Obfuscation of Bloom Filter Queries from Ring-LWE

Alex Davidson
Email: alex.davidson.2014@rhul.ac.uk

Royal Holloway, University of London

Abstract. We devise a virtual black-box (VBB) obfuscator for querying whether set elements are stored within Bloom filters, with security based on the Ring Learning With Errors (RLWE) problem and strongly universal hash functions. Our construction uses an abstracted encoding scheme that we instantiate using the Gentry, Gorbunov and Halevi (GGH15) multilinear map, with an explicit security reduction to RLWE. This represents an improvement on the functionality and security guarantees compared with the conjunction obfuscator introduced by Brakerski et al. (ITCS 2016), where security follows from a non-standard RLWE variant. Immediate applications of our work arise from any common usage of Bloom filters, such as efficient set intersection testing. Our obfuscated program allows this functionality to be executed in a non-interactive manner, whilst preventing the natural leakage that occurs when providing offline access to a Bloom filter. Compared to more general obfuscators for evasive functions, we demonstrate a significant asymptotic reduction in size and required computation for obfuscating set intersection queries. The obfuscator of Wichs and Zirdelis (EPRINT 2017) requires $O(4^{n \log n})$ encodings for obfuscating circuits computing the intersection of sets of size n, requiring the usage of additional primitives such as FHE to allow sets of polynomial size. Our construction requires only $O(kn)$ total encodings and operations for evaluation, where $k << n$. Moreover, the size of our obfuscator is independent of the size of the elements that are contained in the set. Our results, alongside recent and concurrent work, can be seen as another step forward in obfuscating wider classes of evasive functions using standard assumptions and models.

1 Introduction

Program obfuscation [38, 5, 31] has become a central theoretical primitive in constructing "encrypted" programs that reveal nothing about their internal workings, whilst preserving their functionality. There are several definitions of security for a proposed obfuscator, the strongest (and perhaps most intuitive) is known as Virtual Black-Box (VBB) obfuscation. Informally, a circuit obfuscator satisfying VBB security is indistinguishable from an interaction with a simulator which only has oracle access to the underlying functionality. Thus, any information contained within the original circuit is naturally hidden by the obfuscation. Unfortunately, it was shown by Barak et al. [5] that VBB obfuscation is impossible to achieve for general programs. Since this initial work, there have been a number of diverging strands of research investigating what is possible with regards to program obfuscation.

Indistinguishability obfuscation. Firstly, the work of Garg et al. [31] developed an instantiation of an obfuscator for all circuits in **P/POLY** that is secure with respect to the security definition of 'indistinguishability' set out in [5]. This definition gives indistinguishability guarantees for an obfuscator acting on two circuits that compute equivalent functionalities. There have been several follow-up constructions such as [4, 34, 32] that use similar techniques to this initial work. These constructions are known as IO obfuscators. The usage of indistinguishability obfuscation as a central primitive has increased rapidly (for example [54, 15]) with various works achieving milestones of important cryptographic value.

All constructions of IO obfuscators rely on so-called cryptographic multilinear maps (MMAPs) [30, 20, 33] with security proven in associated generic models. Furthermore, there have been polynomial-time attacks on most of these IO constructions [50, 18, 19, 2] exploiting flaws in the generic models, and highlighting disparities between the model and the current constructions. Currently standing constructions of IO are based on little more than heuristics and it is a major open problem to solve whether we can build IO obfuscators for all circuits using standard assumptions and models.

VBB obfuscation. Undeterred by the impossibility result, many recent works have constructed VBB obfuscators when restricted to specific functionality classes. For example, Canetti [12] and Lynn et al. [45]

gave point-function obfuscators in the random oracle model, while Wee [56] developed a point-function obfuscator based on strong one-way functions. Other work has seen developments in hyperplane obfuscators [16] based on strong DDH, and more recently obfuscators for various families of evasive functions based on strong assumptions over MMAPs [8, 3]. Brakerski et al. [10] gave a construction of a VBB conjunction obfuscator based on a variant of Ring-LWE known as 'entropic RLWE'. Very recently and concurrently to this work, Wichs and Zirdelis [57] showed how to obfuscate a wide class of evasive 'compute-and-compare' functions based on the hardness of the LWE problem. Both of these works use the 'directed encodings' abstraction of the GGH15 [33] MMAP for obtaining their construction.

While uncertainty continues to surround constructions of IO obfuscators for general circuits from cryptographic multilinear maps, advances in more specific constructions based on well-known assumptions represent a separate and useful direction for future work.

Hardness assumptions over GGH15. As mentioned above, Brakerski et al. [10] constructed a directed encoding scheme with DDH-like and entropic security properties. Furthermore, they show that the GGH15 [33] MMAP instantiates such a scheme based on the 'entropic' Ring-LWE problem. This involves distinguishing RLWE samples from uniform where secrets are constructed multiplicatively from public Gaussian-sampled elements. This assumption is not derived directly from standard Ring-LWE and so it is unknown whether we can base the hardness of it on well-known lattice problems.

Numerous subsequent works have explored the possibility of instantiating complex primitives from standard assumptions using GGH15 as the foundation. These include obfuscating 'compute-and-compare' functionalities from LWE [57], constructing constraint-hiding, constrained PRFs from LWE [13]. Separately, Goyal et al. [37] demonstrated the separation of circular and IND-CPA security for symmetric-key bit encryption schemes from LWE using a variant of GGH15.

1.1 Our results

We devise a VBB obfuscator for evaluating queries on Bloom filters [6] with hardness based solely on the Ring-LWE problem and strongly universal hash functions. Our obfuscator constructs a public API for data storage mechanisms; allowing a user to query whether elements are already stored or not without giving away internal information. In particular, a direct application of our work allows for computing private set intersections (PSI) non-interactively. Our security guarantees are meaningful in the same manner as [10]; i.e. when the set that is hidden is sampled with sufficient min-entropy (specifically $\alpha(\lambda) = \lambda^{\epsilon}$ for a security parameter λ and some constant $\epsilon > 0$) relative to the universe that it is sampled from. There are several applications pertaining to private set intersection computation. For instance, using this functionality we are able to instantiate the class of evasive circuits where the obfuscator has knowledge of all satisfying inputs.

Our methods make use of the Graded External Diffie-Hellman (GXDH) hardness assumption over directed encoding schemes, first introduced by Brakerski et al. [10]. It is shown that the GXDH security property holds when instantiating the directed encoding scheme using GGH15, based on the hardness of standard Ring-LWE. In addition, the work of [10] requires a non-standard variant of Ring-LWE for proving that 'entropic' security of their scheme holds. In this work, we require a similar property that we name 'linear entropic security', the main difference being that our property can be instantiated in an additive manner; see Section 4 for full details. The advantage of this is that we are able to derive security directly from the hardness of the Ring-LWE problem in a sufficiently general case[1] – see Section 3. As a result, our construction makes advances in security over the work of [10] by removing the requirement for a non-standard variant of RLWE. In summary, security of our VBB obfuscator rests upon a directed encoding scheme that is computationally secure with respect to the GXDH and linear α-entropic (α-LE) security properties. We show that such a scheme can be securely instantiated from GGH15 based on the hardness of Ring-LWE alone. The techniques that used in constructing our obfuscator shares similarities with the design of [3] for obfuscating polynomial root evaluation.

Finally, while the concurrent and more general results of Wichs and Zirdelis [57] could potentially be leveraged to compute the same functionality – our work makes tangible gains in the efficiency of evaluation

[1] We require a large number of public samples and specific assumptions of the width of the Gaussian.

and in the size of the obfuscated program. In [57] the construction requires the conversion of a functionality into a branching program before obfuscation takes place. Such a conversion requires usage of Barrington's theorem, resulting in branching programs of depth $O(4^d)$ for depth d circuits. Clearly, this is a huge blow-up in the size of the program as well as the amount of computation required for evaluation. The 'sort-compare-shuffle' set intersection circuit developed by Huang et al. [41] has depth $\Theta(n \log(n))$, where n is the size of the set being queried. For this circuit, the work of [57] could only be used for obfuscating the intersections of sets of logarithmic size with respect to the entire universe. Bootstrapping their obfuscator for polynomial-depth circuits requires usage of fully homomorphic encryption on top of the existing construction. The obfuscator of Brakerski et al. [10] could also be leveraged for set intersection queries by obfuscating (at worst-case) a conjunction for each element in the set to be queried. Evaluating the intersection would then amount to evaluating n obfuscated conjunctions for each set element. That is, the size of the obfuscator is $O(\delta n)$ and the number of evaluations required is therefore $O(\delta n^2)$, where δ is the size of the string in each conjunction obfuscator.

Our work constructs an obfuscator for data storage queries that is of size $O(kn)$ without using Barringtom's theorem. We eliminate the dependence on the size δ, replacing this with a statistical security parameter k that determines the probability of false-positives occurring. Typically we have that $k < \delta$ and thus an improvement. Specifically, the number of encodings required is $3L + 2$ where L is the length of the Bloom filter (optimally we have that $L = k \cdot n \log e$). Furthermore, an honest evaluation requires worst-case evaluation of $L + 3k - 2$ multiplications of encodings, which is also $O(kn)$. Our protocol presents the first method of obfuscating the set intersection functionality that is both independent of the size of elements and able to obfuscate a polynomially-sized set efficiently (without additional primitives). We view our efficiency gains that we have on the more general obfuscator of [57] as mirroring the distinction between works in secure computation literature that construct practical, specific set intersection functionality and those constructing more general MPC frameworks with steeper computational overheads. We believe that the investigation of asymptotically efficient obfuscators (without using Barrington's theorem, for example) from standard assumptions to be a valuable pursuit in creating obfuscators that are closer to practical realisation.

Entropy requirements. Goldwasser et al. [36] (in brief) showed that LWE with n-dimensional secrets drawn from a distribution with min-entropy α over $\mathbb{Z}_q$ is at least as hard as standard LWE with altered dimensions, in particular $O(\alpha/\log q)$. No such result is known for RLWE.

