Batch-OT with Optimal Rate

Zvika Brakerski[1], Pedro Branco[2], Nico Döttling[3], and Sihang Pu[3]

[1]Weizmann Institute of Science
[2]IT, IST University of Lisbon
[3]Helmholtz Center for Information Security (CISPA)

Abstract

We show that it is possible to perform n independent copies of 1-out-of-2 oblivious transfer in two messages, where the communication complexity of the receiver and sender (each) is $n(1 + o(1))$ for sufficiently large n. Note that this matches the information-theoretic lower bound. Prior to this work, this was only achievable by using the heavy machinery of rate-1 fully homomorphic encryption (Rate-1 FHE, Brakerski et al., TCC 2019).

To achieve rate-1 both on the receiver's and sender's end, we use the LPN assumption, with slightly sub-constant noise rate $1/m^\epsilon$ for any $\epsilon > 0$ together with either the DDH, QR or LWE assumptions. In terms of efficiency, our protocols only rely on linear homomorphism, as opposed to the FHE-based solution which inherently requires an expensive "bootstrapping" operation. We believe that in terms of efficiency we compare favorably to existing batch-OT protocols, while achieving superior communication complexity. We show similar results for Oblivious Linear Evaluation (OLE).

For our DDH-based solution we develop a new technique that may be of independent interest. We show that it is possible to "emulate" the binary group $\mathbb{Z}_2$ (or any other small-order group) inside a prime-order group $\mathbb{Z}_p$ *in a function-private manner*. That is, $\mathbb{Z}_2$ operations are mapped to $\mathbb{Z}_p$ operations such that the outcome of the latter do not reveal additional information beyond the $\mathbb{Z}_2$ outcome. Our encoding technique uses the discrete Gaussian distribution, which to our knowledge was not done before in the context of DDH.

1 Introduction

Oblivious Transfer (OT) [Rab05, EGL82] is one of the most basic cryptographic primitives. In the simple 1-out-of-2 OT, a receiver holds a bit $b \in \{0, 1\}$ and a sender holds two bits x_0, x_1. In the end of the protocol, the receiver should learn x_b, but nothing about x_{1-b}, and the sender should learn nothing about the value of b. In most applications, one OT is not enough and one is required to perform many OT operations in parallel. We let n denote the number of parallel executions. Various techniques have been developed to address this task of *batch-OT* [IKNP03, BCG+19b, BCG+19a]. For the most part, they involve a preprocessing "offline" phase where the parties generate random OT correlations.[1] Given such correlations, executing the OT protocol in the so-called "online phase" is computationally very simple. This approach is very useful for purposes of computational efficiency, since the offline phase can be carried out even before the actual inputs of the computation are known. However, in terms of communication complexity, there is an inherent cost, even just in the online phase, of n receiver bits and $2n$ sender bits. In contrast, the insecure implementation only requires n bits to be sent from each party in a two-message protocol: the receiver sends its input, and the sender returns all of the appropriate x_b values. As always in cryptography, we wish to understand what

[1]That is, a protocol in which the receiver obtains b, x_b and the sender obtains x_0, x_1, where b, x_0, x_1 are all (pseudo-)randomly sampled.

is the "cost of privacy", namely can we approach the information theoretic minimum without losing privacy. Note that we can only hope to achieve this for a sufficiently large n, due to the security parameter overhead.[2]

In prior work, Döttling et al. [DGI+19] showed that if the same receiver bit is used for multiple OT instances, then the sender's response can be compressed to $n(1+o(1))$, achieving an optimal amortized rate. This was shown under a variety of computational assumptions: Decisional Diffie-Hellman (DDH), Quadratic Residuosity (QR), or Learning with Errors (LWE). It was also shown by Brakerski et al. [BDGM19] and by Gentry and Halevi [GH19] that fully homomorphic encryption (FHE) can achieve optimal communication complexity, which in particular implies that under the LWE assumption, optimal rate batch-OT is achievable. However, the FHE-based protocol inherently requires the use of a computationally exorbitant "bootstrapping" mechanism in order to compress the receiver's message.

1.1 Our Contribution

We show that optimal-rate[3] batch-OT can be achieved from various computational assumptions, and without giving up on computational efficiency. In particular, we require the LPN assumption with a small-inverse-polynomial noise[4], in addition to one of the assumptions DDH, QR or LWE. In terms of computational cost, our protocol does not require heavy operations such as bootstrapping and relies on linear homomorphism only. We believe that in terms of overall cost it compares favorably even with random-OT based methods. All of our results are in the semi-honest (honest-but-curious) setting.

We further extend our results to the task of Oblivious Linear Evaluation (OLE) [IPS09, CDI+19, GNN17, BDM22], where the sender holds a linear function over a ring and the receiver holds an input for the function, and we wish for the receiver to learn the output on its input and nothing more, and the sender learns nothing as usual. OLE has been shown to be useful in various settings [GMW19, CDI+19].

Our techniques rely mostly on linear homomorphism, namely on the ability to evaluate linear functions on encrypted data (see Section 2 below). Notably, we require a linearly homomorphic scheme over $\mathbb{Z}_2$ (more generally $\mathbb{Z}_q$ for OLE) where the evaluation is *function-private*. Namely, the output ciphertext should not reveal any information about the linear function that was evaluated. This was not known to be achievable from DDH prior to this work, and we introduce a new technique that we believe may be of independent interest. The reason for this is that DDH works "natively" over the group $\mathbb{Z}_p$ where p is a super-polynomially large prime. Furthermore, we only have access to the $\mathbb{Z}_p$ elements in the exponent of a group generator g. Indeed, one can encode $0 \to g^0$, $1 \to g^1$, and linear $\mathbb{Z}_2$ homomorphism will follow in the sense that after applying a linear function in the exponent, we obtain g^x, where $x \pmod 2$ is the desired $\mathbb{Z}_2$ output. This creates two obstacles: first we need to be able to efficiently map $g^x \to x$, which means that x must come from a polynomially-bounded domain, and second that recovering x reveals more information than just $x \pmod 2$. We develop a new method to resolve this issue using *discrete Gaussian variables*. A technique that was used in the context of the LWE assumption but to the best of our knowledge not for DDH. We view this as an additional contribution of this work, which may find additional applications. In particular we show that it can be used to enhance the key-dependent-message security properties of the well-known encryption scheme [BHHO08].

For more details on all of our contributions, see the technical overview in Section 2.

1.2 Related Work

The communication complexity of OT has been extensively studied throughout the decades. Here we present a brief overlook of previous works.

[2] In more detail, since 2-message OT implies a public-key encryption scheme, the messages must have length that relates to the security parameter of the underlying computation assumption. This is the case even for single-bit OT.

[3] Achieving optimal rate (or any rate above 1/2) seems to involve a "phase-transition" and should be viewed as more than a "constant factor" improvement. For example, OT beyond this threshold implies the existence of lossy trapdoor functions (see discussion in [DGI+19], Section 6.3). Therefore one could expect such a protocol to inherently be heavier on public-key operations.

[4] This is still a regime where LPN alone is not known to imply public-key encryption.

OT from Pseudorandom Correlations. A recent line of research studies the feasibility of efficiently extending OTs in a *silent* manner [BCG+19b, BCG+19a]. In these works, a setup phase is performed to distributed some *shares* between the parties. These shares can later be expanded into random OT correlations. In the most efficient scheme [BCG+19a] the setup phase can be performed in just two rounds assuming just a pseudorandom generator and an OT scheme. Using this scheme for performing the setup together with the standard transformations form random OT to chosen-input OT, [BCG+19a] shows that n independent instances of OT for s-bit strings can be performed with communication complexity $(2s+1)n + o(n)$. For bit OT, this yields a communication complexity $3n + o(n)$ bits.

Download rate-1 OT. We say that an OT protocol has download rate 1 if the rate of the sender's message is asymptotically close to 1. OT protocols with download rate 1 were presented in [DGI+19, GHO20, CGH+21].However, these protocols do not achieve upload rate 1, that is, the rate of the receiver's message is far from being 1. Moreover, it is not clear how we can extend these protocols to achieve upload rate 1.

Using rate-1 FHE. As mentioned before, optimal-rate OT can be achieved using the recent scheme for rate-1 fully homomorphic encryption (FHE) of [BDGM19, GH19] together with (semi-honest) circuit-privacy techniques for FHE (e.g. [BdMW16]). However this can only be instantiated using LWE.

Laconic OT. Laconic OT [CDG+17, QWW18, GVW20, ABD+21] is a flavor of two-round OT where the first message sent by the receiver is sublinear (ideally polylogarithmically) in the size of its input. However, by a simple information-theoretical argument, the sender's message has size at least as large as the size of the sender's input. Note that, if this is not the case, then we would have an OT protocol with asymptotically better communication than an insecure OT protocol.

2 Technical Overview

2.1 Oblivious Transfer from Homomorphic Encryption

Our starting point is a textbook construction of oblivous transfer from simple homomorphic encryption schemes, such as ElGamal. For a cryptographic group $\mathbb{G} = \langle g \rangle$ of prime order p, recall that an ElGamal public key is of the form $\mathsf{pk} = (g, h = g^x) \in \mathbb{G}^2$, where $x \xleftarrow{\$} \mathbb{Z}_p$ is the secret key. Ciphertexts are of the form $c = (c_1, c_2) = (g^r, h^r \cdot g^b)$, where $r \xleftarrow{\$} \mathbb{Z}_p$ is uniformly random and $b \in \{0,1\}$ is the encrypted message. Given such a ciphertext c, the public key pk and two bits $m_0, m_1 \in \{0,1\}$, anyone can homomorphically compute a new ciphertext c' which is distributed identically to a fresh encryption of m_b, by *homomorphically evaluating* the linear function $f(x) = (1-x) \cdot m_0 + x \cdot m_1 = (m_1 - m_0) \cdot x + m_0$ on the ciphertext c and rerandomizing the resulting ciphertext. Note that if $b \in \{0,1\}$ is a bit, then it holds that $f(b) = m_b$. This homomorphic evaluation can be achieved by computing

$$\begin{aligned} c_1' &\leftarrow g^{r^*} \cdot c_1^{m_1 - m_0} \\ c_2' &\leftarrow h^{r^*} \cdot c_2^{m_1 - m_0} \cdot g^{m_0}, \end{aligned}$$

where $r^* \xleftarrow{\$} \mathbb{Z}_p$ is chosen uniformly random. Note that it holds that

$$\begin{aligned} c_1' &= g^{r^* + r \cdot (m_1 - m_0)} \\ c_2' &= h^{r^* + r \cdot (m_1 - m_0)} \cdot g^{(m_1 - m_0) \cdot b + m_0} = h^{r^* + r \cdot (m_1 - m_0)} \cdot g^{m_b}. \end{aligned}$$

Since $r^* \xleftarrow{\$} \mathbb{Z}_p$ is chosen uniformly random, it holds that $r' = r^* + r \cdot (m_1 - m_0)$ is distributed uniformly random and we can conclude that $c' = (c_1', c_2')$ is distributed identical to a fresh encryption of m_b. Since

c' does not reveal more than the function value $f(b) = m_b$, we call the above homomorphic evaluation procedure function private.

This immediately implies an OT protocol: An OT-receiver holding a choice-bit $b \in \{0,1\}$ generates a pair $(\mathsf{pk}, \mathsf{sk})$ of ElGamal public and secret keys, encrypts the bit b under pk and sends the resulting ciphertext to the OT-sender. The OT-sender, holding messages m_0, m_1, homomorphically computes a ciphertext c' encrypting m_b and sends c' back to the OT-receiver, who decrypts c' to m_b. Security against semi-honest senders follows from the IND-CPA security of ElGamal, whereas security against semi-honest receivers follows from the function privacy property established above.

2.2 Download-Rate Optimal String OT

While the above OT protocol is simple and efficient, it suffers from a very poor communication rate. While the receiver's message encrypts just a single bit, he needs to send 4 group elements, whereas the sender sends 2 group elements, each of size $\mathsf{poly}(\lambda)$.

Döttling et al. [DGI+19] proposed a compression technique for *batched ElGamal ciphertexts* based on the share-conversion technique of [BGI16]. A batched ElGamal ciphertext is of the form $\mathbf{c} = (c_0, c_1, \ldots, c_\ell) = (g^r, h_1^r \cdot g^{b_1}, \ldots, h_\ell^r \cdot g^{b_\ell})$, where $\mathsf{pk} = (g, h_1, \ldots, h_\ell)$ is the corresponding public key and $\mathsf{sk} = (s_1, \ldots, s_\ell)$ with $h_i = g^{s_i}$ is the secret key. The compression technique of [DGI+19] keeps c_0 compresses each of the $c_1, \ldots, c_\ell$ into just a single bit. The idea is instead of sending each $c_i \in \mathbb{G}$ (for $i \geq 1$) in full, to first compute the distance d to the next pseudorandom *break-point* in $\mathbb{G}$, and then only send its parity $d \mod 2$. The break points $\mathcal{P} \subseteq \mathbb{G}$ are the set of all points $h \in \mathbb{G}$ satisfying $\mathsf{PRF}_K(h) = 0^t$, where $\mathsf{PRF} : \mathbb{G} \to \{0,1\}^t$ is a pseudorandom function with a range of size $2^t = \mathsf{poly}(\lambda)$. Thus, the distance $d = d(c_i)$ of a group element c_i to the nearest break point is the smallest non-negative d such that $c_i \cdot g^d \in \mathcal{P}$. Given that neither c_i nor $c_i \cdot g^{-1}$ is a breakpoint, we can recover the bit b_i from $c_0 = g^r$, $\beta_= d(c_i) \mod 2$ and the secret key component s_i. It was shown in [BBD+20] that for a given ciphertext $c = (c_0, c_1, \ldots, c_\ell)$, the PRF-key K can be (efficiently) chosen such that all c_i are good, in the sense that neither c_i nor $c_i \cdot g^{-1}$ is a breakpoint. This ensures that a receiver can recover the $b_1, \ldots, b_\ell$ from $c' = (K, c_0, \beta_1, \ldots, \beta_\ell)$, where $\beta_i = d(c_i) \mod 2$. Since all the β_i are bits, such a compressed ciphertext only has additive size-overhead consisting of K, c_0. For a sufficiently large ℓ, this fixed overhead becomes insignificant and the ciphertext rate approaches 1.

The compressed batched ElGamal we've outlined leads to a batch bit-oblivious transfer protocol with *download-rate 1*: The receiver generates a key-pair pk, sk for batched ElGamal, and encrypts his choice-bits $b_1, \ldots, b_\ell$ into

$$\mathbf{c}_1 = \mathsf{Enc}_{\mathsf{pk}}(b_1, 0, \ldots, 0), \ldots, \mathbf{c}_\ell = \mathsf{Enc}_{\mathsf{pk}}(0, \ldots, 0, b_\ell),$$

i.e. $\mathbf{c}^{(i)}$ encrypts a vector which is b_i in index i and 0 everywhere else. The OT-receiver now sends $\mathsf{pk}, \mathbf{c}_1, \ldots, \mathbf{c}_\ell$ to the OT-sender, whose input are messages $(m_{1,0}, m_{1,1}), \ldots, (m_{\ell,0}, m_{\ell,1})$. Using circuit private homomorphic evaluation, the sender computes ciphertexts $\mathbf{c}'_1, \ldots, \mathbf{c}'_\ell$ encrypting $(m_{1,b_1}, 0, \ldots, 0)$, $\ldots$, $(0, \ldots, 0, m_{\ell,b_\ell})$. Homomorphically computing the sum of the ciphertexts $\mathbf{c}'_1, \ldots, \mathbf{c}'_\ell$, we obtain a ciphertext $\mathbf{c}'$ encrypting $(m_{1,b_1}, \ldots, m_{\ell,b_\ell})$. Finally, compressing $\mathbf{c}'$ with the compression technique outlined above we obtain a compressed ciphertext $\bar{\mathbf{c}} = (K, c_0, \beta_1, \ldots, \beta_\ell)$ which the OT-sender sends back to the OT-receiver, who can decrypt $(m_{1,b_1}, \ldots, m_{\ell,b_\ell})$.

Note that the size of the sender's message $\bar{\mathbf{c}}$ in this batch OT-protocol is $\mathsf{poly}(\lambda) + \ell$, which means that the *amortized communication cost* per bit-OT approaches 1 bit, and is therefore asymptotically optimal. Even in terms of concrete complexity this seems hard to beat, as the only additional information sent by the sender are the PRF key K and the ciphertext header c_0.

However, in terms of the upload rate, i.e. in terms of the size of the receiver's message, this protocol performs poorly. Specifically, to encrypt ℓ bits $b_1, \ldots, b_\ell$, the receiver needs to send ciphertexts $\mathbf{c}_1, \ldots, \mathbf{c}_\ell$ of total size $\ell^2 \cdot \mathsf{poly}(\lambda)$, which has a worse dependence on ℓ than just repeating the simple protocol from the last paragraph ℓ times.

Clearly, we need a mechanism to compress the receiver's message. Applying the same ElGamal compression technique as for the sender's message quickly runs into problems: Once an ElGamal ciphertext is compressed, the scheme loses its homomorphic capabilities, i.e. we cannot perform any further homomorphic

operations on compressed ciphertexts and currently we don't know if it is possible to publicly *decompress* such ciphertexts into "regular" ElGamal ciphertexts.

2.3 Our Approach: Recrypting the Receiver's Message

Instead, our approach will be to encrypt the receiver's message under a different encryption scheme, specifically one which achieves ciphertext rate approaching 1 but at the same time can be decrypted by the homomorphic capabilities of batched ElGamal. Specifically, the decryption procedure of this encryption scheme should be a linear function in the secret key. We can get an encryption scheme which almost fulfills these requirements from the Learning Parity with Noise (LPN) assumption. The LPN assumption states that for a random $m \times n$ matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_2^{m \times n}$, a random vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$ and a ρ-Bernoulli distributed [5] $\mathbf{e} \in \mathbb{Z}_2^m$, it holds that

$$(\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{u}),$$

where $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_2^m$ is chosen uniformly at random. This gives rise to the following simple symmetric-key encryption scheme with *approximate correctness*: Assume that $\mathbf{A}$ is a fixed public parameter, the secret key is a uniformly random $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$. To encrypt a message $m \in \mathbb{Z}_2^m$, we compute a ciphertext $\mathbf{d} \leftarrow \mathbf{A}\mathbf{s}+\mathbf{e}+\mathbf{m}$, where $\mathbf{e} \in \mathbb{Z}_2^m$ is chosen via a ρ-Bernoulli distribution. To decrypt such a ciphertext, we compute $\mathbf{m}' \leftarrow \mathbf{d} - \mathbf{A} \cdot \mathbf{s}$.

Note that this scheme is only approximately correct in the sense that it holds that $\mathbf{m}' = \mathbf{m} + \mathbf{e}$, i.e. in most coordinates $\mathbf{m}'$ is identical to $\mathbf{m}$, but only in few coordinates $\mathbf{m}'$ and $\mathbf{m}$ differ. Furthermore, one-time security of this encryption scheme follows from the LPN assumption.

The high level strategy to use this symmetric key encryption scheme is now as follows: Assume the matrix $\mathbf{A} \in \mathbb{Z}_2^{m \times n}$ is known to both the sender and the receiver. In the actual protocol this matrix will be chosen by the receiver, and the communication cost of sending $\mathbf{A}$ will be amortized by reusing $\mathbf{A}$ many times.

The OT-receiver chooses a symmetric key $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^n$ uniformly at random and encrypts his vector of choice bits $\mathbf{b} = (b_1, \ldots, b_\ell)$ to $\mathbf{d} = \mathbf{A}\mathbf{s} + \mathbf{e} + \mathbf{b}$ (where again, $\mathbf{e} \in \mathbb{Z}_2^\ell$ is ρ-Bernoulli distributed). Furthermore, the receiver will encrypt the LPN secret under ElGamal, i.e. he encrypts $\mathbf{s}$ to $\mathbf{c} = \mathsf{Enc}(\mathsf{pk}, \mathbf{s})$. For the moment, assume that $\mathbf{s}$ is encrypted bit-wise with standard ElGamal rather than batched ElGamal. The OT-receiver now sends the ElGamal public key pk and the ciphertexts $\mathbf{c}$ and $\mathbf{d}$ to the OT-sender.

Now, given these values, the sender can homomorphically decrypt the $\mathbf{d}$ into ElGamal, effectively *key-switching* from the ciphertext $\mathbf{d}$ into an ElGamal ciphertext. Concretely: The sender homomorphically evaluates the linear function $f(\mathbf{x}) = \mathbf{d} - \mathbf{A}\mathbf{x}$ on the ElGamal ciphertext $\mathbf{c} = \mathsf{Enc}(\mathsf{pk}, \mathbf{s})$. This produces an ElGamal encryption $\mathbf{c}'$ encrypting $f(\mathbf{s}) = \mathbf{d} - \mathbf{A}\mathbf{s} = \mathbf{b} + \mathbf{e} = \mathbf{b}'$. In other words, the OT-sender has now obtained an ElGamal encryption of a vector $\mathbf{b}'$ which agrees with $\mathbf{b}$ in most locations.

The high-level idea is now to let the OT-sender use this ciphertext $\mathbf{c}'$ as the encryption of the receiver's choice bits and proceed as in the ElGamal-based OT-protocol above. If we were to naively use $\mathbf{c}'$ in this way, the receiver would obtain the correct output m_{i,b_i} in locations where $\mathbf{b}$ and $\mathbf{b}'$ agree, but would get the wrong output $m_{i,1-b_i}$ in locations where $\mathbf{b}$ and $\mathbf{b}'$ disagree. While there certainly are applications in which a small amount of faulty locations are tolerable, in general this leads to insecure protocols.

There is, however, another issue with this approach. In this paragraph we have implicitly assumed that ElGamal is homomorphic for linear functions modulo 2. However, since the group we implement ElGamal over is of large prime order p, when we evaluate linear functions such as $f(\mathbf{x}) = \mathbf{d} - \mathbf{A}\mathbf{x}$ over a ciphertext encrypting a $\mathbf{s} \in \{0,1\}^n$, the result of this evaluation is *not* reduced modulo 2, and the resulting ciphertext in fact encrypts $f(\mathbf{s})$ as an integer. This does not cause major problems in terms of correctness, as this integer will still be small (at most of size m), and hence decryption will still be efficient.

However, this does cause major problems in terms of sender-privacy, as we can only guarantee sender privacy for receiver messages that are guaranteed to encrypt a bit $b \in \{0,1\}$.

For now, we will bypass this problem by relying on a homomorphic encryption scheme which is in fact homomorphic over $\mathbb{Z}_2$ (rather than $\mathbb{Z}_p$), offers function privacy for linear functions modulo 2 and is compatible

[5] i.e. every component of e_i of $\mathbf{e}$ is independently 0 with probability $1 - \rho$ and 1 with probability ρ

with ciphertext compression. Such an encryption can in fact be constructed from the Quadratic Residuosity assumption [DGI+19].

Another small issue we haven't addressed here is that the compression mechanisms for the sender and the receiver are somewhat orthogonal, in the sense that the sender's message is compressed by compressing a batched ElGamal ciphertext (which generally does not allow homomorphic evaluation across different components), whereas the receiver's compression strategy requires the homomorphic evaluation of linear functions with multiple (i.e. vector-valued) inputs. In the main body (Section 7) we will show a tradeoff which allows to reconcile these requirements, leading to a batch OT protocol with overall rate 1.

We will first discuss how to deal with the issue of errors in the key-switched ciphertext, and then return to the issue of implementing our approach with ElGamal instead of QR-based encryption.

2.4 Dealing with LPN Errors

To deal with the LPN errors in the key-switched ciphertext $\mathbf{c}'$, we will pursue the following high-level strategy: The sender will introduce an additional masking on the receiver's output, which can only be removed in error-free locations. This masking effectively erases the receiver's output in locations in which the receiver's output is corrupted.

To communicate the correct outputs in the locations with errors, the parties will rely on an additional protocol which is run in parallel. Given that the number of errors is sufficiently small, the communication cost of this additional protocol will be insubstantial and not affect the overall asymptotic rate.

