode. In simulation mode, we setup keys in the appropriate simulation mode. The ciphertext randomness equivocation property of the PEPE scheme serves as

[8] A somewhat similar notion is the one of Somewhere Equivocal Encryption [15]. However, Somewhere Equivocal Encryption is a purely symmetric-key primitive and equivocation is performed by finding a new secret key. On the other hand, PEPE is a public-key primitive and equivocation is achieved by finding new random coins for the key generation and encryption algorithms.

a drop-in replacement for the gaussian sampling property of the packed Regev scheme in [16]. The remaining aspects are essentially identical to the [16] such as the use of error correcting codes and set partitions.

Assuming that we have a PEPE scheme which achieves constant rate, then the rate of this NCE construction is dominated by the rate penalty of the error correcting code ECC. Consequently, given that ECC has constant rate, this transformation results in an NCE scheme with constant rate.

In the remainder of this outline we briefly discuss constructing rate-1 PEPE schemes from LWE, DDH, and QR.

2.2 Construction from LWE

Before presenting our construction for PEPE from the LWE assumption, we recall a compression technique for Regev's scheme, recently introduced in [3]. Recall that in packed Regev encryption, a ciphertext is of the form

$$\mathsf{ct} = (\mathbf{c}_1, (w_1, \ldots, w_\ell)) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$$

where $\mathbf{c}_1$ is a *ciphertext header* and $w_1, \ldots, w_\ell$ are the ciphertext payload components. As explained above, given a component secret key $\mathbf{s}_i$ a component w_i can be decrypted by computing $w_i - \mathbf{s}_i \cdot \mathbf{c}_1$ and rounding the result to either 0 or $q/2$. Given that the modulus q is sufficiently large, we can compress such a ciphertext by choosing an offset z such that for all indices i

$$w_i + z \notin [q/4 - B, q/4 + B] \cup [-q/4 - B, -q/4 + B]\,,$$

where B is a bound on the decryption noise. Given that the modulus q is large enough, we can ensure that such an offset z always exists and can be found efficiently. Note that z is computed from the ciphertext only, i.e. without the knowledge of the corresponding plaintexts. Given such a z, we can compress the w_i into single bits by computing $c_i = \lfloor w_i + z \rceil_2$. The new compressed ciphertext is composed by $(\mathbf{c}_1, \{c_i\}_{i\in[\ell]}, z)$. To decrypt such a compressed ciphertext, we compute $c_i - \lfloor \mathbf{s}_i\mathbf{c}_1 + z \rceil_2$. A routine calculation shows that, given that z satisfies the constraints above, decryption is always correct.

PEPE from LWE. Recasting the construction of [16] in terms of PEPE, immediately gives us a PEPE scheme of polylogarithmic rate. Since the scheme obtained in this way is a packed Regev scheme, it is naturally compatible with the ciphertext compression technique provided above.

To see that the resulting scheme still supports public key and ciphertext equivocation, note first that we leave the public key unmodified. On the other hand, note that ciphertext compression is merely a public post-processing operation on a ciphertext. Consequently, to equivocate a compressed ciphertext, all we have to do is to equivocate the underlying uncompressed ciphertext. Thus, given that the message length ℓ is sufficiently large, we obtain a PEPE scheme with rate 1 under the same assumptions as above.

2.3 Construction from DDH

We will now outline our DDH-based construction, which follows the same blueprint as the LWE-based construction. We first construct a PEPE scheme with poor rate ($\mathcal{O}(\lambda)$), and then combine it with a public ciphertext compression technique.

We will first explain our novel ciphertext compression technique for the discrete logarithm settings. This algorithm, can be seen as the computational analog of the one described above and it is inspired by recent techniques developed in the domain of homomorphic secret sharing [2]. The scheme is perfectly correct, however the caveat is that the compression algorithm will run in *expected* polynomial time (or, alternatively, will introduce a decryption error with negligible probability). Let $\mathbb{G}$ be a prime order group with generator g and let

$$(h_1 = g^{s_1}, \ldots, h_\ell = g^{s_\ell})$$

be a set of public keys. The ciphertexts that we want to compress are of the form

$$(g^r, (h_i^r g^{M_1}, \ldots, h_i^r g^{M_\ell})) = (c_1, (w_1, \ldots, w_\ell)) \in \mathbb{G}^{\ell+1}$$

where $r \leftarrow_\$ \mathbb{Z}_p$ and $M \in \{0,1\}^\ell$, which is an extended version of the El-Gamal scheme. Decryption is performed component-wise by computing $w_i / c_1^{s_i}$ and checking if the result is equal to 1 (in which case, $M_i = 0$) or g ($M_i = 1$).

Let T be a polynomial in the security parameter. Our compression algorithm uses a pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \mathbb{G} \to \{0,1\}^\tau$. On input a ciphertext $(c_1, (w_1, \ldots, w_\ell))$, the compression algorithm samples a random key K for the PRF until the following two conditions are simultaneously satisfied: For all $i \in [\ell]$ it holds that

(1) $\mathsf{PRF}(K, w_i/g) \neq 0$.
(2) There exists a $\delta_i \in [T-1]$ such that $\mathsf{PRF}(K, w_i \cdot g^{\delta_i}) = 0$.

The compressed ciphertext ct is composed by $\mathsf{ct} = (K, c_1, \delta_1 \mod 2, \ldots, \delta_\ell \mod 2) \in \{0,1\}^\lambda \times \mathbb{G} \times \{0,1\}^\ell$ where δ_i is the smallest integer that satisfies condition (2). In order to decrypt, one needs to find, for every $i \in [\ell]$, the smallest γ_i such that $\mathsf{PRF}(K, c_1^{s_i} \cdot g^{\gamma_i}) = 0$ by exhaustive search. Finally it outputs $M_i = \delta_i \oplus \mathsf{LSB}(\gamma_i)$, where LSB denotes the least significant bit of an integer. Note that the scheme is correct with probability 1, since condition (1) ensures that there is no ambiguity in the decoding of the bit M_i. By setting the parameters appropriately, we can guarantee that K can always be found in polynomial time, except with negligible probability.

PEPE from DDH. We will now outline our uncompressed DDH-based PEPE construction, which shares some ideas with the LWE based construction above. Assume that the underlying group $\mathbb{G}$ supports oblivious sampling, i.e. we can sample uniformly random group elements without knowledge of any discrete logarithm relation. For a vector $\mathbf{a} \in \mathbb{Z}_p^n$ we will use the notation $[\mathbf{a}]$ to denote $g^{\mathbf{a}}$ (i.e. the component-wise exponentiation). In real mode, the public key $\mathsf{pk} = ([\mathbf{a}], \{[\mathbf{v}]\})$ of our DDH-based PEPE is chosen as follows. Choose the vector $[\mathbf{a}]$

and all $[\mathbf{v}_i]$ for $i \in [\ell] \setminus I$ obliviously. For all indices $i \in I$ choose a uniformly random $s_i \leftarrow_\$ \mathbb{Z}_p$ and set $[\mathbf{v}_i] = s_i \cdot [\mathbf{a}] = [s_i \cdot \mathbf{a}]$ (where we write exponentiation multiplicatively). The secret key consists of the component keys $\{s_i\}_{i \in I}$.

To encrypt a message $M \in \{0,1\}^\ell$, first choose a uniformly random $\mathbf{r} \leftarrow_\$ \mathbb{Z}_p^n$ and compute $[\mathbf{c}_1] = [\mathbf{a}] \cdot \mathbf{r} = [\mathbf{a} \cdot \mathbf{r}]$ and for all $i \in [\ell]$ $[w_i] = [\mathbf{v}_i] \cdot \mathbf{r} + [M_i]$. The vector $\mathbf{r} \in \mathbb{Z}_p^n$ constitute the random coins for encryption. To decrypt the i-th ciphertext component, compute $[w_i] - s_i \cdot [\mathbf{c}_1]$, output 0 if this equals 1 and 1 if it equals $g = [1]$.

We will now briefly outline how ciphertext equivocation works for this scheme. In the ideal mode, all elements of the public key are computed *non-obliviously* with respect to a single generator $g = [1]$. That is, we sample $[\mathbf{a}]$ by choosing a uniformly random $\mathbf{a}' \leftarrow_\$ \mathbb{Z}_p^n$ and setting $[\mathbf{a}] = \mathbf{a}' \cdot [1]$. For all $i \in I$ we choose a random $s_i \leftarrow_\$ \mathbb{Z}_p$ and set $[\mathbf{v}_i] = s_i \cdot \mathbf{a}' \cdot [1]$. For all $i \in [\ell] \setminus I$ we choose a uniformly random $\mathbf{v}_i' \leftarrow_\$ \mathbb{Z}_p^n$ and set $[\mathbf{v}_i] = \mathbf{v}_i' \cdot [1]$. The simulator will keep all non-obliviously sampled ring elements as equivocation trapdoor. Notice that obliviously sampled public keys and non-obliviously sampled public keys are identically distributed.

We will finally describe how ciphertexts are equivoked. For a given a ciphertext $\mathsf{ct} = ([\mathbf{c}_1], [w_1], \ldots, [w_\ell])$ encrypting a message $M \in \{0,1\}^\ell$, the simulator knows the random coins $\mathbf{r}$ that were used to generate this ciphertext. I.e. it knows (in $\mathbb{Z}_p$) that

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{a} \cdot \mathbf{r} \\ w_1 &= \mathbf{v}_1 \cdot \mathbf{r} + M_1 \\ &\vdots \\ w_\ell &= \mathbf{v}_\ell \cdot \mathbf{r} + M_\ell \end{aligned}$$

Now, given a message $M' \in \{0,1\}^\ell$ which agrees with M on the index set I we can equivoke the ciphertext ct as an encryption of M' by by uniformly choosing a solution $\bar{\mathbf{r}} \in \mathbb{Z}_p^n$ for the linear equation system

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{a} \cdot \bar{\mathbf{r}} \\ w_1 &= \mathbf{v}_{i_1} \cdot \bar{\mathbf{r}} + M'_{i_1} \\ &\vdots \\ w_{i_k} &= \mathbf{v}_{i_k} \cdot \bar{\mathbf{r}} + M'_{i_k} \end{aligned}$$

where $[\ell] \setminus I = \{i_1, \ldots, i_k\}$. Notice that since for $i \in [\ell] \setminus I$ the $\mathbf{v}_i$ are chosen uniformly at random, given that $k + 1 \leq n$ this system has full rank with overwhelming probability. Consequently, we can sample a uniform solution $\bar{\mathbf{r}}$ via basic linear algebra. Finally, note that since for $i \in I$ the $\mathbf{v}_i$ are of the form $s_i \cdot \mathbf{a}$, it also holds that $w_i = \mathbf{v}_i \cdot \bar{\mathbf{r}} + M'_i$, as $\mathbf{a}\mathbf{r} = \mathbf{c}_1 = \mathbf{a}\bar{\mathbf{r}}_1$ and $M_i = M'_i$. Thus this scheme has perfect ciphertext equivocality.