In this work, we detail a linear variant of the entropic RLWE assumption [10] with a security reduction to standard RLWE. Now that we can base security on standard RLWE, valuable future research would assess the security of RLWE when secrets are drawn from specifically chosen distributions, as in [36].

Obfuscation for evasive functionalities. VBB obfuscators with security based on non-standard assumptions [8, 3] pre-dating [57, 10] showed that security was possible to achieve for evasive functions. An evasive functionality requires that, for a function $f : \{0,1\}^\ell \mapsto \{0,1\}$, that there is only a small chance of finding a 'satisfying' input $\mathbf{x}$, such that $f(\mathbf{x}) = 1$.[2] The accepted security model for proving VBB obfuscators secure is with respect to *distribution ensembles* $D = \{D_\lambda\}$ that sample some evasive function f along with a set S of satisfying inputs $\mathbf{x}$ from D_λ. Evasive in this context means that S has sufficient min-entropy $\alpha(\lambda) \geq \lambda^\epsilon$ even when f is known. The simulator in the VBB security proofs is not given access to f and hence f must be evasive.

Obfuscation for data storage. In this work, we make the same entropic requirements as above when presenting distributional VBB security. Specifically, we consider a class of distributions $\mathcal{D}_\alpha$ consisting of distribution ensembles $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ of min-entropy $\alpha(\lambda)$; sampling $(F^S, S) \leftarrow\!\!\$\, D_\lambda$, for a function F^S and some set $S \subset \mathcal{S}$, taken from a large universe $\mathcal{S}$. For $y \in \mathcal{S}$ we have that $F^S(y) = 1$ if $y \in S$ with all but negligible probability. Recall, that $\alpha(\lambda)$ is the min-entropy of the set S when given access to F^S.

We define a function $F^S_{\boldsymbol{BF}} = (\boldsymbol{BF}, (h_0, \ldots, h_{k-1}))$ sampling $(F^S_{\boldsymbol{BF}}, S) \leftarrow\!\!\$\, D_\lambda$. Here, $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}))$ is a description of a Bloom filter, where $\boldsymbol{BF} \in \{0,1\}^L$ is a bit array and $h_0, \ldots, h_{k-1}$ are the hash functions $h_i : \mathcal{S} \mapsto [L]$ used to compute queries. The function $F^S_{\boldsymbol{BF}}$ is a symbolic representation of the

[2] This can be reversed, leaving the computational challenge as finding an input that results in an output of 0.

functionality presented for evaluating queries.[3] In this work, we will assume that $h_i \leftarrow_{\$} \mathcal{H}$ for some strongly universal hash family $\mathcal{H}$ to guarantee that $\boldsymbol{BF}$ is uniformly distributed. In the following, we may write $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$ to include the set S that is being implicitly considered.

Set intersection for evasive sets. By obfuscating Bloom filters we provide a natural interface for computing set intersection queries in an offline (or non-interactive) mode. However, sampling sets with sufficient min-entropy enough to make F^S evasive is likely to make it computationally difficult for an evaluator to find any sets that intersect with the one hidden in the obfuscated data storage. As such, it is not amiss to suggest that such a functionality is unlikely to be useful.

Our obfuscator becomes more meaningful when considering that there may be some small subsets of users who are able to lower the min-entropy of the hidden set. Such a user may know some information that allows them to predict some elements that may be contained in the obfuscated storage structure. In this case, our obfuscator provides more meaningful functionality in terms of correctness – since the user will learn some information about the intersection of their sets. However, we sacrifice the security guarantees provided by the obfuscator in these cases since the real-world execution will be trivially distinguishable from a simulation with no oracle access. Fortunately, for users without this knowledge security is still maintained. As is noted by Wichs and Zirdelis [57], users can be separated into those who are provided with functionality and those where security is guaranteed.

All current private set intersection (PSI) protocols require online computation between multiple parties – this line of work was initiated by Freedman et al. [28] as a key facet of secure computation research (recent works such as [41, 21, 51, 52, 27, 44, 42] have made current protocols very efficient). Our work could be seen as a non-interactive alternative where PSI can be carried out offline. For example, one player can obfuscate a Bloom filter representing their set and then disseminate this to other users. Unfortunately, we cannot formalise the non-interactive PSI security requirement, since an adversary can test any set of any (polynomially-bounded) length in an offline setting. However, our obfuscator naturally prevents learning any information outside of those elements that are queried.

Obfuscating set intersection via hashing. We could provide a VBB obfuscator for evaluating the intersection of sets that are sampled with sufficient min-entropy by simply publishing the hash output of each set element. By modelling the hash function as a random oracle, we could guarantee that each output was uniformly distributed across a finite domain. Moreover, providing that the distribution that samples the set had sufficient min-entropy, then the evaluation would still be 'evasive'.

In this work, we create a VBB obfuscator in the *standard model* since our hash functions only need to satisfy strong universality. Moreover, the set intersection functionality is a subset of the functionality offered by Bloom filters and so there is the possibility of building extensions to our design for computing more expressive outputs.

1.2 Overview of technique

We provide a brief overview of the individual components underpinning our construction, along with an intuition for the design itself.

Linear Entropic RLWE. Note that the Ring-LWE (RLWE) assumption, for a ring $\mathcal{R}$, broadly states that when $s \leftarrow_{\$} \mathcal{R}$, then the pair $(A, sA+e)$ is indistinguishable from (A, u), for uniformly chosen $u \in \mathcal{R}^m$, uniform $A \in \mathcal{R}^m$ and short error $e \in \mathcal{R}^m$. Alternatively, let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be a distribution ensemble sampling strings $x \in \{0,1\}^n$ with min-entropy $\alpha(\lambda)$. The linear entropic Ring-LWE assumption (LERLWE) states that when $s_0, \ldots, s_{n-1} \leftarrow_{\$} \mathcal{R}$ are short ring elements; then the pair $(A, sA+e)$ is indistinguishable from uniform where $s = \sum_i x_i \cdot s_i$ is used as the secret. Note that in the original (multiplicative) entropic RLWE assumption from [10], the secret is constructed using $s = \prod_i s_i^{x_i}$. We actually consider a variant of the LERLWE assumption where there are multiple bit strings $x^{(j)} \leftarrow_{\$} D_\lambda$ for $0 \le j \le M-1$ and a fixed element $\tilde{s} \in \mathcal{R}$ such that $\tilde{s} = \sum_i x_i^{(j)} \cdot s_i$ and the adversary receives $(s_0, \ldots, s_{n-1})$ and $b = \tilde{s}A + e$ or $b \leftarrow_{\$} \mathcal{R}^m$.

[3] Note that S is not presented with $F^S_{\boldsymbol{BF}}$.

Bloom filters. As in [10, 57], we consider a distributional VBB (or average-case) obfuscation security model [40, 39, 26, 5]. This broadly states that a VBB function obfuscator is valid for a particular distribution class $\mathcal{D}_\alpha$. In our case, $\mathcal{D}_\alpha$ is the class of distribution ensembles that sample Bloom filters 'representing' (we define this term explicitly in Definition 1) sets $S \in \mathcal{S}$ with min-entropy $\alpha(\lambda)$, for some large universe $\mathcal{S}$. For $D = \{D_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{D}_\alpha$, we sample a Bloom filter as the tuple $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S) \leftarrow\!\!\$\, D_\lambda$ where $\boldsymbol{BF}$ is a binary array of length L encoding the set S and $(h_0, \ldots, h_{k-1})$ are the set of hash functions that enable queries to be computed, i.e. $h_i : \mathcal{S} \mapsto [0, L-1]$ for all $i \in [0, k-1]$. Concretely, we sample $\{h_i\}_{i\in[k-1]}$ from the strongly universal hash family $\mathcal{H}$.[4]

Ring-based garbled Bloom filters. While the function we obfuscate works with respect to a standard Bloom filter, the underlying construction is similar in appearance to the garbled Bloom filter defined in [27], used for constructing an efficient PSI protocol. Our design is different in that the operations in our 'garbled' Bloom filter take place in a polynomial ring and that the query functionality is also substantially changed.

For a standard Bloom filter, we define a query function $\mathsf{Query}()$ such that

$$\mathsf{Query}(y) = \bigwedge_{i=0}^{k-1} \boldsymbol{BF}[h_i(y)] = 1.$$

The garbled Bloom filter abstraction used by Dong et al. [27] preserves the functionality of the Bloom filter but alters the user interface. In particular, for querying an element y against a garbled Bloom filter $\boldsymbol{GBF}$, representing a set S, we have (with high probability) that $y \in S$ if

$$\mathsf{Query}(y) = \bigoplus_{i=0}^{k-1} \boldsymbol{GBF}[h_i(y)] = y$$

where $\oplus$ denotes the XOR operation. The authors show that this functionality can be instantiated using an XOR-based secret sharing scheme.

In this work, we define a 'ring-based garbled' Bloom filter, $\boldsymbol{RGBF}$, associated with some ring $\mathcal{R}$ and with a fixed parameter $\tilde{\rho} \in \mathcal{R}$. Now each entry in $\boldsymbol{RGBF}$ is an element of $\mathcal{R}$ and we have that some query y satisfies $y \in S$ (with high probability) if

$$\mathsf{Query}(y) = \sum_{i=0}^{k-1} \boldsymbol{RGBF}[h_i(y)] = \tilde{\rho}.$$

Our definition seems more restrictive than the original garbled Bloom filter definition, but it is necessary for the proof of security for our obfuscator. The formal definition of this construct is written below.

Definition 1. *(Represented elements) Let $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$ be a Bloom filter. An element y is* represented *in a Bloom filter (with high probability), $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$, if*

$$\boldsymbol{BF}[h_i(y)] = 1$$

for all $0 \leq i \leq k-1$. If y is represented as such then we say that $y \in S$.