We will first address the problem of erasing the receiver's output in corrupted locations. First observe that the receiver knows the locations with errors (i.e. the support of the error vector $\mathbf{e}$). Assume that the LPN error vector $\mathbf{e}$ has a fixed hamming weight $t \approx \rho m$, and note that hardness of fixed-weight LPN follows routinely from the hardness of Bernoulli LPN[6]. A t-puncturable pseudorandom function [BGI14, BCG+19b] is a pseudorandom function [GGM84] which supports punctured keys. That is, given a PRF key K and t inputs $x_1, \ldots, x_t$, we can efficiently compute a *punctured key* K' of size $t \cdot \mathsf{poly}(\lambda)$ which allows to evaluate the PRF on all inputs *except* $x_1, \ldots, x_t$. Furthermore, the key K' does not reveal the function values at $x_1, \ldots, x_t$, i.e. $\mathsf{PRF}(K, x_1), \ldots, \mathsf{PRF}(K, x_t)$ are pseudorandom given the punctured key K'.

The approach to erase the receiver's outputs in erroneous locations is now as follows. The sender chooses a PRF key K and masks both $m_{i,0}$ and $m_{i,1}$ with $\mathsf{PRF}(K, i)$, i.e. instead of using $(m_{i,0}, m_{i,1})$ as OT-inputs, he uses $m'_{i,0} = m_{i,0} \oplus \mathsf{PRF}(K, i)$ and $m'_{i,1} = m_{i,1} \oplus \mathsf{PRF}(K, i)$. Assuming that the sender can somehow communicate a punctured key K' which is punctured at the locations $i_1, \ldots, i_t$ of the errors (i.e. $\mathbf{e}_{i_j} = 1$ and $\mathbf{e}$ is 0 everywhere else), the receiver will be able to remove the mask from error-free locations by computing $m_{i,b_i} = m'_{i,b_i} \oplus \mathsf{PRF}(K', i)$. In the erroneous locations however, $m_{i,1-b_i}$ will be hidden from the view of the receiver as $\mathsf{PRF}(K, i)$ is pseudorandom even given the punctured key K'.

How can we communicate the punctured key K' to the receiver with small communication cost in such a way that the sender does not learn the error-locations $i_1, \ldots, i_t$? This could be achieved generically by relying on the punctured PRF construction of [BGI14] and transferring keys using a sublinear private information retrieval (PIR) scheme [CGKS95, DGI+19]. However, recently [BCG+19b] provided a protocol to achieve this task very efficiently via a two round protocol communicating only $t\mathsf{poly}(\lambda)$ bits. In the main body (Section 6), we will refer to this primitive as *co-PIR*, since effectively it allows to communicate a large pseudorandom database to a receiver except in a few locations chosen by the receiver.

Finally, to communicate the correct outputs to the receiver in the locations with errors, we will in fact rely on a two-message PIR scheme with polylogarithmic communication. Such schemes are known e.g. from LWE [BV11] and were recently constructed from a wide variety of assumptions [DGI+19], such as DDH and QR. The idea is as follows: For each error location i_j the receiver sends an additional OT message $OT_1(b_{i_j})$ using an off-the-shelf low-rate OT protocol (e.g. the basic ElGamal based protocol sketched above), as well as a PIR message $PIR_1(i_j)$. The sender speculatively completes this OT protocol for each index i (since the index i_j is not known to the sender), collects his OT responses in a database of size ℓ, runs the PIR sender algorithm on this database, and sends the response back to the receiver. The receiver will now be able to

[6] See e.g. [Döt15, BCG+19b]

recover the correct OT_2 message via PIR, complete the OT and recover $m_{i_j,b_{i_j}}$. We remark that for this protocol to be secure against semi-honest senders, we need a PIR protocol with sender privacy. However, e.g. the protocols provided in [DGI+19] readily have this feature.

Carefully putting all these components together, we obtain a batch bit-OT protocol with rate-1, for both the sender and the receiver.

2.5 Emulating Small Subgroups

We now return to the issue that ElGamal does not provide function privacy for linear functions modulo 2. Recall that the issue essentially boils down to the fact that the plaintext space of ElGamal is *natively* $\mathbb{Z}_p$, and when we encode messages in the least significant bits, i.e. encoding a bit b as g^b, then for all practical purposes homomorphic evaluations of linear functions with $\{0,1\}$ coefficients are over $\mathbb{Z}_2$, i.e. the resulting ciphertext encodes the result of the function evaluation *without* reduction modulo 2.

From an algebraic perspective, this problem is rooted in the fact that since p is prime, $\mathbb{Z}_p$ has no non-trivial subgroup, i.e. it just does not support modular reductions with respect to anything else than p.

To approach this problem, we will take inspiration from the domain of lattice cryptography [Reg05]. There, messages are typically encoded in the high order bits of group elements, i.e. to encode b in $\mathbb{Z}_p$, we would like to encoded it as $b \cdot \frac{p}{2}$. However, since p is odd, first have to round $\frac{p}{2}$ to the nearest integer in order to get a proper $\mathbb{Z}_p$ element, i.e. we encode b via $b \cdot \lceil \frac{p}{2} \rfloor$. If we could encode b with respect to $\frac{p}{2} \notin \mathbb{Z}_p$, we would get a subgroup of order 2, i.e. for bits $b, b' \in \{0,1\}$ it holds that

$$\left(b \cdot \frac{p}{2} + b' \cdot \frac{p}{2}\right) \bmod p = (b + b' \bmod 2) \cdot \frac{p}{2}.$$

However, once we round $\frac{p}{2}$ to the next integer, we get essentially the same problem as before: If we perform group operations on $b \lceil \frac{p}{2} \rfloor$ and $b' \lceil \frac{p}{2} \rfloor$, then the rounding errors start to accumulate information about b and b' which is cannot be obtained from $b + b' \bmod 2$. Specifically

$$\begin{aligned} b \left\lceil \frac{p}{2} \right\rfloor + b' \left\lceil \frac{p}{2} \right\rfloor \bmod p &= b \left(\frac{p}{2} + \frac{1}{2} \right) + b' \left(\frac{p}{2} + \frac{1}{2} \right) \bmod p \\ &= (b + b' \bmod 2) \frac{p}{2} + (b + b') \frac{1}{2} \bmod p. \end{aligned}$$

Thus, now the least significant bit of $b \lceil \frac{p}{2} \rfloor + b' \lceil \frac{p}{2} \rfloor \bmod p$ e.g. leaks if $b = b' = 1$, something which cannot be learned from $b + b' \bmod 2$.

Consequently, at first glance the idea of encoding a bit b in the "high-order" bits of a $\mathbb{Z}_p$ element seems ineffective. However, the lattice toolkit still has more to offer. In particular, in the context of sampling discrete gaussians from lattices, Peikert [Pei10] considered a technique called *randomized rounding*. The basic idea is, given a a real number $r \in \mathbb{R}$ to not always round to the same value e.g. $\lceil r \rceil$, but to sample a an integer z close to r. In [Pei10], this distribution is a discrete gaussian Z on $\mathbb{Z}$ centered at r, i.e. the expectation of Z is r. Such a discrete gaussian is parametrized by a gaussian parameter σ, which essentially controlls the standard deviation of the discrete gaussian. We denote Z by $\lceil r \rfloor_\sigma$.

Now, given any two $r, r' \in \mathbb{R}$ and $\sigma_1, \sigma_2 > \omega(\sqrt{\log(\lambda)})$ (more generally the *smoothing parameter* of $\mathbb{Z}$), Peikert [Pei10] shows that

$$\lceil r \rfloor_{\sigma_1} + \lceil r' \rfloor_{\sigma_2} \approx_s \lceil r + r' \rfloor_{\sqrt{\sigma_1^2 + \sigma_2^2}}.$$

In other words, while $\lceil r \rfloor_{\sigma_1} + \lceil r' \rfloor_{\sigma_2}$ and $\lceil r + r' \rfloor_{\sqrt{\sigma_1^2+\sigma_2^2}}$ are note the same, they are statistically close. This means that anything that can be learned from $\lceil r \rfloor_{\sigma_1} + \lceil r' \rfloor_{\sigma_2}$ could have as well been learned from $\lceil r + r' \rfloor_{\sqrt{\sigma_1^2+\sigma_2^2}}$! While this comes at the expense of an increase "error" term with parameter $\sqrt{\sigma_1^2 + \sigma_2^2}$, this *additive error* is very small (of size approx σ) controlling the growth of this error term can be handled by standard techniques.

Returning to our goal of emulating small subgroups in $\mathbb{Z}_p$, our approach follows almost instantly: Instead of encoding a bit $b \in \mathbb{Z}_2$ as $b \cdot \lceil \frac{p}{2} \rceil$, we will encode it as $\lceil b \cdot \frac{p}{2} \rfloor_\sigma$ (for a $\sigma > \omega(\sqrt{\log(\lambda)})$). For $b, b' \in \{0,1\}$ this ensures that

$$\left\lceil b \cdot \frac{p}{2} \right\rfloor_\sigma + \left\lceil b' \cdot \frac{p}{2} \right\rfloor_\sigma \mod p \approx_s \left\lceil (b + b' \mod 2) \cdot \frac{p}{2} \right\rfloor_{\sqrt{2}\sigma} \mod p.$$

Thus, we have ensured that $\lceil b \cdot \frac{p}{2} \rfloor_\sigma + \lceil b' \cdot \frac{p}{2} \rfloor_\sigma \mod p$ does not leak more information than $b + b' \mod 2$.

Function-Private Evaluation for ElGamal We will now briefly discuss how this idea leads to a modulo 2 function private homomorphic evaluation procedure for ElGamal. Say we have two ElGamal ciphertexts $c_1 = (g^{r_1}, h^{r_1} \cdot g^{b_1})$ and $c_2 = (g^{r_2}, h^{r_2} \cdot g^{b_2})$ for a public key $\mathsf{pk} = (g, h)$ and we want to homomorphically evaluate the function $f(x_1, x_2) = a_1 x_1 + a_2 x_2 \mod 2$ (for $a_1, a_2 \in \{0,1\}$) on this pair of ciphertexts. In the first step, we *randomly encode* the function f as

$$f'(x_1, x_2) = x_1 \cdot \left\lceil a_1 \frac{p}{2} \right\rfloor_\sigma + (1 - x_1) \cdot \lceil 0 \rfloor_\sigma + x_2 \cdot \left\lceil a_2 \frac{p}{2} \right\rfloor_\sigma + (1 - x_2) \cdot \lceil 0 \rfloor_\sigma ,$$

noting that this is still a linear function (chosen from a distribution). Homomorphically evaluating f' on the ciphertexts c, c_2 we obtain a ciphertext c' encrypting

$$\begin{aligned} f'(b_1, b_1) &= b_1 \cdot \left\lceil a_1 \frac{p}{2} \right\rfloor_\sigma + (1 - b_1) \cdot \lceil 0 \rfloor_\sigma + b_1 \cdot \left\lceil a_2 \frac{p}{2} \right\rfloor_\sigma + (1 - b_1) \cdot \lceil 0 \rfloor_\sigma \\ &= \left\lceil b_1 a_1 \frac{p}{2} \right\rfloor_\sigma + \left\lceil b_1 a_2 \frac{p}{2} \right\rfloor_\sigma \\ &\approx_s \left\lceil (b_1 a_1 + b_1 a_2 \mod 2) \frac{p}{2} \right\rfloor_{\sqrt{2}\sigma} . \end{aligned}$$

In other words, this ciphertext could have been simulated knowing only the function result $f(b_1, b_1) = b_1 a_1 + b_1 a_2 \mod 2$, establishing that this homomorphic evaluation procedure is function private.

One aspect to note is that while the messages b_1, b_1 are encoded in c_1, c_2 in the "low-order-bits" via g^{b_1} and g^{b_2}, the function result $f(b_1, b_2)$ encrypted in c' is encoded in the high order bits, i.e. it is encoded as $\approx g^{f(b_1, b_2)\frac{p}{2}}$. This makes it necessary to change the decryption procedure: Let $c' = (c'_1, c'_2)$ and s be the secret key. To decrypt c' we compute $f = c'_2 \cdot (c'_1)^{-s} \approx_s g^{\lceil f(s_1, s_2) \cdot \frac{p}{2} \rfloor}$, we test if f is close to $g^0 = 1$ or $g^{\lceil p/2 \rceil}$. This recovers $f(s_1, s_2)$, as the error introduced by the rounding operation is of size at most $\mathsf{poly}(\lambda)$ via standard gaussian tail bounds.

Finally, we remark this this "high-order-bit" encoding is still compatible with ElGamal ciphertext compression, i.e. we can still compress homomorphically evaluated batch ElGamal ciphertexts down asymptotically optimal size, using a slightly different compression mechanism. This mechanism is discussed in Section 5. We expect this technique to have additional applications. As one immediate application, it allows to upgrade the key-dependent message secure encryption scheme of Boneh et al. [BHHO08] to support arbitrary linear functions modulo 2.

3 Preliminaries

The acronym PPT denotes "probabilistic polynomial time". Throughout this work, λ denotes the security parameter. By $\mathsf{negl}(\lambda)$, we denote a negligible function in λ, that is, a function that vanishes faster than any inverse 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 denote by $S[i]$ the i-th element of S (where the elements are ordered by ascending order except when explicitly stated otherwise)

For two probability distributions X, Y, we use the notation $X \approx_s Y$ to state that the distributions are statistically indistinguishable.

For two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$ over a finite field $\mathbb{F}$, we denote by $\mathbf{u} \odot \mathbf{v}$ their component-wise multiplication. We denote by $\mathsf{Supp}(\mathbf{u})$ the support of $\mathbf{u}$, that is, the set of indices where $\mathbf{u}$ is different from 0.[7] For $S \subseteq [n]$, $\mathbf{u}_S$ denotes the vector $(\mathbf{u}_i)_{i \in S}$. Finally, $\mathbf{u}^T$ denotes the transpose of $\mathbf{u}$ and $\mathsf{hw}(\mathbf{u})$ denotes the hamming weight of $\mathbf{u}$ (that is, the number of coordinates of $\mathbf{u}$ different from 0).

3.1 UC Security

In terms of security, we work in the standard UC-framework [Can01]. Let $\mathcal{F}$ be a functionality, π a protocol that implements $\mathcal{F}$ and $\mathcal{E}$ be a environment, an entity that oversees the execution of the protocol in both the real and the ideal worlds. Let $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$ be a random variable that represents the output of $\mathcal{E}$ after the execution of $\mathcal{F}$ with adversary Sim. Similarly, let $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{E}}$ be a random variable that represents the output of $\mathcal{E}$ after the execution of π with adversary $\mathcal{A}$.

In this work, we only consider semi-honest adversaries.

Definition 1. *A protocol* π implements $\mathcal{F}$ *if for every PPT adversary* $\mathcal{A}$ *there is a PPT simulator* Sim *such that for all PPT environments* $\mathcal{E}$*, the distributions* $\mathsf{IDEAL}_{\mathcal{F},\mathsf{Sim},\mathcal{E}}$ *and* $\mathsf{REAL}_{\pi,\mathcal{A},\mathcal{E}}$ *are computationally indistinguishable.*

3.2 Learning Parity with Noise

The LPN assumption is closely related to the problem of decoding a random linear code. Informally, it states that it is hard to find a solution for a noisy system of linear equations over $\mathbb{Z}_2$.

Definition 2 (LPN assumption)**.** *Let* $n, m, t \in \mathbb{N}$ *such that* $n \in \mathsf{poly}(\lambda)$ *and let* $\chi_{m,t}$ *be uniform distribution over the set of error vectors of size* m *and hamming weight* t*. The* Learning Parity with Noise *(LPN) assumption* $\mathsf{LPN}(n, m, \rho)$ *holds if for any PPT adversary* $\mathcal{A}$ *we have that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\, \{0,1\}^{n \times m} \\ \mathbf{s} \leftarrow\!\!\$\, \{0,1\}^{n} \\ \mathbf{e} \leftarrow\!\!\$\, \chi_{m,t} \end{array} \right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{y}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\, \{0,1\}^{n \times m} \\ \mathbf{y} \leftarrow\!\!\$\, \{0,1\}^{m} \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

where $\rho = m/t$ *(*ρ *is called the noise rate).*

In this work, we assume that the noise rate ρ is $m^{1-\varepsilon}$ for any constant $\varepsilon > 0$. The LPN assumption is believed to be hard for that noise rate (see e.g. [BCG+19a] and references therein).

LPN over larger fields. Following [BCG+19a, JLS21], we define the LPN assumption over larger fields $\mathbb{Z}_q$ where $q > 2$ is a prime number. In the following, let $\chi_{m,t,q}$ be the uniform distribution over $\{\mathbf{v} \in \mathbb{Z}_q^m : \mathsf{hw}(\mathbf{v}) = t\}$. In other words, $\chi_{m,t,q}$ is the uniform distribution over the set of vectors in $\mathbb{Z}_q$ which have $m - t$ null-coordinates.

Definition 3 (LPN over larger fields assumption)**.** *Let* $n, m, t, q \in \mathbb{N}$ *such that* $n \in \mathsf{poly}(\lambda)$ *and* q *is a prime number, and let* $\chi_{m,t,q}$ *be as above. The* LPN over larger fields *assumption* $\mathsf{LPN}(n, m, \rho, q)$ *holds if for any PPT adversary* $\mathcal{A}$ *we have that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{sA} + \mathbf{e}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{n \times m} \\ \mathbf{s} \leftarrow\!\!\$\, \mathbb{Z}_q^{n} \\ \mathbf{e} \leftarrow\!\!\$\, \chi_{m,t,q} \end{array} \right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{y}) : \begin{array}{c} \mathbf{A} \leftarrow\!\!\$\, \mathbb{Z}_q^{n \times m} \\ \mathbf{y} \leftarrow\!\!\$\, \mathbb{Z}_q^{m} \end{array} \right] \right| \leq \mathsf{negl}(\lambda)$$

where $\rho = m/t$*.*

[7] If there is only one index different from zero, $\mathsf{Supp}(\mathbf{u})$ denotes this index.

3.3 Cryptographic Primitives

We now present several cryptographic primitives that will be needed in our constructions. Namely, we present the definitions of puncturable PRFs, private information retrieval and oblivious transfer.

3.3.1 Puncturable Pseudorandom Functions

A pseudorandom function (PRF) is a pair of functions $\mathsf{KeyGen}, \mathsf{Eval}$ where $\mathsf{Eval} : \mathcal{K} \times \{0,1\}^\alpha \to \{0,1\}^\beta$ (for some $\alpha, \beta = \mathsf{poly}(\lambda)$) is computed by a deterministic polynomial time algorithm: On input $(\mathsf{K}, \mathbf{x}) \in \mathcal{K} \times \{0,1\}^\alpha$ the algorithm outputs $\mathsf{Eval}(\mathsf{K}, \mathbf{x}) = \mathbf{y} \in \{0,1\}^\beta$. In terms of security, the value $\mathbf{y}$ is pseudorandom.

Puncturable pseudorandom functions (PPRFs) [BW13, KPTZ13, BGI14] are a special case of PRFs where a punctured key allows one to evaluate the PRF at all points except one.

Definition 4 (Puncturable PRF)**.** *Let $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$ be two polynomials. A puncturable PRF (PPRF) scheme $\mathsf{PPRF}_{\alpha,\beta} = \mathsf{PPRF}$ is composed by the following algorithms:*

- $\mathsf{KeyGen}(1^\lambda)$ *takes as input a security parameter λ. It outputs a key K.*
- $\mathsf{Eval}(\mathsf{K}, \mathbf{x})$ *takes as input a key K and $x \in \{0,1\}^\alpha$. It outputs $\mathbf{y} \in \{0,1\}^\beta$.*
- $\mathsf{Punct}(\mathsf{K}, S)$ *takes as input a key K and a subset $S \subseteq \{0,1\}^\alpha$. It outputs a punctured key K_S.*
- $\mathsf{EvalPunct}(\mathsf{K}_S, \mathbf{x})$ *takes as input a punctured key K_S and $\mathbf{x} \in \{0,1\}^\alpha$. It outputs $\mathbf{y} \in \{0,1\}^\beta$.*

Definition 5 (Correctness)**.** *A PPRF scheme PPRF is said to be correct if for all $\lambda \in \mathbb{N}$, for all $S \subseteq (\{0,1\}^\alpha)^t$ (for $t = \mathsf{poly}(\lambda)$), all $\mathbf{x} \notin S$ we have that*

$$\Pr\left[\mathsf{Eval}(\mathsf{K}, \mathbf{x}) = \mathsf{EvalPunct}(\mathsf{K}_S, \mathbf{x}) : \begin{array}{c} \mathsf{K} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{K}_S \leftarrow \mathsf{Punct}(\mathsf{K}, S) \end{array}\right] = 1.$$

Definition 6 (Pseudorandomness)**.** *A PPRF scheme PPRF is said to be pseudorandom at punctured points if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have that*

$$\left| \Pr\left[1 \leftarrow \mathcal{A}_2(\mathsf{K}_S, S, T, \mathsf{aux}) : \begin{array}{c} (S, \mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda);\ \mathsf{K} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{K}_S \leftarrow \mathsf{Punct}(\mathsf{K}, S);\ T \leftarrow \mathsf{Eval}(\mathsf{K}, S) \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}_2(\mathsf{K}_S, S, T, \mathsf{aux}) : \begin{array}{c} (S, \mathsf{aux}) \leftarrow \mathcal{A}_1(1^\lambda);\ \mathsf{K} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{K}_S \leftarrow \mathsf{Punct}(\mathsf{K}, S);\ T \leftarrow\$ \{0,1\}^{\beta|S|} \end{array}\right] \right| \leq \mathsf{negl}(\lambda).$$

PPRFs can be built solely based on any length-doubly pseudorandom generators (PRG)[8] via (a variant of) the tree-based construction of [GGM86]. Throughout this work, we call the term GGM-PPRF to this scheme and denote it by $\mathsf{PPRF}_{\mathsf{GGM}}$.

3.3.2 Private Information Retrieval

Private Information Retrieval (PIR) schemes[CGKS95] allow a user to retrieve the i-th bit of an n-bit database, without revealing to the database holder the value of i. Besides, we require an additional privacy property in our schemes: sender privacy (or data privacy)[DMO00].

Definition 7 (PIR)**.** *A private information retrieval (PIR) scheme PIR is composed by the following algorithms:*

- $\mathsf{Query}(n, i)$ *takes as input an index $i \in [n]$. It outputs a query q and a state st_i.*
- $\mathsf{Send}(DB, \mathsf{q})$ *takes as input a database $DB \in \{0,1\}^n$ and a message q. It outputs a response r.*
- $\mathsf{Retrieve}(\mathsf{r}, \mathsf{st}_i)$ *takes as input a response r and a state st_i. It retrieves the entry DB_i.*

[8] Which in turn, can be based on LWE, DDH or QR assumptions.

Definition 8 (Correctness)**.** *A PIR scheme* PIR *is said to be correct if for any* $n \in \mathbb{N}$, $DB \in \{0,1\}^n$ *and* $i \in [n]$*, we have that*

$$\Pr\left[DB_i = \mathsf{Retrieve}(\mathsf{st_i}, \mathsf{r}) : \begin{array}{c}(\mathsf{st_i}, \mathsf{q}) \leftarrow \mathsf{Query}(n, i) \\ \mathsf{r} \leftarrow \mathsf{Send}(DB, \mathsf{q})\end{array}\right] = 1.$$

Definition 9 (User privacy)**.** *A PIR scheme* PIR *is said to be user private if for any PPT adversary* $\mathcal{A}$*, any* $n, \lambda \in \mathbb{N}, DB \in \{0,1\}^n$ *and* $i, j \in [n]$*, we have that*

$$\left|\begin{array}{c}\Pr[1 \leftarrow \mathcal{A}(1^\lambda, DB, \mathsf{q_i}) : (\mathsf{st_i}, \mathsf{q_i}) \leftarrow \mathsf{Query}(\mathsf{n}, \mathsf{i})] - \\ \Pr[1 \leftarrow \mathcal{A}(1^\lambda, DB, \mathsf{q_j}) : (\mathsf{st_j}, \mathsf{q_j}) \leftarrow \mathsf{Query}(\mathsf{n}, \mathsf{j})]\end{array}\right| \leq \mathsf{negl}(\lambda).$$

Definition 10 (Sender privacy)**.** *A PIR scheme* PIR *is said to be sender private if for any* $\lambda \in \mathbb{N}$*, any* $n = \mathsf{poly}(\lambda)$*, any* $i \in [n]$ *and any two databases* $DB^x, DB^y \in \{0,1\}^n$ *such that* $DB^x_i = DB^y_i$ *we have that for all PPT adversaries* $\mathcal{A}$

$$\left|\begin{array}{c}\Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, i, n, \mathsf{st}_i, \mathsf{r}_i) : \begin{array}{c}(\mathsf{st}_i, \mathsf{q}_i) \leftarrow \mathsf{Query}(n, i) \\ \mathsf{r}_i \leftarrow \mathsf{Send}(DB^x, \mathsf{q}_i)\end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, i, n, \mathsf{st}_i, \mathsf{r}_i) : \begin{array}{c}(\mathsf{st}_i, \mathsf{q}_i) \leftarrow \mathsf{Query}(n, i) \\ \mathsf{r}_i \leftarrow \mathsf{Send}(DB^y, \mathsf{q}_i)\end{array}\right]\end{array}\right| \leq \mathsf{negl}(\lambda).$$

Black-box constructions for PIR exist LWE, DDH or QR assumptions [DGI+19].