Finally, applying the oblivious ciphertext compression algorithm described above yields a PEPE scheme of rate 1.

2.4 Construction from QR

We conclude our overview by briefly sketching how to adapt the above developed techniques to construct PEPE from the QR assumption.[9] Similarly to the DDH case, the public is composed by

$$\mathsf{pk} = \left([\mathbf{a}], \{[\mathbf{v}_i]\}_{i\in[\ell]}\right)$$

where $[\mathbf{a}] \leftarrow_\$ \mathbb{QR}_N^n$ and $[\mathbf{v}_i] = s_i[\mathbf{a}]$ for $i \in I$ and $[\mathbf{v}_i] \leftarrow_\$ \mathbb{QR}_N^n$. To encrypt a message $M \in \{0,1\}^\ell$, we compute

$$\tilde{\mathsf{ct}} = ([\mathbf{a}\mathbf{r}^T], ((-1)^{M_1} \cdot [\mathbf{v}_1\mathbf{r}^T], \ldots, (-1)^{M_\ell} \cdot [\mathbf{v}_\ell\mathbf{r}^T]) \in \mathbb{QR}_N{}^n \times \mathbb{G}^\ell$$

with a uniformly chosen $\mathbf{r} \leftarrow_\$ \mathbb{Z}_{(N-1)/2}^n$, and compress it into $\mathsf{ct} = ([\mathbf{a}\mathbf{r}^T], (b_1, \ldots, b_\ell)) \in \mathbb{QR}_N^n \times \{0,1\}^\ell$ via the compressing procedure of [10]. When generating a public key in the equivocal mode, the simulator keeps $\mathbf{a}$ and the vectors $\mathbf{v}_i$, for $i \notin I$, to himself. The vectors $\mathbf{a}$ and $\mathbf{v}_i$ will allow him to equivocate by solving a linear system of equations in a similar fashion as in the DDH case.

3 Preliminaries

Throughout this work, λ denotes the natural security parameter. By $\mathsf{negl}(\lambda)$, we denote a negligible function in λ, that is, a function that vanishes faster than any polynomial in λ.

Let $n \in \mathbb{N}$. Then, $[n]$ denotes the set $\{1, \ldots, n\}$. If $\mathcal{A}$ is an algorithm, we denote by $y \leftarrow \mathcal{A}(x)$ the output y after running $\mathcal{A}$ on input x. If S is a (finite) set, we denote by $x \leftarrow_\$ S$ the experiment of sampling uniformly at random an element x from S. If D is a distribution over S, we denote by $x \leftarrow_\$ D$ the element x sampled from S according to D. We say that D is B-bounded if for every $\mathbf{x} \leftarrow_\$ D$, we have $\|\mathbf{x}\| < B$, except with negligible probability, and where $\|\mathbf{x}\|$ is the usual ℓ_2 norm. We will usually use bold upper-case letters (e.g., $\mathbf{M}$) to denote matrices and lower-case letters (e.g., $\mathbf{v}$) to denote vectors, unless explicitly state otherwise. Let $q \in \mathbb{N}$. We define the rounding function $\lfloor\cdot\rceil_2 : \mathbb{Z}_q \to \mathbb{Z}_2$ as $\lfloor x\rceil_2 = \lfloor x \cdot 2/q\rceil \mod 2$.

We say that two distributions are computationally indistinguishable if no probabilistic polynomial-time (PPT) adversary can distinguish them.

The following lemma will be useful and provides a tail bound for the hypergeometric distribution.

Lemma 3 *Let $\mathsf{H}(a,b,n)$ be a hypergeometric distribution, with $a = \alpha n$ and $b = \beta n$, and let X be a random variable sampled from $\mathsf{H}(a,b,n)$. Then*

$$\Pr\left[X \leq (\alpha\beta - \varepsilon)n\right] \leq \mathsf{negl}(n)$$

for some constant $0 < \varepsilon < 1$.

[9] The QR-based construction is presented in the full version of this paper.

3.1 Coding Theory

We present some basic coding theory definitions and results that will be useful for our work.

Definition 4 (Error-Correcting Code) *A (binary) Error-Correnting Code (ECC) consists of a pair of algorithms* $\mathsf{ECC}_{N,n} = (\mathsf{Encode}, \mathsf{Decode})$ *such that:*

- $\mathbf{c} \leftarrow \mathsf{Encode}(M \in \{0,1\}^n)$ *takes as input a message* $M \in \{0,1\}^n$ *to be encoded. It outputs a codeword* $\mathbf{c} \in \{0,1\}^N$.
- $M \leftarrow \mathsf{Decode}(\mathbf{c}')$ *takes as input a corrupted codeword* $\mathbf{c}' \in \{0,1\}^N$. *It outputs* M *if* $\mathbf{c}'$ *and* $\mathbf{c} \leftarrow \mathsf{Decode}(M)$ *differ in at most* t *positions.*

Let $\mathsf{ECC}_{N,n}$ be a ECC. We call $R = n/N$ the rate of a code C. The error rate is defined as $E = t/N$. A list-decoding ECC [14] is a ECC such that the Decode algorithm outputs a list S of polynomial size (in the security parameter), one of which is the correct original encoded message. Constructions for list-decoding ECC with constant rate and that correct a large amount of errors (say, $1/2 - \zeta$ for any constant $\zeta > 0$) are known to exist [14].

Lemma 5 ([18]) *Let* C *be a list-decoding ECC with rate* R *and error rate* E. *Then, there exists a unique-decoding error correction code with rate* R *and error rate* E *given that One-Way Functions exist.*

In particular, there exists a code with constant rate R *and error rate of* $1/2-\zeta$, *for any constant* $\zeta > 0$, *given that One-Way Functions exist.*

3.2 Hardness Assumptions

In the following, we present the hardness assumptions that we use in this work.

Learning with Errors The Learning with Errors (LWE) problem was firstly presented in [21]. We now present the decisional version of the problem. In the following, let D_σ be a discrete Gaussian distribution with parameter σ.

Definition 6 (Learning with Errors) *Let* $k, q \in \mathbb{Z}$ *and let* D_σ *be an error distribution. The LWE assumption holds if for any PPT adversary*

$$|\Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e})] - \Pr[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{u})]| \leq \mathsf{negl}(\lambda)$$

for all $n \in \mathbb{Z}$, *where* $\mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{k \times n}$, $\mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^k$, $\mathbf{e} \leftarrow\!\!\$\, D_\sigma^n$ *and* $\mathbf{u} \leftarrow\!\!\$\, \mathbb{Z}_q^n$.

In this work, we assume the hardness of the LWE with superpolynomial modulus-to-noise ratio. That is, we assume that the problem remains hard even when $B/q = \mathsf{negl}(\lambda)$ where the error $\mathbf{e}$ comes from a B-bounded distribution.

The following lemma states that we can *drown* (i.e., statistically hide) an error vector with a much wider distribution.

Lemma 7 *Let q, B, σ such that $q = \lambda^{\omega(1)}$ and $\sigma/B = \lambda^{\omega(1)}$. Then the distributions D_σ and $D_\sigma + e$ are statistically close, where e is sampled from a B-bounded distribution.*

The following lemma states that there are matrices statistically close to uniform and for which we can sample low-norm pre-images with the help of a trapdoor [12,19].

Lemma 8 ([19]) *There exists a pair of algorithms* $(\mathsf{TrapGen}, \mathsf{SampleD})$ *such that:*

- $(\mathbf{B}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^\lambda, k, n, q)$ *takes as input the security parameter λ and $n, k, q \in \mathbb{Z}$. It outputs a matrix $\mathbf{B} \in \mathbb{Z}_q^{k \times n}$ and a trapdoor* td*. The matrix $\mathbf{B}$ is 2^{-k} close to uniform.*
- $\mathbf{r} \leftarrow \mathsf{SampleD}(\mathsf{td}, \mathbf{B}, \mathbf{y}, \sigma)$ *takes as input a trapdoor* td*, a matrix $\mathbf{B}$ and a vector $\mathbf{y} \in \mathbb{Z}_q^k$. It outputs $\mathbf{r} \in \mathbb{Z}_q^n$ such that $\mathbf{r} \leftarrow\!\!\$\, D_{\Lambda_\mathbf{y}^\perp(\mathbf{B}),\sigma}$, where $D_{\Lambda_\mathbf{y}^\perp(\mathbf{B}),\sigma}$ is the discrete Gaussian distribution with standard deviation σ over the lattice $\Lambda_\mathbf{y}^\perp(\mathbf{B}) = \{\mathbf{r} \in \mathbb{Z}_q^n : \mathbf{A}\mathbf{r}^T = \mathbf{y}\}$.*

Decisional Diffie-Hellman A (prime-order) *group generator* is an algorithm $\mathcal{G}$ that takes as an input a security parameter 1^λ and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, p is the order of the group which is always a prime number unless differently specified, and g is a generator of the group. In the following we state the decisional version of the Diffie-Hellman (DDH) assumption [9].

Definition 9 (Decisional Diffie-Hellman Assumption) *A group generator algorithm $\mathcal{G}$ satisfies the DDH assumption (or is DDH-hard) if for any PPT adversary $\mathcal{A}$*

$$\left|\Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), (g^a, g^b, g^{ab}))] - \Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), (g^a, g^b, g^c))]\right| \leq \mathsf{negl}(\lambda)$$

where $(\mathbb{G}, p, g) \leftarrow\!\!\$\, \mathcal{G}(1^\lambda)$ and $(a, b, c) \leftarrow\!\!\$\, \mathbb{Z}_p$.

In this work, we use the matrix version of the DDH assumption, called the Matrix Decisional Diffie-Hellman Assumption (MDDH), which generalizes the DDH assumption (and other number-theoretic assumptions). Let $g \in \mathbb{G}$ and let $\mathbf{M} \in \mathbb{Z}_p^{k \times n}$. We denote by $[\mathbf{M}] \in \mathbb{G}^{k \times n}$ the matrix

$$g^\mathbf{M} = \begin{pmatrix} g^{\mathbf{M}_{1,1}} & \dots & g^{\mathbf{M}_{1,n}} \\ \vdots & \ddots & \vdots \\ g^{\mathbf{M}_{k,1}} & \dots & g^{\mathbf{M}_{k,n}} \end{pmatrix}.$$

Definition 10 (Matrix Decisional Diffie-Hellman Assumption [11]) *A group generator algorithm $\mathcal{G}$ satisfies the MDDH if for any PPT algorithm $\mathcal{A}$ such that*

$$|\Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), ([\mathbf{A}], [\mathbf{w}\mathbf{A}]))] - \Pr[1 \leftarrow \mathcal{A}((\mathbb{G}, p, g), ([\mathbf{A}], [\mathbf{u}]))]| \leq \mathsf{negl}(\lambda)$$

where $(\mathbb{G}, p, g) \leftarrow\!\!\$\, \mathcal{G}(1^\lambda)$, $k < n$, $\mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_p^{k \times n}$, $\mathbf{w} \leftarrow\!\!\$\, \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow\!\!\$\, \mathbb{Z}_p^n$.