Definition 2. *(Ring-based garbled Bloom filters) Let $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$ be a Bloom filter, where $\boldsymbol{BF}$ has length $L = L(\lambda)$ (i.e. $\boldsymbol{BF} \in \{0,1\}^L$). Let $\boldsymbol{RGBF} \in \mathcal{R}^L$ be the array in the* ring-based garbled Bloom filter *for the ring $\mathcal{R}$ with fixed parameter $\tilde{\rho} \in \mathcal{R}$. Then for y represented in the Bloom filter, $\boldsymbol{RGBF}$ satisfies the following:*

$$\sum_{i=0}^{k-1} \boldsymbol{RGBF}[h_i(y)] = \tilde{\rho}.$$

We can redefine the tuple for the Bloom filter as $(\boldsymbol{RGBF}, (h_0, \ldots, h_{k-1}), S, \tilde{\rho})$.

[4] See Section 2.2 for details.

Remark 1. An explicit $\boldsymbol{RGBF}$ object is never presented to an evaluator. This representation can merely be thought of as an intermediate step towards obfuscating a Bloom filter query function $F^S_{\boldsymbol{BF}}$. Therefore, in cases where $\boldsymbol{BF}[i] = 0$ we simply mirror these as zeroes in $\boldsymbol{RGBF}$, since we do not stipulate valid conditions for these entries until obfuscation.

Directed encodings. Brakerski et al. [10] devise an abstract interpretation of the GGH15 [33] MMAP known as a directed encoding scheme. Directed encoding schemes are specialised to computations along a line, rather than a more general graph-based topology as enforced in GGH15. For some ring $\mathcal{R}$ and public keys $\boldsymbol{A}, \boldsymbol{A}' \in \mathcal{R}^m$, we define an encoding of a short ring element $s \in \mathcal{R}$ with respect to a path $\boldsymbol{A} \to \boldsymbol{A}'$ as a short matrix $\boldsymbol{R} \in \mathcal{R}^{m \times m}$, such that

$$\boldsymbol{AR} = s\boldsymbol{A}' + e$$

for some short Gaussian error $\boldsymbol{e} \in \mathcal{R}^m$. From this stand point, we are able to multiply encodings as long as they form a connected path in a graph. For example, encodings $\boldsymbol{R}_1, \boldsymbol{R}_2$ with respect to paths $\boldsymbol{A}_0 \to \boldsymbol{A}_1$ and $\boldsymbol{A}_1 \to \boldsymbol{A}_2$. The result of performing $\boldsymbol{R}_1 \cdot \boldsymbol{R}_2$ is a matrix $\boldsymbol{R}_\times$ encoded with respect to the path $\boldsymbol{A}_0 \to \boldsymbol{A}_2$. The work of [10] did not require additions but noted that it was possible in the directed encoding scheme paradigm. Indeed, we can define additions of any encodings that are encoded with respect to the same path segment.

Lastly, we can detect if a matrix $\boldsymbol{R}$ encodes the value of zero under $\boldsymbol{A} \to \boldsymbol{A}'$ by checking whether $\boldsymbol{AR}$ is a short vector in $\mathcal{R}^m$. This also allows for equality testing on encodings indexed with respect to the same path.

Obfuscator overview. Ignoring technical details about the explicit rings for now, we give an overview of how the obfuscation procedure works for a Bloom filter. Let $\boldsymbol{RGBF}$ be empty (each entry initialised to zero) and defined as in Definition 2. Sample a short ring element $\tilde{r}$ and use this as the fixed parameter $\tilde{\rho} = \tilde{r}$.

Construction of the Bloom filter. To represent some element y in $\boldsymbol{RGBF}$ we first compute $I = \{i_0, \ldots, i_{k-1}\} = \{h_0(y), \ldots, h_{k-1}(y)\}$. Then we sample $k-1$ short ring elements s_{i_j} for $0 \leq j \leq k-2$ and finally calculate

$$s_{i_{k-1}} = \tilde{\rho} - \left(\sum_{j=0}^{k-2} s_{i_j} \right) \tag{1}$$

so that

$$\sum_{j=0}^{k-1} s_{i_j} = \tilde{\rho}.$$

We then place each s_{i_j} into the entry given by $h_j(y)$, i.e. $\boldsymbol{RGBF}[h_j(y)] = s_{i_j}$. When representing future elements we can re-use entries used for previous elements. For example, suppose that we have $I' = \{i'_0, \ldots, i'_{k-1}\}$ where there exists a strict subset $J \subset I'$ such that for all $l \in J$ then $\boldsymbol{RGBF}[l] \neq 0$. Then for $t \in (I' \setminus J)$ we can sample $|I' \setminus J| - 1$ new short elements s_{i_t}. We can then sample the final element as in Equation (1). We replay this procedure each time we add new elements to $\boldsymbol{RGBF}$; reusing samples in entries that are non-zero. We require that $|J| \leq k-2$ to ensure that each sum contains some independently sampled elements. We discuss this requirement in greater detail in Section 2.4.

Generating the obfuscator. Once $\boldsymbol{RGBF}$ has been constructed, we can employ our directed encoding scheme to encode each of the entries. Firstly we sample $2L$ short random ring elements $r_{i,b}$ for $0 \leq i \leq L-1$ and $b \in \{0, 1\}$ – two for each entry in the Bloom filter. Finally, let $T < L$ be the number of entries in $\boldsymbol{RGBF}$ still set to zero, then sample T short ring elements $\{r'_0, \ldots, r'_{T-1}\}$.

To construct the obfuscator, firstly we associate an element $r_{i,0}$ with each entry in $\boldsymbol{RGBF}$. For entries that are zero in $\boldsymbol{RGBF}$ we also associate a pair of the form:

$$(r_{i,1}, r_{i,1} r'_i) \tag{2}$$

for r'_i randomly chosen from $\{r'_0, \ldots, r'_{T-1}\}$, and for entries that are non-zero we associate pairs:

$$(r_{i,1}, r_{i,1} s_i), \tag{3}$$

where $\boldsymbol{RGBF}[i] = s_i \neq 0$. Let $[x]_{a\to b}$ denote an encoding of x with respect to the path $a \to b$. We now sample $L+2$ public keys $\boldsymbol{A}_0, \ldots, \boldsymbol{A}_{L+1}$ and compute encodings $\zeta_i = [r_{i,0}]_{i\to i+1}$ for each $i \in [L-1]$. We also encode both components of each of the pairs from Equations (2) and (3) such that:

$$v_i = ([r_{i,1}]_{i\to i+1}, [r_{i,1} f_i]_{i\to i+1}),$$

where $f_i = s_i$ for $\boldsymbol{RGBF}[i] \neq 0$ and $f_i = r'_i$ otherwise. Finally, we sample a short ring element $\hat{r}$ and construct a last pair $(\hat{r}, \hat{r}\tilde{\rho})$ and then produce a pair of encodings $p_{zt} = [(\hat{r}, \hat{r}\tilde{\rho})]_{L\to L+1}$ with respect to the path $L \to L+1$. This last pair will be used to perform the final equality test that determines whether an element is represented in the Bloom filter, or not.

The obfuscation of $\boldsymbol{RGBF}$ is then written as:

$$(\boldsymbol{A}_0, \{\zeta_i\}_{i=0}^{L-1} \{v_i\}_{i=0}^{L-1}, p_{zt}, \{h_j\}_{j=0}^{k-1}), \tag{4}$$

where we only need the first public key for evaluation (see Appendix A).

Querying the obfuscator. An entity who is given control of the obfuscator from Equation (4) is able to evaluate elements y of their choosing and see if y is represented in the underlying Bloom filter. Firstly, evaluate $J = \{h_0(y), \ldots, h_{k-1}(y)\}$ and let $I = ([L-1] \setminus J)$ be the set of indices that are not chosen. We define an operation $\boxplus$, that operates over two pairs of encodings $\bar{x} = ([x']_{a\to b}, [x'x]_{a\to b})$, $\bar{y} = ([y']_{b\to c}, [y'y]_{b\to c})$ in the following way:

$$\bar{x} \boxplus \bar{y} = ([x'y']_{a\to c}, [x'y'(x+y)]_{a\to c})$$

and we can define a subtraction operation $\boxminus$ analogously. Roughly speaking, we multiply the encodings in the first component of the pair together and with the opposed second component. Finally, we add the second components together to retrieve an encoding containing a random multiple of $(x+y)$ along the union of the two paths. Now compute:

$$v_\theta = ([r_\theta]_{0\to L}, [r_\theta \theta]_{0\to L}) = \left(\boxplus_{j\in J} v_j\right) \cdot \left(\prod_{i\in I} \zeta_i\right) \tag{5}$$

where the computation is interleaved such that operations take place in the order defined by the sequentially chosen paths $0 \to 1$, $1 \to 2$, ... with respect to the encodings.[5] We give a diagrammatic representation in Figure 1 of how operations along a path are computed.

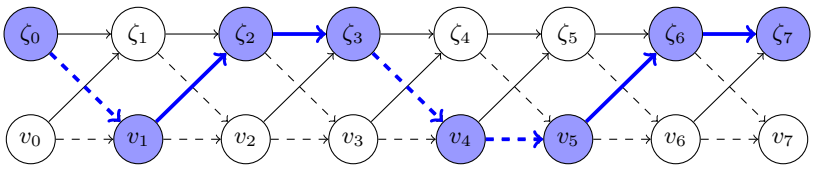

Fig. 1. Example of evaluation for $J = \{1, 4, 5\}$ and $I = \{0, 2, 3, 6, 7\}$. We use dashed arrows to indicate that $\boxplus$ operation is used, full arrows indicate standard multiplication of encodings. Note that the first time we move from the top branch to the lower branch we simply compute a multiplication – i.e. in this example we would compute $\zeta_0 \cdot v_1$ but we compute $(\zeta_0 \cdot v_1 \cdot \zeta_2 \cdot \zeta_3) \boxplus v_4$ on the second occasion.