3.3.3 Distributed GGM-PPRF Correlation

Let $\mathsf{PPRF_{GGM}} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{EvalPunct})$ be the GGM-PPRF scheme based on [GGM86]. The distributed GGM-PPRF correlation functionality [BCG+19a] considers two parties: A receiver with input $\alpha \in \{0,1\}^\ell$ and a sender with input $\beta \in \mathbb{F}_{p^r}$ and a GGM-PPRF key K. The functionality outputs a punctured key K_α and a hardwired value $\beta - \mathsf{PPRF.Eval}(\mathsf{K}, \alpha)$ to the receiver. We now present the formal definition of the functionality.

Distributed GGM-PPRF correlation functionality. The functionality $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ is parametrized by integers $\ell, p, r \in \mathbb{N}$. Moreover, let $\mathsf{PPRF_{GGM}} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{EvalPunct})$ be the GGM PPRF scheme with input space $\{0,1\}^\ell$ and output space $\mathbb{F}_{p^r}$. The functionality works as follows:

- **Receiver phase.** R sends α to $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ where $\alpha \in \{0,1\}^\ell$.
- **Sender phase.** S sends (β, K) to $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ where $\beta \in \mathbb{F}_{p^r}$ and $\mathsf{K} \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$. $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ sends $\mathsf{K}_\alpha \leftarrow \mathsf{PPRF.Puncture}(\mathsf{K}, \alpha)$ and $\gamma \leftarrow \beta - \mathsf{PPRF.Eval}(\mathsf{K}, \alpha)$ to R.

A protocol that implements the functionality $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ is presented in [BCG+19a]. The protocol uses a pseudorandom generator (PRG) and an oblivious transfer (OT)[9] protocol in a black-box way. Moreover, security is proven against semi-honest adversaries. Finally, the protocol presented runs in two rounds (assuming that the OT runs in two rounds) and achieves communication complexity of $\mathsf{poly}(\lambda, \ell)$.

For convenience, we will denote such a protocol by $\mathsf{PPRF\text{-}GGM} = (\mathsf{PPRF\text{-}GGM.R}_1, \mathsf{PPRF\text{-}GGM.S}, \mathsf{PPRF\text{-}GGM.R}_2)$ where:

- $\mathsf{PPRF\text{-}GGM.R}_1(\alpha)$ receives as input $\alpha \in \{0,1\}^\ell$. It outputs a message $\mathsf{pprf\text{-}ggm}_1$ and a state st.
- $\mathsf{PPRF\text{-}GGM.S}(\beta, \mathsf{K}, \mathsf{pprf\text{-}ggm}_1)$ receives as input $\beta \in \mathbb{F}_{p^r}$, a key $\mathsf{K} \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$ and a message $\mathsf{pprf\text{-}ggm}_1$. It outputs $\mathsf{pprf\text{-}ggm}_2$.
- $\mathsf{PPRF\text{-}GGM.R}_2(\mathsf{st}, \mathsf{pprf\text{-}ggm}_2)$ receives as input a state st and a message $\mathsf{pprf\text{-}ggm}_2$. It outputs a punctured key K_α and a value $\gamma \in \mathbb{F}_{p^r}$.

Using the two-round OT scheme of [PVW08], we can obtain a black-box construction for distributed GGM-PPRF correlation scheme under the LWE, DDH or QR assumptions.

[9] The OT protocol is not required to have overall rate 1.

3.3.4 Two-message Oblivious Transfer

In this work, we consider the two-message oblivious transfer (OT) with overall (almost) optimal rate; where the sender has input $(m_0, m_1) \in \{0,1\}^2$ and receiver a choice bit $b \in \{0,1\}$. At the end, the receiver learns the bit m_b and nothing else; the sender learns nothing about b. We define the OT in plain model as follows.

Definition 11 (Two-message OT)**.** *A two-message OT protocol between a sender and a receiver can be defined as a tuple of three PPT algorithms* $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$*. Let* λ *be the security parameter and* $k = \mathsf{poly}(\lambda)$*. The receiver computes* $(\mathsf{ot}_1, \mathsf{st}) \leftarrow \mathsf{OTR}(1^\lambda, \mathbf{b})$ *with his input* $\mathbf{b} = (b_1, \ldots, b_k) \in \{0,1\}^k$ *and sends* ot_1 *to the sender. The sender computes* $\mathsf{ot}_2 \leftarrow \mathsf{OTS}(1^\lambda, \mathsf{ot}_1, (\mathbf{m}_0, \mathbf{m}_1))$ *where* $(\mathbf{m}_0, \mathbf{m}_1) = ((m_{0,1}, \ldots, m_{0,k})(m_{1,1}, \ldots, m_{1,k}) \in (\{0,1\}^k)^2$ *and sends to the receiver* ot_2*. At the end, the receiver decodes the message to get* $\mathbf{m_b} = (m_{b_1,1}, \ldots, m_{b_k,k}) \leftarrow \mathsf{OTD}(\mathsf{ot}_2, \mathsf{st})$*.*

In terms of security, OT should implement the following functionality.

OT functionality. The functionality $\mathcal{F}_{\mathsf{OT}}$ is parametrized by a integer $k = \mathsf{poly}(\lambda)$ and works as follows:

- **Receiver phase.** R sends $\mathbf{b}$ to $\mathcal{F}_{\mathsf{OT}}$ where $\mathbf{b} \in \{0,1\}^k$.
- **Sender phase.** S sends $(\mathbf{m}_0, \mathbf{m}_1)$ to $\mathcal{F}_{\mathsf{OT}}$ where $\mathbf{m}_0, \mathbf{m}_1 \in \{0,1\}^k$. $\mathcal{F}_{\mathsf{OT}}$ sends $\{\mathbf{m}_{b_i,i}\}_{i\in[k]}$ to R.

3.4 Lattices and Gaussians

We now review some basic notions of lattices and gaussian distributions.

Let $\mathbf{B} \in \mathbb{R}^{k\times n}$ be a matrix. We denote the lattice generated by $\mathbf{B}$ by $\Lambda = \Lambda(\mathbf{B}) = \{\mathbf{xB} : \mathbf{x} \in \mathbb{Z}^k\}$.[10] The dual lattice Λ^* of a lattice Λ is defined by $\Lambda^* = \{\mathbf{x} \in \mathbb{R}^n : \forall y \in \Lambda, \mathbf{x}\cdot\mathbf{y} \in \mathbb{Z}\}$. It holds that $(\Lambda^*)^* = \Lambda$. The orthogonal lattice $\Lambda_q^\perp$ is defined by $\{\mathbf{y} \in \mathbb{Z}_q^n : \mathbf{A}\mathbf{y}^T = 0 \mod q\}$.

Let $\rho_s(\mathbf{x})$ be probability distribution of the Gaussian distribution over $\mathbb{R}^n$ with parameter s and centered in 0. We define the discrete Gaussian distribution $D_{S,s}$ over S and with parameter s by the probability distribution $\rho_s(\mathbf{x})/\rho(S)$ for all $\mathbf{x} \in S$ (where $\rho_s(S) = \sum_{\mathbf{x}\in S}\rho_s(\mathbf{x})$).

For $\varepsilon > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ of a lattice Λ is the least real $\sigma > 0$ such that $\rho_{1/\sigma}(\Lambda^* \setminus \{0\}) \leq \varepsilon$ [MR04].

Lemma 1 ([Ban93])**.** *For all* $\alpha \in \mathbb{R}$*,* $\|\mathbf{x}\| \leq \alpha\sqrt{n}$ *for* $\mathbf{x} \leftarrow\!\!\$\, D^n_{\mathbb{Z},\alpha}$*, except with negligible probability in* n*.*

We will make use of the following convolution property of discrete gaussians.

Lemma 2 ([GMPW20], Corollary 4.8)**.** *Let* $\Lambda_1, \Lambda_2 \subseteq \mathbb{R}^n$ *be lattices, let* $\sigma_1, \sigma_2 > 0$ *be such that* $1/\sqrt{1/\sigma_1^2 + 1/\sigma_2^2} > \eta_\epsilon(\Lambda_1 \cap \Lambda_2)$ *for some* $\epsilon = \mathsf{negl}(\lambda)$*. Then it holds for all* $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ *that* $D_{\Lambda_1+\mathbf{a},\sigma_1} + D_{\Lambda_2+\mathbf{b},\sigma_2}$ *is statistically close to* $D_{\Lambda_1+\Lambda_2+\mathbf{a}+\mathbf{b},\sqrt{\sigma_1^2+\sigma_2^2}}$*.*

We just need the following simple corollary of Lemma 2, which can be obtained by setting $\Lambda_1 = \Lambda_2 = \mathbb{Z}$.

Corollary 1. *Let* σ_1, σ_2*,* $\sigma_3 = \sqrt{\sigma_1^2 + \sigma_2^2}$ *be such that* $\sigma_1\sigma_2/\sigma_3 > \eta_\epsilon(\mathbb{Z})$ *for a negligible* ϵ *and let* $a, b \in \mathbb{Z}$*. Then* $D_{\mathbb{Z}+a,\sigma_1} + D_{\mathbb{Z}+b,\sigma_2}$ *and* $D_{\mathbb{Z}+a+b,\sigma_3}$ *are statistically close.*

Gadget matrix. For given parameters $n, q \in \mathbb{Z}$, let $\mathbf{g}$ be the vector $\left(1, 2, 2^2, \ldots, 2^{\lceil \log q\rceil - 1}\right)$ and $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n$ where $\mathbf{I}_n$ is the identity matrix of size n. The matrix $\mathbf{G}$ is usually called the gadget matrix [MP12].

Moreover, let $\bar{\mathbf{G}}_n = \sum_i \mathbf{G}_i \in \mathbb{Z}^{n\times\lceil\log q\rceil}$ where $\mathbf{G}_i$ is the matrix which is zero everywhere but its i-th row is $\mathbf{g}$.

The function $\mathbf{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^m$, where $m = \lceil \log q\rceil$, receives a value $v \in \mathbb{Z}_q$ and outputs its binary decomposition. Note that $\mathbf{g}\cdot\mathbf{g}^{-1}(v) = v \mod q$. Following [BdMW16], we define $\mathbf{g}^{-1}_{\mathsf{rnd}}$ to be the function that, on input $v \in \mathbb{Z}_q$, outputs $\mathbf{x} \leftarrow\!\!\$\, D_{\Lambda_q^\perp(\mathbf{g})+\mathbf{g}^{-1}(v),r}$, where $r = \tilde{\mathcal{O}}(1)$. It holds that $\mathbf{g}\cdot\mathbf{g}^{-1}_{\mathsf{rnd}}(v) = v \mod q$.

[10] The matrix $\mathbf{B}$ is called a basis of $\Lambda(\mathbf{B})$.

4 Compression-friendly Subgroup Emulation via Gaussian Rounding

We will now provide our new subgroup emulation technique. We first define the gaussian rounding functionality.

Definition 12. *Let $\sigma > 0$. For any $x \in \mathbb{R}$, the gaussian rounding $\lceil x \rfloor_\sigma$ is a random variable supported on $\mathbb{Z}$ defined by*

$$\lceil x \rfloor_\sigma = x + D_{\mathbb{Z}-x,\sigma}.$$

In other words, $\lceil x \rfloor_\sigma$ is a discrete gaussian centered on $x \in \mathbb{R}$ but supported on $\mathbb{Z}$.

We will use the following convolution lemma which provides a *simulation property* for gaussian rounding.

Lemma 3. *Let $\epsilon > 0$ be bounded by a sufficiently small constant and let $\sigma_1, \sigma_2 \geq \eta_\epsilon(\mathbb{Z})$. Then it holds for all $x, y \in \mathbb{R}$ that*

$$\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} \approx_s \lceil x + y \rfloor_{\sqrt{\sigma_1^2+\sigma_2^2}}.$$

It immediately follows from Lemma 3 that it holds for every integer $p \geq 2$ that

$$\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} \mod p \approx_s \lceil x + y \rfloor_{\sqrt{\sigma_1^2+\sigma_2^2}} \mod p.$$

Proof. The lemma follows routinely from Corollary 1, by definition of $\lceil \cdot \rfloor_\sigma$ it holds that

$$\begin{aligned}\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} &= x + y + D_{\mathbb{Z}-x,\sigma_1} + D_{\mathbb{Z}-y,\sigma_2} \\ &\approx_s x + y + D_{\mathbb{Z}-x-y,\sigma_3} \\ &= \lceil x + y \rfloor_{\sigma_3}.\end{aligned}$$

□

Lemma 4. *Let $p > q \geq 2$ be integers with $q \leq 2^k$, and let $\sigma > \eta_\epsilon(\mathbb{Z})$ for a negligible ϵ. Let $f : \mathbb{Z}_q^n \to \mathbb{Z}_q$ be given by $f(x_1, \ldots, x_n) = \sum_{i=1}^n a_i x_i + c$ for $a_1, \ldots, a_n, c \in \mathbb{Z}_q$. Define the randomized function $\hat{f} : \{0,1\}^{nk} \to \mathbb{Z}_p^n$ via*

$$\hat{f}(x_{1,1}, \ldots, x_{n,k}) = \sum_{i=1}^n \sum_{j=1}^k \left(x_{i,j} \cdot \left\lceil 2^j \cdot \frac{p}{q} a_i \right\rfloor_\sigma + (1 - x_{i,j}) \lceil 0 \rfloor_\sigma \right) + \left\lceil \frac{p}{q} c \right\rfloor_\sigma.$$

Then it holds for all $x_{1,1}, \ldots, x_{n,k} \in \{0,1\}$ that

$$\hat{f}(x_{1,1}, \ldots, x_{n,k}) \approx_s \left\lceil \frac{p}{q} \cdot f\left(\sum_{j=1}^k x_{1,j} 2^j, \ldots, \sum_{j=1}^k x_{n,j} 2^j \right) \right\rfloor_{\sqrt{2nk+1}\sigma}.$$

Proof. It holds routinely that

$$\begin{aligned}
\hat{f}(x_{1,1},\ldots,x_{n,k}) &= \sum_{i=1}^{n}\sum_{j=1}^{k}\left(x_{i,j}\cdot\left\lceil 2^j\cdot\frac{p}{q}a_i\right\rfloor_{\sigma}+(1-x_{i,j})\left\lceil 0\right\rfloor_{\sigma}\right)+\left\lceil\frac{p}{q}c\right\rfloor_{\sigma}\\
&\approx_s \sum_{i=1}^{n}\sum_{j=1}^{k}\left\lceil x_{i,j}2^j\cdot\frac{p}{q}a_i\right\rfloor_{\sqrt{2}\sigma}+\left\lceil\frac{p}{q}c\right\rfloor_{\sigma} \qquad (1)\\
&\approx_s \left\lceil\sum_{i=1}^{n}\sum_{j=1}^{k}x_{i,j}2^j\cdot\frac{p}{q}a_i+\frac{p}{q}c\right\rfloor_{\sqrt{2nk+1}\sigma} \qquad (2)\\
&= \left\lceil\frac{p}{q}\cdot\left(\sum_{i=1}^{n}a_i\left(\sum_{j=1}^{k}x_{i,j}2^j\right)+c\right)\right\rfloor_{\sqrt{2nk+1}\sigma}\\
&= \left\lceil\frac{p}{q}\cdot f\left(\sum_{j=1}^{k}x_{1,j}2^j,\ldots,\sum_{j=1}^{k}x_{n,j}2^j\right)\right\rfloor_{\sqrt{2nk+1}\sigma},
\end{aligned}$$

where in equations (1) and (2) we have used Lemma 3. □

5 Rate-1 Circuit-Private Linearly Homomorphic Encryption

In this section we define circuit-private LHE and present constructions based on LWE, DDH or QR. All constructions achieve rate 1.

Definition 13. *A (packed)* linearly homomorphic encryption *(LHE) scheme* LHE *over a finite group* $\mathbb{G}$ *is composed by a tuple of algorithms* $(\mathsf{Keygen},\mathsf{Enc},\mathsf{Eval},\mathsf{Shrink},\mathsf{DecShrink})$ *such that:*

- $\mathsf{KeyGen}(1^\lambda,k)$ *takes as input a security parameter* λ *and* $k\in\mathbb{N}$*. It outputs a pair of public and secret keys* $(\mathsf{pk},\mathsf{sk})$*.*
- $\mathsf{Enc}(\mathsf{pk},\mathbf{m}=(m_1,\ldots,m_k))$ *takes as input a public key* pk *and a message* $\mathbf{m}=(m_1,\ldots,m_k)\in\mathbb{G}^k$*. It outputs a ciphertext* ct*.*
- $\mathsf{Eval}(\mathsf{pk},f,(\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell))$ *takes as input a public key* pk*, a linear function* $f:(\mathbb{G}^k)^\ell\to\mathbb{G}^k$ *and* ℓ *ciphertexts* $(\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell)$*. It outputs a new ciphertext* $\tilde{\mathsf{ct}}$*.*
- $\mathsf{Shrink}(\mathsf{pk},\mathsf{ct})$ *takes as input a public key* pk *and a ciphertext* ct*. It outputs a new shrunken ciphertext* ct'*.*
- $\mathsf{DecShrink}(\mathsf{sk},\mathsf{ct})$ *takes as input a secret key* sk *and a shrunken ciphertext* ct*. It outputs a message* $\mathbf{m}$*.*

For simplicity, we define the algorithm $\mathsf{Eval\&Shrink}(\mathsf{pk},f,(\mathsf{ct}_1\ldots,\mathsf{ct}_\ell))$ which outputs a ciphertext $\tilde{\mathsf{ct}}$ and is defined as

$$\mathsf{Eval\&Shrink}(\mathsf{pk},f,(\mathsf{ct}_1\ldots,\mathsf{ct}_\ell))=\mathsf{Shrink}(\mathsf{pk},\mathsf{Eval}(\mathsf{pk},f,(\mathsf{ct}_1,\ldots,\mathsf{ct}_\ell)))$$

for any linear function f.

We require the following properties from a (circuit-private) packed LHE: Correctness, semantic security, compactness and circuit-privacy.

Definition 14 (Correctness)**.** *A packed LHE scheme* LHE *is said to be correct if for any* $\ell\in\mathbb{N}$*, any messages* $\mathbf{m}_1,\ldots,\mathbf{m}_\ell$ *and any linear function* $f:(\mathbb{G}^k)^\ell\to\mathbb{G}^k$ *we have that*

$$\Pr\left[\tilde{\mathbf{m}}\leftarrow\mathsf{DecShrink}(\mathsf{sk},\tilde{\mathsf{ct}}):\begin{array}{c}(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KeyGen}(1^\lambda,k)\\ \mathsf{ct}_i\leftarrow\mathsf{Enc}(\mathsf{pk},\mathbf{m}_i)\ \textit{for}\ i\in[\ell]\\ \tilde{\mathsf{ct}}\leftarrow\mathsf{Eval\&Shrink}(\mathsf{pk},,f,(\mathsf{ct}_1\ldots,\mathsf{ct}_\ell))\end{array}\right]=1$$

where $\tilde{\mathbf{m}} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

Definition 15 (Semantic Security)**.** *A packed LHE scheme* LHE *is said to be semantically secure if for all* $\lambda \in \mathbb{N}$*, all* $k = \mathsf{poly}(\lambda)$ *and all adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *we have that*

$$\left| \Pr\left[b \leftarrow \mathcal{A}_1(\mathsf{st}, \mathsf{ct}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ (\mathbf{m}_0, \mathbf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pk}) \\ b \leftarrow\$ \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Definition 16 (Compactness)**.** *We require that a packed LHE scheme* LHE *has the following compactness properties:*

- *For* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k)$*, the size of the public key* $|\mathsf{pk}|$ *is bounded by* $k \cdot \mathsf{poly}(n)$*.*
- *For any linear function* $f : (\mathbb{G}^k)^\ell \to \mathbb{G}^k$ *and any* $(\mathbf{m}_1, \ldots, \mathbf{m}_\ell) \in (\mathbb{G}^k)^\ell$ *we have that*

$$\lim_{\lambda \to \infty} \inf \frac{|f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)|}{|\mathsf{Eval\&Shrink}(\mathsf{pk}, , f, (\mathsf{ct}_1 \ldots, \mathsf{ct}_\ell))|} \to 1$$

  for sufficiently large k*, where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k)$ *and* $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i)$ *for* $i \in [\ell]$*. In this case, we say that the scheme has rate 1.*

We also need that the packed LHE scheme fulfills circuit privacy (in the semi-honest case).

Definition 17 (Circuit Privacy)**.** *A packed LHE scheme* LHE *is said to be circuit-private if for all messages* $(\mathbf{m}_1, \ldots, \mathbf{m}_\ell) \in (\mathbb{G}^k)^\ell$ *and all linear functions* $f : (\mathbb{G}^k)^\ell \to \mathbb{G}^k$*, there exists a simulator* Sim *such that for all adversaries* $\mathcal{A}$ *we have that*

$$\left| \begin{array}{c} \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}, \tilde{\mathsf{ct}}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i) \text{ for } i \in [\ell] \\ \tilde{\mathsf{ct}} \leftarrow \mathsf{Eval\&Shrink}(\mathsf{pk}, , f, (\mathsf{ct}_1 \ldots, \mathsf{ct}_\ell)) \end{array} \right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk}, \tilde{\mathsf{ct}}) : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k) \\ \tilde{\mathsf{ct}} \leftarrow \mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}}) \end{array} \right] \end{array} \right| \leq \mathsf{negl}(\lambda)$$

where $\tilde{\mathbf{m}} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

In other words, since Sim does not use f to compute $\tilde{\mathsf{ct}}$, no information about it is leaked from $\tilde{\mathsf{ct}}$ (apart from what is trivially leaked by f).

Encryption of matrices. Above, we defined LHE that supports encryption of vectors $\mathbf{m} \in \mathbb{G}^k$. We can easily extend the definition to support encryption of matrices $\mathbf{M} \in \mathbb{G}^{k \times \alpha}$ for any $\alpha = \mathsf{poly}(\lambda)$: Given a public key pk, an encryption $\mathsf{Enc}(\mathsf{pk}, \mathbf{M})$ of $\mathbf{M}$ is defined as

$$\mathsf{Enc}(\mathsf{pk}, \mathbf{M}) = \begin{pmatrix} | & & | \\ \mathsf{Enc}\left(\mathsf{pk}, \mathbf{m}^{(1)}\right) & \ldots & \mathsf{Enc}\left(\mathsf{pk}, \mathbf{m}^{(\alpha)}\right) \\ | & & | \end{pmatrix}$$

where $\mathbf{m}^{(i)}$ is the i-th column of $\mathbf{M}$.

5.1 Construction from LWE

Before sketching the scheme, we present the LWE assumption [Reg05].

Definition 18 (Learning with Errors)**.** *Let* $n, q \in \mathbb{Z}$*. The LWE assumption holds if for any PPT adversary* $\mathcal{A}$

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

for all $m = \mathsf{poly}(n)$*, where* $\mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n \times m}$*,* $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^n$*,* $\mathbf{e} \leftarrow\$ D_{\mathbb{Z},\sigma}^m$ *and* $\mathbf{u} \leftarrow\$ \mathbb{Z}_q^m$*.*

When we consider $\sigma = \zeta q \geq 2\sqrt{n}$, the LWE problem is at least as hard as solving the approximate shortest independent vector problem to within a factor of $\tilde{\mathcal{O}}(n/\zeta)$ [Reg05].