Observe that anyone can compute $s[\mathbf{A}] = [s\mathbf{A}]$, $\mathbf{w}[\mathbf{A}] = [\mathbf{w}\mathbf{A}]$ or $[\mathbf{A}]\mathbf{v}^T = [\mathbf{A}\mathbf{v}^T]$ knowing $[\mathbf{A}]$, s, $\mathbf{w}$ and $\mathbf{v}$, for any $\mathbf{A} \in \mathbb{Z}_q^{k \times n}$, $s \in \mathbb{Z}_q$, $\mathbf{w} \in \mathbb{Z}_q^k$ and $\mathbf{v} \in \mathbb{Z}_q^n$.

Quadratic Residuosity In this version, we omit the QR assumption description due to space restrictions.

3.3 Non-Committing Encryption

The formal definition of Non-Committing Encryption, as well as its security requirements, are presented below.

Definition 11 (Non-Committing Encryption) *A Non-Committing Encryption (NCE) scheme is composed by a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, r_G)$ *takes as input a security parameter* λ *and some randomness* r_G*. It outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*.*
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r_E)$ *takes as input a public key* pk*, a message* M *and randomness* r_E*. It outputs a ciphertext* c*.*
- $M/\perp \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ *takes as input a secret key* sk *and a ciphertext* c*. It outputs either a message* M *or an error message* $\perp$*.*
- $(\mathsf{pk}, c, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda)$ *takes as input a security parameter* λ*. It outputs a simulated public key* pk*, a ciphertext* c *and an internal state* st*.*
- $(r_G, r_E) \leftarrow \mathsf{Sim}_2(M, \mathsf{st})$ *takes as input a message* M *and an internal state* st*. It outputs a pair of randomness for key generation and for encryption* (r_G, r_E)*.*

A NCE scheme should have the following properties:

- **Correctness.** A NCE scheme is said to be correct if

$$\Pr\left[M \leftarrow \mathsf{Dec}(\mathsf{sk}, c) : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, M)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

- **Simulatability.** Let $\mathcal{A}$ be any PPT adversary. A NCE scheme is said to be simulatable if the distributions IDEAL and REAL are computationally indistinguishable to $\mathcal{A}$, where

$$\mathsf{IDEAL} = \left\{(M, \mathsf{pk}, c, r_G, r_E) : \begin{array}{c}(\mathsf{pk}, c, \mathsf{st}) \leftarrow \mathsf{Sim}_1(1^\lambda) \\ M \leftarrow \mathcal{A}(\mathsf{pk}) \\ (r_G, r_E) \leftarrow \mathsf{Sim}_2(M, \mathsf{st})\end{array}\right\}$$

and

$$\mathsf{REAL} = \left\{(M, \mathsf{pk}, c, r_G, r_E) : \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, r_G) \\ M \leftarrow \mathcal{A}(\mathsf{pk}) \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, M, r_E)\end{array}\right\}.$$

4 Ciphertext Shrinking Algorithms

In this section we discuss how we can shrink the ciphertext of certain cryptosystems based on LWE, DDH or QR. Every procedure presented in this section is a post-processing operation that is applied to a ciphertext in order to reduce its size.

4.1 Ciphertext Shrinking Algorithm for LWE-based Encryption Schemes

The following technique to shrink ciphertexts of LWE-based PKE schemes was firstly introduced in [3]. This is a post-processing technique that can be applied to every decrypt-and-multiply PKE scheme (see [3] for details). In particular, it can be applied to the usual Regev's scheme [21] which we use to construct our NCE scheme.

Construction 1 *Consider a PKE scheme with ciphertexts of the form* $(\mathbf{c}_1, (w_{2,1}, \ldots, w_{2,\ell})) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^\ell$*, secret key* $\mathbf{S} \in \mathbb{Z}_q^{\ell \times n}$ *and where decryption is computed by multiplying* $\lfloor (w_{2,1}, \ldots, w_{2,\ell}) - \mathbf{S}\mathbf{c}_1^T \rceil_2 = \lfloor M + e \rceil_2$ *where* e *is sampled from a* B*-bounded distribution. We describe the shrinking algorithms in detail:*

$\mathsf{Shrink}(\mathsf{pk}, (\mathbf{c}_1, (w_{2,1}, \ldots, w_{2,\ell})))$*:*

- *Choose* $z \leftarrow\!\!\$\, \mathbb{Z}_q \setminus U$ *where*

$$U = \bigcup_{i=1}^{\ell} \left(\left[-\frac{q}{4} - w_{2,i} - B, -\frac{q}{4} - w_{2,i} + B \right] \cup \left[\frac{q}{4} - w_{2,i} - B, \frac{q}{4} - w_{2,i} + B \right] \right).$$

- *Compute* $c_{2,i} = \lfloor w_{2,i} + z \rceil_2 \in \mathbb{Z}_2$ *for every* $i \in [\ell]$.
- *Output* $\mathsf{ct} = (\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z)$.

$\mathsf{ShrinkDec}(\mathsf{sk} = \mathbf{S}, \mathsf{ct})$:

- *Parse* ct *as* $(\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z)$.
- *Compute* $M_i \leftarrow \left(c_{2,i} - \lfloor \mathbf{s}_i \mathbf{c}_1^T + z \rceil_2 \right) \mod 2$ *where* $\mathbf{s}_i$ *is the* i*-th row of* $\mathbf{S}$.
- *Output* $M = (M_1, \ldots, M_\ell)$.

Note that each bit of M is independently recovered from the other ones. Hence, we can relax the definition of $\mathsf{ShrinkDec}$ in order to output only a partial decryption of M. More precisely, if a subset $I \subseteq [\ell]$ is given as input to $\mathsf{ShrinkDec}$, then it outputs $\{M_i\}_{i \in I}$.

The following lemma guarantees the correctness of the shrinking procedure presented above.

Lemma 12 ([3]) *Let* $B = B(\lambda)$ *and* $q > 4\ell B$*. Then the shrinking algorithm described in Construction 1 is correct up to noise* B*.*

4.2 Ciphertext Shrinking Algorithm for DDH-based Encryption Schemes

Before presenting the shrinking procedure compatible with DDH-based encryption schemes, recall the definition of Pseudorandom Functions (PRF).

Definition 13 (Pseudorandom Function) *Let $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$. A* Pseudorandom Function *(PRF) is defined by a keyed function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\alpha \to \{0,1\}^\beta$ such that, for any adversary $\mathcal{A}$*

$$|\Pr[1 \leftarrow \mathcal{A}(y,x) : y \leftarrow \mathsf{PRF}(K,x)] - \Pr[1 \leftarrow \mathcal{A}(y,x) : y \leftarrow f(x)]| \leq \mathsf{negl}(\lambda)$$

for any $x \in \{0,1\}^\alpha$, where $f : \{0,1\}^\alpha \to \{0,1\}^\beta$ is a uniformly chosen random function and the key K is sampled uniformly at random from $\{0,1\}^\lambda$.

We now explain how one can compress ciphertexts of ElGamal-based encryption schemes. The following technique is a variant of the compression technique introduced in [2,10]. However, in this variant we achieve perfect correctness.

Construction 2 *Below we show our DDH-based scheme, with message space $\mathbb{Z}_q^\ell$, for some polynomials $q = q(\lambda)$ and $\ell = \ell(\lambda)$. The scheme is parametrized by two polynomials $\tau = \tau(\lambda)$ and $T = T(\lambda)$ that influence the runtime of the evaluation algorithm, whose exact value will be fixed later. The scheme assumes the existence of a pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \mathbb{G} \to \{0,1\}^\tau$. We also assume that we have ciphertexts of the form $(c_1, (w_{2,1}, \ldots, w_{2,\ell})) \in \mathbb{G} \times \mathbb{G}^\ell$ and that the secret key is of the form $(x_1, \ldots x_\ell) \in \mathbb{Z}_p^\ell$. Decryption is done by computing $w_{2,i}/c_1^{x_i} = g^{M_i}$ and recovering $M_i \in \mathbb{Z}_q$, for each $i \in [\ell]$.*

$\mathsf{Shrink}(\mathsf{pk}, (c_1, (w_{2,1}, \ldots, w_{2,\ell}))):$

- *Set $d_0 = c_1$ and $d_i = w_{2,i}$, for all $i = 1, \ldots, \ell$.*
- *Sample a uniform key $K \leftarrow_\$ \{0,1\}^\lambda$ such that the following conditions are simultaneously satisfied:*
  (1) For all $i = 1, \ldots, \ell$ and for all $k = 1, \ldots, (q-1)$ it holds that

$$\mathsf{PRF}(K, d_i/g^k) \neq 0^\tau.$$

  (2) For all $i = 1, \ldots, \ell$ there exists some $k = 0, \ldots, (T-1)$ such that $\mathsf{PRF}(K, d_i \cdot g^k) = 0^\tau$.
- *For all $i = 1, \ldots, \ell$ let δ_i be the smallest non-negative integer such that $\mathsf{PRF}(K, d_i \cdot g^{\delta_i}) = 0^\tau$.*
- *Return $\mathsf{ct} = (K, d_0, \delta_1 \bmod q, \ldots, \delta_\ell \bmod q)$.*

$\mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{ct})$:

- *Parse sk as $(x_1, \ldots, x_\ell)$ and ct as $(K, d_0, \delta_1 \bmod q, \ldots, \delta_\ell \bmod q)$.*
- *Compute for all $i = 1, \ldots, \ell$ the smallest non-negative integer γ_i such that $\mathsf{PRF}(K, d_0^{x_i} \cdot g^{\gamma_i}) = 0^\tau$*
- *Set $M_i = \delta_i - \gamma_i \bmod q$.*
- *Return $M = (M_1, \ldots, M_\ell)$.*

Again, note that each element M_i can be independently decrypted. Thus, if the $\mathsf{ShrinkDec}$ algorithm receives as input a subset $I \subseteq [\ell]$, it outputs $\{M_i\}$ for $i \in I$.

Analysis. The more interesting aspects of this scheme concern its correctness and the runtime of the subroutines.