To learn the output of the query in Equation (5) we finally compute:

$$v_\theta \boxminus p_{zt} = ([r_\theta \hat{r}]_{0\to L+1}, [r_\theta \hat{r}(\theta - \tilde{\rho})]_{0\to L+1})$$

where $\theta = \tilde{\rho}$ if and only if y is represented in $\boldsymbol{RGBF}$ by Definition 2 (with high probability). Therefore, we can now use the $\mathsf{ZeroTest}(\cdot)$ procedure, provided by our encoding scheme, to test if $[r_\theta \hat{r}(\theta - \tilde{\rho})]$ encodes

[5] For multiplying $\zeta_i = ([r_{i,0}])$ with $v_{i+1} = ([r_{i+1,1}], [r_{i+1,1} f_{i+1}])$ we multiply ζ_i directly with both components in $v_{i,1}$, and vice-versa if the path was reversed.

zero or not (returning '1' and '0' in these respective cases). If y is not represented in $\boldsymbol{RGBF}$ then we are likely to find that $\theta \neq \tilde{\rho}$ and thus

$$\mathsf{ZeroTest}([r_\theta \hat{r}(\theta - \tilde{\rho})]_{0 \to L+1}) = 0,$$

otherwise it will return 1.

Security of the obfuscator. The security of our obfuscator relies on the fact that the function we are computing is evasive, the security guarantee erodes when satisfying inputs are found.[6] Subsequently, the probability of selecting an element y that satisfies the obfuscator is necessarily small. The distribution class that we can obfuscate translates naturally to those that sample sparse sets with sufficient min-entropy from a large universe. As noted before, we can use the GXDH and linear α-entropic security properties (Section 4) of our directed encoding scheme to prove security in this setting.

2 Preliminaries

2.1 Notation

For a set $N = \{0, \ldots, n\}$ we write $x \in [n]$ for representing that $x \in N$ (we count from 0 unless stated otherwise). This should not be confused with encodings in the directed encoding framework, where we write $[a]_{b \to c}$ for a value a that is encoded along the path $b \to c$. If a is an array then each component of the array is encoded individually; see Section 4 for more details. For a message distribution $\mathcal{M}$, we use $x \leftarrow_\$ \mathcal{M}$ to denote the sampling of x uniformly from the distribution. We use the acronym PPT to denote 'probabilistic polynomial time'. Vectors are denoted by bold-face, e.g. $\mathbf{x}$. For two vectors, $\mathbf{x}$ and $\mathbf{y}$, we denote their inner product by $\langle \mathbf{x}, \mathbf{y} \rangle$. We use the infinity norm ($\|\cdot\|_\infty$) over a vector $\mathbf{x}$ to characterise the size of $\mathbf{x}$. For a string $x \in \{0,1\}^L$ we denote the i^{th} character by x_i. For a distribution D, we denote by $\mathsf{poly}(D^n)$ the distribution sampling n values from D and using these as coefficients in a degree-n polynomial.

2.2 Entropy

Definition 3. (Average min-entropy [25]) *Let X and Z be random variables that are possibly dependent on each other. The* average min-entropy *of X conditioned on Z is:*

$$\widetilde{\boldsymbol{H}}_\infty(X|Z) = -\log\left(\mathbb{E}[2^{-\boldsymbol{H}_\infty(X|z=Z)}]\right)$$

Definition 4. (Strongly universal hash family) *Let $\mathcal{H}$ be a family of hash functions $h : X \mapsto [m-1]$ for some universe X and a $m \in \mathbb{Z}$. Then $\mathcal{H}$ is a* strongly universal hash family *if, for all $x, y \in X$ such that $x \neq y$ and $h \leftarrow_\$ \mathcal{H}$:*

$$\Pr[h(x) = h(y)] = 1/m^2.$$

This is equivalent to the statement that, for $h \leftarrow_\$ \mathcal{H}$, then $\Pr[h(x) = t] = 1/m$ for $x \in X$ and any $t \in [m-1]$.

Definition 5. (Classes of distributions) *We consider classes $\mathcal{D}$ of efficiently samplable distribution ensembles $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ sampling functions F along with sets $S \in \mathcal{S}$. That is, an ensemble $D = \{D_\lambda\}_{\lambda \in \mathbb{N}} \in \mathcal{D}$ samples $((F, S), \mathsf{aux}) \leftarrow_\$ D_\lambda$. Furthermore, for a function $\alpha(\lambda)$, the ensemble D satisfies $\widetilde{\boldsymbol{H}}_\infty(S|(F, \mathsf{aux})) \geq \alpha(\lambda)$. We say that such ensembles $D \in \mathcal{D}$* have min-entropy $\alpha(\lambda)$.

[6] As mentioned previously, if we intentionally provide knowledge to some users that reduces the min-entropy of the set then we can provide more meaningful functionality.

2.3 VBB Obfuscation

We discuss the notion of *distributional VBB* obfuscation, using the same definition as in [10]. We use the distributional notion of security since our obfuscator satisfies security for specific choices of distribution ensemble.

Definition 6. (Distributional VBB obfuscator) *Consider a circuit family $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ with input size n and let $\mathcal{O}$ be a PPT algorithm, which takes as input a circuit $C \in \mathcal{C}$, a security parameter $\lambda \in \mathbb{N}$, and outputs a boolean circuit $\mathcal{O}(1^\lambda, C)$ – this circuit does not have to be in $\mathcal{C}$. Let $\mathcal{D}$ be a class of distribution ensembles $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ that sample $(C, \mathsf{aux}) \leftarrow\!\!\$\, D_\lambda$ with $C \in \mathcal{C}$ and aux representing arbitrary auxiliary information. $\mathcal{O}$ is an obfuscator for $\mathcal{D}$ over the circuit family $\mathcal{C}$, if it satisfies the following properties:*

1. Functionality*: For a negligible function* negl*, then for all circuits $C \in \mathcal{C}$, we have*

$$\Pr\left[\forall x \in \{0,1\}^n \;:\; C(x) = \mathcal{O}(1^\lambda, C)(x)\right] \geq 1 - \mathsf{negl}(\lambda)$$

   where the probability is taken over the random coin tosses of $\mathcal{O}$.
2. Polynomial slowdown*: For every $\lambda \in \mathbb{N}$ and $C \in \mathcal{C}$, the circuit $\mathcal{O}(1^\lambda, C)$ is of size at most $poly(|C|, \lambda)$.*
3. Distributional virtual black-box*: For every (non-uniform) polynomial size adversary $\mathcal{A}$, there exists a (non-uniform) polynomial-size simulator,* Sim*, such that for every distribution ensemble $D = \{D_\lambda\} \in \mathcal{D}$, and every (non-uniform) polynomial-size predicate $P : \mathcal{C} \mapsto \{0,1\}$, then:*

$$\left| \Pr_{(C,aux)\sim D_\lambda, O, A}[\mathcal{A}(\mathcal{O}(1^\lambda, C), \mathsf{aux}) = P(C)] - \Pr_{(C,aux)\sim D_\lambda, Sim}[\mathsf{Sim}^C(1^\lambda, 1^{|C|}, \mathsf{aux}) = P(C)] \right| < \mathsf{negl}(\lambda) \tag{6}$$

2.4 Bloom filter preliminaries

Bloom filters are commonly used as efficient data storage objects. They have been used extensively in private set operations research [27, 22, 23, 43] in order to lower the overheads for participating entities. In short, they are useful in any situation where quick data storage and querying is required. In this section, we give a formal overview of the functionality offered by these constructs.

Let $\mathcal{H}$ be the space of hash functions $h : \mathcal{R} \mapsto [L-1]$, we require that $\mathcal{H}$ satisfies the strongly universal hash family property from Definition 4. Let $\boldsymbol{BF}$ be an array of length L bits along with a uniformly sampled set of k functions $\{h_i\}_{i\in[k-1]} \leftarrow\!\!\$\, \mathcal{H}$. We desire the following functionality.

- *Storage* ($\mathsf{Store}()$): For elements $x \in \mathcal{R}$, evaluate $x_{h_i} = h_i(x)$ and thus set $\boldsymbol{BF}[x_{h_i}] = 1$ for all i.
- *Querying* ($\mathsf{Query}()$): For some element $y \in \mathcal{R}$, evaluate $y_{h_i} = h_i(y)$ and check that $\boldsymbol{BF}[y_{h_i}] = 1$ for all i, if so return 1, else return 0.

We require that $\mathcal{H}$ is strongly universal to ensure that the bits that are set to 1 in the Bloom filter are uniformly distributed. Since instantiations of strongly of strongly universal hash families exist [17], then we can situate our construction in the standard model. We define a Bloom filter to be the tuple $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$ where S is the set represented in the array $\boldsymbol{BF}$ and $(h_0, \ldots, h_{k-1})$ is the collection of hash functions used for querying the array.

Parameter choices. A feature of Bloom filters is that there is a non-trivial false-positive probability associated with querying elements. That is, it is possible to find $y' \in \mathcal{R}$ such that $\mathsf{Query}(y', \boldsymbol{BF}) \to 1$ but where $y \notin S$. However, as shown in [27], if $p = 1 - (1 - 1/L)^{kn}$ is the probability that a particular bit in $\boldsymbol{BF}$ is set to 1 for a set of size n, then the upper bound of the false-positive probability is given by

$$\epsilon = p^k \times \left(1 + O\left(\frac{k}{p}\sqrt{\frac{ln\, L - k\, ln\, p}{L}}\right)\right),$$

which is negligible in k, the number of hash functions. In practice, one will select the values of k and L for a set of size n such that ϵ is capped at a specific low value. In [27] it is claimed that performance optimality is achieved when

$$k = \frac{L}{n} \ln 2, \tag{7}$$

$$L \geq n \log_2 e \cdot \log_2 1/\epsilon, \tag{8}$$

where e is the base of the natural logarithm. By minimising L, we get the optimal value of k to be

$$k = \log_2 1/\epsilon. \tag{9}$$

It is helpful to know that parameters are always chosen optimally, at least as defined in Equations (7) and (8) (as in [27]). The proof that these values are optimal can be found in [7].