5.1.1 Shrinking ciphertexts

The work of [BDGM19] shows how to shrink LWE-based ciphertexts of the form $(\mathbf{Ar}, \mathbf{b}_1\mathbf{r}+\lceil q/2\rfloor m_1, \ldots, \mathbf{b}_k\mathbf{r}+\lceil q/2\rfloor m_k)$ (where $\mathbf{A} \leftarrow\!\$\, \mathbb{Z}^{n\times m}$, $\mathbf{b}_i$ are LWE samples and $\mathbf{r}$ is a short vector). The resulting shrunken cipherthext is composed by $(\mathbf{Ar}, b_1, \ldots, b_k)$ where $b_1, \ldots, b_k \in \{0,1\}$ and thus the rate tends to 1 when we consider large k.

Before presenting the result of [BDGM19], we first need to define *relaxed correctness* for a standard LHE (as in [BDGM19]). A standard LHE is an LHE where the algorithms Shrink and $\mathsf{DecShrink}$ are replaced by a decryption algorithm $\mathbf{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$, and it is not required to have rate 1.

Definition 19 (Relaxed correctness)**.** *Let* $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a (standard) LHE. We say that* LHE *is correct with* B*-noise if*

$$\mathbf{Tm} + \mathbf{e} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{Enc}(\mathsf{pk}, \mathbf{m}_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_\ell)))$$

where $\mathbf{T}$ *is an encoding matrix and* $\|\mathbf{e}\| \leq B$*.*

Lemma 5 ([BDGM19])**.** *Let* $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be a (standard) LHE that is correct with* B*-noise. Additionally, assume that the ciphertexts of the scheme are of the form* $(\mathbf{c}_1, \mathbf{c}_2)$*, secret key is of the form* $\mathbf{S} \in \mathbb{Z}_q^{k\times n}$ *and noisy decryption works by computing* $\mathbf{c}_2 - \mathbf{Sc}_1$*. If* $q > 4kB$ *then there exist a correct shrinking algorithm* $(\mathsf{Shrink}_{\mathsf{LWE}}, \mathsf{DecShrink}_{\mathsf{LWE}})$ *for the packed LWE-based LHE scheme.*

5.1.2 Circuit-private LHE from LWE.

We now present the circuit-private LHE from LWE. The scheme is a hybrid between the packed Regev PKE [GPV08] and GSW PKE [GSW13], together with the circuit-privacy technique of [BdMW16].

We present a scheme supporting plaintext space $\{0,1\}^k$. We later briefly explain how we can extend the scheme to any $q = \mathsf{poly}(\lambda)$.

We will need the following ingredients: Let $(\mathsf{Shrink}_{\mathsf{LWE}}, \mathsf{DecShrink}_{\mathsf{LWE}})$ be the pair of algorithms from Lemma 5. Let $\sigma, \alpha, \beta, q, m, n, t, k$ be polynomials in λ. Let $\mathbf{g}_{\mathsf{rnd}}^{-1}$ be the (randomized) function defined in Section 3 that receives $v \in \mathbb{Z}_p$ as input and outputs $x \leftarrow\!\$\, D_{\Lambda_q^\perp(\mathbf{g})+\mathbf{g}^{-1}(v),\gamma}$ for some $\gamma = \tilde{\mathcal{O}}(1)$. As defined in Section 3, let $\mathbf{G}_i$ be the matrix with k rows which is zero everywhere except for the i-th row which is equal to $\mathbf{g} = (1, 2, 2^2, \ldots, 2^t)$, and let $\bar{\mathbf{G}}_j$ be the matrix with j rows and where every row is equal to $\mathbf{g}$.

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

- Sample $\mathbf{A} \leftarrow\!\$\, \mathbb{Z}_q^{n\times m}$, $\mathbf{S} \leftarrow\!\$\, \mathbb{Z}_q^{k\times n}$ and $\mathbf{E} \leftarrow\!\$\, D_{\mathbb{Z},\sigma}^{k\times m}$. Compute $\mathbf{B} = \mathbf{SA} + \mathbf{E}$.
- Output $\mathsf{pk} = (\mathbf{A}, \mathbf{B})$ and $\mathsf{sk} = \mathbf{S}$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \ldots, m_k) \in \{0,1\}^k)$:

- Parse pk as $(\mathbf{A}, \mathbf{B})$.
- Sample $\mathbf{R} \leftarrow\!\$\, D_{\mathbb{Z},\alpha}^{m\times t}$. Compute $\mathbf{C}_1 = \mathbf{AR}$ and $\mathbf{C}_2 = \mathbf{BR} + \sum_{i=1}^k m_i \mathbf{G}_i$.
- Output $\mathsf{ct} = (\mathbf{C}_1, \mathbf{C}_2)$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$

- Parse pk as , f as $f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \sum_{i=1}^\ell a_i \mathbf{x}_i + \mathbf{b}$, where $\mathbf{a} = (a_1, \ldots, a_\ell) \in \mathbb{Z}_2^\ell$, $\mathbf{b} \in \{0,1\}^k$ and ct_i as $(\mathbf{C}_{1,i}, \mathbf{C}_{2,i})$.
- Compute

$$\mathbf{c}_1 = \sum_{j=1}^{\ell} \mathbf{C}_{1,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2} a_j\right)^T + \left(\bar{\mathbf{G}}_n - \mathbf{C}_{1,j}\right) \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{A}\mathbf{y}_j^T$$

and

$$\mathbf{c}_2 = \sum_{j=1}^{\ell} \left(\mathbf{C}_{2,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T + \left(\bar{\mathbf{G}}_k - \mathbf{C}_{2,j}\right) \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{B}\mathbf{y}_j^T \right) + \frac{q}{2}\mathbf{b}$$

where $\mathbf{y}_j \leftarrow\!\!\$\, D_{\mathbb{Z},\beta}^m$.

- Output $\tilde{\mathsf{ct}} = (\mathbf{c}_1, \mathbf{c}_2)$.

$\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$: Output $\tilde{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{LWE}}(\mathsf{pk}, \mathsf{ct})$.

$\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$: Output $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{LWE}}(\mathsf{sk}, \mathsf{ct})$.

We now analyze the construction presented above. We start by showing that the scheme is correct.

Lemma 6 (Correctness)**.** *Let $q = 2q' > 4k\left(2\alpha\gamma t + \beta\right)\ell\sigma m\sqrt{k}$ for some $q' \in \mathbb{Z}$. Then the scheme presented above is correct.*

Proof. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda, k)$ and $\mathsf{ct}_i = (\mathbf{C}_{1,i}, \mathbf{C}_{2,i}) \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_i)$ be well-formed ciphertexts for all $i \in [\ell]$. We have to show that

$$\tilde{\mathbf{m}} \leftarrow \mathsf{DecShrink}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))))$$

where $\tilde{\mathbf{m}} = \sum_{j=1}^{\ell} a_j\mathbf{m}_j + \mathbf{b} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

Let $(\mathbf{c}_1, \mathbf{c}_2) \leftarrow \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$. A routine calculation shows that

$$\mathbf{c}_1 = \mathbf{A}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right)$$

and

$$\mathbf{c}_2 = \mathbf{B}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right) + \frac{q}{2}\left(\sum_{j=1}^{\ell} a_j\mathbf{m}_j + \mathbf{b}\right)$$

where the last equality holds because $2|q$.

We first show that the scheme meets the conditions of Lemma 5. After computing $\mathbf{c}_2 - \mathbf{S}\mathbf{c}_1$ we obtain

$$\frac{q}{2}\left(\sum_{j=1}^{\ell} a_j\mathbf{m}_j + \mathbf{b}\right) + \mathbf{E}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right).$$

Let $\mathbf{e}' = \mathbf{E}\left(\sum_{j=1}^{\ell} \mathbf{R}_j \left(\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right) - \mathbf{g}_{\mathsf{rnd}}^{-1}(0)\right)^T + \mathbf{y}_j^T\right)$. By Lemma 1, each row of $\mathbf{R}_j$ has norm at most $\alpha\sqrt{t}$, the vectors $\mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right), \mathbf{g}_{\mathsf{rnd}}^{-1}(0)$ have norm at most $\gamma\sqrt{t}$, $\mathbf{y}_j$ has norm at most $\beta\sqrt{m}$ and each row of $\mathbf{E}$ has norm at most $\sigma\sqrt{m}$. Hence

$$\|\mathbf{e}'\| \leq \left(2\alpha\gamma t + \beta\right)\ell\sigma m\sqrt{k}.$$

Since $q > 4k\left(2\alpha\gamma t + \beta\right)\ell\sigma m\sqrt{k}$ then we are in the conditions of Lemma 5. Thus, we conclude that

$$\tilde{\mathbf{m}} \leftarrow \mathsf{DecShrink}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, (\mathbf{c}_1, \mathbf{c}_2)))$$

where $\tilde{\mathbf{m}} = \sum_{j=1}^{\ell} a_j\mathbf{m}_j + \mathbf{b} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$ and the scheme is correct. □

Semantic security can be established by relying on the smoothing lemma together with the LWE assumption [Reg05, MR04].

Lemma 7 (Semantic security)**.** *Assume that the LWE assumption holds for $\sigma = \zeta q \geq 2\sqrt{n}$ for some $\zeta \in \mathbb{R}$ and $m = \mathsf{poly}((n+k)\log q)$. Also, let $\alpha \geq \omega(\sqrt{\log m})$. Then the scheme is semantically secure.*

Proof (Sketch). In the first hybrid, we replace the public key $(\mathbf{A}, \mathbf{B})$ by $(\mathbf{A}, \mathbf{U})$ for a uniformly chosen $\mathbf{U}$ and this change goes unnoticed by the LWE assumption. Next, we can use the smoothing lemma [Reg05, MR04, GPV08] to replace $(\mathbf{A}, \mathbf{U}, \mathbf{AR}, \mathbf{UR})$ by $(\mathbf{A}, \mathbf{U}, \mathbf{V}_1, \mathbf{V}_2)$ where $\mathbf{V}_1, \mathbf{V}_2$ are uniformly chosen. Finally, we can conclude that the encrypted message is statistically hidden and the result follows. □

Before presenting the proof that the scheme is circuit private, we present a lemma that we will need.

Lemma 8 ([BdMW16, AR16])**.** *For any $a \in \mathbb{Z}_q$ and any matrix $\mathbf{E} \in \mathbb{Z}^{k \times m}$, let $r = \tilde{\Theta}(\max_i \|\mathbf{e}_i\| \sqrt{\lambda})$ (where $\mathbf{e}_i$ are the rows of $\mathbf{E}$). Then*

$$\mathbf{E} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(a)^T + \mathbf{y}^T \approx_s \mathbf{f}^T$$

where $\mathbf{y} \leftarrow\!\!\$\, D_{\mathbb{Z},r}^k$ $\mathbf{f} \leftarrow\!\!\$\, D_{\mathbb{Z},r'}^k$ and $r' = r\sqrt{1 + \max_i \|\mathbf{e}_i\|^2}$.

Lemma 9 (Circuit-privacy)**.** *Let $\beta = \tilde{\Theta}(\alpha\sqrt{2\lambda t})$. Then the scheme presented above is circuit private.*

Proof. To prove circuit-private, we show how we can simulate evaluated ciphertexts. We first present the simulator $\mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}})$:

$\mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}})$:

- Sample $\bar{\mathbf{r}} \leftarrow\!\!\$\, D_{\mathbb{Z},\mu}^m$. Compute $\mathbf{c}_1 = \mathbf{A}\bar{\mathbf{r}}^T$ and $\mathbf{c}_2 = \mathbf{B}\bar{\mathbf{r}}^T + \frac{q}{2}\tilde{\mathbf{m}}$ where $\mu = \beta\sqrt{\ell\left(1 + \left(\alpha\sqrt{t}\right)^2\right)}$ and $\tilde{\mathbf{m}} = f(\mathbf{m}_1, \ldots, \mathbf{m}_t)$.
- Output $\tilde{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{LWE}}(\mathsf{pk}, (\mathbf{c}_1, \mathbf{c}_2))$.

We now prove that the simulated ciphertext is indistinguishable from a evaluated ciphertext.

First, note that evaluated ciphertexts are of the form

$$\mathbf{c}_1 = \sum_{j=1}^{\ell} \mathbf{C}_{1,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T - (\bar{\mathbf{G}}_n - \mathbf{C}_{1,j}) \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{A}\mathbf{y}_j^T$$

and

$$\mathbf{c}_2 = \sum_{j=1}^{\ell} \left(\mathbf{C}_{2,j} \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T + (\bar{\mathbf{G}}_k - \mathbf{C}_{2,j}) \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{B}\mathbf{y}_j^T\right) + \frac{q}{2}\mathbf{b}.$$

Writing in matrix form, each term of the sum is of the form

$$\begin{aligned}
&\begin{pmatrix}\mathbf{C}_{1,j}\\ \mathbf{C}_{2,j}\end{pmatrix} \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a_j\right)^T + \left(\begin{pmatrix}\bar{\mathbf{G}}_n\\ \bar{\mathbf{G}}_k\end{pmatrix} - \begin{pmatrix}\mathbf{C}_{1,j}\\ \mathbf{C}_{2,j}\end{pmatrix}\right) \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \mathbf{y}_j^T \\
&= \begin{pmatrix}\mathbf{A}\mathbf{R}_j\\ \mathbf{B}\mathbf{R}_j + \sum_{i=1}^k m_{j,i}\mathbf{G}_i\end{pmatrix} \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T + \left(\begin{pmatrix}\bar{\mathbf{G}}_n\\ \bar{\mathbf{G}}_k\end{pmatrix} - \begin{pmatrix}\mathbf{A}\mathbf{R}_j\\ \mathbf{B}\mathbf{R}_j + \sum_{i=1}^k m_{j,i}\mathbf{G}_i\end{pmatrix}\right) \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \mathbf{y}_j^T \\
&= \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \left(\mathbf{R}_j \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T - \mathbf{R}_j \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{y}_j^T\right) + \begin{pmatrix}\mathbf{0}\\ a_j \cdot \mathbf{m}_j\end{pmatrix}
\end{aligned}$$

where $\mathbf{m}_j = (m_{j,1}, \ldots, m_{j,t})$.

It follows that

$$\begin{aligned}
&\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \left(\mathbf{R}_j \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T - \mathbf{R}_j \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{y}_j^T\right) \\
&\approx_s \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \left(\left(\mathbf{R}_j \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}\left(\frac{q}{2}a\right)^T + \mathbf{y}_{j,1}^T\right) + (-\mathbf{R}_j \cdot \mathbf{g}_{\mathsf{rnd}}^{-1}(0)^T + \mathbf{y}_{j,2}^T)\right) \\
&\approx_s \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} (\mathbf{r}'_{1,j} + \mathbf{r}'_{2,j}) \\
&\approx_s \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix} \mathbf{r}'^T_j
\end{aligned}$$

for $\mathbf{y}_{j,b} \leftarrow\$ D_{\mathbb{Z},\beta/\sqrt{2}}$, $\mathbf{r}'_{j,1}, \mathbf{r}'_{j,2} \leftarrow\$ D^m_{\mathbb{Z},r'/\sqrt{2}}$ and $\mathbf{r}'_j \leftarrow\$ D^m_{\mathbb{Z},r'}$ where $r' = \beta\sqrt{1+\left(\alpha\sqrt{t}\right)^2}$. The first and the last steps follow from the fact that the sum of two independent discrete gaussians is statistically close to a discrete gaussian (that is, $\mathbf{y}_j \approx_s \mathbf{y}_{j,1} + \mathbf{y}_{j,2}$ and $\mathbf{r}'_j \approx_s \mathbf{r}'_{j,1} + \mathbf{r}'_{j,2}$). The second step follows from Lemma 8.

Hence,

$$\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\left(\mathbf{R}_j \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}\left(\frac{q}{2}a\right)^T - \mathbf{R}_j \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}(0)^T + \mathbf{y}_j^T\right) + \begin{pmatrix}\mathbf{0}\\ a_j \cdot \mathbf{m}_j\end{pmatrix} \approx_s \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\mathbf{r}_j'^T + \begin{pmatrix}\mathbf{0}\\ a_j \cdot \mathbf{m}_j\end{pmatrix}.$$

From this, we conclude that

$$\begin{aligned}\begin{pmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{pmatrix} &= \sum_{j=1}^{\ell}\begin{pmatrix}\mathbf{C}_{1,j} \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}\left(\frac{q}{2}a_j\right)^T - \left(\bar{\mathbf{G}}_n - \mathbf{C}_{1,j}\right)\mathbf{g}^{-1}_{\mathsf{rnd}}(0)^T + \mathbf{A}\mathbf{y}_j^T\\ \mathbf{C}_{2,j} \cdot \mathbf{g}^{-1}_{\mathsf{rnd}}\left(\frac{q}{2}a_j\right)^T + \left(\bar{\mathbf{G}}_k - \mathbf{C}_{2,j}\right)\mathbf{g}^{-1}_{\mathsf{rnd}}(0)^T + \mathbf{B}\mathbf{y}_j^T\end{pmatrix} + \begin{pmatrix}\mathbf{0}\\ \frac{q}{2}\mathbf{b}\end{pmatrix}\\ &\approx_s \sum_{j=1}^{\ell}\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\mathbf{r}_j'^T + \begin{pmatrix}\mathbf{0}\\ \frac{q}{2}\left(\sum_{j=1}^{\ell} a_j\mathbf{m}_j + \mathbf{b}\right)\end{pmatrix}\\ &\approx_s \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\bar{\mathbf{r}}^T + \begin{pmatrix}\mathbf{0}\\ \frac{q}{2} \cdot f(\mathbf{m}_1, \ldots, \mathbf{m}_t)\end{pmatrix}\end{aligned}$$

where $\bar{\mathbf{r}} \leftarrow\$ D^m_{\mathbb{Z},\mu}$. The last step follows from the fact that the sum of ℓ independent discrete gaussians with parameter r' (where $r' = \beta\sqrt{1+\left(\alpha\sqrt{t}\right)^2}$) is statistically indistinguishable from a discrete gaussian with parameter $\sqrt{\ell}r'$. □

Ciphertext rate. It is easy to see that the rate of the ciphertext tends to 1 for large enough k by relying on Lemma 5. It is also easy to see that the size of the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{B} = \mathbf{S}\mathbf{A} + \mathbf{E}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{k\times m}$ is bounded by $k \cdot \mathsf{poly}(\lambda)$.

Larger plaintext space. In the construction presented above, the plaintext space is $\mathbb{Z}_2^k$. The construction can be extended to support plaintext space $\mathbb{Z}_p^k$ for any $p = \mathsf{poly}(\lambda)$ by choosing the LWE modulus q of the form $q = pp'$ where p, p' are co-prime, and encoding the encrypted message by q/p.

5.2 Construction from DDH

In the following, let $\mathcal{G}$ be a (prime-order) *group generator*, that is, $\mathcal{G}$ is an algorithm 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.

Definition 20 (Decisional Diffie-Hellman Assumption)**.** *Let $(\mathbb{G}, p, g) \leftarrow\$ \mathcal{G}(1^\lambda)$. We say that the DDH assumption holds (with respect to $\mathcal{G}$) 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 $a, b, c \leftarrow\$ \mathbb{Z}_p$.

5.2.1 Shrinking ciphertexts.

We first present how we can shrink DDH-based ciphertexts to achieve rate 1. The shrinking mechanism presented below is a modification of the one presented in [BBD+20] (which is itself based on previous works [BGI16, DGI+19]).

Let $(\mathbb{G}, p, g) \leftarrow\!\$\, \mathcal{G}(1^\lambda)$ and $k \in \mathbb{Z}$. Consider an El Gamal public key of the form $\mathsf{pk} = (g, (h_1, \ldots, h_k) = (g, (g^{x_1}, \ldots, g^{x_k})) \in \mathbb{G}^{k+1}$ for $x_1, \ldots, x_k \leftarrow\!\$\, \mathbb{Z}_p$ (here, $\mathbf{x} = (x_1, \ldots, x_k)$ is the secret key). Consider the following modified El Gamal encryption algorithm where a ciphertext for $\mathbf{m} = (m_1, \ldots, m_k) \in \{0,1\}^k$ is of the form $\mathsf{ct} = (c_1, (c_{2,1}, \ldots, c_{2,k})) \in \mathbb{G}^{k+1}$ where $c_1 = g^r$ and $c_{2,i} = h_i^r g^{\lceil m_i (p/2) \rfloor_\sigma}$. [11] We now show how to compress ciphertexts of this form.

We will need the following ingredients: Let $B, T \in \mathsf{poly}(\lambda)$ and $\mathsf{PRF} = (\mathsf{KeyGen}, \mathsf{Eval})$ be a PRF that maps $g \in \mathbb{G}$ to $\{0,1\}^\tau$ for some $\tau \in \mathbb{Z}$. We also define the function $\mathsf{LEq}_< : \mathbb{G}^2 \to \{0,1\}$ which receives two group elements g_0, g_1 and outputs 1 if $g_0 < g_1$ and 0 otherwise, for some order relation $<$ (e.g. the lexicographic order).

$\mathsf{Shrink}_{\mathsf{DDH}}(\mathsf{pk}, \mathsf{ct})$:

- Parse $\mathsf{pk} = (g, (h_1, \ldots, h_k))$ and $\mathsf{ct} = (c_1, (c_{2,1}, \ldots, c_{2,k}))$. Let $w = g^{\lfloor p/2 \rfloor}$.
- Sample a PRF key $\mathsf{K} \leftarrow\!\$\, \mathsf{PRF.KeyGen}(1^\lambda)$ such that the following conditions are simultaneously satisfied:
  1. For every $i \in [k]$ and $j \in \{-B, \ldots, B\}$ we have that
  $$\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0.$$
  2. For all $i \in [k]$ there exists $\ell \in \{B+1, \ldots, T\}$ such that
  $$\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^\ell) = 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^\ell) = 0.$$
- For every $i \in [k]$, let $\delta_{0,i}, \delta_{1,i} > 0$ be the smallest integer such that
$$\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^{\delta_{0,i}}) = 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^{\delta_{1,i}}) = 0.$$
Let $\alpha_{0,i} = c_{2,i} \cdot g^{\delta_{0,i}}$ and $\alpha_{1,i} = c_{2,i} \cdot w \cdot g^{\delta_{1,i}}$. If $\mathsf{LEq}_<(\alpha_{0,i}, \alpha_{1,i}) = 0$, then set $b_i = 0$. Else, set $b_i = 1$.
- Output $\bar{\mathsf{ct}} = (c_1, \mathsf{K}, (b_1, \ldots, b_k))$.

$\mathsf{DecShrink}_{\mathsf{DDH}}(\mathsf{sk}, \bar{\mathsf{ct}})$:

- Parse $\mathsf{sk} = \mathbf{x} = (x_1, \ldots, x_k)$ and $\bar{\mathsf{ct}} = (c_1, \mathsf{K}, (b_1, \ldots b_k))$. Let $w = g^{\lfloor p/2 \rfloor}$.
- For every $i \in [k]$, compute $\beta_{0,i} = c_1^{x_i}$ and $\beta_{1,i} = c_1^{x_i} \cdot w$.
- For every $i \in [k]$, find the smallest integers $\gamma_{0,i}, \gamma_{1,i} > 0$ such that
$$\mathsf{PRF.Eval}(\mathsf{K}, \beta_{0,i} \cdot g^{\gamma_{0,i}}) = 0 \text{ and } \mathsf{PRF.Eval}(\mathsf{K}, \beta_{1,i} \cdot g^{\gamma_{1,i}}) = 0.$$
Let $\bar{\alpha}_{0,i} = \beta_{0,i} \cdot g^{\gamma_{0,i}}$ and $\bar{\alpha}_{1,i} = \beta_{1,i} \cdot g^{\gamma_{1,i}}$. If $\mathsf{LEq}_<(\bar{\alpha}_{0,i}, \bar{\alpha}_{1,i}) = b_i$, set $m_i = 0$. Else, set $m_i = 1$.
- Output $\mathbf{m} = (m_1, \ldots, m_k)$.