Lemma 14 *The scheme as described in Construction 2 is perfectly correct.*

Proof. We assume without loss of generality that the decryption algorithm takes as input an evaluated ciphertext $\mathsf{ct} = (K, d_0, \delta_1, \ldots, \delta_\ell)$. Recall that $d_0 = c_1 = g^r$ for a random $r \leftarrow\!\!\$\, \mathbb{Z}_p$. Furthermore, for all $i = 1, \ldots, \ell$ the term δ_i is defined to be the smallest non-negative integer (mod q) such that $\mathsf{PRF}(K, d_i \cdot g^{\delta_i}) = 0^\tau$, where

$$d_i = h_i^r g^{M_i} = g^{x_i r_i} g^{M_i} = d_0^{x_i} g^{M_i}$$

Recall that γ_i is defined to be the smallest non-negative integer such that $\mathsf{PRF}(K, d_0^{x_i} \cdot g^{\gamma_i}) = 0^\tau$. Note that the pair (δ_i, γ_i) is always well defined by condition (2). We claim that

$$d_i \cdot g^{\delta_i} = d_0^{x_i} \cdot g^{\gamma_i}$$

with probability 1. Assume that this is not the case, then we have that $M_i + \delta_i \neq \gamma_i$. We distinguish two cases:

(a) $M_i + \delta_i < \gamma_i$: This case cannot happen since we assumed that γ_i was the smallest non-negative integer such that $\mathsf{PRF}(K, d_0^{x_i} \cdot g^{\gamma_i}) = 0^\tau$.

(b) $M_i + \delta_i > \gamma_i$: This case implies that $\gamma_i < q$ since $M_i \leq q$ and δ_i is the smallest non-negative integer such that $\mathsf{PRF}(K, d_i \cdot g^{\delta_i}) = \mathsf{PRF}(K, d_0^{x_i} \cdot g^{M_i} \cdot g^{\delta_i}) = 0^\tau$. Consequently we have that $\mathsf{PRF}(K, d_i / g^{\gamma_i}) = 0^\tau$ where $\gamma_i < q$, which violates condition (2).

Therefore we have that

$$M_i = \gamma_i - \delta_i \bmod q$$

for all $i = 1, \ldots, \ell$. This concludes our proof.

By condition (2), the values of γ_i always lie within $T - 1$ steps from $d_0^{x_i}$ and therefore $\mathsf{ShrinkDec}$ runs in strict polynomial time. What is left to be shown is that Shrink runs in expected polynomial time.

Lemma 15 *Let* PRF *be a pseudorandom function, let* $\tau = \log_2(2(q-1)\ell)$ *and let* $T = 2^\tau \lambda \log_e(\ell) + (q-1)\ell$. *Then* Shrink *terminates within* λ *iterations except with negligible probability.*

Proof. Observe that all the subroutines of Shrink run in strict polynomial time, except for the sampling of K. It therefore suffices to bound the probability that some K satisfies conditions (1) and (2) simultaneously. Throughout the following analysis we treat $\mathsf{PRF}(K, \cdot)$ as a truly random function (indexed by K) and the same analysis holds true, up to a negligible amount, for the case that $\mathsf{PRF}(K, \cdot)$ is a pseudorandom function by a standard argument.

We first bound from below the probability that a uniform $K \leftarrow\!\!\$\, \{0,1\}^\lambda$ satisfies condition (1), that is,

$$\Pr\left[\forall i \in [\ell], \forall k \in [q-1] : \mathsf{PRF}(K, d_i/g^k) \neq 0^\tau\right] \geq \left(1 - \frac{1}{2^\tau}\right)^{(q-1)\ell}$$
$$\geq 1 - \frac{(q-1)\ell}{2^\tau} = 1 - \frac{(q-1)\ell}{2(q-1)\ell} = \frac{1}{2}$$

* where the probability is taken over the random choice of K. The first inequality comes from the fact that we assume that all points d_i/g^k are distinct (since it minimizes the probability) and therefore the outputs of $\mathsf{PRF}(K, \cdot)$ are uniformly and independently distributed over $\{0,1\}^\tau$. The second inequality is from Bernoulli. We now bound from above the probability that condition (2) is not satisfied, conditioned on the fact that condition (1) is met. Let us denote by $S \subseteq \{0,1\}^\lambda$ the set of all keys K that satisfy condition (1). Then we have

$$\Pr\left[\exists i \in [\ell] \text{ s.t. } \forall k = 0, \ldots, (T-1) : \mathsf{PRF}(K, d_i \cdot g^k) \neq 0^\tau \middle| K \in S\right]$$
$$\leq \sum_{i=1}^{\ell} \Pr\left[\forall k = 0, \ldots, (T-1) : \mathsf{PRF}(K, d_i \cdot g^k) \neq 0^\tau \middle| K \in S\right]$$
$$\leq \sum_{i=1}^{\ell} \left(1 - \frac{1}{2^\tau}\right)^{T-(q-1)\ell} \leq \sum_{i=1}^{\ell} e^{-\frac{T-(q-1)\ell}{2^\tau}} = \sum_{i=1}^{\ell} e^{-\lambda \log_e(\ell)} = e^{-\lambda}$$

where the probability is taken over the random choice of K. The first inequality comes from a union bound whereas the second inequality is derived by observing that the constraint $K \in S$ fixes the value of $\mathsf{PRF}(K, \cdot)$ on at most $(q-1)\ell$ points.

To conclude, the probability that condition (1) is not satisfied after λ uniform choices of K is at most $2^{-\lambda}$ and the probability that condition (2) is not satisfied constrained on meeting condition (1) is $e^{-\lambda}$. By a union bound, the probability that Shrink does not terminate after λ iterations is at most $2^{-\lambda} + e^{-\lambda}$.

Rate-1 Linearly Homomorphic Encryption from DDH. An interesting consequence of our algorithm is that it yields a linearly homomorphic encryption scheme with rate approaching 1 from the DDH assumption. To see why this is the case, we recall the packed version of ElGamal encryption: The public key of the scheme consists of the tuple $(g, h_1, \ldots, h_\ell) = (g, g^{x_1}, \ldots, g^{x_\ell})$, and a ciphertext for a message $(M_1, \ldots, M_\ell)$ is of the form

$$(g^r, h_1^r \cdot g^{M_1}, \ldots, h_\ell^r \cdot g^{M_\ell})$$

for some uniformly chosen $r \leftarrow\!\!\$\, \mathbb{Z}_p$. This scheme can be shown secure by ℓ invocations of the DDH assumption and satisfies the structural requirements to apply our shrinking algorithm as described above. Furthermore note that the scheme supports the homomorphic evaluation of linear functions $f : \mathbb{Z}_q^\ell \to \mathbb{Z}_q$. One caveat of this scheme is that the runtime of the shrinking algorithm is polynomial q and therefore the function f has a polynomial-size range (we stress

that q is a bound on the output size and not the order of the DDH-hard group p). Yet these homomorphic capabilities suffice for many interesting applications, such as constructing rate-1 oblivious transfer, or semi-compact homomorphic encryption for branching programs [10].

4.3 Ciphertext Shrinking Algorithm for QR-based Encryption Schemes

The ciphertext shrinking algorithm for QR-based encryption schemes is the one presented in [10]. We omit it here due to space restrictions.

5 Packed Encryption with Partial Equivocality

We begin this section by presenting the formal definition of PEPE as well as its security properties. We then show how to construct this primitive under several hardness assumptions. Then, we present constructions of PEPE from LWE, DDH and QR assumptions.

Definition 16 *A Packed Encryption with Partial Equivocality (PEPE) scheme that encrypts messages in* $\{0,1\}^\ell$ *is composed by a tuple of algorithms* $(\mathsf{KG}, \mathsf{E}, \mathsf{D}, \mathsf{EquivPK}, \mathsf{EquivCT})$ *where:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, b \in \{0,1\}, I, r)$ *takes as input a security parameter* λ*, a bit* b*, a set of indexes* $I \in [\ell]$ *and random coins* r*.*[10] *It outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*. When* $b = 0$ *we say that the keys were generated in the real mode. Otherwise, if* $b = 1$*, we say that the keys were generated in the ideal mode.*
- $\mathsf{ct} \leftarrow \mathsf{E}(\mathsf{pk}, M \in \{0,1\}^\ell, r)$ *takes as input a public key* pk*, a message* M *and random coins* r*, and outputs a ciphertext* ct*.*
- $(M_i)_{i \in I} \leftarrow \mathsf{D}(\mathsf{sk}, \mathsf{ct})$ *takes as input a secret key* sk *and a ciphertext* ct*. It outputs bits* M_i*, for* $i \in I$*.*
- $r' \leftarrow \mathsf{EquivPK}(\mathsf{sk}, b, (I, r), I')$ *takes as input a secret key* sk*, a bit* b*, subsets* $I, I' \subseteq [\ell]$ *and randomness* r*. It outputs randomness* r'*.*
- $r' \leftarrow \mathsf{EquivCT}(\mathsf{sk}, (M, r), \{M'_i\}_{i \notin I})$ *takes as input a secret key* sk*, a pair of message and randomness* (M, r) *and and some bits* $\{M'_i\}_{i \notin I}$ *together with a subset* $I \subseteq [\ell]$*. It outputs random coins* r'*.*

A PEPE scheme should fulfill correctness for decryption and for equivocality. Also, the random coins used in the key generation and encryption algorithms should be indistinguishable from random coins outputted by the equivocality algorithms.

[10] When the random coins r are omitted, it means they are chosen uniformly at random during the execution of the algorithm. In this case, the algorithm also outputs r. The same happens for algorithm E.

– **Correctness.** For any message $M \in \{0,1\}^\ell$ and any subset $I \subset [\ell]$, we have that

$$\mathsf{Pr}\left[\{M_i\}_{i\in I} = \{M'_i\}_{i\in I} : \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 0, I, r_G)\\ \mathsf{ct} \leftarrow \mathsf{E}(\mathsf{pk}, M, r_E)\\ \{M'_i\}_{i\in I} \leftarrow \mathsf{D}(\mathsf{sk}, \mathsf{ct})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

– **Public key randomness indistinguishability.** The random coins outputted by the algorithm $\mathsf{EquivPK}$ should be *computationally indistinguishable* from true random coins. That is, the distributions $\mathsf{IDEAL}_{\mathsf{pk}}$ and $\mathsf{REAL}_{\mathsf{pk}}$ should be computationally indistinguishable, where

$$\mathsf{IDEAL}_{\mathsf{pk}_b} = \left\{r_G : \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, b, I', r'_G)\\ r_G \leftarrow \mathsf{EquivPK}(\mathsf{sk}, b, (I', r'_G), I)\end{array}\right\}$$

and

$$\mathsf{REAL}_{\mathsf{pk}} = \left\{r_G : (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 0, I, r_G)\right\}$$

for any subsets $I, I' \subset [\ell]$ such that $I \subset I'$ and any $b \in \{0,1\}$.
Note that this also ensures that no adversary can distinguish public keys created in the ideal mode or in the real mode as the distribution of both keys are indistinguishable.