Ring-based garbled Bloom filters. As noted in the introduction we use a ring-based variant of a Bloom filter for a ring $\mathcal{R}$ that is inspired by the garbled Bloom filters introduced in [27]. Our variant requires a fixed parameter $\tilde{\rho}$ and storing elements in the array $\boldsymbol{RGBF}$ requires sampling a set of indices I and then for all $i \in I$ we randomly sample $s_i \in \mathcal{R}$ and set $\boldsymbol{RGBF}[i] = s_i$ such that the following holds:

$$\sum_{i=0}^{k-1} \boldsymbol{RGBF}[h_i(y)] = \tilde{\rho}.$$

In the above procedure certain elements in $\boldsymbol{RGBF}$ are reused for multiple storage iterations. We noted previously that it is important that for $y \neq y'$ then the set $J = \{h_0(y), \ldots, h_{k-1}(y)\}$ should only coincide with the set $J' = \{h_0(y'), \ldots, h_{k-1}(y')\}$ on only $k-2$ points; i.e. $|J \cap J'| \leq k-2$. In standard (and garbled) Bloom filters, we can allow these sets to coincide on $k-1$ points. The intuition behind this requirement is that the non-zero entries in our Bloom filter are dependent upon the single target parameter $\tilde{\rho}$. Therefore, this can be interpreted as giving away an extra secret share of the garbled elements. If sets coincide on $k-1$ points then we can reconstruct the k^{th} element using this knowledge. This would have harmful consequences for arguing the security of our construction. By limiting to only coinciding on $k-2$ points we are able to guarantee independently chosen elements for the two entries that do not coincide. Consequently, we can guarantee that each possible set of entries representing some stored element $y \in S$ contain at least one independently sampled element from $\mathcal{R}_q$.

In order to exercise this requirement, we can merely pick parameters and replace k in Equations (7), (8) and (9) by $k+1$. Thus maintaining a level false-positive probability requires increasing the size of the Bloom filter slightly.

Bloom filter obfuscators. Let $\mathcal{C}^{\mathcal{S}}_{\boldsymbol{BF}}$ be the class of Bloom filters of length L representing a set $S \subset \mathcal{S}$, where $h_0, \ldots, h_{k-1} \leftarrow\!\!\$\, \mathcal{H}$ are the set of k corresponding hash functions. As mentioned previously, we require that the set of hash functions $\{h_i\}_{i\in[k-1]}$ is sampled from a strongly universal hash family $\mathcal{H}$. Intuitively, sampling from a distribution over $\mathcal{C}^{\mathcal{S}}_{\boldsymbol{BF}}$ returns both the set, S, and the Bloom filter representation, $F^S_{\boldsymbol{BF}}$, of that set.

Let $\mathcal{D}_\alpha$ be a class of distribution ensembles $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ sampling Bloom filters $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$ from $\mathcal{C}^{\mathcal{S}}_{\boldsymbol{BF}}$, where $S \in \mathcal{S}$ has min-entropy $\alpha(\lambda)$ given $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}))$. Define a function $F^S_{\boldsymbol{BF}} : \mathcal{S} \mapsto \{0,1\}$ where, for some query $y \in \mathcal{S}$:

$$\Pr\left[y \in S \mid F^S_{\boldsymbol{BF}}(y) = 1\right] > 1 - \mathsf{negl}(\lambda),$$

and

$$\Pr\left[y \notin S \mid F^S_{\boldsymbol{BF}}(y) = 0\right] = 1.$$

We abuse notation and typically write $F^S_{\boldsymbol{BF}} = (\boldsymbol{BF}, (h_0, \ldots, h_{k-1}))$ since it is completely defined by the array $\boldsymbol{BF}$ and the hash functions $\{h_i\}_{i\in[k-1]}$. We also redefine $\mathcal{D}_\alpha$ to be the class of distribution ensembles sampling $(F^S_{\boldsymbol{BF}}, S)$ where S has min-entropy $\alpha(\lambda)$ given $F^S_{\boldsymbol{BF}}$. The entropic requirement over the distribution class $\mathcal{D}_\alpha$ ensures that $F^S_{\boldsymbol{BF}}$ is an evasive function; i.e. $F^S_{\boldsymbol{BF}}(y) = 1$ occurs with very low probability for any arbitrarily chosen $y \in \mathcal{S}$.

Definition 7. (Bloom filter obfuscator) *Let $\mathcal{D}_\alpha$ be the class of distribution ensembles $D = \{D_\lambda\}_{\lambda\in\mathbb{N}}$, sampling $(F^S_{\boldsymbol{BF}}, S) \leftarrow_\$ D_\lambda$ where $S \in \mathcal{S}$, that have min-entropy $\alpha(\lambda)$ given $F^S_{\boldsymbol{BF}}$. We say that $\mathcal{O}$ is an $\alpha(\lambda)$-distributional VBB obfuscator for Bloom filters if it is a distributional VBB obfuscator for the class $\mathcal{D}_\alpha$.*

Definition 8. ($\alpha(\lambda)$-entropic security) *A Bloom filter obfuscator $\mathcal{O}$ satisfies $\alpha(\lambda)$-entropic security if there exists a PPT simulator* Sim *such that, for all efficiently samplable distributions $D \in \mathcal{D}_\alpha$ with min-entropy $\alpha(\lambda)$, we have:*

$$(\mathcal{O}(1^\lambda, F^S_{\boldsymbol{BF}}), \mathsf{aux}) \overset{c}{\approx} (\mathsf{Sim}(1^\lambda, 1^L, 1^k), \mathsf{aux})$$

where $(F^S_{\boldsymbol{BF}}, S) \leftarrow_\$ D$, and where aux *is some auxiliary information.*

As in [10], the simulator does not have access to the distribution D; so we can think of an obfuscator satisfying this definition as hiding all properties of the underlying distribution. The simulator also has no oracle access to $F^S_{\boldsymbol{BF}}$ and therefore it cannot learn anything about the Bloom filter that is being queried. If the function that we evaluate is not evasive then the real world setting is trivially distinguishable from the simulated world.

2.5 Lattice preliminaries

Lattices. An n-dimensional lattice of rank n is the set Λ of integer combinations of n linearly independent vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1} \in \mathbb{R}^n$, that is:

$$\Lambda = \left\{ \sum_{i=0}^{n-1} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z} \right\}.$$

The matrix $\boldsymbol{B} = [\mathbf{b}_0 \mid \ldots \mid \mathbf{b}_{n-1}]$ is called a basis for Λ. The dual of a lattice Λ is the set $\Lambda^* = \{\mathbf{x} \in \mathsf{span}(\Lambda) \ : \ \forall \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$.

Gaussian function. The Gaussian function $\rho_s : \mathbb{R}^m \mapsto \mathbb{R}$ with parameter s is defined as

$$\rho_s(x) = \exp(-\pi ||\mathbf{x}||^2 / s^2),$$

when s is omitted, it is assumed to be 1. The discrete Gaussian distribution $D_{\mathbb{Z},\Lambda+\mathbf{c}}$ with parameter s over a lattice coset $\Lambda+\mathbf{c}$ is the distribution that samples each element $\mathbf{x} \in \Lambda+\mathbf{c}$ with probability $\rho_s(\mathbf{x})/\rho_s(\Lambda+\mathbf{c})$, where

$$\rho_s(\Lambda + \mathbf{c}) = \sum_{y \in \Lambda+\mathbf{c}} \rho_s(\mathbf{y})$$

is a normalization factor. In the following, we will explicitly use $D_{\mathbb{Z},\sigma}$ to denote the discrete Gaussian distribution over the integers, centred at 0.

B-bounded distributions. We say that a distribution D is B-bounded if for $x \leftarrow_\$ D$ then $\Pr[\|x\|_\infty \leq B] = 1$. In this work, we require sampling vectors of length n where each component is sampled from a truncated Gaussian over $\mathbb{Z}$, $\overline{D_{\mathbb{Z},\sigma}}$, centred at zero, with parameter σ. It is shown in [49] (Lemma 4.4), that if we take $B = 6\sigma$ all samples from the original discrete Gaussian are B-bounded with very high probability.

Smoothing parameter. For any $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\Lambda)$ is the smallest $s > 0$ such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \epsilon$. When ϵ is omitted, it is some unspecified negligible function $\epsilon = n^{-\omega(1)}$ of the lattice dimension or security parameter n, which may vary.

Trapdoor sampling. We detail lemmas that describe procedures for performing lattice trapdoor sampling, more can be read of these procedures in the works of [35, 47, 1].

Lemma 1. (Trapdoor sampling) *There is an efficient, randomised algorithm* $\mathsf{TrapSamp}(1^n, 1^m, q)$ *such that, given integers $n > 1, q > 2$ and sufficiently large $m = \Omega(n \log q)$, outputs a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and a trapdoor matrix $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ where the distribution of $\boldsymbol{A}$ is computationally indistinguishable from uniform.*

Lemma 2. (Pre-image sampling) *There is an efficient algorithm* $\mathsf{GaussSamp}()$ *that, with overwhelming probability over all random choices, does the following. For any* $\mathbf{u} \in \mathbb{Z}_q^n$*, and large enough* $s = \Omega(\sqrt{n \log q})$*, then* $\mathsf{GaussSamp}(\boldsymbol{A}, \boldsymbol{T}, \mathbf{u}, s)$ *outputs a vector* $\mathbf{r} \in \mathbb{Z}^m$ *with norm* $\|\mathbf{r}\|_\infty \leq \|\mathbf{r}\|_2 \leq s\sqrt{n}$ *such that* $\boldsymbol{A} \cdot \mathbf{r} = \mathbf{u}$*.*

Lemma 3. (Indistinguishability of samples) *The following distributions are within* $\mathsf{negl}(n)$ *statistical distance of each other for any* $l \in \mathbb{N}$*:*

- $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$; $\boldsymbol{U} \leftarrow \mathbb{Z}_q^{n \times l}$; $\boldsymbol{R} \leftarrow \mathsf{GaussSamp}(\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{U}, s)$.
- $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \mathsf{TrapSamp}(1^n, 1^m, q)$; $\boldsymbol{R} \leftarrow\!\!\$\ (D_{\mathbb{Z}^m, s})^l$; $\boldsymbol{U} \coloneqq \boldsymbol{A}\boldsymbol{R}\ (\mod q)$.