Lemma 10 (Correctness)**.** *Let $B = \mathsf{poly}(\lambda)$ be such that $B > \lambda\sigma + 1$. Then the shrinking procedure presented above is correct.*

Proof. We have to show that $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{DDH}}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, \mathsf{ct}))$ for $\mathsf{ct} = (c_1, (c_{2,1}, \ldots, c_{2,k}))$ where $c_1 = g^r$ and $c_{2,i} = h_i^r g^{\lceil m_i (p/2) \rfloor_\sigma}$ for $i \in [k]$.

For that, we will first show that

$$\begin{array}{c} (\bar{\alpha}_{0,i} = \alpha_{0,i} \wedge \bar{\alpha}_{1,i} = \alpha_{1,i}) \\ \vee \\ (\bar{\alpha}_{0,i} = \alpha_{1,i} \wedge \bar{\alpha}_{1,i} = \alpha_{0,i}) \end{array}.$$

[11] Note that $\lceil \cdot \rfloor_\sigma$ is defined in section 4.

We have that $\alpha_{0,i} \neq \alpha_{1,i}$.

The first observation is that $0 \in \{z_0 - (B-1), \ldots, z_0 + (B-1)\}$ where $z_0 = \lceil 0 \rfloor_\sigma$ except with negligible probability. This is because $B > \lambda\sigma + 1$ and $|z_0| < \lambda\sigma$ except with negligible probability.[12] Thus $0 \in \{z_0 - B, \ldots, z_0 + B\}$

Likewise, $p/2 \in \left[z_{p/2} - (B-1), z_{p/2} + (B-1)\right]$ where $z_{p/2} = \lceil p/2 \rfloor_\sigma$ and thus $\lfloor p/2 \rfloor \in \{z_{p/2} - B, \ldots, z_{p/2} + B\}$.

We divide the proof in two cases: Either $m_i = 0$ or $m_i = 1$. We start by analyzing the case where $m_i = 0$.

Case $m_i = 0$. Assume that $m_i = 0$. We first note that $\bar{\alpha}_{0,i} = \beta_{0,i} \cdot g^{\gamma_{0,i}} = c_1^{x_i} \cdot g^{\gamma_{0,i}} = h_i^r \cdot g^{\gamma_{0,i}}$, and $\alpha_{0,i} = h_i^r \cdot g^{z_{0,i} + \delta_{0,i}}$ where $z_{0,i} = \lceil m_i(p/2) \rfloor_\sigma = \lceil 0 \rfloor_\sigma$.We prove that $\bar{\alpha}_{0,i} = \alpha_{0,i}$. To prove this, it is enough to show that $\gamma_{0,i} = z_{0,i} + \delta_{0,i}$. Observe that, if this is not the case, then one of the two cases must be true:

(i) $\gamma_{0,i} < \lceil m_i(p/2) \rfloor_\sigma + \delta_{0,i} = \lceil 0 \rfloor_\sigma + \delta_{0,i}$: If this happens then one of the three cases must hold:

  (a) $\gamma_{0,i} < z_{0,i} - B$: This case cannot hold since $0 \in \{z_{0,i} - B, \ldots, z_{0,i} + B\}$ (except with negligible probability) and $\gamma_{0,i} \geq 0$. This implies that $\gamma_{0,i} \geq z_{0,i} - B$, except with negligible probability.

  (b) $z_{0,i} - B \leq \gamma_{0,i} \leq z_{0,i} + B$: This case cannot hold since it violates condition 1.

  (c) $z_{0,i} + B < \gamma_{0,i} < z_{0,i} + \delta_{0,i}$: This case violates condition 2 since $\delta_{0,i} > B$ is the smallest integer that fulfills $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r \cdot g^{z_{0,i} + \delta_{0,i}})$.

(ii) $\gamma_{0,i} > z_{0,i} + \delta_{0,i}$: This case cannot happen as $\gamma_{0,i}$ is the smallest integer (greater than 0) such that $\mathsf{PRF.Eval}(K, h_i^r \cdot g^{\gamma_{0,i}}) = 0$.

Showing that, if $m_i = 0$, then $\bar{\alpha}_{1,i} = \alpha_{1,i}$ follows an identical reasoning. We conclude that, if $m_i = 0$, then $\bar{\alpha}_{0,i} = \alpha_{0,i} \wedge \bar{\alpha}_{1,i} = \alpha_{1,i}$.

Case $m_i = 1$. Now assume that $m_i = 1$. In this case, we show that $\bar{\alpha}_{1,i} = \alpha_{0,i}$ and $\bar{\alpha}_{0,i} = \alpha_{1,i}$.

First, note that $\bar{\alpha}_{1,i} = h_i^r \cdot g^{\lfloor p/2 \rfloor + \gamma_{1,i}}$ and $\alpha_{0,i} = h_i^r \cdot g^{z_{p/2,i} + \delta_{0,i}}$ where $z_{p/2,i} = \lceil m_i(p/2) \rfloor_\sigma = \lceil p/2 \rfloor_\sigma$. We prove that $\bar{\alpha}_{1,i} = \alpha_{0,i}$. To show this, we have to prove that $\lfloor p/2 \rfloor + \gamma_{1,i} = z_{p/2,i} + \delta_{0,i}$. Assume that this is not true, then one of the two cases must happen:

(i) $\lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} + \delta_{0,i}$: If this is the case, then one of the three cases must happen:

  (a) $\lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} - B$: This case cannot happen because $\lfloor p/2 \rfloor \in \{z_{p/2,i} - B, \ldots, z_{p/2,i} + B\}$ except with negligible probability and $\gamma_{1,i} > 0$.

  (b) $z_{p/2,i} - B < \lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} + B$: This case violates 1 since for any value $j \in \{z_{p/2,i} - B, \ldots, z_{p/2,i} + B\}$, we have that $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r \cdot g^{z_{p/2,i}} \cdot g^j) \neq 0$.

  (c) $z_{p/2,i} + B < \lfloor p/2 \rfloor + \gamma_{1,i} < z_{p/2,i} + \delta_{0,i}$: This case is also impossible since, by condition 2, $\delta_{0,i}$ is the smallest integer (greater than 0) such that $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r \cdot g^{z_{p/2,i}} \cdot g^{\delta_{0,i}}) = 0$.

(ii) $\lfloor p/2 \rfloor + \gamma_{1,i} > z_{p/2,i} + \delta_{0,i}$: Assume that this is the case. Then $\gamma_{1,i}$ is not the smallest integer greater than 0 such that $\mathsf{PRF.Eval}(\mathsf{K}, h_i^r g^{\lfloor p/2 \rfloor + \gamma_{1,i}}) = 0$.

If $m_i = 1$, showing that $\bar{\alpha}_{0,i} = \alpha_{1,i}$ follows an identical reasoning as above.

[12] Recall that for a gaussian random variable X centered on 0 and with parameter σ, the probability that $|X| > \lambda\sigma$ is negligible in λ.

Wrapping up. We proved that if $m_i = 0$ then $\alpha_{0,i} = \bar{\alpha}_{0,i}$ and $\alpha_{1,i} = \bar{\alpha}_{1,i}$. On the other hand, if $m_i = 1$, then $\alpha_{0,i} = \bar{\alpha}_{1,i}$ and $\alpha_{1,i} = \bar{\alpha}_{0,i}$. Thus, if the encrypted value is $m_i = 0$

$$b_i = \mathsf{LEq}_<(\alpha_{0,i}, \alpha_{1,i}) = \mathsf{LEq}_<(\bar{\alpha}_{0,i}, \bar{\alpha}_{1,i})$$

and the value output by $\mathsf{DecShrink}_{\mathsf{DDH}}$ is 0. Else if $m_i = 0$

$$b_i = \mathsf{LEq}_<(\alpha_{0,i}, \alpha_{1,i}) \neq \mathsf{LEq}_<(\bar{\alpha}_{0,i}, \bar{\alpha}_{1,i})$$

and the value output by $\mathsf{DecShrink}_{\mathsf{DDH}}$ is 1. □

Lemma 11 (Runtime)**.** *Let* PRF *be a PRF,* $\tau = \log(8Bk)$ *and* $T = 2^\tau \lambda \log_e(k) + B(1 + 4k)$*. Then, the shrinking algorithm* $\mathsf{Shrink}_{\mathsf{DDH}}$ *described above terminates in polynomial time, except with negligible probability.*

Proof. The analysis of the runtime follows the same reasoning as the analysis of the runtime of the shrinking procedure from [BBD+20].

We have to show that the algorithm $\mathsf{Shrink}_{\mathsf{DDH}}$ is able to find a PRF key K that fulfills both conditions in expected polynomial time, since all other subroutines run in polynomial time. Here, we treat $\mathsf{PRF.Eval}$ as a truly random function. The same analysis is true for the case where $\mathsf{PRF.Eval}$ is a PRF except with negligible probability.

We first lower-bound the probability that a certain PRF key $\mathsf{K} \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$ satisfies condition 1. That is,

$$\begin{aligned}\Pr\left[\forall i \in [k], \forall j \in \{-B, \ldots, B\}, \begin{array}{c}\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \\ \wedge \\ \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0\end{array}\right] &\geq \left(1 - \frac{1}{2^\tau}\right)^{4Bk} \\ &\geq 1 - \frac{4Bk}{2^\tau} \\ &= 1 - \frac{1}{2} = \frac{1}{2}.\end{aligned}$$

Here, the first inequality comes from the fact that the outputs of $\mathsf{PRF.Eval}(K, \cdot)$ are uniform and independent over $\{0,1\}^\tau$ and the second inequality is simply Bernoulli's inequality.

We now upper-bound the probability that condition 2 is not met given that condition 1 happens. Let S be the set of PRF keys for which condition 1 is satisfied. Then

$$\begin{aligned}&\Pr\left[\exists i \in [k] \forall j \in \{B+1, \ldots, T\} : \begin{array}{c}\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \\ \vee \\ \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0\end{array} \middle| K \in S\right] \\ &\leq \sum_{i=1}^{k} \Pr\left[\forall j \in \{B+1, \ldots, T\} : \begin{array}{c}\mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot g^j) \neq 0 \\ \vee \\ \mathsf{PRF.Eval}(\mathsf{K}, c_{2,i} \cdot w \cdot g^j) \neq 0\end{array} \middle| K \in S\right] \\ &\leq \sum_{i=1}^{k} \left(1 - \frac{1}{2^\tau}\right)^{T-B-4Bk} \leq \sum_{i=1}^{k} e^{-(T-B-4Bk)/2^\tau} = \sum_{i=1}^{k} e^{-\lambda \log_e(k)} = e^{-\lambda}.\end{aligned}$$

Here, the first inequality is a simple consequence of the union bound and the second inequality follows from observing that K fixes $\mathsf{PRF.Eval}(\mathsf{K}, \cdot)$ on at most $4Bk$ points.

We conclude that, after λ iterations of the protocol, the probability that all the keys do not fulfill both conditions is negligible in λ. □

Ciphertext rate. After applying $\mathsf{Shrink}_{\mathsf{DDH}}$ we obtain a ciphertext composed by $\tilde{\mathsf{ct}} = (c_1, \mathsf{K}, (b_1, \ldots, b_k)) \in \mathbb{G} \times \mathcal{K} \times \{0,1\}^k$. Hence,

$$\frac{|\tilde{\mathsf{ct}}|}{|\mathbf{m}|} = \frac{|c_1| + |\mathsf{K}| + |(b_1, \ldots, b_k)|}{k} = \frac{2\lambda + k}{k} = 1 + \frac{2\lambda}{k}$$

which tends to 1 for large enough k.

5.2.2 Function-private LHE from DDH.

We now present our circuit-private LHE over $\mathbb{Z}_2$ based on DDH.

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

- $(\mathbb{G}, p, g) \leftarrow\!\$\, \mathcal{G}(1^\lambda)$
- Sample $x_1, \ldots, x_k \leftarrow\!\$\, \mathbb{Z}_p$. Compute $h_i = g^{x_i}$.
- Output $\mathsf{pk} = (\mathbb{G}, p, g, h_1, \ldots, h_k)$ and $\mathsf{sk} = \mathbf{x} = (x_1, \ldots, x_k)$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \ldots, m_k))$:

- Parse pk as $(\mathbb{G}, p, g, h_1, \ldots, h_k)$.
- Sample $r \leftarrow\!\$\, \mathbb{Z}_p$. Compute $c_1 = g^r$ and $c_{2,i} = h_i^r g^{m_i}$ for $i \in [k]$.
- Output $\mathsf{ct} = (c_1, (c_{2,1}, \ldots, c_{2,k}))$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$

- Parse pk as $(\mathbb{G}, p, g, h_1, \ldots, h_k)$, f as $f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \sum_{i=1}^{\ell} a_i \mathbf{x}_i + \mathbf{b}$ for $\mathbf{a} = (a_1, \ldots, a_\ell) \in \mathbb{Z}_2^\ell$ and $\mathbf{b} \in \mathbb{Z}_2^k$ and ct_i as $(c_{1,i}, \mathbf{c}_{2,i})$ where $\mathbf{c}_{2,i} = (c_{2,1,i}, \ldots, c_{2,k,i}))$ for $i \in [\ell]$.
- Compute $\bar{\mathsf{ct}} = (\bar{c}_1, (\bar{c}_{2,1}, \ldots, \bar{c}_{2,1}))$ where

$$\bar{c}_1 = \prod_{i=1}^{\ell} \left(c_{1,i}^{\lceil a_i \frac{p}{2} \rfloor_\sigma} \cdot (g \cdot c_{1,i}^{-1})^{\lceil 0 \rfloor_\sigma} \right) \cdot g^t$$

  and

$$\bar{\mathbf{c}}_2 = \bigodot_{i=1}^{\ell} \left(\mathbf{c}_{2,i}^{\lceil a_i \frac{p}{2} \rfloor_\sigma} \odot (g \cdot \mathbf{c}_{2,i}^{-1})^{\lceil 0 \rfloor_\sigma} \right) \odot \left(g^{\lceil b_1 \frac{p}{2} \rfloor_\sigma}, \ldots, g^{\lceil b_k \frac{p}{2} \rfloor_\sigma} \right) \odot (h_1^t, \ldots, h_k^t).$$

  for $t \leftarrow\!\$\, \mathbb{Z}_p$ and where $\odot$ denotes the component-wise multiplication.
- Output $\bar{\mathsf{ct}}$.

$\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$: Output $\bar{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{DDH}}(\mathsf{pk}, \mathsf{ct})$.

$\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$: Output $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{DDH}}(\mathsf{sk}, \bar{\mathsf{ct}})$.

Correctness and expected polynomial runtime of the LHE described above is guaranteed by Lemma 10 and Lemma 11 by setting $B > \lambda(\sigma(\sqrt{2\ell}+1))$. Semantic security of the scheme can be established by a simple reduction to the DDH assumption in a similar way as in many previous works (the reduction is similar to the one that proves that El Gamal is semantically secure). It is also easy to see that the scheme has rate-1 for large enough k.

We now show that the scheme is circuit private. Essentially, circuit privacy can be established by resorting to Lemma 4.

Lemma 12 (Circuit-privacy)**.** *The scheme presented above is circuit private.*

Proof. We need to show that we can simulate evaluated ciphertexts. We first present the simulator that receives $\tilde{\mathbf{m}} \leftarrow f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$.

$\mathsf{Sim}(\mathsf{pk}, \tilde{\mathbf{m}})$

- Sample $t \leftarrow\!\!\$\, \mathbb{Z}_q$ and $\alpha_i = \left\lceil \tilde{m}_i \frac{p}{2} \right\rfloor_{\sqrt{2\ell+1}\sigma}$.
- Compute $\tilde{\mathsf{ct}} = (\tilde{c}_1, (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,1}))$ where $\tilde{c}_1 = g^t$ and $\tilde{c}_{2,i} = h_i^t g^{\alpha_i}$. Output $\mathsf{ct}' \leftarrow \mathsf{Shrink}_{\mathsf{DDH}}(\mathsf{pk}, \tilde{\mathsf{ct}})$.

We now show that simulated ciphertexts are statistically indistinguishable from the ones output by Eval. Let $\mathsf{ct}_i = (g^{r_i}, (h_1^{r_i} g^{m_{1,i}}, \ldots, h_k^{r_i} g^{m_{k,i}})$. The output of Eval is $(\tilde{c}_1, (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,k}))$ where

$$\tilde{c}_1 = g^{\sum_i^\ell \left(r_i \left\lceil a_i \frac{p}{2} \right\rfloor_\sigma - r_i \lceil 0 \rfloor_\sigma\right) + t}$$

and

$$\tilde{c}_{2,j} = h^{\sum_i^\ell \left(r_i \left\lceil a_i \frac{p}{2} \right\rfloor_\sigma - r_i \lceil 0 \rfloor_\sigma\right) + t} \cdot g^{\sum_i^\ell \left(m_{j,i} \left\lceil a_i \frac{p}{2} \right\rfloor_\sigma + (1 - m_{j,i}) \lceil 0 \rfloor_\sigma\right) + \left\lceil b_i \frac{p}{2} \right\rfloor_\sigma}.$$

By Lemma 4 we have that

$$\sum_{i=1}^{\ell} \left(m_{j,i} \left\lceil a_i \frac{p}{2} \right\rfloor_\sigma + (1 - m_{j,i}) \lceil 0 \rfloor_\sigma\right) + \left\lceil b_i \frac{p}{2} \right\rfloor_\sigma \approx_s \left\lceil \tilde{m}_j \frac{p}{2} \right\rfloor_{\sqrt{2\ell+1}\sigma}$$

where $\tilde{m}_j$ is the j-th coordinate of $f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. Hence,

$$g^{\sum_{i=1}^\ell \left(m_{j,i} \left\lceil a_i \frac{p}{2} \right\rfloor_\sigma + (1 - m_{j,i}) \lceil 0 \rfloor_\sigma\right) + \left\lceil b_i \frac{p}{2} \right\rfloor_\sigma} \approx_s g^{\left\lceil f(\mathbf{m}_1, \ldots, \mathbf{m}_\ell) \frac{p}{2} \right\rfloor_{\sqrt{2\ell+1}\sigma}}.$$

Moreover, since $t \leftarrow\!\!\$\, \mathbb{Z}_p$ then

$$(g^{z+t}, h^{z+t}) \approx_s (g^t, h^t)$$

for any $z \in \mathbb{Z}_p$. We conclude that

$$(\tilde{c}_1, (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,k})) \approx_s (g^t, (h_1^t g^{\alpha_1}, \ldots, h_k^t g^{\alpha_k})$$

where $\alpha_i = \left\lceil \tilde{m}_i \frac{p}{2} \right\rfloor_{\sqrt{2\ell+1}\sigma}$. □

Larger plaintext space. As in the LWE case, in the construction presented above, the plaintext space is $\mathbb{Z}_2^k$. Both the shrinking algorithm and the function-private LHE schemes can be extended to support plaintext space $\mathbb{Z}_q^k$ where $q = \mathsf{poly}(\lambda)$ and $q = 2^\nu$ for some $\nu \in \mathbb{Z}$ (the constrain of q being a power of 2 comes from Lemma 4)

5.3 Construction from QR

The scheme presented in this section is the packed version of the scheme from [BG10] together with the shrinking technique from [DGI+19].

In the following, let N is a Blum integer if $N = p \cdot q$ for some primes p and q such that $p \pmod 4 = q \pmod 4 = 3$. Moreover, we say p and q are safe primes if $p = 2p' + 1$ and $q = 2q' + 1$ for some prime numbers p', q'. We denote by $\mathbb{J}_N$ the multiplicative group of the elements in $\mathbb{Z}_N^*$ with Jacobi symbol $+1$ and by $\mathbb{QR}_N$ the multiplicative group of quadratic residues modulo N with generator g. Note that $\mathbb{QR}_N$ is a subgroup of $\mathbb{J}_N$ and they have order $\frac{\varphi(N)}{4}$ and $\frac{\varphi(N)}{2}$, respectively, where $\varphi(\cdot)$ is Euler's totient function. It is useful to write $\mathbb{J}_N \simeq \mathbb{H} \times \mathbb{QR}_N$, where $\mathbb{H}$ is the multiplicative group $(\pm 1, \cdot)$ of order 2. Note that if N is a Blum integer then $\gcd\left(2, \frac{\varphi(N)}{4}\right) = 1$ and $-1 \in \mathbb{J}_N \setminus \mathbb{QR}_N$. We recall the quadratic residuosity (QR) assumption [GM82].

Definition 21 (Quadratic Residuosity Assumption)**.** *Let N be a uniformly sampled Blum integer and let $\mathbb{QR}_N$ be the multiplicative group of quadratic residues modulo N with generator g. We say the QR assumption holds with respect to $\mathbb{QR}_N$ if for any PPT adversary $\mathcal{A}$*

$$|\Pr[1 \leftarrow \mathcal{A}(N, g, a)] - \Pr[1 \leftarrow \mathcal{A}(N, g, (-1) \cdot a)]| \leq \mathsf{negl}(\lambda)$$

where $a \leftarrow\!\!\$\, \mathbb{QR}_N$.

5.3.1 Shrinking ciphertexts.

We recall the shrinking mechanism of [DGI+19]. Let a (packed) ciphertext $\mathsf{ct} = (g^r, (-1)^{b_1} h_1^r, \ldots, (-1)^{b_k} h_k^r) = (c_1, c_{2,1}, \ldots, c_{2,k})$ and let $<$ be an order over $\mathbb{J}_N$ (e.g., the lexicographic order). The shrinking mechanism of [DGI+19] simply outputs 0 if $c_{2,i} < -c_{2,i}$ and outputs 1 otherwise.

Lemma 13 ([DGI+19])**.** *There exists a correct shrinking procedure* $\mathsf{Shrink}_{\mathsf{QR}}, \mathsf{DecShrink}_{\mathsf{QR}}$ *for the packed QR-based PKE.*

5.3.2 Circuit-private LHE from QR.

We now present the scheme which is essentially the same as the one from [BG10] together with the shrinking technique of Lemma 13.

In the following, let $(\mathsf{Shrink}_{\mathsf{QR}}, \mathsf{DecShrink}_{\mathsf{QR}})$ be the pair of algorithms from Lemma 13.

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

- Choose two safe primes $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are primes and compute $N = pq$. Choose a generator g of $\mathbb{QR}_N$.
- Sample $\mathbf{s} \leftarrow\!\$\, \mathbb{Z}^k_{\phi(N)/2}$ and compute $\mathbf{h} = g^{\mathbf{s}}$.
- Output $\mathsf{pk} = (N, g, \mathbf{h})$ and $\mathsf{sk} = \mathbf{s}$.

$\mathsf{Enc}(\mathsf{pk}, \mathbf{m} = (m_1, \ldots, m_k))$:

- Parse pk as $(N, g, \mathbf{h} = (h_1, \ldots, h_k))$.
- Sample $r \leftarrow\!\$\, \mathbb{Z}_{(N-1)/2}$. Compute $c_1 = g^r$ and $c_{2,i} = (-1)^{m_i} h_i^r$ for $i \in [k]$.
- Output $\mathsf{ct} = (c_1, \mathbf{c}_2 = (c_{2,1}, \ldots, c_{2,k}))$.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell))$

- Parse pk as $(N, g, \mathbf{h} = (h_1, \ldots, h_k))$, f as $f(\mathbf{x}_1, \ldots, \mathbf{x}_\ell) = \sum_{j=1}^{\ell} a_j \mathbf{x}_j + \mathbf{b}$, where $a_1, \ldots, a_\ell \in \mathbb{Z}_2$, $\mathbf{b} \in \mathbb{Z}_2^k$ and ct_j as $(c_{1,j}, \mathbf{c}_{2,j} = (c_{2,1,j}, \ldots, c_{2,k,j}))$.
- Compute $\tilde{\mathsf{ct}} = (\tilde{c}_1, \tilde{\mathbf{c}}_2 = (\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,k}))$ where $\tilde{c}_1 = g^t \prod_{j=1}^{\ell} c_{1,j}^{a_j}$ and $\tilde{c}_{2,i} = h_i^t \cdot (-1)^{b_i} \cdot \prod_{j=1}^{\ell} c_{2,i,j}^{a_j}$ where $t \leftarrow\!\$\, \mathbb{Z}_{(N-1)/2}$. Output $\tilde{\mathsf{ct}}$.

$\mathsf{Shrink}(\mathsf{pk}, \mathsf{ct})$: Output $\tilde{\mathsf{ct}} \leftarrow \mathsf{Shrink}_{\mathsf{QR}}(\mathsf{pk}, \mathsf{ct})$.