– **Ciphertext randomness indistinguishability.** The random coins outputted by the algorithm $\mathsf{EquivCT}$ should be *statistically close* to true random coins. That is, for any subset $I \subset [\ell]$ and any message $M' \in \{0,1\}^\ell$, the distributions $\mathsf{IDEAL}_{\mathsf{ct}}$ and $\mathsf{REAL}_{\mathsf{ct}}$ should be statistically close, where

$$\mathsf{IDEAL}_{\mathsf{ct}} = \left\{(\mathsf{pk}, M, r_E) : \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 1, I, r'_G)\\ \mathsf{ct} \leftarrow \mathsf{E}(\mathsf{pk}, M', r'_E)\\ M \leftarrow \mathcal{A}(\mathsf{pk})\\ r_E \leftarrow \mathsf{EquivCT}(\mathsf{sk}, (M', r'_E), \{M_i\}_{i\notin I})\end{array}\right\}$$

and

$$\mathsf{REAL}_{\mathsf{ct}} = \left\{(\mathsf{pk}, M, r_E) : \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 0, I, r'_G)\\ M \leftarrow \mathcal{A}(\mathsf{pk})\\ \mathsf{ct} \leftarrow \mathsf{E}(\mathsf{pk}, M, r_E)\end{array}\right\}$$

where $\mathcal{A}$ is an unbounded adversary which outputs a message M such that $M_i = M'_i$ for $i \in I$.

5.1 Packed Encryption with Partial Equivocality from LWE

We now present a PEPE scheme from the LWE assumption. The construction is similar to the one in [16], except that we use the compression technique introduced in [3] to achieve better rate.

Construction 3 *Let* $(\mathsf{TrapGen}, \mathsf{SampleD})$ *be the pair of algorithms described in Lemma 8, let* $(\mathsf{Shrink}, \mathsf{ShrinkDec})$ *the pair of algorithms described in Construction 1 and let* $\sigma, \sigma' \in \mathbb{R}$ *such that* $\sigma/\sigma' = \mathsf{negl}(\lambda)$.

$\mathsf{KG}(1^\lambda, b \in \{0,1\}, I, r_G)$*:*

- *If* $b = 0$*, do the following:*
  - *Choose* $\mathbf{A} \leftarrow_\$ \mathbb{Z}_q^{k \times n}$
  - *For* $i \in I$*, set* $\mathbf{v}_i = \mathbf{s}_i\mathbf{A} + \mathbf{e}_i$ *where* $\mathbf{s}_i \leftarrow_\$ \mathbb{Z}_q^k$ *and* $\mathbf{e}_i \leftarrow_\$ D_\sigma^n$*.*
  - *For* $i \notin I$*, set* $\mathbf{v}_i \leftarrow_\$ \mathbb{Z}_q^n$*.*
  - *Set* $\mathsf{pk} = (\mathbf{A}, \{\mathbf{v}_i\}_{i\in[\ell]})$ *and* $\mathsf{sk} = (I, \{\mathbf{s}_i\}_{i\in I})$
  - *Set the random coins* $r_G = \{\mathbf{e}_i\}_{i\in I}$*.*
- *Else if* $b = 1$*, do the following:*
  - *Run* $(\mathbf{B}, \mathsf{td}_\mathbf{B}) \leftarrow \mathsf{TrapGen}(1^\lambda, k + \ell - |I|, n, q)$ *and parse* $\mathbf{B}$ *as* $\begin{pmatrix}\mathbf{A}\\ \mathbf{V}\end{pmatrix} \in \mathbb{Z}_q^{(k+\ell-|I|)\times n}$*.*
  - *For* $i \in I$*, set*
    $$\mathbf{v}_i = \mathbf{s}_i\mathbf{A} + \mathbf{e}_i$$
    where $\mathbf{s}_i \leftarrow_\$ \mathbb{Z}_q^k$ *and* $\mathbf{e}_i \leftarrow_\$ D_\sigma^n$*.*
  - *For* $i \notin I$*, set* $\mathbf{v}_i = \mathbf{V}_i$*, where* $\mathbf{V}_i$ *is the* i*-th row of* $\mathbf{V}$*.*
  - *Set* $\mathsf{pk} = (\mathbf{A}, \{\mathbf{v}_i\}_{i\in[\ell]})$ *and* $\mathsf{sk} = (I, \{\mathbf{s}_i\}_{i\in I}, \mathsf{td}_\mathbf{B})$*.*
  - *Set the random coins* $r_G = \{\mathbf{e}_i\}_{i\in I}$*.*
- *Output* $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{E}(\mathsf{pk}, M \in \{0,1\}^\ell, r_E)$*:*

- *Parse* $\mathsf{pk} = (\mathbf{A}, \{\mathbf{v}_i\}_{i\in[\ell]})$*.*
- *Sample* $\mathbf{r} \leftarrow_\$ D_\sigma^n$*.*
- *Compute* $\mathbf{c}_1 \leftarrow \mathbf{A}\mathbf{r}^T$ *and* $w_{2,i} = \mathbf{v}_i\mathbf{r}^T + e_i + \lfloor q/2 \rceil \cdot M_i \in \mathbb{Z}_q$*, for every* $i \in [\ell]$*, where* $e_i \leftarrow_\$ D_{\sigma'}$*.*
- *Compress* $(\mathbf{c}_1, (w_{2,1}, \ldots, w_{2,\ell}))$ *into*
  $$(\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z) \leftarrow \mathsf{Shrink}(\mathbf{c}_1, (w_{2,1}, \ldots, w_{2,\ell})).$$
- *Set the random coins* r_E *to be* $(\mathbf{r}, \{e_i\}_{i\in[\ell]})$*.*
- *Output* $\mathsf{ct} = (\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z)$*.*

$\mathsf{D}(\mathsf{sk}, \mathsf{ct})$*:*

- *Parse* sk *as* $(I, \{\mathbf{s}_i\}_{i\in I})$ *and* ct *as* $(\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z)$*.*
- *Compute* $\{M_i\}_{i\in I} \leftarrow \mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{ct}, I)$*.*
- *Output* $\{M_i\}_{i\in I}$*.*

$\mathsf{EquivPK}(\mathsf{sk}, b, (I, r), I')$*:*

- *If* $I' \not\subseteq I$*, then abort the protocol. Else, continue.*
- *Parse* r *as* $\{\mathbf{e}_i\}_{i_\in I}$*.*
- *If* $b = 0$*, parse* sk *as* $(I, \{\mathbf{s}_i\}_{i\in I})$*. Else, parse* $\mathsf{sk} = (I, \{\mathbf{s}_i\}_{i\in I}, \mathsf{td}_\mathbf{B})$*.*
- *Set* $r' = \{\mathbf{e}_i\}_{i\in I'}$ *and* $\mathsf{sk} = (I', \{\mathbf{s}_i\}_{i\in I'})$
- *Output* (sk', r')

$\mathsf{EquivCT}((\mathsf{sk}, r_G), (M, r), (M'_i)_{i \notin I})$*:*

- *Parse* sk *as* $(I, \{\mathbf{s}_i\}_{i \in I}, \mathsf{td}_{\mathbf{B}})$ *and* $r_G = \{\mathbf{e}_i\}_{i \in I}$*. Set* $\mathsf{ct} = (\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z) \leftarrow \mathsf{E}(\mathsf{pk}, M, r)$ *where* $r = (\mathbf{r}, \{e^*_i\}_{i \in [\ell]})$*.*
- *Sample* $e'_i \leftarrow\!\!\$\, D_{\sigma'}$ *for* $i \notin I$*, and* $\bar{\mathbf{r}} \leftarrow \mathsf{SampleD}(\mathsf{td}_{\mathbf{B}}, \mathbf{B}, \mathbf{y}, \sigma)$ *where* $\mathbf{y} = (\mathbf{c}_1, \{w_{2,i} - \lfloor q/2 \rceil M_i - e'_i\}_{i \notin I})$
- *For* $i \in I$*, set* $e'_i = e^*_i + \mathbf{e}_i(\mathbf{r} - \bar{\mathbf{r}})^T$*.*
- *Output* $r' = (\bar{\mathbf{r}}, \{e'_i\}_{i \in [\ell]})$*.*

Analysis. Correctness for decryption follows from the correctness of the usual Regev's scheme and from Lemma 12.

Lemma 17 (Public-key randomness indistinguishability) *The scheme in Construction 3 is public key randomness indistinguishable given that the LWE assumption holds.*

Proof. Assume that $b = 1$ in the experiment $\mathsf{IDEAL}_{\mathsf{pk}}$ (the case where $b = 0$ is just a particular case of this one). The proof follows from the following sequence of hybrids:

Hybrid $\mathcal{H}_0$*.* This is the experiment $\mathsf{IDEAL}_{\mathsf{pk}}$ between a challenger C and an adversary $\mathcal{A}$:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 1, I', r'_G)$ where $\mathsf{pk} = (\mathbf{A}, \{\mathbf{v}_i\}_{i \in [\ell]})$ and $\mathsf{sk} = (I', \{\mathbf{s}_i\}_{i \in I'}, \mathsf{td}_{\mathbf{B}})$.
- Run $r_G \leftarrow \mathsf{EquivPK}(\mathsf{sk}, 1, (I', r'_G), I)$.
- $b \leftarrow \mathcal{A}(r_G)$.

Hybrid $\mathcal{H}_1$*.* In this hybrid, we replace the matrix $\mathbf{A}$ and the vectors $\mathbf{v}_i$, when $i \notin I'$, for uniform ones.

- C chooses $\mathbf{A} \leftarrow \mathbb{Z}_q^{k \times n}$ and $\mathbf{v}_i \leftarrow\!\!\$\, \mathbb{Z}_q^n$ for $i \notin I'$. For $i \in I'$, it computes $\mathbf{v}_i \leftarrow \mathbf{s}_i\mathbf{A} + \mathbf{e}_i$. For $I \subset I'$, set $r_G = \{(\mathbf{s}_i, \mathbf{e}_i)\}_{i \in I}$. It sends r_G to $\mathcal{A}$.
- $b \leftarrow \mathcal{A}(r_G)$.

Claim. $|\Pr[1 \leftarrow \mathcal{A} : \mathcal{A} \text{ plays } \mathcal{H}_0] - \Pr[1 \leftarrow \mathcal{A} : \mathcal{A} \text{ plays } \mathcal{H}_1]| \leq \mathsf{negl}(\lambda)$.

By Lemma 8, $\mathbf{A}$ is statistically close to a uniform matrix. Using the same lemma, each $\mathbf{v}_i$, for $i \notin I'$, is also statistically close to a uniform vector. The claim follows.

Hybrid $\mathcal{H}_2$*.* In this hybrid, we replace each $\mathbf{v}_i$ for $i \in I' \setminus I$ by a uniform vector.