Note that these lemmas also extend to the ring setting with $\boldsymbol{A} \in \mathcal{R}_q^m$, $\boldsymbol{T} \in \mathcal{R}^{m \times m}$, $\boldsymbol{U} \in \mathcal{R}^k$ and $\boldsymbol{R} \in \mathcal{R}^{m \times k}$.

Finally, we give the convolution theorem introduced by Micciancio and Peikert [48].

Theorem 1. (Convolution theorem [48]) *Let* Λ *be an* n*-dimensional lattice,* $\mathbf{z} \in \mathbb{Z}^m$ *a non-zero integer vector,* $s_i \geq \sqrt{2}\|\mathbf{z}\|_\infty \cdot \eta(\Lambda)$*, and* $\Lambda + \mathbf{c}_i$ *arbitrary cosets of* Λ *for* $0, \ldots, m-1$*. Let* $\mathbf{y}_i$ *be independent vectors with distributions* $D_{\lambda + \mathbf{c}_i, s_i}$*, respectively. Then the distribution of*

$$\mathbf{y} = \sum_{i=0}^{m-1} z_i \mathbf{y}_i$$

is statistically close to $D_{Y,s}$ *where* $Y = \gcd(\mathbf{z})\Lambda + \mathbf{c}$*,* $\mathbf{c} = \sum_i z_i \mathbf{c}_i$*, and* $s = \sqrt{\sum_i (z_i s_i)^2}$*. When* $\gcd(\mathbf{z}) = 1$ *and* $\sum_i z_i \mathbf{c}_i \in \Lambda$*, then* $\mathbf{y}$ *is distributed statistically close to* $D_{\Lambda, s}$*.*

2.6 The Ring Learning with Errors problem

We work with a simple variant of the Ring-LWE (RLWE) problem [46] (sometimes more commonly known as the Polynomial LWE problem introduced in [9]). We will denote our formulation of the problem by $\mathsf{RLWE}_{n,m,q,\chi}$ for parameters n, m, q, χ.

Definition 9. (RLWE) *Let* n *be a power of two, let* $\phi(x) = x^n + 1$ *and let* $\mathcal{R} = \mathbb{Z}[x]/\langle \phi(x) \rangle$*. Let* q *be such that* $q \equiv 1 \mod 2n$ *and define* $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$*. Let* $m \in \mathbb{N}$ *and let* χ *be a distribution over the integers. The* $\mathsf{RLWE}_{n,m,q,\chi}$ *problem is the problem of distinguishing* $\{(a_i, a_i \cdot s + e_i \pmod{(\phi(x), q)})\}_{i \in [m]}$ *from* $\{(a_i, u_i)\}_{i \in [m]}$*, where* $s, e_i \leftarrow\!\!\$\ \mathsf{poly}(\chi^n)$*,* $a_i, u_i \leftarrow\!\!\$\ \mathcal{R}_q$*.*

3 Linear entropic RLWE

In this section, we detail a variant of the RLWE assumption known as the linear entropic RLWE assumption (LERLWE). We need this variant for proving the security of our scheme; though we show that we can derive LERLWE from standard RLWE in a sufficiently general case.

Definition 10. (LERLWE) *Let* n, m, q, χ, L *be parameters of* λ *and* $\mathcal{R}_q$ *as defined for the standard RLWE assumption. Let* $D = \{D_\lambda\}$ *be an efficiently samplable distribution ensemble sampling strings* x *from* $\{0,1\}^L$ *with min-entropy* $\alpha(\lambda)$*. The linear entropic* $\mathsf{RLWE}_{n,m,q,\chi}$ (LERLWE) *problem is to distinguish between:*

$$(\{s_j\}_{j \in [L-1]}, \{(a_i, s \cdot a_i + e_i)\}_{i \in [m-1]})$$

and

$$(\{s_j\}_{j \in [L-1]}, \{(a_i, u_i)\}_{i \in [m-1]})$$

where $s_j, e_i \leftarrow\!\!\$\ \mathsf{poly}(\chi^n)$*,* $s = \sum_{j \in [L-1]} x_j \cdot s_j$*, and* a_i, u_i *are sampled uniformly from* $\mathcal{R}_q$ *(for all* i, j *as above).*

Secondly, we define a stronger and more general interpretation of the LERLWE assumption where there are M strings $x^{(\beta)}$ of hamming weight k. Then we sample a fixed element $\tilde{s}$ such that $\tilde{s} = \sum_{j=0}^{L-1} x_j^{(\beta)} s_j$. Intuitively, first sample a fixed element $\tilde{s}$ from $\mathsf{poly}(\chi^n)$ and sample M different strings $x^{(0)}, \ldots, x^{(M-1)}$ from $\{0,1\}^L$ with the same entropic restriction as before. Let P be an array of L slots initialised to 0 and let $I^{(\beta)} = \{i_0, \ldots, i_{k-1}\}$ be the set of indices such that $x_{i_j}^{(\beta)} = 1$. Then, do the following for each $\beta \in [M-1]$:

1. Let j^* be some index such that $P[i_{j^*}] = 0$ for some $i_{j^*} \in I^{(\beta)}$.
2. For $j \in I^{(\beta)}$ where $j \neq j^*$, if $P[i_j] = 0$ then sample $s_{i_j} \leftarrow_\$ \mathsf{poly}(\chi^n)$, else if $P[i_j] \neq 0$ then let $s_{i_j} = P[i_j]$.
3. Let $s_{i_{j^*}} = \tilde{s} - \sum_{i_j \in I, j \neq j^*} x_{i_j}^{(\beta)} s_{i_j}$ and set $P[i_{j^*}] = s_{i_{j^*}}$.

For indices l such that $P[l]$ is still equal to 0, sample $s'_l \leftarrow_\$ \mathsf{poly}(\chi^n)$ and set $P[l] = s'_l$. The adversary is then given P and either receives samples of the form $(a_i, \tilde{s} \cdot a_i + e_i)$ for $i \in [m-1]$; or (a_i, u_i) for uniformly sampled a_i, u_i from $\mathcal{R}_q$, as before.

We name the variant above the 'Fixed' LERLWE assumption (FLERLWE^{M,k}) since we fix a consistent summed secret $\tilde{s}$ for M different strings of hamming weight k.

Definition 11. (FLERLWE^{M,k}) *Let $n, m, q, \chi, \mathcal{R}_q$ be as before; let $L = L(\lambda), M = M(\lambda), k = k(\lambda)$ be some additional parameters. Let P be an array of length L with all entries set to 0 and let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be an efficiently samplable distribution ensemble sampling strings $x \in \{0,1\}^L$ of hamming weight k with min-entropy $\alpha(\lambda)$. Sample $x^{(\beta)} \leftarrow D_\lambda$ for $\beta \in [M-1]$ and $\tilde{s} \leftarrow_\$ \mathsf{poly}(\chi^n)$. For each $\beta \in [M-1]$ let $I^{(\beta)} = \{i_0, \ldots, i_{k-1}\}$ be indices where $x_i^{(\beta)} = 1$ and compute the following:*

- *let $j^* \in I^{(\beta)}$ be some index where $P[i_{j^*}] = 0$, for $j \neq j^*$ if $P[i_j] = 0$ sample a short ring element $s_{i_j} \leftarrow_\$ \mathcal{R}_q$, else let $s_{i_j} = P[i_j]$;*
- *let $s_{j^*} = \tilde{s} - \sum_{i_j \in I, j \neq j^*} s_{i_j}$, and let $P[i_j] = s_{i_j}$ for all $i_j \in I^{(\beta)}$.*[7]

For any indices l such that $P[l] = 0$ still, sample $s_l \leftarrow_\$ \mathsf{poly}(\chi^n)$ and let $P[l] = s_l$. The $\mathsf{FLERLWE}^{M,k}_{n,m,q,\chi}$ problem is to distinguish between:

$$(P, (a_i, g_i = \tilde{s} \cdot a_i + e_i)), \text{ and } (P, (a_i, g_i = u_i))$$

where $e_i \leftarrow_\$ \mathsf{poly}(\chi^n)$, and a_i, u_i are sampled uniformly from $\mathcal{R}_q$ (for all $i \in [m-1]$).

3.1 RLWE $\implies$ FLERLWE

The work of Brakerski et al [10] introduced a multiplicative variant of the RLWE assumption, our work introduces the first usage of the linear variant. Our confidence in the hardness of FLRLWE can be based on the hardness of RLWE in a generalised case.

Informally, we can construct a distinguishing problem that is equivalent to the one seen in the FLERLWE^{M,k} problem based on the samples seen in the RLWE problem. Our proof makes use of the 'convolution' theorem proposed by Micciancio and Peikert [48] (Theorem 3.3), restated in Theorem 1. Let $\tilde{s}$ be the secret in an instance of the RLWE problem, with a distinguisher $\mathcal{A}_\mathsf{R}$. $\mathcal{A}_\mathsf{R}$ samples $S = (s_0, \ldots, s_{L-1})$ from a Gaussian distribution with associated parameter σ. We assume that $\tilde{s}$ is sampled from a truncated Gaussian with standard deviation $k\sigma$ where k is the size of subsets that we consider for our subset sum.

Theorem 1 essentially states that for $x \in \mathbb{Z}^L$, the distribution induced by $\sum_i x_i s_i$, for $s_i \leftarrow_\$ D_{\mathbb{Z},\sigma}$, is statistically close to $D_{\mathbb{Z},k\sigma}$. Recall that $\tilde{s} \leftarrow_\$ \overline{D_{\mathbb{Z},k\sigma}}$, i.e. sampled from a truncated Gaussian distribution. Let $X_{\tilde{s}}$ be a set such that, for all $x \in X_{\tilde{s}}$ we have $\sum_i x_i s_i = \tilde{s}$, for any $\tilde{s}$ that can be sampled from $\overline{D_{\mathbb{Z},k\sigma}}$. Then, by taking L sufficiently large, we can guarantee that $|X_{\tilde{s}}| \geq M$ for all such $\tilde{s}$. Note, that the sampling of strings does not change, and so the entropic requirement for sampling is unaffected.