$\mathsf{DecShrink}(\mathsf{sk}, \mathsf{ct})$: Output $\mathbf{m} \leftarrow \mathsf{DecShrink}_{\mathsf{QR}}(\mathsf{sk}, \mathsf{ct})$.

It is easy to see that correctness and compactness holds due to Lemma 13. Semantic security also follows easily from the QR assumption.

To see that the scheme is circuit private, note that $g^{(\sum_j r_j a_j)+t} \approx_s g^t$ for a uniformly chosen $t \leftarrow\!\$\, \mathbb{Z}_{(N-1)/2}$ (this holds since the uniform distribution over $\mathbb{Z}_{(N-1)/2}$ is statistically indistinguishable from the uniform distribution over $\mathbb{Z}_{\phi(N)/2}$). Similarly, we have that $h_i^{(\sum_j r a_j)+t} \approx_s h_i^t$. Thus,

$$\left(g^{(\sum_j r_j a_j)+t}, (-1)^{f(\mathbf{m})} h^{(\sum_j r_j a_j)+t}\right) \approx_s \left(g^t, (-1)^{f(\mathbf{m})} h^t\right)$$

and, thus, the distributions of an evaluated ciphertext and a fresh ciphertext are statistically indistinguishable.

6 Co-Private Information Retrieval

In this section, we present a new cryptographic primitive that we call *co-PIR*. In a co-PIR scheme, a receiver (with input a set of indices S) and a sender (with no input) interact such that, at the end, the sender obtains a string $\mathbf{y} \in \mathbb{Z}_q^m$ and receiver obtains $\mathbf{y}_{-S}$ (all positions of y except for the indices in S).

In terms of security, we require that the sender learns nothing about S, whereas the string $\mathbf{y}_S$ looks pseudorandom to the receiver. In terms of efficiency, we require that the total communication of the protocol scales only with $|S|\mathsf{poly}(\lambda)\mathsf{polylog}(m)$ (that is, it scales only poly-logarithmically with m). We present a construction for Co-PIR from the distributed GGM-PPRF correlation (as shown in [BCG+19a]) in Appendix E.1 of the full version paper; We also present another construction with black-box usage of PPRF and PIR in Appendix E.2 of the full version paper.

6.1 Definition

We start by defining Co-PIR and present its security properties.

Definition 22 (Co-PIR)**.** *A (two-round) Co-PIR scheme* CoPIR *over* $\mathbb{Z}_q$ *is parametrized by an integer* m *where* $m = \mathsf{poly}[\lambda]$*, and is composed by a tuple of algorithms* $(\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ *such that*

- $\mathsf{Query}(1^\lambda, S)$ *takes as input a set of indices* $S \subseteq [m]$*. It outputs a message* copir_1 *and a private state* st*.*
- $\mathsf{Send}(\mathsf{copir}_1)$ *takes as input a first message* copir_1*. It outputs a second message* copir_2 *and a string* $\mathbf{y} \in \mathbb{Z}_q^m$*.*
- $\mathsf{Dec}(\mathsf{copir}_2, \mathsf{st})$ *takes as input a second message* copir_2 *and a state* st*. It outputs a string* $\tilde{\mathbf{y}} \in \mathbb{Z}_q^m$*.*

Definition 23 (Correctness)**.** *A Co-PIR scheme* CoPIR *is said to be correct if for any* $m = \mathsf{poly}(\lambda)$ *and* $S \subseteq [m]$ *we have that*

$$\Pr\left[\mathbf{y}_{[m]\setminus S} = \tilde{\mathbf{y}}_{[m]\setminus S} : \begin{array}{c}(\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S) \\ (\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1) \\ \tilde{\mathbf{y}} \leftarrow \mathsf{Retrieve}(\mathsf{copir}_2, \mathsf{st})\end{array}\right] = 1.$$

In other words, the strings $\mathbf{y}$ *and* $\tilde{\mathbf{y}}$ *match for every coordinate* $i \in [m] \setminus S$*.*

In terms of security, we require two properties: receiver security and sender security.

Definition 24 (Receiver security)**.** *A Co-PIR scheme* CoPIR *is said to be receiver secure if for all* $m = \mathsf{poly}(\lambda)$*, any subsets* $S_1, S_2 \subseteq [m]$ *we have that for any adversary* $\mathcal{A}$

$$\left| \begin{array}{c}\Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{copir}_1) : (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S_1)\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{copir}_1) : (\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S_2)\right]\end{array} \right| \leq \mathsf{negl}(\lambda).$$

Definition 25 (Sender security)**.** *A Co-PIR scheme* CoPIR *is said to be sender secure if for any* $m = \mathsf{poly}(\lambda)$*, any subset* $S \subseteq [m]$ *we have that for all adversaries* $\mathcal{A}$

$$\left| \begin{array}{c}\Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{st}, \mathsf{copir}_2, \mathbf{y}_S) : \begin{array}{c}(\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S) \\ (\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1, \mathbf{x})\end{array}\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(k, \mathsf{st}, \mathsf{copir}_2, \mathbf{y}'_S) : \begin{array}{c}(\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S) \\ (\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1, \mathbf{x}) \\ \mathbf{y}'_S \leftarrow\$ \mathbb{Z}_q^{|S|}\end{array}\right]\end{array} \right| \leq \mathsf{negl}(\lambda).$$

Definition 26 (Compactness)**.** *A Co-PIR scheme* CoPIR *is said to be compact if* $|\mathsf{copir}_1|, |\mathsf{copir}_2| = |S| \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$ *for any* $S \subseteq [m]$ *where* $(\mathsf{copir}_1, \mathsf{st}) \leftarrow \mathsf{Query}(1^\lambda, S)$ *and* $(\mathsf{copir}_2, \mathbf{y}) \leftarrow \mathsf{Send}(\mathsf{copir}_1)$*. In other words, the communication complexity depends only poly-logarithmically in* m*.*

6.2 Co-PIR from Distributed GGM-PPRF Correlation

We now present a scheme for Co-PIR from the distributed GGM-PPRF correlation which is proposed by Boyle et al. [BCG+19a]. For the sake of simplicity we present the scheme for $q = 2$. Let $\mathsf{PPRF}_{\mathsf{GGM}} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Punct}, \mathsf{EvalPunct})$ be a GGM puncturable PRF which maps from $[m]$ to $\{0,1\}$ and let $\mathsf{PPRF\text{-}GGM} = (\mathsf{R}_1, \mathsf{S}, \mathsf{R}_2)$ be a distributed GGM-PPRF correlation scheme.

$\mathsf{Query}(1^\lambda, S)$:

- Parse $S = \{a_1, \ldots, a_t\} \subseteq [m]^t$ where $t = |S|$.
- For $j \in [t]$ compute $(\mathsf{pprf\text{-}ggm}_{1,j}, \mathsf{state}_j) \leftarrow \mathsf{PPRF\text{-}GGM.R}_1(\bar{a}_j)$.
- Output $\mathsf{copir}_1 = \{\mathsf{pprf\text{-}ggm}_{1,j}\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.

$\mathsf{Send}(\mathsf{copir}_1)$:

- Parse $\mathsf{copir}_1 = \{\mathsf{pprf\text{-}ggm}_{1,j}\}_{j\in[t]}$.
- For $j \in [t]$ compute $\mathsf{K}_j \leftarrow \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{KeyGen}(1^\lambda)$ and $\mathbf{z}_j \leftarrow \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_j, *)$.
- For $j \in [t]$ compute $\mathsf{pprf\text{-}ggm}_{2,j} \leftarrow \mathsf{PPRF\text{-}GGM.S}(0, \mathsf{K}_j, \mathsf{pprf\text{-}ggm}_1)$.
- Output $\mathsf{copir}_2 = \{\mathsf{pprf\text{-}ggm}_{2,j}\}_{j\in[t]}$ and $\mathbf{y} = \sum_{i=1}^{t} \mathbf{z}_j$

$\mathsf{Dec}(\mathsf{copir}_2, \mathsf{st})$:

- Parse $\mathsf{copir}_2 = \{\mathsf{pprf\text{-}ggm}_{2,j}\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.
- For $j \in [t]$ compute $\bar{\mathsf{K}}_j \leftarrow \mathsf{PPRF\text{-}GGM.R}_2(\mathsf{state}_j, \mathsf{pprf\text{-}ggm}_{2,j})$.
- For $i \in [m] \setminus S$, set $\tilde{y}_i = \sum_{i=1}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\bar{\mathsf{K}}_j, i)$. For $i \in S$, set $\tilde{y}_i = 0$. Output $\tilde{\mathbf{y}} = (\tilde{y}_1 \ldots, \tilde{y}_m)$.

We now analyze the scheme presented above starting with correctness.

Lemma 14 (Correctness)**.** *Assume that* PPRF-GGM *and* PPRF *are correct. Then the scheme presented above is correct*

Proof. We have to prove that $\tilde{\mathbf{y}}_{[m]\setminus S} = \mathbf{y}_{[m]\setminus S}$. Let $\mathbf{y} = (y_1, \ldots, y_m) \in \{0,1\}^m$. Note that $y_i = \sum_{j=1}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_j, i)$.

First, by the correctness of the underlying distributed GGM-PPRF correlation scheme, $\bar{\mathsf{K}}_j \leftarrow \mathsf{PPRF.Punct}(a_j)$ for all $j \in [t]$ and $a_j \in S$. Also,

$$\tilde{y}_i = \sum_{j=1}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\bar{\mathsf{K}}_j, i)$$

for all $i \in [m] \setminus S$. By the correctness of the PPRF, $\mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\bar{\mathsf{K}}_j, i) = \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{EvalPunct}(\mathsf{K}_j, i)$ for all $i \in [m] \setminus S$. Then $\tilde{y}_i = y_i$ for all $i \in [m] \setminus S$. □

Lemma 15 (Receiver security)**.** *Assume that* PPRF-GGM *implements* $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$*. Then the scheme presented above is receiver secure.*

The proof follows directly from the receiver security of PPRF-GGM.

Lemma 16 (Sender security)**.** *Assume that* PPRF-GGM *implements* $\mathcal{F}_{\mathsf{PPRF\text{-}GGM}}$ *and* PPRF *is a pseudorandom PPRF. Then the scheme presented above is sender secure.*

Proof. Let $\mathsf{Sim}_{\mathsf{PPRF}}\text{-}\mathsf{GGM}$ be the simulator of PPRF-GGM for sender security. The proof of security follows the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real protocol.

For all $j \in [t]$ consider the following sub-hybrids.

Hybrid $\mathcal{H}_{1,j}$. In this hybrid, we replace $\mathsf{pprf\text{-}ggm}_{2,j}$ by the message generated by $\mathsf{Sim}_{\mathsf{PPRF}\text{-}\mathsf{GGM}}$. Indistinguishability of hybrids follows from the sender security of PPRF-GGM.

Hybrid $\mathcal{H}_{2,j}$. In this hybrid, we replace $\mathsf{PRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_j, a_j)$ by a uniform bit $u_j \leftarrow\$ \{0,1\}$. Indistinguishability of hybrids follows from the pseudorandomness of PPRF.

Hybrid $\mathcal{H}_{3,j}$. In this hybrid, we replace y_{a_j} by $v_j \leftarrow\$ \{0,1\}$. Statistical indistinguishability follows because

$$y_{a_j} = \sum_{i=1}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_i, a_j) = \sum_{i=1, i\neq j}^{t} \mathsf{PPRF}_{\mathsf{GGM}}.\mathsf{Eval}(\mathsf{K}_i, a_j) + u_j \approx_s v_j.$$

Finally, note that in hybrid $\mathcal{H}_{3,t}$ the string $\mathbf{y}_S$ is uniformly random to the receiver and we conclude the proof □

Compactness. To conclude, we analyze compactness of the scheme. Assuming that the distributed GGM-PPRF correlation scheme has polynomial communication complexity in $|a_j| = \log m$ and in λ, and $|S| = t$, we conclude that the receiver's and the sender's message are of size $t \cdot \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(m)$.

Extending to Co-PIR over any $\mathbb{Z}_q$. The scheme can be easily extended to any q by taking a PPRF that maps $x \in [m]$ to $\mathbb{Z}_q$. It is easy to see that the resulting scheme has total communication complexity of $t \cdot \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(m, q)$.

Hardness assumptions for Co-PIR. Since the distributed GGM-PPRF correlation scheme and the GGM-PPRF can be based on LWE, DDH or QR assumptions (using only black-box techniques), then the Co-PIR scheme presented above can also be based on these assumptions. Moreover, the resulting scheme uses only black-box techniques.

6.3 Co-PIR from PPRF and PIR

The construction for Co-PIR from Section 6 uses a distributed GGM-PPRF correlation scheme which can be built from a GGM-PPRF and an OT. In this section we present a construction for Co-PIR from any PPRF (not necessarily the GGM-PPRF) and a PIR in a black-box way.

6.3.1 The Protocol

For the sake of simplicity we present the scheme for $q = 2$.

For our Co-PIR construction, we will need the following ingredients: Let $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ be a PIR scheme with poly-logarithmic communication complexity and sender privacy and let $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Punct}, \mathsf{EvalPunct})$ be a puncturable PRF which maps from $[m]$ to $\{0,1\}$. We use the notation $\mathsf{PPRF.Eval}(\mathsf{K}, *)$ to denote the vector $(\mathsf{PPRF.Eval}(\mathsf{K}, 1), \ldots, \mathsf{PPRF.Eval}(\mathsf{K}, m)) \in \{0,1\}^m$.

$\mathsf{Query}(1^\lambda, S)$:

- Parse $S = \{a_1, \ldots, a_t\}$ where $t = |S|$.
- For $j \in [t]$ compute $(\mathsf{q}_j, \mathsf{state}_j) \leftarrow \mathsf{PIR.Query}(a_j)$.
- Output $\mathsf{copir}_1 = \{\mathsf{q}_j\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.

$\mathsf{Send}(\mathsf{copir}_1)$:

- Parse $\mathsf{copir}_1 = \{\mathsf{q}_j\}_{j\in[t]}$.

- For $j \in [t]$ compute $\mathsf{K}_j \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$ and $\mathbf{z}_j \leftarrow \mathsf{PPRF.Eval}(\mathsf{K}_j, *)$.
- For $i \in [j]$ and $\ell \in [m]$, set $\dot{\mathsf{K}}_{j,\ell} \leftarrow \mathsf{PPRF.Punct}(\mathsf{K}_j, \ell)$.
- For $j \in [t]$ set $\mathbf{DB}_j = (\dot{\mathsf{K}}_{j,1}, \ldots, \dot{\mathsf{K}}_{j,m})$. Compute $\mathsf{r}_j \leftarrow \mathsf{PIR.Send}(\mathbf{DB}_j, \mathsf{q}_j)$.
- Output $\mathsf{copir}_2 = \{\mathsf{r}_j\}_{j\in[t]}$ and $\mathbf{y} = \sum_{i=1}^{t} \mathbf{z}_j$

$\mathsf{Dec}(\mathsf{copir}_2, \mathsf{st})$:

- Parse $\mathsf{copir}_2 = \{\mathsf{r}_j\}_{j\in[t]}$ and $\mathsf{st} = \{\mathsf{state}_j\}_{j\in[t]}$.
- For $j \in [t]$ compute $\bar{\mathsf{K}}_j \leftarrow \mathsf{PIR.Retrieve}(\mathsf{r}_j, \mathsf{state}_j)$.
- For $i \in [k] \setminus S$, set $\tilde{y}_i = \sum_{i=1}^{t} \mathsf{PPRF.EvalPunct}(\bar{\mathsf{K}}_j, i)$. For $i \in S$, set $\tilde{y}_i = 0$. Output $\tilde{\mathbf{y}} = (\tilde{y}_1 \ldots, \tilde{y}_m)$.

6.3.2 Analysis

We now analyze the scheme presented above starting with correctness.

Lemma 17 (Correctness)**.** *Assume that* PIR *and* PPRF *are correct. Then the scheme presented above is correct*

Proof. We have to prove that $\tilde{\mathbf{y}}_{[k]\setminus S} = \mathbf{y}_{[k]\setminus S}$. Let $\mathbf{y} = (y_1, \ldots, y_m) \in \{0,1\}^m$. Note that $y_i = \sum_{j=1}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_j, i)$.

First, by the correctness of the underlying PIR scheme, $\bar{\mathsf{K}}_j = \dot{\mathsf{K}}_{j,a_j}$ for all $j \in [t]$ and $a_j \in S$. Also,

$$\tilde{y}_i = \sum_{j=1}^{t} \mathsf{PPRF.EvalPunct}(\bar{\mathsf{K}}_j, i) = \sum_{i=1}^{t} \mathsf{PPRF.EvalPunct}(\dot{\mathsf{K}}_{j,a_j}, i)$$

for all $i \in [k] \setminus S$. By the correctness of the PPRF, $\mathsf{PPRF.EvalPunct}(\dot{\mathsf{K}}_{j,a_j}, i) = \mathsf{PPRF.EvalPunct}(\mathsf{K}_j, i)$ for all $i \in [k] \setminus S$. Then $\tilde{y}_i = y_i$ for all $i \in [k] \setminus S$. □

Lemma 18 (Receiver security)**.** *Assume that* PIR *is user secure. Then the scheme presented above is receiver secure.*

The proof follows from a simple reduction from the receiver security of CoPIR to user security of PIR.

Lemma 19 (Sender security)**.** *Assume that* PIR *is sender secure and* PPRF *is a pseudorandom PPRF. Then the scheme presented above is sender secure.*

Proof. The proof of security follows the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This is the real protocol.

For all $j \in [t]$ consider the following sub-hybrids.

Hybrid $\mathcal{H}_{1,j}$. In this hybrid, we replace DB_j by $\overline{DB}_j$ which is 0 everywhere but its a_j-th coordinate is equal to DB_{j,a_j}. Indistinguishability of hybrids follows from the sender security of PIR.

Hybrid $\mathcal{H}_{2,j}$. In this hybrid, we replace $\mathsf{PRF.Eval}(\mathsf{K}_j, a_j)$ by a uniform bit $u_j \leftarrow\!\!\$\, \{0,1\}$. Indistinguishability of hybrids follows from the pseudorandomness of PPRF.

Hybrid $\mathcal{H}_{3,j}$. In this hybrid, we replace y_{a_j} by $v_j \leftarrow\!\!\$\, \{0,1\}$. Statistical indistinguishability follows because

$$y_{a_j} = \sum_{i=1}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_i, a_j) = \sum_{i=1, i\neq j}^{t} \mathsf{PPRF.Eval}(\mathsf{K}_i, a_j) + u_j \approx_s v_j.$$

Finally, note that in hybrid $\mathcal{H}_{3,t}$ the string $\mathbf{y}_S$ is uniformly random to the receiver and we conclude the proof □

Compactness. To conclude, we analyze compactness of the scheme. Assuming that the PIR scheme has poly-logarithmic communication complexity and $|S| = t$, we conclude that the receiver's and the sender's message are of size $t \cdot \mathsf{poly}(\lambda) \cdot \mathsf{polylog}(m)$.

7 Oblivious Transfer with Overall Rate 1

We will now provide our construction of an oblivious transfer protocol with overall rate 1.

Ingredients. We will make use of the following ingredients.

- A packed linearly homomorphic encryption scheme $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Shrink}, \mathsf{DecShrink})$ with plaintext space $\{0,1\}^\ell$ and a post homomorphism shrinking procedure Shrink which converts ciphertexts into a rate 1 representation.[13]
- The binary $\mathsf{LPN}(n, m, \rho)$ problem with dimension $n = \mathsf{poly}(n)$, $m = n \cdot \ell \cdot \mathsf{poly}(n)$ samples and slightly sub-constant noise-rate $\rho = m^{1-\epsilon}$.
- A 2-round PIR scheme $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ with poly-logarithmic communication complexity and sender privacy.
- A 2-round Co-PIR scheme $\mathsf{CoPIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ over $\mathbb{Z}_2$ parametrized by m.

Additional Notation. Furthermore, to declutter notation we define the following embedding functions.

$\mathsf{RowMatrix}(\ell, n, \mathbf{v}_1, \ldots, \mathbf{v}_\ell)$**:** Takes row-vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \{0,1\}^n$ and outputs a matrix

$$\mathbf{V} = \begin{pmatrix} — & \mathbf{v}_1 & — \\ & \vdots & \\ — & \mathbf{v}_\ell & — \end{pmatrix},$$

i.e. for every $i \in [\ell]$ the i-th row of $\mathbf{V}$ is the row-vector $\mathbf{v}_i$.

$\mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{v})$**:** Takes a row-vector $\mathbf{v} \in \{0,1\}^n$ and outputs a matrix

$$\mathbf{V} = \begin{pmatrix} 0 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & 0 \\ — & \mathbf{v} & — \\ 0 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & 0 \end{pmatrix},$$

i.e. the i-th row of $\mathbf{V}$ is $\mathbf{v}$, but $\mathbf{V}$ is 0 everywhere else.

$\mathsf{Diag}(n, \mathbf{v})$**:** Takes a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \{0,1\}^n$ and outputs a matrix

$$\mathbf{D} = \begin{pmatrix} v_1 & & 0 \\ & \ddots & \\ 0 & & v_n \end{pmatrix},$$

i.e. $\mathbf{D} \in \{0,1\}^{n \times n}$ is a diagonal matrix with the components of $\mathbf{v}$ on its diagonal.

[13] Recall that we use the notation $\mathsf{Eval}\&\mathsf{Shrink}$ to denote the composition of algorithms Eval and Shrink.

We observe the following:

- For any $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \{0,1\}^n$ it holds that

$$\mathsf{RowMatrix}(\ell, n, \mathbf{v}_1, \ldots, \mathbf{v}_\ell) = \sum_{i=1}^{\ell} \mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{v}_i).$$

- For $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ it holds that

$$\mathbf{x} \cdot \mathsf{Diag}(n, \mathbf{y}) = \mathbf{x} \odot \mathbf{y},$$

  where $\odot$ denotes component-wise multiplication.

7.1 The Protocol

The protocol $\mathsf{OT} = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ is given as follows.

$\mathsf{OTR}(\mathbf{b} \in \{0,1\}^{m\ell})$:

- Parse $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$, where the $\mathbf{b}_i \in \{0,1\}^m$ are blocks of size m.
- Choose $\mathbf{A} \leftarrow\!\!\$\, \{0,1\}^{n \times m}$ uniformly at random and compute a pair of public and secret key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, \ell)$.
- For all $i \in [\ell]$, choose $\mathbf{s}_i \leftarrow\!\!\$\, \{0,1\}^n$, and $\mathbf{e}_i \leftarrow\!\!\$\, \chi_{m,t}$, compute $\mathbf{c}_i \leftarrow \mathbf{s}_i\mathbf{A} + \mathbf{e}_i + \mathbf{b}_i$, and set $\mathbf{S}_i \leftarrow \mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{s}_i)$. Compute a matrix-ciphertext $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_i)$.
- For all $i \in [\ell]$ set $J_i = \mathsf{Supp}(\mathbf{e}_i)$ to be the support of $\mathbf{e}_i$. Compute $(\mathsf{copir}_{1,i}, \mathsf{st}_i) \leftarrow \mathsf{CoPIR.Query}(J_i)$. Additionally, for $j \in [t]$ compute $(\mathsf{q}_{i,j}, \hat{\mathsf{st}}_{i,j}) = \mathsf{PIR.Query}(J_i[j])$.
- Output $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}_{1,i}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$ and $\mathsf{st} = (\mathsf{sk}, \{\mathsf{st}_i, J_i\}_{i\in[\ell]}, \{\hat{\mathsf{st}}_{i,j}\}_{i\in[\ell],j\in[t]]})$.