- C chooses $\mathbf{A} \leftarrow \mathbb{Z}_q^{k \times n}$ and $\mathbf{v}_i \leftarrow\!\!\$\, \mathbb{Z}_q^n$ for $i \notin I'$ and for $i \in I' \setminus I$. For $i \in I$, it computes $\mathbf{v}_i \leftarrow \mathbf{s}_i\mathbf{A} + \mathbf{e}_i$. For $I \subset I'$, it sets $r_G = \{\mathbf{s}_i, \mathbf{e}_i\}_{i \in I}$. It sends r_G to $\mathcal{A}$.
- $b \leftarrow \mathcal{A}(r_G)$.

Claim. Assume that the LWE assumption holds. Then

$$|\Pr[1 \leftarrow \mathcal{A} : \mathcal{A} \text{ plays } \mathcal{H}_1] - \Pr[1 \leftarrow \mathcal{A} : \mathcal{A} \text{ plays } \mathcal{H}_2]| \leq \mathsf{negl}(\lambda).$$

It is straightforward to build an algorithm that decides the LWE assumption given an adversary that is able to distinguish hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$. The claim follows.

Finally, note that hybrid $\mathcal{H}_2$ is exactly the experiment $\mathsf{REAL}_{\mathsf{pk}}$. Hence, the distributions are computationally indistinguishability given that the LWE assumption holds.

Lemma 18 (Ciphertext randomness indistinguishability) *The scheme in Construction 3 is ciphertext randomness indistinguishable.*

Proof. Let $\mathsf{ct} = (\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell})) \leftarrow \mathsf{E}(\mathsf{pk}, M, r_E)$ for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 1, I, r_G)$ where $\mathsf{pk} = (\mathbf{A}, \{\mathbf{v}_i\}_{i\in[\ell]})$, $\mathsf{sk} = (I, \{\mathbf{s}_i\}_{i\in I}, \mathsf{td}_{\mathbf{B}})$, and $r_E = (\mathbf{r}, \{e_i\}_{i\in[\ell]})$ is the randomness used in E to encrypt the message $M = (M_1, \ldots, M_\ell)$. Now let $M' = (M'_1, \ldots, M'_\ell)$ such that $M_i = M'_i$, for all $i \in I$, and $M_i \neq M'_i$ otherwise. After running $\mathsf{EquivCT}(\mathsf{sk}, (M, r_E), (M'_i)_{i\notin I})$ we obtain

$$r'_E = (\bar{\mathbf{r}}, \{e'_i\}_{i\notin I}).$$

Let $\mathsf{ct}' = (\mathbf{c}'_1, (c'_{2,1}, \ldots, c'_{2,\ell})) \leftarrow \mathsf{E}(\mathsf{pk}, M', r'_E)$. First, note that by definition of the algorithm $\mathsf{SampleD}$ (Lemma 8) we have that $\mathbf{A}\mathbf{r}^T = \mathbf{A}\bar{\mathbf{r}}^T$. Hence $\mathbf{c}_1 = \mathbf{c}'_1$.

For $i \in I$, we have that

$$\mathbf{v}_i\mathbf{r}^T + e_i^* + \left\lfloor \frac{q}{2} \right\rceil M_i = \mathbf{v}_i\bar{\mathbf{r}}^T + e'_i + \left\lfloor \frac{q}{2} \right\rceil M_i,$$

hence the rounded values are the same.

Finally, for $i \notin I$, by definition of $\mathsf{SampleD}$, we have that

$$\mathbf{v}_i\mathbf{r} + e_i + \left\lfloor \frac{q}{2} \right\rceil M_i = \mathbf{v}_i\bar{\mathbf{r}} + e'_i + \lfloor q/2 \rceil M'_i.$$

Hence,

$$c_{2,i} = \lfloor \mathbf{v}_i\mathbf{r} + e_i + \lfloor q/2 \rceil M_i + z \rceil_2 = \lfloor \mathbf{v}_i\bar{\mathbf{r}} + e'_i + \lfloor q/2 \rceil M'_i + z \rceil_2 = c'_{2,i}.$$

We conclude that $\mathsf{ct} = \mathsf{ct}'$.

By Lemma 7, we have that $e'_i \leftarrow\!\!\$\, D_{\sigma'} + \mathbf{e}_i(\mathbf{r} - \bar{\mathbf{r}})^T$ and $e_i^* \leftarrow\!\!\$\, D_{\sigma'}$ are statistically close.

5.2 Packed Encryption with Partial Equivocality from DDH

The DDH-based construction for PEPE is presented below as well as the corresponding security proofs.

Construction 4 *Let $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$, $n \in \mathbb{N}$ and $(\mathsf{Shrink}, \mathsf{ShrinkDec})$ be the algorithms from Construction 2. The DDH-based PEPE scheme is defined as follows:*

$\mathsf{KG}(1^\lambda, b \in \{0,1\}, I, r_G)$:

- *If* $b = 0$*, do the following:*
  - *Choose* $[\mathbf{a}] = g^{\mathbf{a}}$ *where* $\mathbf{a} \leftarrow_\$ \mathbb{Z}_p^n$*, (here* $[\mathbf{a}]$ *is chosen obliviously).*
  - *For* $i \in I$*, set* $[\mathbf{v}_i] = s_i[\mathbf{a}]$ *where* $s_i \leftarrow_\$ \mathbb{Z}_p$*.*
  - *For* $i \notin I$*, set* $[\mathbf{v}_i] \leftarrow_\$ \mathbb{G}^n$*.*
  - *Set* $\mathsf{pk} = ([\mathbf{a}], \{[\mathbf{v}_i]\}_{i\in[\ell]})$ *and* $\mathsf{sk} = (I, \{s_i\}_{i\in I})$*.*
- *Else if* $b = 1$*, do the following:*
  - *Choose* $\mathbf{a} \leftarrow_\$ \mathbb{Z}_p^n$ *and compute* $[\mathbf{a}] = g^{\mathbf{a}}$*.*
  - *For* $i \in I$*, set* $[\mathbf{v}_i] = s_i[\mathbf{a}]$ *where* $s_i \leftarrow_\$ \mathbb{Z}_p^k$*.*
  - *For* $i \notin I$*, set* $[\mathbf{v}_i] \leftarrow_\$ \mathbb{G}^n$
  - *Set* $\mathsf{pk} = ([\mathbf{a}], \{[\mathbf{v}_i]\}_{i\in[\ell]})$ *and* $\mathsf{sk} = (I, \mathbf{a}, \{s_i\}_{i\in I}, \{\mathbf{v}_i\}_{i\notin I})$*.*
- *Output* $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{E}(\mathsf{pk}, M \in \{0,1\}^\ell, r_E)$:

- *Parse* $\mathsf{pk} = ([\mathbf{a}], \{[\mathbf{v}_i]\}_{i\in[\ell]})$*.*
- *Choose* $\mathbf{r} \leftarrow_\$ \mathbb{Z}_p^n$*.*
- *Compute* $[c_1] = [\mathbf{a}\mathbf{r}^T]$ *and* $w_{2,i} = [\mathbf{v}_i\mathbf{r}^T] \cdot g^{M_i}$ *for every* $i \in [\ell]$*.*
- *Compress* $([c_1], (w_{2,1}, \ldots, w_{2,\ell}))$ *into*

$$(K, [c_1], (c_{2,1}, \ldots, c_{2,\ell})) \leftarrow \mathsf{Shrink}([\mathbf{c}_1], (w_{2,1}, \ldots, w_{2,\ell}))$$

  where $(c_{2,1}, \ldots, c_{2,\ell}) = (\delta_1 \bmod 2, \ldots, \delta_\ell \bmod 2)$*.*
- *Set the random coins* r_E *to be* $\mathbf{r}$*.*
- *Output* $\mathsf{ct} = (K, [c_1], (c_{2,1}, \ldots, c_{2,\ell}))$*.*

$\mathsf{D}(\mathsf{sk}, \mathsf{ct})$:

- *Parse* sk *as* $(I, \{s_i\}_{i\in I})$ *and* ct *as* $(K, [c_1], (c_{2,1}, \ldots, c_{2,\ell}))$*.*
- *Compute* $\{M_i\}_{i\in I} \leftarrow \mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{ct}, I)$*.*
- *Output* $\{M_i\}_{i\in I}$*.*

$\mathsf{EquivPK}(\mathsf{sk}, b, (I, r), I')$:

- *If* $I' \not\subseteq I$*, then abort the protocol. Else, continue.*
- *If* $b = 0$*, parse* sk *as* $(I, \{s_i\}_{i\in I})$*. Else, parse* sk *as* $(I, \mathbf{a}, \{s_i\}_{i\in I}, \{\mathbf{v}_i\}_{i\notin I})$*.*
- *Set* $\mathsf{sk}' = (I', \{s_i\}_{i\in I'})$
- *Output* sk'

$\mathsf{EquivCT}(\mathsf{sk}, (M, r), \{M'_i\}_{i\notin I})$:

- *Parse* sk *as* $(I, \mathbf{a}, \{s_i\}_{i\in I}, \{\mathbf{v}_i\}_{i\notin I})$ *and* $r = \mathbf{r} \in \mathbb{Z}_p^n$*. Let* $\{i_1, \ldots, i_\alpha\} = [\ell] \setminus I$
- *Sample uniformly at random a solution* $\bar{\mathbf{r}} \in \mathbb{Z}_p^n$ *for*

$$\begin{pmatrix} \mathbf{a} & 0 \\ \mathbf{v}_{i_1} & M'_{i_1} \\ \vdots & \vdots \\ \mathbf{v}_{i_\alpha} & M'_{i_\alpha} \end{pmatrix} \begin{pmatrix} \bar{\mathbf{r}}^T \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{a}\mathbf{r}^T \\ \mathbf{v}_{i_1}\mathbf{r}^T + M_{i_1} \\ \vdots \\ \mathbf{v}_{i_\alpha}\mathbf{r}^T + M_{i_\alpha} \end{pmatrix}.$$

- *Output* $r_E = \bar{\mathbf{r}}$*.*

Analysis. We now proceed to the analysis of the construction above.

Lemma 19 *The scheme in Construction 4 is correct.*

Correctness for decryption follows from the correctness of the matrix version of the El Gamal scheme and from Lemma 14.

Lemma 20 (Public-key randomness indistinguishability) *The scheme in Construction 4 is public key randomness indistinguishable given that the MDDH assumption holds.*

Proof. The proof follows from the following sequence of hybrids:

Hybrid $\mathcal{H}_0$. This is the experiment $\mathsf{IDEAL}_{\mathsf{pk}}$ between a challenger C and an adversary $\mathcal{A}$:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KG}(1^\lambda, 1, I', r'_G)$ where $\mathsf{pk} = (\mathbf{a}, \{[\mathbf{v}_i]\}_{i\in[\ell]})$ and $\mathsf{sk} = (I', \{s_i\}_{i\in I'})$.
- Run $r_G \leftarrow \mathsf{EquivPK}(\mathsf{sk}, 1, (I', r'_G), I)$.
- $b \leftarrow \mathcal{A}(r_G)$.