[7] If $P[i_j] \neq 0$ for all $i_j \in I^{(\beta)}$ then sample $x^{(\beta)}$ again.

Finally, if $\mathcal{A}_{\mathsf{F}}$ is an adversary attempting to break $\mathsf{FLERLWE}^{M,k}$, $\mathcal{A}_{\mathsf{R}}$ provides $P = S$ along with a sample (a, g) where either $g = a \cdot \tilde{s} + e$ or $g = u$ for uniformly sampled u. In the first case, there are at least M possible (uniformly distributed) strings $x^{(\beta)} \in \{0,1\}^L$ that satisfy $\tilde{s} = \sum_i x_i^{(\beta)} \cdot s_i$. Therefore, the two cases are equivalent to the two cases seen by $\mathcal{A}_{\mathsf{F}}$ in the real $\mathsf{FLERLWE}^{M,k}$ problem and thus we can bound the advantage by that of $\mathcal{A}_{\mathsf{R}}$. We provide a formal proof of this argument in Lemma 4.

Lemma 4. ($\mathsf{RLWE} \implies \mathsf{FLERLWE}^{M,k}$) *Let n, m, q be parameters of the ring $\mathcal{R}_q$ and let $L = L(\lambda)$, $M = M(\lambda)$, $k = k(\lambda)$ be additional chosen parameters. Let $\chi_k = \overline{D_{\mathbb{Z},k\sigma}}$ be the B-bounded discrete Gaussian for parameter σ. Let χ be the discrete Gaussian $D_{\mathbb{Z},\sigma}$. Let $\mathcal{A}_{\mathsf{R}}$ be a distinguisher attempting to distinguish samples in the $\mathsf{RLWE}_{n,m,q,\chi_k}$ problem, and likewise $\mathcal{A}_{\mathsf{F}}$ an adversary attempting to solve the $\mathsf{FLERLWE}^{M,k}_{n,m,q,\chi_k}$ problem. Then we have that:*

$$\mathsf{Adv}(\mathcal{A}_{\mathsf{F}}) \leq \mathsf{Adv}(\mathcal{A}_{\mathsf{R}}),$$

where L, k, M, B are chosen such that $\binom{L}{k} \cdot f_s >> M$ for all $s \in \mathsf{poly}(\chi_k^n)$, where $f_s = \Pr[X = s]$ for a random variable distributed over χ_k^n.

Proof. $\mathcal{A}_{\mathsf{R}}$ receives a sample of the form (a, g) where $a \in \mathcal{R}_q$ is a uniformly sampled ring element and either $g = a\tilde{s} + e$ for $\tilde{s}, e \leftarrow_{\$} \mathsf{poly}(\chi_k^n)$ or $g = u$ where u is uniformly sampled from $\mathcal{R}_q$.

$\mathcal{A}_{\mathsf{R}}$ samples $s_0, \ldots, s_{L-1} \leftarrow_{\$} \mathsf{poly}(\chi^n)$. Let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be a distribution ensemble that samples $x \in \{0,1\}^L$ with hamming weight k, with min-entropy $\alpha(\lambda)$. By Theorem 1 the distribution produced by

$$\sum_{i=0}^{L-1} x_i \cdot s_i$$

is statistically close to $D_{\mathbb{Z},k\sigma}$ (since $k\sigma = \sqrt{(\sum_i x_i \sigma)^2}$ for x with hamming weight k). Therefore, we can naturally say that it is statistically close to χ_k by appropriate choice of bound B.

Let Y be a random variable associated with χ_k^n, then the probability $\Pr[Y = s]$, for some s, is statistically close to the probability $\Pr[X = x_s]$ where $x_s \in \{0,1\}^L$ satisfies $s = \sum_i x_{s,i} s_i$ for a random variable X. Let X_s denote the set of values x_s that satisfy this property. Notice, that there are $\binom{L}{k}$ possible strings (of hamming weight k) that the random variable X is distributed over and $f_s = \Pr[X = x_s]$ denotes the probability mass of sampling a string that sums to $s \in \mathsf{poly}(\chi_k^n)$. Since f_s is only dependent on k, we can choose L with respect to M, B (and fix k large enough), such that:

$$\binom{L}{k} f_s >> M,$$

where, with overwhelming probability, we have that $|X_s| \geq M$ for all s samplable from χ_k^n (since this is a finite set and $\binom{L}{k}$ increases exponentially). The same applies for the secret $\tilde{s}$ that is sampled in the original RLWE sample (a, g).

Finally, $\mathcal{A}_{\mathsf{R}}$ invokes $\mathcal{A}_{\mathsf{F}}$ with $P = \{s_0, \ldots, s_{L-1}\}$ and the sample (a, g). In both cases for g this simulates the $\mathsf{FLERLWE}^{M,k}_{n,m,q,\chi}$ game. Specifically, when $g = a\tilde{s} + e$ there are at least M possible strings $x \in \{0,1\}^L$ such that $\tilde{s} = \sum_i x_i s_i$, and these are uniformly distributed. In the second case the distribution is trivially simulated. Thus the proof is complete. □

Remark 2. Note that Lemma 4 constructs a reduction with a slightly different structure to that shown in Definition 11. That is, in the statement of FLERLWE there are M elements within P that are sampled as sums of other elements and $\tilde{s}$, but these are replaced with L independently sampled elements in the reduction.

4 Directed encoding schemes

In this section, we detail the abstract interface that we use for our directed encoding scheme. Directed encoding schemes can be thought of as a special case of GGH15 where the graded encoding is specialised

to a line rather than a graph topology. The definition that we use is similar to those used in [10, 57] except that we define additions between encodings that lie on the same path segment. It is shown in [10] that the GGH15 MMAP [33] securely implements the directed encoding paradigm. That is, that the GXDH and entropic security properties are fulfilled. In Appendix A we show that the same paradigm also instantiates the α-LE security property (Definition 14) that we require, assuming the hardness of $\mathsf{FLERLWE}^{M,k}$.

Let $\mathcal{R}_q$ be a ring and let $\mathcal{M} \subseteq \mathcal{R}_q$ be a message distribution over $\mathcal{R}_q$.

Definition 12. (Directed encoding scheme) *A directed encoding scheme associated with a message space $\mathcal{M} \subseteq \mathcal{R}_q$ is a tuple of PPT algorithms:*

$$(\mathsf{Setup}(\cdot), \mathsf{Encode}(\cdot), \mathsf{REncode}(\cdot), \mathsf{Mult}(\cdot), \mathsf{Add}(\cdot), \mathsf{ZeroTest}(\cdot))$$

which work in the following way.

- $(\mathsf{pk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$: *On input a security parameter λ and an upper bound κ on the number of 'levels' in the encoding scheme, outputs a public key pk and a private encoding key ek.*
- $v \leftarrow \mathsf{Encode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, s)$: *On input a key pair $(\mathsf{pk}_0, \mathsf{ek}_0)$, a 'target' public key pk_1 and a message $s \in \mathcal{M}$, outputs an encoding $v = [s]_{0\to 1}$.*
- $v \leftarrow \mathsf{REncode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, 1^\lambda)$: *Same as above, except that no input message is specified and the output v encodes a randomly sampled message $m \in \mathcal{M}$.*
- $v \leftarrow \mathsf{Mult}(v_1, v_2)$: *On input encodings $v_1 = [s_1]_{i-1\to i}$ and $v_2 = [s_2]_{i\to i+1}$, outputs an encoding $v = [s_1 \cdot s_2]_{i-1\to i+1}$.*
- $v \leftarrow \mathsf{Add}(v_1, v_2)$: *On input encodings $v_1 = [s_1]_{i\to i+1}$ and $v_2 = [s_2]_{i\to i+1}$, outputs an encoding $v = [s_1 + s_2]_{i\to i+1}$*
- $b \leftarrow \mathsf{ZeroTest}(\mathsf{pk}_0, v)$: *On input an encoding $v = [s]_{0\to 1}$ and source public key pk_0, outputs a bit $b \in \{0, 1\}$ where $b = 1$ if and only if $s = 0$ and $b = 0$ otherwise.*

Remark 3. In the work of [10] they alternatively define an equality test rather than the zero test that we provide. The difference between these tests is purely semantic.

Correctness. We require the following correctness properties to hold with probability $1 - \mathsf{negl}(\lambda)$.

- For all i, j for which $i < j < l \leq \kappa$, let $v_1 = [s_1]_{i\to j}, v_2 = [s_2]_{j\to l}$. Then $\mathsf{Mult}(v_1, v_2) = v^*$ where $v^* = [s_1 \cdot s_2]_{i\to l}$.
- For all $i < \kappa$ and $v_1, v_2 \leftarrow \mathsf{Encode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, s_1), \mathsf{Encode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, s_2)$, we have $\mathsf{ZeroTest}(\mathsf{pk}_0, v_1 - v_2) = 1$ iff $s_1 = s_2$.

Note that we only require the first public key pk_0 for performing equality/zero testing. If an encoding is malformed we do not provide any correctness guarantees.

Security properties. Below we define the GXDH2 and GXDH1 security properties, these are derivations of the original GXDH security game that is shown in [10]. We also give a formal description of the linear α-entropic property (α-LE) that we use for proving security of our obfuscator in Section 5.

GXDH security. Define the GXDH2 security game using the $\mathsf{exp}^{\mathsf{gxdh}}$ and $\mathsf{exp}^{\mathsf{rand}}$ experiments shown in Figure 2.