$\mathsf{OTS}((\mathbf{m}_0, \mathbf{m}_1) \in (\{0,1\}^{m\ell})^2, \mathsf{ot}_1)$:

- Parse $\mathbf{m}_0 = (\mathbf{m}_{0,1}, \ldots, \mathbf{m}_{0,\ell})$ and $\mathbf{m}_1 = (\mathbf{m}_{1,1}, \ldots, \mathbf{m}_{1,\ell})$, where each $\mathbf{m}_{b,i} = (m_{b,i,1}, \ldots, m_{b,i,m}) \in \{0,1\}^m$. Parse $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}_{1,i}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.
- For $i \in [\ell]$ $(\mathbf{y}_i, \mathsf{copir}_{2,i}) \leftarrow \mathsf{CoPIR.Send}(\mathsf{copir}_{1,i})$ where $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,m})$. Set $\mathbf{z}_i = \mathbf{m}_{0,i} + \mathbf{y}_i$.
- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- For all $i \in [\ell]$ set $\mathbf{C}_i = \mathsf{SingleRowMatrix}(\ell, m, i, \mathbf{c}_i)$ and $\mathbf{D}_i = \mathsf{Diag}(m, \mathbf{m}_{1,i} - \mathbf{m}_{0,i})$.
- Define the $\mathbb{Z}_2$-linear function $f : (\{0,1\}^{\ell\times n})^\ell \to \{0,1\}^{\ell\times m}$ via

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_\ell) = \left(\sum_{i=1}^{\ell} (-\mathbf{X}_i\mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i\right) + \mathbf{Z}.$$

- Compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{LHE.Eval\&Shrink}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
- For $i \in [\ell]$ set $DB_i = (y_{i,1} + (m_{1,i,1} - m_{0,i,1}), \ldots, y_{i,m} + (m_{1,i,m} - m_{0,i,m}))$. For all $j \in [t]$ compute $\mathsf{r}_{i,j} \leftarrow \mathsf{PIR.Send}(DB_i, \mathsf{q}_{i,j})$.
- Output $\mathsf{ot}_2 = \big(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.

$\mathsf{OTD}(\mathsf{ot}_2, \mathsf{st})$**:**

- Parse $\mathsf{ot}_2 = \big(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$ and $\mathsf{st} = (\mathsf{sk}, \{\mathsf{st}_i, J_i\}_{i\in[\ell]}, \{\hat{\mathsf{st}}_{i,j}\}_{i\in[\ell],j\in[t]]})$.
- For all $i \in [\ell]$ compute $\tilde{\mathbf{y}}_i = (\tilde{y}_{i,1}, \ldots, \tilde{y}_{i,m}) \leftarrow \mathsf{CoPIR.Retrieve}(\mathsf{copir}_{2,i}, \mathsf{st}_i)$.
- For $i \in [\ell]$ and $j \in [t]$ compute $\tilde{z}_{i,j} \leftarrow \mathsf{PIR.Retrieve}(\mathsf{r}_{i,j}, \hat{\mathsf{st}}_{i,j})$.

- For $i \in [\ell]$ set $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m})$ where

$$z_{i,l} = \begin{cases} \tilde{z}_{i,j} & \text{if } l = J_i[j] \\ \tilde{y}_{i,\ell} & \text{otherwise} \end{cases}.$$

- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- Compute $\tilde{\mathbf{W}} \leftarrow \mathsf{LHE.DecShrink}(\mathsf{sk}, \tilde{\mathsf{ct}})$ and $\mathbf{W} = \tilde{\mathbf{W}} - \mathbf{Z}$.
- Let $\mathbf{w}_1, \ldots, \mathbf{w}_\ell$ be the rows of $\mathbf{W}$. Output $\mathbf{w} = (\mathbf{w}_1 \| \ldots \| \mathbf{w}_\ell) \in \{0,1\}^{m\ell}$.

Correctness. We will first show that OT is correct, given that LHE, CoPIR and PIR are correct.

Theorem 1. *Assume that* LHE*,* CoPIR *and* PIR *are correct. Then the scheme presented above is correct.*

Proof. First note that by linear-homomorphic correctness of LHE it holds that

$$\begin{aligned} \tilde{\mathbf{W}} &= \mathsf{LHE.DecShrink}(\mathsf{sk}, \mathsf{LHE.Eval\&Shrink}(\mathsf{pk}, f, \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_1), \ldots, \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_\ell)) \\ &= f(\mathbf{S}_1, \ldots, \mathbf{S}_\ell) \\ &= \left(\sum_{i=1}^{k} (-\mathbf{S}_i \mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i \right) + \mathbf{Z} \end{aligned}$$

Let $\tilde{\mathbf{w}}_i$ be the i-th row of $\tilde{\mathbf{W}}$. It holds by definition $\mathbf{S}_i$, $\mathbf{C}_i$ and $\mathbf{Z}_i$ that

$$\begin{aligned} \tilde{\mathbf{w}}_i &= (-\mathbf{s}_i \mathbf{A} + \mathbf{c}_i)\mathbf{D}_i + \mathbf{z}_i \\ &= (-\mathbf{s}_i \mathbf{A} + \mathbf{s}_i \mathbf{A}_i + \mathbf{e}_i + \mathbf{b}_i)\mathbf{D}_i + \mathbf{m}_{0,i} + \mathbf{y}_i \\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{y}_i. \end{aligned}$$

where $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,m})$ is part of the output of $\mathsf{CoPIR.Send}$.

Let J_i be the support of $\mathbf{e}_i$ and let $\tilde{y}_i = (\tilde{y}_{i,1}, \ldots, \tilde{y}_{i,m}) \leftarrow \mathsf{CoPIR.Retrieve}(\mathsf{copir}_{2,i}, \mathsf{st}_i)$. By the correctness of the Co-PIR scheme CoPIR we have that $\tilde{y}_{i,j} = y_{i,j}$ for all $j \notin J_i$. On the other hand, by the correctness of the PIR scheme PIR it holds that

$$\tilde{z}_{i,j} = y_{i,j} + (m_{1,i,j} - m_{0,i,j})$$

for all $j \in J_i$. Consequently, we have that

$$z_{i,j} = \begin{cases} y_{i,j} + (m_{1,i,j} - m_{0,i,j}) & \text{if } l = J_i[j] \\ y_{i,j} & \text{otherwise} \end{cases}.$$

In other words, the term $(m_{1,i,j} - m_{0,i,j})$ only appears in the coordinates where $\mathbf{e}_i$ is equal to one. Then, it holds that

$$\mathbf{z}_i = \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{y}_i.$$

We conclude that

$$\mathbf{w} = \tilde{\mathbf{w}}_i - \mathbf{z}_i = \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i}.$$

Since $\mathbf{w} = (\mathbf{w}_1 \| \ldots \| \mathbf{w}_\ell)$ it follows that

$$\mathbf{w} = \mathbf{b} \odot (\mathbf{m}_1 - \mathbf{m}_0) + \mathbf{m}_0,$$

i.e. OT is correct. □

Communication complexity. We will now analyze the communication complexity of OT and show which choice of parameters leads to an overall rate approaching 1.

The bit-size of the message $\mathsf{ot}_1 = \left(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}_{1,i}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\right)$ can be bounded as follows.

- $|\mathsf{pk}| = \ell \cdot \mathsf{poly}(\lambda)$
- $|\mathbf{A}| = n \cdot m$
- $|\{\mathsf{ct}_i\}_{i\in[\ell]}| = \ell^2 \cdot n \cdot \mathsf{poly}(\lambda)$
- $|\{\mathbf{c}_i\}_{i\in[\ell]}| = \ell \cdot m$
- $|\{\mathsf{copir}_{1,i}\}_{i\in[\ell]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$.

Consequently, the overall upload-rate ρ_{up} can be bounded by

$$\begin{aligned}\rho_{\text{up}} &= \frac{|\mathsf{pk}| + |\mathbf{A}| + |(\mathsf{ct}_i)_{i\in[\ell]}| + |(\mathbf{c}_i)_{i\in[\ell]}| + |\{\mathsf{copir}_{1,i}\}_{i\in[\ell]}| + |(\mathsf{q}_{i,j})_{i\in[\ell],j\in[t]}|}{\ell m} \\ &\leq 1 + \frac{\mathsf{poly}(\lambda)}{m} + \frac{n}{\ell} + \frac{\ell \cdot n \cdot \mathsf{poly}(\lambda)}{m} + \frac{t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m} \\ &\leq 1 + \frac{n}{\ell} + \frac{\ell \cdot n \cdot \mathsf{poly}(\lambda)}{m} + \frac{t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m}.\end{aligned}$$

We get an overall upload rate of $\rho_{\text{up}} = 1 + O(1/\lambda)$ by choosing $\ell = \lambda \cdot n$ and $m = n^2 \cdot \mathsf{poly}(\lambda)$ for a sufficiently large $\mathsf{poly}(\lambda)$ depending on ϵ (where $t = m^{1-\epsilon}$).

The bit-size of the message $\mathsf{ot}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\right)$ can be bounded as follows.

- $|\tilde{\mathsf{ct}}| = \ell m(1 + \rho_{\mathsf{LHE}})$, where $1 + \rho_{\mathsf{LHE}}$ is the ciphertext rate of LHE.
- $|\{\mathsf{copir}_{2,i}\}_{i\in[\ell]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}| = \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$

Thus, the download-rate ρ_{down} can be bounded by

$$\rho_{\text{down}} = \frac{|\tilde{\mathsf{ct}}| + |\{\mathsf{copir}_{2,i}\}_{i\in[\ell]}| + |\{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}|}{\ell m} \leq 1 + \rho_{\mathsf{LHE}} + \frac{\ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m}.$$

By the above choice of m this comes down to $\rho_{\text{down}} \leq 1 + \rho_{\mathsf{LHE}} + O(1/\lambda)$.

7.2 Security

Receiver Security We now focus on the security of the scheme. We start by proving that the scheme is secure against semi-honest senders.

Theorem 2. *Assume that* LHE *is a semantic secure LHE scheme,* PIR *is a user-private PIR scheme,* CoPIR *is a receiver secure Co-PIR scheme and that the* $\mathsf{LPN}(n, m, \rho)$ *assumption holds for* $\rho = m^{1-\varepsilon}$ *for* $\varepsilon > 0$. *Then the scheme presented in Section 7.1 is receiver secure against semi-honest adversaries.*

Recall that the receiver's message is composed by LHE ciphertexts, LPN samples, Co-PIR and PIR first messages. In a nutshell, receiver security follows from the fact that the ciphertexts hide the LPN secret, the LPN samples hide the receiver's input $\mathbf{b}$ and finally the Co-PIR and PIR first messages hide the indices J_i.

Proof. We first present the simulator for a semi-honest sender. The simulator Sim receives the sender's input and sends it to the ideal functionality. Then it simulates the receiver as follows:

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

- Choose $\mathbf{A} \leftarrow\!\!\$\, \{0,1\}^{n\times m}$ and compute $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, \ell)$.
- For all $i \in [\ell]$, choose $\mathbf{c}_i \leftarrow\!\!\$\, \{0,1\}^m$ and compute $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{0})$.
- For all $i \in [\ell]$, let $J_i \subset [m]$ be a random subset of size t. Compute $\mathsf{copir}_{1,i} \leftarrow \mathsf{CoPIR}(J_i)$. Additionally, for $j \in [t]$ compute $(\mathsf{q}_{i,j}, \hat{\mathsf{st}}_{i,j}) = \mathsf{PIR.Query}(a_{i,j})$ where $a_{i,j} \leftarrow\!\!\$\, [m]$.
- Output $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.

We now show that the ideal-world and real-world executions are indistinguishable. The proof follows from the following sequence of hybrids.

Hybrid $\mathcal{H}_0$. This hybrid is the real experiment.

For $i \in [\ell]$, consider the following sub-hybrids.

Hybrid $\mathcal{H}_{1,i}$. Let $\mathcal{H}_{1,0} = \mathcal{H}_0$. This hybrid is identical to the previous one, except that the receiver computes $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{0})$.

Indistinguishability of hybrids $\mathcal{H}_{1,i-1}$ and $\mathcal{H}_{1,i}$ are indistinguishable, for $i = 1, \ldots, \ell$ and where $\mathcal{H}_{1,0} = \mathcal{H}_0$. follows from the semantic security of LHE.

Hybrid $\mathcal{H}_{2,i}$. Let $\mathcal{H}_{2,0} = \mathcal{H}_{1,\ell}$. This hybrid is identical to the previous one, except that the receiver computes $(\mathsf{copir}_{1,i}, \mathsf{st}_i) = \mathsf{CoPIR.Query}(J_i'[j])$ where J_i' is a uniformly subset of $[m]$ of size t.

Indistinguishability of hybrids $\mathcal{H}_{2,i-1}$ and $\mathcal{H}_{2,i}$, for $i = 1, \ldots, \ell$ and where $\mathcal{H}_{2,0} = \mathcal{H}_{1,\ell}$, follows directly from the receiver security of the underlying CoPIR.

Let $\phi : [\ell t] \to [\ell] \times [t]$ be a bijective function. For $i' \in [\ell t]$ consider the following hybrids.

Hybrid $\mathcal{H}_{3,i'}$. Let $\mathcal{H}_{3,0} = \mathcal{H}_{2,\ell}$. Let $\phi(i') = (i, j)$. This hybrid is identical to the previous one, except that the receiver computes $(\mathsf{q}_{i,j}, \hat{\mathsf{st}}_{i,j}) = \mathsf{PIR.Query}(a_{i,j})$ where $a_{i,j} \leftarrow\!\!\$\, [m]$.

Indistinguishability of hybrids $\mathcal{H}_{3,i-1}$ and $\mathcal{H}_{3,i}$, for $i = 1, \ldots, \ell t$ and where $\mathcal{H}_{3,0} = \mathcal{H}_{2,\ell}$, follows directly from the receiver security of the underlying PIR.

Finally for $i \in [\ell]$ consider the following sub-hybrids.

Hybrid $\mathcal{H}_{4,i}$. Let $\mathcal{H}_{4,0} = \mathcal{H}_{3,\ell t}$. This hybrid is identical to the previous one, except that the receiver samples $\mathbf{c}_i \leftarrow\!\!\$\, \{0,1\}^m$.

Indistinguishability of hybrids $\mathcal{H}_{4,i-1}$ and $\mathcal{H}_{4,i}$, for $i = 1, \ldots, \ell$ and where $\mathcal{H}_{4,0} = \mathcal{H}_{3,\ell t}$, follows directly from the LPN assumption.

Finally, note that hybrid $\mathcal{H}_{4,\ell}$ is identical to the ideal-world execution. This concludes the proof of receiver security. □

Sender Security

Theorem 3. *Assume that* LHE *is a statistically function-private LHE scheme,* PIR *is a sender-private PIR scheme and* CoPIR *is a sender-private Co-PIR scheme. Then the scheme presented in Section 7.1 is sender secure.*

Proof. We begin by presenting the simulator Sim against a semi-honest receiver. Recall that, in the semi-honest case, the simulator has access to the receiver's internal state. The simulator sends $\mathbf{b} = (b_1, \ldots, b_{m\ell})$ to the ideal functionality and receives $\tilde{\mathbf{m}} = (\tilde{m}_1, \ldots, \tilde{m}_{m\ell})$.

$\mathsf{Sim}(1^\lambda, \tilde{\mathbf{m}} \in \{0,1\}^{m\ell}, \mathbf{e} \in \{0,1\}^{m\ell}, \mathsf{ot}_1)$:

- Parse $\mathsf{ot}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i, \mathsf{copir}\}_{i\in[\ell]}, \{\mathsf{q}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.
- It sets $m_{b_i,i} = \tilde{m}_i$ and $m_{1-b_i,i} = 0$. Finally, it sets $\mathbf{m}_0 = (m_{0,1}, \ldots, m_{0,m\ell})$ and $\mathbf{m}_1 = (m_{1,1}, \ldots, m_{1,m\ell})$.
- For $i \in [\ell]$, let $J_i = \mathsf{Supp}(\mathbf{e}_i) := \{J_i[1], \ldots, J_i[t]\}$. Compute $(\mathsf{copir}_{2,i}, \mathbf{y}_i) \leftarrow \mathsf{CoPIR}(\mathsf{copir}_{1,i})$ where $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,m})$.
- For $i \in [\ell]$ and $j \in [t]$, choose $y'_{i,J_i[j]} \leftarrow\!\!\$\, \{0,1\}$ restricted to $y_{i,J_i[j]} = y'_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]})$. Set $\overline{\mathbf{DB}}_{i,j} = (0, \ldots, y'_{i,J_i[j]}, \ldots, 0)$. Compute $\mathbf{r}_{i,j} \leftarrow \mathsf{PIR.Send}(DB_{i,j}, \mathsf{q}_{i,j})$.
- Compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{LHE.Sim}(\mathsf{pk}, \mathbf{w}^*)$ where $\mathbf{w}^* := (\mathbf{w}^*_1, \ldots, \mathbf{w}^*_m)$ and $\mathbf{w}^*_i = \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{z}'_i$. Here, $\mathbf{z}'_i = (z_{i,1}, \ldots, z_{i,m})$ such that

$$z'_{i,j'} = \begin{cases} y'_{i,j} & \text{if } j' = J_i[j] \\ y_{i,j'} & \text{otherwise} \end{cases}$$

- Output $\mathsf{ot}_2 = \big(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i}\}_{i\in[\ell]}, \{\mathsf{r}_{i,j}\}_{i\in[\ell],j\in[t]}\big)$.

We will establish security via the following sequence of hybrids to show that the ideal-world experiment and the real-world one are indistinguishable.

Hybrid $\mathcal{H}_0$. This is the real experiment.

For $i' \in [\ell t]$ consider the following sub-hybrid. Let $\phi : [\ell t] \to [\ell] \times [t]$.

Hybrid $\mathcal{H}_{1,i'}$. Let $\mathcal{H}_{1,0} = \mathcal{H}_0$. Let $\phi(i') = (i,j)$. This hybrid is identical to the previous one except that we set $\overline{\mathbf{DB}}_{i,j} = (0, \ldots, y_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]}), \ldots, 0)$, i.e, $\overline{\mathbf{DB}}_{i,j}$ is set to 0 everywhere except for position $J_i[j]$ where it assumes the value $y_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]})$ as in the previous hybrid. Additionally, we compute $\mathsf{r}_{i,j} \leftarrow \mathsf{PIR.Send}(\overline{\mathbf{DB}}_{i,j}, \mathsf{q}_{i,j})$.

Indistinguishability of hybrids $\mathcal{H}_{1,i'-1}$ and $\mathcal{H}_{1,i'}$ follows from the sender security of PIR.

For $i \in [\ell]$ consider the following hybrid.

Hybrid $\mathcal{H}_{2,i}$. Let $\mathcal{H}_{2,0} = \mathcal{H}_{1,\ell t}$. This hybrid is identical to the previous one except that for all $j \in [t]$, choose $y'_{i,J_i[j]} \leftarrow \{0,1\}$ such that $y_{i,J_i[j]} = y'_{i,J_i[j]} - (m_{1,i,J_i[j]} - m_{0,i,J_i[j]})$.

Indistinguishability of hybrids $\mathcal{H}_{2,i-1}$ and $\mathcal{H}_{2,i}$ follows from the sender security of CoPIR.

Note that, in this hybrid $\overline{\mathbf{DB}}_{i,j}$ is of the form

$$\overline{\mathbf{DB}}_{i,j} = (0, \ldots, y'_{i,J_i[j]}, \ldots, 0).$$

Furthermore, note that we can write $\mathbf{z}_i$ as $\mathbf{z}_i = \mathbf{z}'_i - \mathbf{e} \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i})$, where $\mathbf{z}'_i = (z'_{i,1}, \ldots, z'_{i,m})$ is defined by

$$z'_{i,j'} = \begin{cases} y'_{i,j} & \text{if } j' = J_i[j] \\ \tilde{y}_{i,j'} & \text{otherwise} \end{cases}$$

where $\tilde{y}_{i,j} = y_{i,j}$ for $j \notin J_i$ by the correctness of CoPIR.

Finally, consider the remaining sub-hybrids.

Hybrid $\mathcal{H}_3$. In this hybrid we compute ct as follows: Set $\mathbf{W}^* \leftarrow f(\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$ and compute $\mathsf{ct} \leftarrow \mathsf{LHE.Sim}(\mathsf{pk}, \mathbf{W}^*)$, where $\mathsf{LHE.Sim}$ is the function-privacy simulator for LHE. Statistical indistinguishability between $\mathcal{H}_{2,\ell}$ and $\mathcal{H}_3$ follows from the statistical function privacy of LHE.

Hybrid $\mathcal{H}_4$. In this hybrid, we compute $\mathbf{W}^* = (\mathbf{w}_1^*, \ldots, \mathbf{w}_m^*)$ via

$$\mathbf{w}_i^* = \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{z}'_i.$$

Finally, note that

$$\begin{aligned}\mathbf{W}^* &= f(\mathbf{S}_1, \ldots, \mathbf{S}_\ell) \\ &= \left(\sum_{i=1}^{k} (-\mathbf{S}_i\mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i\right) + \mathbf{Z}.\end{aligned}$$

Let $\mathbf{w}_i^*$ be the i-th row of $\mathbf{W}^*$. It holds by definition $\mathbf{S}_i$, $\mathbf{C}_i$ and $\mathbf{Z}_i$ that

$$\begin{aligned}\mathbf{w}_i^* &= (-\mathbf{s}_i\mathbf{A} + \mathbf{c}_i)\mathbf{D}_i + \mathbf{z}_i \\ &= (-\mathbf{s}_i\mathbf{A} + \mathbf{s}_i\mathbf{A}_i + \mathbf{e}_i + \mathbf{b}_i)\mathbf{D}_i + \mathbf{m}_{0,i} + \mathbf{z}_i \\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{z}_i \\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + (\mathbf{z}'_i - \mathbf{e}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i})) \\ &= \mathbf{b}_i \odot (\mathbf{m}_{1,i} - \mathbf{m}_{0,i}) + \mathbf{m}_{0,i} + \mathbf{z}'_i\end{aligned}$$

Consequently, $\mathcal{H}_4$ is identical to the ideal experiment. □

Hardness assumptions for optimal-rate OT. When we instantiate the LHE with one of the schemes from Section 5, the Co-PIR with the construction from Section 6 and the PIR with a known black-box construction based on LWE, DDH or QR [DGI+19], we obtain the following corollary

Corollary 1. *Assuming the LWE, DDH or QR assumptions together with the* $\mathsf{LPN}(n, m, \rho)$*, there is a black-box construction for optimal-rate OT.*

8 Oblivious Matrix-Vector Product and Oblivious Linear Evaluation with Overall Rate 1

In this section we show how we can extend the techniques from the previous section to build protocols for OMV and OLE that achieve optimal rate.

8.1 OMV Protocol

We start by presenting a secure protocol for oblivious matrix-vector product (OMV). In an OMV functionality there is a sender, with input a matrix $\mathbf{M} \in \mathbb{Z}_q^{m \times m}$ and a vector $\mathbf{v} \in \mathbb{Z}_q^m$, and a receiver with input $\mathbf{b} \in \mathbb{Z}_q^m$. In the end, the receiver gets the value $\mathbf{bM} + \mathbf{v}$ but learns nothing about $\mathbf{M}$ and $\mathbf{v}$ whereas the sender learns nothing about $\mathbf{b}$.

We start by defining the functionality:

OMV functionality. The functionality $\mathcal{F}_{\mathsf{OMV}}$ is parametrized by integers $m = \mathsf{poly}(\lambda)$ and q and works as follows:

- **Receiver phase.** R sends $\mathbf{b}$ to $\mathcal{F}_{\mathsf{OMV}}$ where $\mathbf{b} \in \mathbb{Z}_q^m$.
- **Sender phase.** S sends $(\mathbf{M}, \mathbf{v})$ to $\mathcal{F}_{\mathsf{OMV}}$ where $\mathbf{M} \in \mathbb{Z}_q^{m \times m}$ and $\mathbf{v} \in \{0,1\}^m$. $\mathcal{F}_{\mathsf{OMV}}$ sends $\mathbf{bM}+\mathbf{v} \in \mathbb{Z}_q^m$ to R.

Below, we present a protocol for OMV that supports a sublinear number of multiplications in the size of the matrix. That is, all columns and rows of the matrix M should have bounded (sublinear in m) hamming weight.[14]

[14]Recall that hamming weight is used to count non-zero elements.

8.1.1 The Protocol

We start by presenting the ingredients that we need for our OMV protocol.

Ingredients. Let $q = \mathsf{poly}(\lambda)$. We will need the following ingredients.