Hybrid $\mathcal{H}_1$. In this hybrid, we replace the vectors $\mathbf{v}_i$, when $i \in I' \setminus I$, for uniform ones.

- C chooses $\mathbf{a} \leftarrow \mathbb{Z}_p^n$ and $\mathbf{v}_i \leftarrow_\$ \mathbb{Z}_p^n$ for $i \notin I'$ and for $i \in I' \setminus I$. For $i \in I$, it computes $[\mathbf{v}_i] \leftarrow s_i[\mathbf{a}]$. For $I \subset I'$, set $r_G = \{s_i\}_{i\in I}$. It sends r_G to $\mathcal{A}$.
- $b \leftarrow \mathcal{A}(r_G)$.

Claim. $|\Pr[1 \leftarrow \mathcal{A} : \mathcal{A} \text{ plays } \mathcal{H}_0] - \Pr[1 \leftarrow \mathcal{A} : \mathcal{A} \text{ plays } \mathcal{H}_1]| \leq \mathsf{negl}(\lambda)$.

It is straightforward to build a distinguisher for the MDDH assumption if we are given an algorithm $\mathcal{A}$ that can distinguish both hybrids.

Finally, note that hybrid $\mathcal{H}_1$ is exactly the experiment $\mathsf{REAL}_{\mathsf{pk}}$. Hence, the distributions are computationally indistinguishability given that the MDDH assumption holds.

Lemma 21 (Ciphertext randomness indistinguishability) *The scheme in Construction 4 is ciphertext randomness indistinguishable, if* $1 + \alpha \leq n$.

Proof. Let $M, M' \in \{0,1\}^\ell$ be any two messages such that $M_i = M'_i$, for $i \in I$, and $M_i \neq M'_i$ otherwise. We prove that, if $1 + \alpha \leq n$, then the equation

$$\begin{pmatrix} \mathbf{a} & 0 \\ \mathbf{v}_{i_1} & M'_{i_1} \\ \vdots & \vdots \\ \mathbf{v}_{i_\alpha} & M'_{i_\alpha} \end{pmatrix} \begin{pmatrix} \bar{\mathbf{r}}^T \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{a}\mathbf{r}^T \\ \mathbf{v}_1\mathbf{r}^T + M_{i_1} \\ \vdots \\ \mathbf{v}_{i_\alpha}\mathbf{r}^T + M_{i_\alpha} \end{pmatrix} \tag{1}$$

has a solution $\bar{\mathbf{r}} \in \mathbb{Z}_p^n$, except with negligible probability.

First, note that the equation in 1 is equivalent to

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{v}_{i_1} \\ \vdots \\ \mathbf{v}_{i_\alpha} \end{pmatrix} \left(\bar{\mathbf{r}}^T\right) = \begin{pmatrix} \mathbf{a}\mathbf{r}^T \\ \mathbf{v}_1\mathbf{r}^T + M_{i_1} - M'_{i_1} \\ \vdots \\ \mathbf{v}_{i_\alpha}\mathbf{r}^T + M_{i_\alpha} - M'_{i_\alpha} \end{pmatrix}$$

We now prove that the rank β of the matrix on the left side is maximal, that is, $\beta = 1+\alpha$. Note that, every row of this matrix is uniformly chosen at random.

By a simple counting argument, we have that the rank of the matrix on the left side is maximal, except with probability $1/|\mathbb{G}|$. Since $|\mathbb{G}| \in \mathcal{O}(2^{\omega(\log \lambda)})$, then

$$\Pr\left[\beta = 1 + \alpha\right] \geq 1 - \frac{1}{|\mathbb{G}|} \geq 1 - \mathsf{negl}(\lambda)\,.$$

If the rank of the matrix is equal to the rank of the augmented matrix, then the system of equations has solutions. Hence, we can find a solution $\bar{\mathbf{r}}$ for equation 1, except with negligible probability.

We now prove that, given $\bar{\mathbf{r}}$ satisfying Equation 1, $\mathsf{ct} = \mathsf{ct}'$, where $\mathsf{ct} = (K, [c_1], (c_{2,1}, \ldots, c_{2,\ell})) \leftarrow \mathsf{E}(\mathsf{pk}, M, r_E)$ and $\mathsf{ct}' = (K, [c'_1], (c'_{2,1}, \ldots, c'_{2,\ell})) \leftarrow \mathsf{E}(\mathsf{pk}, M', r'_E)$ where M' is such that $M'_i = M_i$ for $i \in I$ and $r_E = \mathbf{r}$, $r'_E = \bar{\mathbf{r}}$.

First, note that by Equation 1 we have that $\mathbf{a}\mathbf{r}^T = \mathbf{a}\bar{\mathbf{r}}^T$. Hence,

$$\left[\mathbf{a}\mathbf{r}^T\right] = \left[\mathbf{a}\bar{\mathbf{r}}^T\right] \Leftrightarrow [c_1] = [c'_1]\,. \tag{2}$$

A direct consequence of Equation 2 is that

$$s_i\mathbf{a}\mathbf{r}^T + M_i = s_i\mathbf{a}\bar{\mathbf{r}}^T + M_i \Leftrightarrow c_{2_i} = c'_{2,i}$$

for $i \in I$. It remains to show that $c_{2,i} = c'_{2,i}$ for $i \notin I$. Observe that

$$\mathbf{v}_i\mathbf{r}^T + M_i = \mathbf{v}_i\bar{\mathbf{r}}^T + M'_i$$

for $i \notin I$, from Equation 1. Hence, $c_{2,i} = c'_{2,i}$ for $i \notin I$.

Finally, the random coins r_E used in the encryption algorithm E (in the real mode) and the random coins r'_E outputted by the equivocation algorithm $\mathsf{EquivCT}$ have exactly the same distribution.

5.3 Packed Encryption with Partial Equivocality from QR

The construction of PEPE from QR is follows the same blueprint as the DDH construction. We omit the construction in this version.

6 From PEPE to constant ciphertext-rate NCE

Finally, we present the generic construction for NCE from PEPE, which generalizes the construction of [16]. Then, we analyze the security and efficiency of the construction.

The following lemma is adapted from [16] and will help us to prove security for the construction.

Lemma 22 ([16]) *Let* $\mathsf{ECC}_{\ell,\ell'} = (\mathsf{ECC.Encode}, \mathsf{ECC.Decode})$ *be an error-correcting code and let* $\mathsf{PEPE} = (\mathsf{PEPE.KG}, \mathsf{PEPE.E}, \mathsf{PEPE.D}, \mathsf{PEPE.Equiv})$ *be a PEPE scheme. There exists an algorithm* $\mathsf{F_{id}}$ *such that*

$$(I_R, I_S, \mathbf{z}') \leftarrow \mathsf{F_{id}}(I_g, \mathbf{y}, \mathbf{z})$$

where I_R, I_S, I_g *are subsets of* $[\ell]$ *and* $\mathbf{y}, \mathbf{z}, \mathbf{z}' \in \{0,1\}^\ell$*. Moreover, the distributions* $\mathsf{IDEAL}_{\mathsf{sets}}$ *and* $\mathsf{REAL}_{\mathsf{sets}}$ *are computationally indistinguishable given that the underlying PEPE scheme is public key randomness indistinguishable, where*

$$\mathsf{IDEAL}_{\mathsf{sets}} = \left\{ (I_R, I_S, \mathbf{z}') : \begin{array}{c} I_g \leftarrow_\$ W_g \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PEPE.KG}(1^\lambda, 1, I_g, r_G) \\ \mathbf{z} \leftarrow_\$ \{0,1\}^\ell \\ M \leftarrow \mathcal{A}(\mathsf{pk}) \\ \mathbf{y} \leftarrow \mathsf{ECC.Encode}(M) \\ (I_R, I_S, \mathbf{z}') \leftarrow \mathsf{F_{id}}(I_g, \mathbf{y}, \mathbf{z}) \end{array} \right\},$$

and

$$\mathsf{REAL}_{\mathsf{sets}} = \left\{ (I_R, I_S, \mathbf{z}') : \begin{array}{c} I_R \leftarrow_\$ W \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PEPE.KG}(1^\lambda, 0, I_R, r_G) \\ M \leftarrow \mathcal{A}(\mathsf{pk}) \\ \mathbf{y} \leftarrow \mathsf{ECC.Encode}(M) \\ I_S \leftarrow_\$ W \\ \mathbf{z}' \leftarrow f(\mathbf{y}, I_S) \end{array} \right\}$$

for any message M*, where* $W_g = \{I \subset [\ell] : |I| = 3\ell/4\}$*,* $W = \{I \subset [\ell] : |I| = \ell/8\}$*,* $I_R \subset I_g$*,* $\mathbf{y} = (y_1, \ldots, y_\ell)$ *and* f *is a function such that if* $\mathbf{z}' = (z'_1, \ldots, z'_\ell) \leftarrow f(\mathbf{y}, I_S)$ *then*

$$z'_i = \begin{cases} y_i, \text{ if } i \in I_S \\ z'_i \leftarrow_\$ \{0,1\}, \text{ otherwise} \end{cases}.$$

Construction 5 *Let* $\mathsf{ECC}_{\ell,\ell'} = (\mathsf{ECC.Encode}, \mathsf{ECC.Decode})$ *be a suitable error-correcting code with constant rate* $\mathcal{O}(1)$ *(Lemma 5) and* $\mathsf{PEPE} = (\mathsf{PEPE.KG}, \mathsf{PEPE.E}, \mathsf{PEPE.D}, \mathsf{PEPE.EquivPK}, \mathsf{PEPE.EquivCT})$ *be a PEPE scheme with message space* $\{0,1\}^{\ell'}$*.* $\mathsf{F_{id}}$ *is the algorithm of Lemma 22. We describe the NCE construction in full detail:*

$\mathsf{KeyGen}(1^\lambda)$*:*

- *Choose a random subset* $I_R \subset [\ell]$ *such that* $|I_R| = \ell/8$.
- *Compute* $(\mathsf{pk}_{\mathsf{pepe}}, \mathsf{sk}_{\mathsf{pepe}}) \leftarrow \mathsf{PEPE.KG}(1^\lambda, 0, I_R, r_{G,\mathsf{pepe}})$*, where* $r_{G,\mathsf{pepe}}$ *are the random coins.*
- *Output* $\mathsf{pk} = \mathsf{pk}_{\mathsf{pepe}}$, $\mathsf{sk} = (\mathsf{sk}_{\mathsf{pepe}}, I_R)$ *and* $r_G = (r_{G,\mathsf{pepe}}, I_R)$.

$\mathsf{Enc}(\mathsf{pk}, M)$*:*

- *Parse* pk *as* $\mathsf{pk}_{\mathsf{pepe}}$.
- *Encode the message by computing* $\mathbf{y} = (y_1, \ldots, y_\ell) \leftarrow \mathsf{ECC.Encode}(M)$.
- *Choose a random subset* $I_S \subset [\ell]$ *such that* $|I_S| = \ell/8$*. For every* $i \in [\ell]$*, set*

$$z_i = \begin{cases} y_i, \text{ if } i \in I_S \\ z_i' \leftarrow_\$ \{0,1\}, \text{ otherwise} \end{cases}$$

  for every $i \in [\ell]$ *and* $\mathbf{z} = (z_1, \ldots, z_\ell)$.
- *Compute* $\mathsf{ct} \leftarrow \mathsf{PEPE.E}(\mathsf{pk}_{\mathsf{pepe}}, \mathbf{z}, r_{E,\mathsf{pepe}})$ *where* $r_{E,\mathsf{pepe}}$ *are random coins*
- *Output* ct *and* $r_E = (\mathbf{z}, r_{E,\mathsf{pepe}}, I_S)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*:*

- *Parse* sk *as* $(\mathsf{sk}_{\mathsf{pepe}}, I_R)$.
- *Compute* $\{z_i\}_{i \in I_R} \leftarrow \mathsf{PEPE.D}(\mathsf{sk}_{\mathsf{pepe}}, \mathsf{ct})$.
- *For* $i \notin I_R$*, set* $z_i \leftarrow_\$ \{0,1\}$.
- *Output* $M \leftarrow \mathsf{ECC.Decode}(\mathbf{z})$ *where* $\mathbf{z} = (z_1, \ldots, z_\ell)$.

$\mathsf{Sim}_1(1^\lambda)$*:*

- *Choose a random subset* $I_g \subset [\ell]$ *such that* $|I_g| = 3\ell/4$.
- *Compute* $(\mathsf{pk}_{\mathsf{pepe}}, \mathsf{sk}_{\mathsf{pepe}}) \leftarrow \mathsf{PEPE.KG}(1^\lambda, 1, I_g, r_{G,\mathsf{pepe}})$*, where* $r_{G,\mathsf{pepe}}$ *are the random coins.*
- *Choose a random encoding* $\mathbf{z} \leftarrow_\$ \{0,1\}^\ell$ *and encrypt it*

$$\mathsf{ct} \leftarrow \mathsf{PEPE.E}(\mathsf{pk}_{\mathsf{pepe}}, \mathbf{z}, r_{E,\mathsf{pepe}}).$$

- *Output* $\mathsf{pk} = \mathsf{pk}_{\mathsf{pepe}}$, ct *and* $\mathsf{st} = (I_g, \mathbf{z}, r_{G,\mathsf{pepe}}, r_{E,\mathsf{pepe}})$.

$\mathsf{Sim}_2(M, \mathsf{st})$*:*

- *Parse* $\mathsf{st} = (\mathsf{pk}_{\mathsf{pepe}}, \mathsf{ct}, \mathsf{sk}_{\mathsf{pepe}}, I_g, \mathbf{z}, r_{G,\mathsf{pepe}}, r_{E,\mathsf{pepe}})$.
- *Encode the message* M *into* $\mathbf{y} \leftarrow \mathsf{ECC.Encode}(M)$.
- *Compute* $(I_R, I_S, \mathbf{z}') \leftarrow \mathsf{F}_{\mathsf{id}}(I_g, \mathbf{y}, \mathbf{z})$.
- *Set* $r'_{G,\mathsf{pepe}} \leftarrow \mathsf{PEPE.EquivPK}(\mathsf{sk}_{\mathsf{pepe}}, 1, (I_g, r_{G,\mathsf{pepe}}), I_R)$ *to be the randomness according to* I_R.
- *Let* $J = \{i \in [\ell] \setminus I_g : z_i \neq z_i'\}$*. Compute*

$$r'_{E,\mathsf{pepe}} \leftarrow \mathsf{PEPE.EquivCT}(\mathsf{sk}_{\mathsf{pepe}}, (\mathbf{z}, r_{E,\mathsf{pepe}}), \{z_i\}_{i \in J}).$$

- *Set* $r_G = (r'_{G,\mathsf{pepe}}, I_R)$ *and* $r_E = (\mathbf{z}', r'_{E,\mathsf{pepe}}, I_S)$*. Output* (r_G, r_E).

Analysis. We now proceed to the analysis of the scheme described above.

Theorem 23 (Correctness). *Let* $\mathsf{ECC}_{\ell,\ell'} = (\mathsf{ECC.Encode}, \mathsf{ECC.Decode})$ *be an ECC with error-rate* $1/2 - \delta$ *for some constant* $\delta > 0$ *(Lemma 5) and* $\mathsf{PEPE} = (\mathsf{PEPE.KG}, \mathsf{PEPE.E}, \mathsf{PEPE.D}, \mathsf{PEPE.EquivPK}, \mathsf{PEPE.EquivCT})$ *be a PEPE scheme. Then the scheme described in Construction 5 is correct.*

Proof. The proof of correctness follows the proof of correctness presented in [16]. Let $\mathbf{z} = (z_1, \ldots, z_\ell)$ be the codeword obtained after running Dec. The key observation is that $|I_R \cap I_S| = \xi$ follows a hypergeometric distribution $\mathsf{H}(1/8, 1/8, \ell)$. Thus, we can bound the maximum value of ξ, using Lemma 3, except with negligible probability. On the other hand, all other positions of $\mathbf{z}$ are correct with probability $1/2$. Thus, we can estimate the number of errors γ of $\mathbf{z}$:

$$\gamma \leq \left(\frac{1}{2} + \varepsilon\right)(\ell - \xi) \leq \ell\left(\frac{1}{2} + \varepsilon\right)\left(1 + \varepsilon - \frac{1}{16\ell^2}\right) \leq \ell\left(\frac{1}{2} - \delta\right)$$

where the second inequality follows from Lemma 3, and the third one follows from considering an appropriate value for the constant $\varepsilon > 0$.

Theorem 24 (Simulatability). *Let* PEPE *be a PEPE scheme. Then the scheme in Construction 5 is simulatable.*

The proof of the theorem above is presented in the full version of this paper.

Ciphertext-rate of the NCE scheme. Let $R = \ell'/\ell$ be the rate of the code used in Construction 5 and $M \in \{0,1\}^{\ell'}$. We now analyze the ciphertext-rate of the scheme for the LWE case when instantiated with the PEPE constructions of Section 5. The analysis for the DDH case follows the same reasoning.

The ciphertext is composed by $\mathsf{ct} = (\mathbf{c}_1, (c_{2,1}, \ldots, c_{2,\ell}), z) \in \mathbb{Z}_q^n \times \{0,1\}^\ell \times \mathbb{Z}_q$. Then, the ciphertext-rate is

$$\frac{(n+1)\log q + \ell}{\ell'} = \frac{(n+1)\log q}{\ell'} + R^{-1}.$$

The ciphertext-rate is equal to R^{-1} when ℓ' tends to infinity. When we use a code as in Lemma 5, then $R = \mathcal{O}(1)$, therefore the whole rate of the NCE scheme is $\mathcal{O}(1)$.

Acknowledgements

Z. Brakerski is supported by the Binational Science Foundation (Grant No. 2016726), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

P. Branco thanks the support from DP-PMI and FCT (Portugal) through the grant PD/BD/135181/2017. This work is supported by Security and Quantum

Information Group of Instituto de Telecomunicações, by the Fundação para a Ciência e a Tecnologia (FCT) through national funds, by FEDER, COMPETE 2020, and by Regional Operational Program of Lisbon, under UIDB/50008/2020.

N. Döttling: This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4).

S. Garg supported in part from AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, DARPA SIEVE Award, and research grants by the Sloan Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). 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 funding agencies.

References

1. Beaver, D.: Plug and play encryption. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO '97. pp. 75–89. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
2. Boyle, E., Gilboa, N., Ishai, Y.: Breaking the circuit size barrier for secure computation under DDH. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 509–539. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
3. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. pp. 407–437. Springer International Publishing, Cham (2019)
4. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. pp. 639–648. STOC '96, ACM, New York, NY, USA (1996), http://doi.acm.org/10.1145/237814.238015
5. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing. p. 494–503. STOC '02, Association for Computing Machinery, New York, NY, USA (2002), https://doi.org/10.1145/509907.509980
6. Canetti, R., Poburinnaya, O., Raykova, M.: Optimal-rate non-committing encryption. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 212–241. Springer International Publishing, Cham (2017)
7. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. pp. 287–302. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
8. Damgård, I., Nielsen, J.B.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. pp. 432–450. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
9. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)

10. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019. pp. 3–32. Springer International Publishing, Cham (2019)
11. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 129–147. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
12. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1374376.1374407
13. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. p. 218–229. STOC '87, Association for Computing Machinery, New York, NY, USA (1987), https://doi.org/10.1145/28395.28420
14. Guruswami, V., Sudan, M.: List decoding algorithms for certain concatenated codes. In: Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing. pp. 181–190. STOC '00, ACM, New York, NY, USA (2000), http://doi.acm.org/10.1145/335305.335327
15. Hemenway, B., Jafargholi, Z., Ostrovsky, R., Scafuro, A., Wichs, D.: Adaptively secure garbled circuits from one-way functions. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part III. Lecture Notes in Computer Science, vol. 9816, pp. 149–178. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016)
16. Hemenway, B., Ostrovsky, R., Richelson, S., Rosen, A.: Adaptive security with quasi-optimal rate. In: Kushilevitz, E., Malkin, T. (eds.) Theory of Cryptography. pp. 525–541. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
17. Hemenway, B., Ostrovsky, R., Rosen, A.: Non-committing encryption from ϕ-hiding. In: Dodis, Y., Nielsen, J.B. (eds.) Theory of Cryptography. pp. 591–608. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
18. Micali, S., Peikert, C., Sudan, M., Wilson, D.A.: Optimal error correction against computationally bounded noise. In: Kilian, J. (ed.) Theory of Cryptography. pp. 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
19. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 700–718. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
20. O'Neill, A., Peikert, C., Waters, B.: Bi-deniable public-key encryption. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science, vol. 6841, pp. 525–542. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)
21. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. pp. 84–93. STOC '05, ACM, New York, NY, USA (2005), http://doi.acm.org/10.1145/1060590.1060603
22. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982). pp. 160–164 (Nov 1982)
23. Yoshida, Y., Kitagawa, F., Tanaka, K.: Non-committing encryption with quasi-optimal ciphertext-rate based on the ddh problem. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 128–158. Springer International Publishing, Cham (2019)