- A packed linearly homomorphic encryption scheme $\mathsf{LHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Shrink}, \mathsf{DecShrink})$ with plaintext space $\mathbb{Z}_q^\ell$ and a post-homomorphism shrinking procedure Shrink which converts ciphertexts into a rate 1 representation.
- The binary $\mathsf{LPN}(n, m, \rho, q)$ problem with dimension $n = \mathsf{poly}(n)$, $m = n \cdot \ell \cdot \mathsf{poly}(n)$ samples and slightly sub-constant noise-rate $\rho = m^{1-\epsilon}$.
- A 2-round PIR scheme $\mathsf{PIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ with poly-logarithmic communication complexity and sender privacy.
- A 2-round Co-PIR scheme $\mathsf{CoPIR} = (\mathsf{Query}, \mathsf{Send}, \mathsf{Retrieve})$ over $\mathbb{Z}_q$ parametrized by $m(q-1)$.

We define the hamming weight of a matrix $\mathbf{D} \in \mathbb{Z}_q^{m \times m}$ to be the value $\mathsf{hw}(\mathbf{D}) = \max_i\{\mathsf{hw}(\mathbf{d}_i)\}, \mathsf{hw}(\mathbf{d}^{(i)})\}$ for all $i \in [m]$, where $\mathbf{d}_i$ and $\mathbf{d}^{(i)}$ are the i-th row and column of $\mathbf{D}$ respectively. In addition to the notation presented in Section 7, we present the following algorithm:

$\mathsf{AffineDecomp}(\mathbf{D} \in \mathbb{Z}_q^{m \times m})$: Takes a matrix $\mathbf{D}$ such that $\mathsf{hw}(\mathbf{D}) \leq \mu$ for all $i \in [m]$. It outputs $\mathbf{T}_1, \ldots, \mathbf{T}_\mu \in \mathbb{Z}_q^{m \times m}$ such that $\mathsf{hw}(\mathbf{T}_i) \leq 1$ for all $i \in [\mu]$ and $\mathbf{D} = \mathbf{T}_1 + \cdots + \mathbf{T}_\mu$.

Protocol. The protocol $\mathsf{OMV} = (\mathsf{OMVR}, \mathsf{OMVS}, \mathsf{OMVD})$ is presented below.

$\mathsf{OMVR}(\mathbf{b} \in \mathbb{Z}_q^{m\ell})$:

- Parse $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$, where the $\mathbf{b}_i \in \mathbb{Z}_q^m$ are blocks of size m.
- Choose $\mathbf{A} \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}$ uniformly at random and compute a pair of public and secret key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{LHE.KeyGen}(1^\lambda, \ell)$.
- For all $i \in [\ell]$, choose $\mathbf{s}_i \leftarrow\!\$\, \mathbb{Z}_q^n$, and $\mathbf{e}_i \leftarrow\!\$\, \chi_{m,t,q}$, compute $\mathbf{c}_i \leftarrow \mathbf{s}_i\mathbf{A} + \mathbf{e}_i + \mathbf{b}_i$, and set $\mathbf{S}_i \leftarrow \mathsf{SingleRowMatrix}(\ell, n, i, \mathbf{s}_i)$. Compute a matrix-ciphertext $\mathsf{ct}_i \leftarrow \mathsf{LHE.Enc}(\mathsf{pk}, \mathbf{S}_i)$.
- For all $i \in [\ell]$ set $J_i = \mathsf{Supp}(\mathbf{e}_i)$ to be the support of $\mathbf{e}_i$.
- For all $i \in [\ell]$ and $k \in [\mu]$ compute $(\mathsf{copir}_{1,i,k}, \mathsf{st}_{i,k}) \leftarrow \mathsf{CoPIR.Query}(J_i)$. Additionally, for all $j \in [t]$ compute $(\mathsf{q}_{i,k,j}, \hat{\mathsf{st}}_{i,k,j}) = \mathsf{PIR.Query}((q-1)(J_i[j]-1) + e_{i,J_i[j]})$.
- Output $\mathsf{omv}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i\}_{i\in[\ell]}, \{\mathsf{copir}_{1,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{q}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\big)$ and

$$\mathsf{st} = \big(\mathsf{sk}, \{J_i\}_{i\in[\ell]}, \{\mathsf{st}_{i,k}\}_{i\in[\ell],k\in[\mu]}, \{\hat{\mathsf{st}}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\big)\,.$$

$\mathsf{OMVS}((\mathbf{D}, \mathbf{v}) \in \mathbb{Z}_q^{m\times m\ell} \times \mathbb{Z}_q^{m\ell}, \mathsf{omv}_1)$:

- Parse $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_\ell)$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_\ell)$. If $\mathsf{hw}(\mathbf{D}) > \mu$ abort the protocol. Parse $\mathsf{omv}_1 = \big(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i\}_{i\in[\ell]}, \{\mathsf{copir}_{1,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{q}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\big)$.
- For $i \in [\ell]$ and $k \in [\mu]$ $(\mathbf{y}_{i,k}, \mathsf{copir}_{2,i,k}) \leftarrow \mathsf{CoPIR.Send}(\mathsf{copir}_{1,i,k})$ where $\mathbf{y}_{i,k} = (y_{i,k,1}, \ldots, y_{i,k,m})$.
- For all $i \in [\ell]$ set $\mathbf{z}_i = \mathbf{v}_i + \sum_{k=1}^{\mu} \mathbf{y}_{i,k}$.
- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- For all $i \in [\ell]$ set $\mathbf{C}_i = \mathsf{SingleRowMatrix}(\ell, m, i, \mathbf{c}_i)$.

- Define the $\mathbb{Z}_q$-linear function $f : (\mathbb{Z}_q^{\ell \times n})^\ell \to \mathbb{Z}_q^{\ell \times m}$ via

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_\ell) = \left(\sum_{i=1}^{\ell} (-\mathbf{X}_i \mathbf{A} + \mathbf{C}_i) \cdot \mathbf{D}_i \right) + \mathbf{Z}.$$

- Compute $\tilde{\mathsf{ct}} \leftarrow \mathsf{LHE.Eval\&Shrink}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
- For all $\in [\ell]$, set $(\mathbf{T}_{i,1}, \ldots, \mathbf{T}_{i,\mu}) \leftarrow \mathsf{AffineDecomp}(\mathbf{D}_i)$. Moreover, for all $k \in [\mu]$ and all $l \in [m]$, let $t_{i,k,l}$ be the only non-zero element in the l-th row of $\mathbf{T}_{i,k}$. If its l-th row is a zero vector, set $t_{i,k,l} = 0$.
- For all $i \in [\ell]$ and $k \in [\mu]$ set

$$DB_{i,k} = (y_{i,k,1} + t_{i,k,1}, y_{i,k,1} + 2 \cdot t_{i,k,1}, \ldots, y_{i,k,1} + (q-1) \cdot t_{i,k,1}, \ldots, y_{i,k,m} + (q-1) \cdot t_{i,k,m}),$$

  where $DB_{i,k}$ is a $(q-1)m$-sized vector. For all $j \in [t]$ compute $\mathsf{r}_{i,k,j} \leftarrow \mathsf{PIR.Send}(DB_{i,k}, \mathsf{q}_{i,k,j})$.
- Output $\mathsf{omv}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i,k}\}_{i \in [\ell], k \in [\mu]}, \{\mathsf{r}_{i,k,j}\}_{i \in [\ell], k \in [\mu], j \in [t]}\right)$.

$\mathsf{OMVD}(\mathsf{omv}_2, \mathsf{st})$**:**

- Parse $\mathsf{omv}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i,k}\}_{i \in [\ell], k \in [\mu]}, \{\mathsf{r}_{i,k,j}\}_{i \in [\ell], k \in [\mu], j \in [t]}\right)$ and

$$\mathsf{st} = \left(\mathsf{sk}, \{J_i\}_{i \in [\ell]}, \{\mathsf{st}_{i,k}\}_{i \in [\ell], k \in [\mu]}, \{\hat{\mathsf{st}}_{i,k,j}\}_{i \in [\ell], k \in [\mu], j \in [t]}\right).$$

- For all $i \in [\ell]$ and $k \in [\mu]$ compute $\tilde{\mathbf{y}}_{i,k} = (\tilde{y}_{i,k,1}, \ldots, \tilde{y}_{i,k,m}) \leftarrow \mathsf{CoPIR.Retrieve}(\mathsf{copir}_{2,i,k}, \mathsf{st}_{i,k})$.
- For $i \in [\ell]$, $k \in [\mu]$ and $j \in [t]$ compute $\tilde{z}_{i,k,j} \leftarrow \mathsf{PIR.Retrieve}(\mathsf{r}_{i,k,j}, \hat{\mathsf{st}}_{i,k,j})$.
- For all $i \in [\ell]$ and $k \in [\mu]$ set $\mathbf{z}_{i,k} = (z_{i,k,1}, \ldots, z_{i,k,m})$ where

$$z_{i,k,l} = \begin{cases} \tilde{z}_{i,k,j} & \text{if } l = J_i[j] \\ \tilde{y}_{i,k,l} & \text{otherwise} \end{cases}.$$

- For all $i \in [\ell]$ set $\mathbf{z}_i = \sum_{k=1}^{\mu} \mathbf{z}_{i,k}$.
- Set $\mathbf{Z} = \mathsf{RowMatrix}(\ell, m, \mathbf{z}_1, \ldots, \mathbf{z}_\ell)$.
- Compute $\tilde{\mathbf{W}} \leftarrow \mathsf{LHE.DecShrink}(\mathsf{sk}, \tilde{\mathsf{ct}})$ and $\mathbf{W} = \tilde{\mathbf{W}} - \mathbf{Z}$.
- Let $\mathbf{w}_1, \ldots, \mathbf{w}_\ell$ be the rows of $\mathbf{W}$. Output $\mathbf{w} = (\mathbf{w}_1 \| \ldots \| \mathbf{w}_\ell) \in \mathbb{Z}_q^{m\ell}$.

Correctness. We first show that the scheme presented above is correct.

Theorem 4 (Correctness)**.** *Assume that* LHE*,* CoPIR *and* PIR *are correct. Then the scheme presented above is correct.*

The proof follows the same reasoning as the proof of Theorem 1.

Communication complexity. We now analyze the communication complexity of OMV and show which choice of parameters leads to an overall rate approaching 1.

The bit-size of the message $\mathsf{omv}_1 = \left(\mathsf{pk}, \mathbf{A}, \{\mathsf{ct}_i, \mathbf{c}_i\}_{i \in [\ell]}, \{\mathsf{copir}_{1,i,k}\}_{i \in [\ell], k \in [\mu]}, \{\mathsf{q}_{i,k,j}\}_{i \in [\ell], k \in [\mu], j \in [t]}\right)$ can be bounded as follows:

- $q = \mathsf{poly}(\lambda)$
- $|\mathsf{pk}| = \ell \cdot \mathsf{poly}(\lambda)$
- $|\mathbf{A}| = n \cdot m \cdot \log q$

- $|\{\mathsf{ct}_i\}_{i\in[\ell]}| = \ell^2 \cdot n \cdot \mathsf{poly}(\lambda)$
- $|\{\mathbf{c}_i\}_{i\in[\ell]}| = \ell \cdot m \cdot \log q$
- $|\{\mathsf{copir}_{1,i,k}\}_{i\in[\ell]}| = \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m,q) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{q}_{i,k,j}\}_{i\in[\ell],j\in[t]}| = q \cdot \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$.

Thus, the overall upload-rate ρ_{up} can be bounded by

$$\rho_{\text{up}} \leq 1 + \frac{n \log q}{\ell} + \frac{\ell \cdot n \cdot \mathsf{poly}(\lambda)}{m} + \frac{\mu \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m}.$$

We get an overall upload rate of $\rho_{\text{up}} = 1 + O(1/\lambda)$ by choosing $\ell = \lambda \cdot n \log q$, $\mu = m^{1-\zeta}$ (for some $\zeta > 0$ such that $\zeta + \epsilon > 1$) and $m = n^2 \cdot \log q \cdot \mathsf{poly}(\lambda)$ for a sufficiently large $\mathsf{poly}(\lambda)$ depending on ϵ (where $t = m^{1-\epsilon}$).

The bit-size of the message $\mathsf{omv}_2 = \left(\tilde{\mathsf{ct}}, \{\mathsf{copir}_{2,i,k}\}_{i\in[\ell],k\in[\mu]}, \{\mathsf{r}_{i,k,j}\}_{i\in[\ell],k\in[\mu],j\in[t]}\right)$ can be bounded as follows:

- $q = \mathsf{poly}(\lambda)$
- $|\tilde{\mathsf{ct}}| = \ell m(1 + \rho_{\mathsf{LHE}})$, where $1 + \rho_{\mathsf{LHE}}$ is the ciphertext rate of LHE
- $|\{\mathsf{copir}_{2,i,k}\}_{i\in[\ell]}| = \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$
- $|\{\mathsf{r}_{i,k,j}\}_{i\in[\ell],j\in[t]}| = q \cdot \mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)$.

Thus, the download-rate ρ_{down} can be bounded by

$$\rho_{\text{down}} \leq 1 + \rho_{\mathsf{LHE}} + \frac{\mu \cdot \ell \cdot t \cdot \mathsf{polylog}(m) \cdot \mathsf{poly}(\lambda)}{m}.$$

By the above choice of m and μ this comes down to $\rho_{\text{down}} \leq 1 + \rho_{\mathsf{LHE}} + O(1/\lambda)$.

Security. Finally, we state the result that guarantees security of the scheme.

Theorem 5 (Security)**.** *The scheme presented above is:*

- *Receiver secure if* LHE *is a semantic secure LHE scheme,* PIR *is a user-private PIR scheme,* CoPIR *is a receiver secure Co-PIR scheme and that the* $\mathsf{LPN}(n, m, \rho, q)$ *assumption holds for* $\rho = m^{1-\varepsilon}$ *for* $\varepsilon > 0$*.*
- *Sender secure if* LHE *is a statistically function-private LHE scheme,* PIR *is a sender-private PIR scheme and* CoPIR *is a sender-private Co-PIR scheme.*

The proof of the theorem follows the same reasoning as the proof of Theorem 2 and Theorem 3.

Again, instantiating the ingredients used in OMV with the constructions from this work, we obtain the following corollary.

Corollary 2. *There exists a black-box construction for* OMV *over* $\mathbb{Z}_q$ *assuming:*

- *LWE and* $\mathsf{LPN}(n, m, \rho, q)$ *for* $q = \mathsf{poly}(\lambda)$
- *DDH and* $\mathsf{LPN}(n, m, \rho, q)$ *for* $q = 2^\mu = \mathsf{poly}(\lambda)$*.*

8.2 OLE Protocol

An oblivious linear evaluation (OLE) is a protocol between a sender, with input an affine function f, and a receiver, with input a point b. It allows for the receiver to obliviously learn $f(b)$. We now show how we can obtain an OLE using the OMV protocol presented in Section 8.1.

We start by defining the functionality:

OLE functionality. The functionality $\mathcal{F}_{\mathsf{OLE}}$ is parametrized by integers $k = \mathsf{poly}(\lambda)$ and q and works as follows:

- **Receiver phase.** R sends $\mathbf{b}$ to $\mathcal{F}_{\mathsf{OLE}}$ where $\mathbf{b} \in \mathbb{Z}_q^k$.
- **Sender phase.** S sends $(\mathbf{u}_0, \mathbf{u}_1)$ to $\mathcal{F}_{\mathsf{OLE}}$ where $\mathbf{u}_0, \mathbf{u}_1 \in \mathbb{Z}_q^k$. $\mathcal{F}_{\mathsf{OLE}}$ sends $\mathbf{b} \odot \mathbf{u}_0 + \mathbf{u}_1 \in \mathbb{Z}_q^k$ to R.

8.2.1 Protocol for Small Fields

We briefly sketch how we can construct an OLE scheme over $\mathbb{Z}_q$ where $q = \mathsf{poly}(\lambda)$. The protocol follows as a particular case of the protocol of Section 8.1. We give a brief overview of the scheme below.

Let Using the notation of Section 8.1, let $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_\ell) \in \mathbb{Z}_q^{m\ell}$ be the receiver's input and let $(\mathbf{u}_0 = (\mathbf{u}_{0,1}, \ldots, \mathbf{u}_{0,\ell}), \mathbf{u}_1 = (\mathbf{u}_{1,1}, \ldots, \mathbf{u}_{1,\ell})) \in (\mathbb{Z}_q^{m\ell})^2$ be the sender's input. To achieve OLE, the sender constructs the matrices $\mathbf{D}_i = \mathsf{Diag}(m, \mathbf{u}_{0,i})$ and sets $\mathbf{v}_i = \mathbf{u}_{1,i}$ for all $i \in [\ell]$. Then they run the OMV protocol where the receiver inputs $\mathbf{b}$ and the sender inputs $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_\ell)$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_\ell)$. It is easy to see that the output of the receiver is $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_\ell)$ where

$$\mathbf{y}_i = \mathbf{b}_i \mathbf{D}_i + \mathbf{v}_i = \mathbf{b}_i \odot \mathbf{u}_{0,i} + \mathbf{u}_{1,i}$$

be the correctness of the OMV protocol.

Moreover, $\mathsf{hw}(\mathbf{D}_i) = 1 \leq m^{1-\zeta}$ for some $\zeta > 0$ such that $\zeta + \epsilon > 1$. Thus the resulting protocol achieves overall rate 1. Finally, in terms of hardness assumptions, the OLE protocol inherits the same security.

8.2.2 Extending OLE to Larger Rings

Following [DGI+19], we briefly explain how we can achieve OLE over larger rings (which can potentially have super-polynomial size in λ).

OLE over $\mathbb{Z}_N = \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_\delta}$. Let $N = \prod_{i=1}^{\delta} q_i$ be an integer (which might be superpolynomial in λ) such that for all $i \in [\delta]$ $q_i = \mathsf{poly}(\lambda)$ are different prime numbers. Then, via the Chinese Remainder Theorem, $\mathbb{Z}_N$ is isomorphic to $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_\delta}$. Thus, performing an OLE over $\mathbb{Z}_N$ boils down to performing δ OLEs over each one of the smaller fields $\mathbb{Z}_{q_i}$. It is easy to see that, if each OLE over $\mathbb{Z}_{q_i}$ has overall rate 1, then the resulting OLE over $\mathbb{Z}_N$ also achieves overall rate 1.

OLE over extension fields. We now show how these techniques can be adapted to perform OLE over an extension field $\mathbb{F}_{q^k}$ of order q^k for a prime q. Here, we rely on the fact that multiplication over $\mathbb{F}_{q^k}$ can be expressed as a linear function over the field $\mathbb{Z}_q$. That is, suppose that an element $\mathbf{x} \in \mathbb{F}_{q^k}$ is of the form $\mathbf{x} = x_1 + x_2\alpha + \cdots + x_k\alpha^{k-1}$ where each $x_i \in \mathbb{Z}_q$ and α is a symbol. Then, for elements $\mathbf{a}, \mathbf{x} \in \mathbb{F}_{q^k}$ the product

$$\mathbf{xa} = f_{1,\mathbf{a}}(\mathbf{x}) + f_{2,\mathbf{a}}(\mathbf{x})\alpha + \cdots + f_{k,\mathbf{a}}(\mathbf{x})\alpha^{k-1}$$

where each $f_{i,\mathbf{a}}$ is a $\mathbb{Z}_q$-linear function which depends solely on $\mathbf{a}$.

Given this, we briefly describe how we can perform several OLEs over $\mathbb{F}_{q^k}$ while preserving overall rate 1. The receiver has input $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_t) \in \mathbb{F}_{q^k}^t$ such that $kt = m\ell$ and $k|m$ (using the same notation as in Section 8.1). It parses each $\mathbf{b}_i$ as a k-dimensional vectors $\bar{\mathbf{b}}_i \in \mathbb{Z}_q^k$. Then, it organizes all t vectors $\mathbf{b}_i$ in blocks $\mathbf{c}_i \in \mathbb{Z}_q^m$ of size m. It inputs $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_\ell)$ into the OMV protocol.

The sender, with input $\mathbf{u}, \mathbf{v} \in \mathbb{F}_{q^k}$ rearranges $\mathbf{u}, \mathbf{v}$ in the same way as the receiver and obtains $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_\ell), \mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_\ell)$ respectively. Then, for each $\mathbf{w}_i = (\mathbf{w}_{i,1}, \ldots, \mathbf{w}_{i,m/k})$, it computes the functions $f_{j,\mathbf{w}_{i,r}}$ for each $j \in [k]$, $i \in [\ell]$ and $r \in [m/k]$. Let $\mathbf{f}_{j,\mathbf{w}_{i,r}}$ be the vector composed by the coefficients of $f_{j,\mathbf{w}_{i,r}}$. The sender computes the matrices

$$\bar{\mathbf{D}}_{i,r} = \begin{pmatrix} | & & | \\ \mathbf{f}_{1,\mathbf{w}_{i,r}} & \ldots & \mathbf{f}_{k,\mathbf{w}_{i,r}} \\ | & & | \end{pmatrix}$$

and then sets

$$\mathbf{D}_i = \begin{pmatrix} \bar{\mathbf{D}}_{i,1} & & \\ & \ddots & \\ & & \bar{\mathbf{D}}_{i,m/k} \end{pmatrix}.$$

It inputs $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_\ell)$ and $\mathbf{z}$ into the OMV protocol.

It is easy to see that the receiver's output will be $\mathbf{b} \odot \mathbf{u} + \mathbf{v}$ where $\odot$ denotes component-wise multiplication over $\mathbb{F}_{q^k}$. Moreover, $\mathsf{hw}(\mathbf{D}_i) = k$. By choosing k such that $k \leq \mu = m^{1-\zeta}$ we achieve a protocol with overall rate 1. In particular, we can set the parameters such that $k = \lambda$ and we achieve an OLE over the field $\mathbb{F}_{q^\lambda}$ of exponential size.

Acknowledgment

Zvika Brakerski is supported by the Israel Science Foundation (Grant No. 3426/21), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

Pedro 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.

Nico Döttling and Sihang Pu were supported by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I- OO1 4).

References

[ABD+21] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 94–125, Cham, 2021. Springer International Publishing.

[AR16] Divesh Aggarwal and Oded Regev. A note on discrete gaussian combinations of lattice vectors. *Chicago Journal of Theoretical Computer Science*, 2016(7), June 2016.

[Ban93] W. Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(4):625–636, 1993.

[BBD+20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In *TCC 2020: 18th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 58–87. Springer, Heidelberg, Germany, March 2020.

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 291–308. ACM Press, November 11–15, 2019.

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 407–437, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[BDM22] Pedro Branco, Nico Döttling, and Paulo Mateus. Two-round oblivious linear evaluation from learning with errors. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 379–408, Cham, 2022. Springer International Publishing.

[BdMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[CDI+19] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 462–488, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[CGH+21] Melissa Chase, Sanjam Garg, Mohammad Hajiabadi, Jialin Li, and Peihan Miao. Amortizing rate-1 ot and applications to pir and psi. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 126–156, Cham, 2021. Springer International Publishing.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science*, pages 41–50, Milwaukee, Wisconsin, October 23–25, 1995. IEEE Computer Society Press.

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 122–138, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[Döt15] Nico Döttling. Low noise LPN: KDM secure public key encryption and sample amplification. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 604–626, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

[EGL82] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 205–210, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[GHO20] Sanjam Garg, Mohammad Hajiabadi, and Rafail Ostrovsky. Efficient range-trapdoor functions and applications: Rate-1 OT and more. In *TCC 2020: 18th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 88–116. Springer, Heidelberg, Germany, March 2020.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press.

[GMPW20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. Improved discrete gaussian and subgaussian analysis for lattice cryptography. In *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, Lecture Notes in Computer Science, pages 623–651. Springer, Heidelberg, Germany, 2020.

[GMW19] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. 2019.

[GNN17] Satrajit Ghosh, Jesper Buus Nielsen, and Tobias Nilges. Maliciously secure oblivious linear function evaluation with constant overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 629–659, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[GVW20] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. New constructions of hinting PRGs, OWFs with encryption, and more. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, Lecture Notes in Computer Science, pages 527–558, Santa Barbara, CA, USA, August 16–20, 2020. Springer, Heidelberg, Germany.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, Heidelberg, Germany, March 15–17, 2009.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 60–73, New York, NY, USA, 2021. Association for Computing Machinery.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press.

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 859–870, Paris, France, October 7–9, 2018. IEEE Computer Society Press.

[Rab05] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptol. ePrint Arch.*, 2005(187), 2005.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.