Definition 13. (GXDH2 security) *For every polynomial $\kappa = \kappa(\lambda)$, the outputs of the experiments detailed in $\mathsf{exp}^{\mathsf{gxdh}}(1^\lambda, 1^\kappa)$ and $\mathsf{exp}^{\mathsf{rand}}(1^\lambda, 1^\kappa)$ should be computationally indistinguishable for all PPT adversaries. That is, the advantage of any such adversary in $\mathsf{exp}^{\mathsf{gxdh}}$ should be bounded by a negligible function.*

Let GXDH1 represent the same two experiments, $\mathsf{exp}^{\mathsf{gxdh}}$ and $\mathsf{exp}^{\mathsf{rand}}$, except where the output is modified to only include $(\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{ek}_0, u)$.

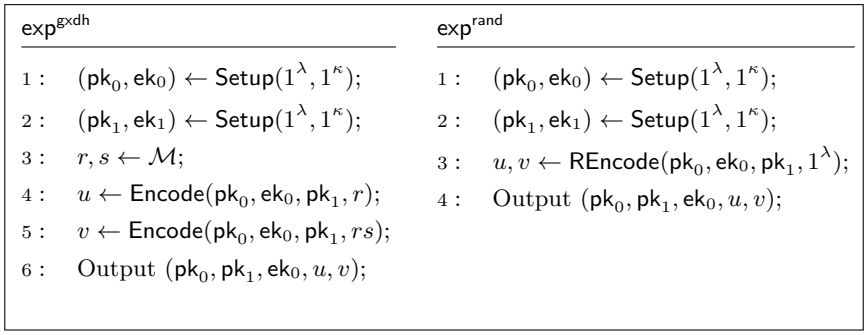

Fig. 2. Experiments $\mathsf{exp}^{\mathsf{gxdh}}$ and $\mathsf{exp}^{\mathsf{rand}}$ for the GXDH2 security game.

Linear α-entropic security. As in [10] we are required to define an additional security requirement for our directed encoding scheme abstraction. We show in Appendix A that our instantiation of a directed encoding scheme satisfies this property based on the hardness of the $\mathsf{FLERLWE}^{M,k}$ problem (or RLWE, by extension).

Definition 14. (Linear α-entropic security) *For parameters $k = k(\lambda)$, $M = M(\lambda)$, $L = L(\lambda)$, $\kappa = L + 1$ and let $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ be an efficiently samplable distribution that samples strings of hamming weight k from $\{0,1\}^L$ with min-entropy $\alpha(\lambda)$. Let $I = \{i_0, \ldots, i_{k-1}\}$ contain the string indices where $x_i^{(\beta)} = 1$. Let P be an array of length L initialised to 0. Sample key pairs $(\mathsf{pk}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$, $(\mathsf{pk}', \mathsf{ek}') \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$. Sample $\tilde{t} \leftarrow_\$ \mathcal{M}$. For each $\beta \in [M-1]$ do the following:*

- *sample $x^{(\beta)} \leftarrow_\$ D_\lambda$ and let $j^* \in [k-1]$ be some index such that $P[i_{j^*}] = 0$;*
- *for $j \neq j^*$: if $P[i_j] = 0$, sample $f_{i_j} \leftarrow_\$ \mathcal{M}$; else if $P[i_j] \neq 0$, let $f_{i_j} = P[i_j]$;*
- *let $f_{i_{j^*}} = \tilde{t} - \sum_{i_j \in I, j \neq j^*} f_{i_j}$ and set $P[i_{j^*}] = f_{i_{j^*}}$*

For any indices l such that $P[l] = 0$ still, sample $f'_l \leftarrow_\$ \mathcal{M}$ and let $P[l] = f'_l$. Shuffle P and let $\mathcal{A}$ be a PPT adversary that receives $(\mathsf{pk}, \mathsf{ek}, \mathsf{pk}', P, v_\kappa)$ where, either $v_\kappa \leftarrow \mathsf{Encode}(\mathsf{pk}, \mathsf{ek}, \mathsf{pk}', \tilde{t})$, or $v_\kappa \leftarrow \mathsf{REncode}(\mathsf{pk}, \mathsf{ek}, \mathsf{pk}', 1^\lambda)$. A PPT adversary $\mathcal{A}$ succeeds if they distinguish between the cases with non-negligible advantage.

The α-LE security property contains a significant amount more structure in comparison to the entropic security property of [10]. In particular, we fix a parameter $\tilde{t}$ and require that there are multiple strings $x^{(\beta)}$ that satisfy a subset sum of the publicly sampled elements up to $\tilde{t}$. Observe for both GXDH and α-LE that we do not give access to the full encoding parameters, by withholding the encoding key ek'. This is in contrast with the GCAN assumption used by Brakerski and Rothblum [8].

5 The obfuscator

In this section, we detail how we can use the abstracted directed encoding scheme shown in Section 4 to instantiate a Bloom filter obfuscator satisfying α-entropic security.

5.1 Construction

Let Γ be a directed encoding scheme as defined in Section 4. We associate Γ with a message distribution $\mathcal{M}$ over the ring $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let $\mathcal{D}_\alpha$ be a class of distribution ensembles $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ sampling $(F^S_{\boldsymbol{BF}}, S) \leftarrow_\$ D_\lambda$ where S has min-entropy $\alpha = \alpha(\lambda)$ given $F^S_{\boldsymbol{BF}}$. Let $(\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S) \in \mathcal{C}^{\mathcal{S}}_{\boldsymbol{BF}}$ be a Bloom filter of length L representing the set S and hash functions $h_j : \mathcal{S} \mapsto [L-1]$ for $j \in [k-1]$ sampled

from a strongly universal hash family $\mathcal{H}$.[8] We abuse notation and write that $F^S_{\boldsymbol{BF}} = (\boldsymbol{BF}, (h_0, \ldots, h_{k-1}))$ though $F^S_{\boldsymbol{BF}}$ actually describes the result of evaluating

$$F^S_{\boldsymbol{BF}}(y) = \bigwedge_{j=0}^{k-1} \boldsymbol{BF}[h_j(y)]$$

for some given input element $y \in \mathcal{S}$.

We define $\mathcal{O}(1^\lambda, F^S_{\boldsymbol{BF}}) \to (\Pi_S, \mathsf{aux})$ to be the obfuscator that takes a functionality $F^S_{\boldsymbol{BF}}$ as input and outputs an obfuscated version of it, Π_S. An evaluator can interact with Π_S using valid inputs for the function $F^S_{\boldsymbol{BF}}$ and receiving correct outputs without learning the underlying set S.

First the obfuscator computes the ring-based, garbled Bloom filter given by $(\boldsymbol{RGBF}, (h_0, \ldots, h_{k-1}), S, \tilde{\rho})$, encoding the set S – we continue to use $F^S_{\boldsymbol{BF}}$ to denote the function that we are obfuscating. This procedure is given in Figure 3.

$\mathcal{O}$: Computation of $\boldsymbol{RGBF}$

- Sample $\tilde{\rho} \leftarrow_\$ \mathcal{M}$.
- For each $y \in S$ compute $J = \{h_j(y)\}_{j \in [k-1]}$ and for $y_j \in J$ ($j \in [k-2]$): if $\boldsymbol{RGBF}[y_j] = 0$, then sample $s_j \leftarrow_\$ \mathcal{M}$ and set $\boldsymbol{RGBF}[y_j] = s_j$, else: let $s_j = \boldsymbol{RGBF}[y_j]$. Finally, compute

$$s_{k-1} = \tilde{\rho} - (\sum_{j=0}^{k-2} s_j)$$

  and set $\boldsymbol{RGBF}[y_{k-1}] = s_{k-1}$. Note, $\boldsymbol{RGBF}[y_{k-1}]$ must be equal to 0; if not, swap for $y_j \in J$ that does satisfy this.
- Output $(\boldsymbol{RGBF}, (h_0, \ldots, h_{k-1}), S, \tilde{\rho})$.

Fig. 3. Method for constructing $\boldsymbol{RGBF}$ from a Bloom filter $\boldsymbol{BF}$ for a set S.

The obfuscator then proceeds with sampling new random elements from $\mathcal{M}$ and encoding these using Γ. We provide the rest of the construction in Figure 4.

5.2 Evaluation

Let $y^* \in \mathcal{S}$ be an element that we wish to query on the obfuscated program. Now that we have shown how to construct Π_S, we must show how we can evaluate the obfuscated program for learning whether y^* satisfies $y^* \in S$. Recall that Π_S is an obfuscated data structure that represents the set S. In particular, the obfuscation should take elements in the universe $\mathcal{S}$ as input and output 0 if $y^* \notin S$ and 1 otherwise.

Observe, that for an encoding $\zeta_0 = [\eta_0]_{0\to 1}$ and $v_1 = ([r_1]_{1\to 2}, [r_1 f_1]_{1\to 2})$, we can compute

$$\zeta_0 \cdot v_1 = ([\eta_0 r_1]_{0\to 2}, [\eta_0 r_1 f_1]_{0\to 2}).$$

The same holds if the paths were reversed, though the order of multiplication must also change.

In addition, for two encodings v_0, v_1 on paths $0 \to 1$, $1 \to 2$ respectively, we define the operation $\boxplus$ where

$$v_0 \boxplus v_1 = ([r_0 r_1]_{0\to 2}, [r_0 r_1 (f_0 + f_1)]_{0\to 2}),$$

this can be thought of as an addition of encodings on sequential paths.[9] As such, we do not define standard multiplications for pairs of this form. In the below, we define an interleaved product; i.e. for some disjoint

[8] Parameters L, k are chosen as in Section 2.4 to prevent false positives occurring.
[9] Subtraction also holds in the same way, we denote this operation by $\boxminus$.

$\mathcal{O}$: Construction of Π_S

- Let $\kappa = L + 1$, choose $(\mathsf{pk}_i, \mathsf{ek}_i) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ for $i \in [\kappa]$
- Sample $\eta_i, r_i \leftarrow\!\!\$\, \mathcal{M}$ for $i \in [L-1]$.
- For j where $\boldsymbol{RGBF}[j] = 0$ sample $\gamma_j \leftarrow\!\!\$\, \mathcal{M}$. For $i \in [L-1]$, let $f_i = \gamma_i$ for $\boldsymbol{RGBF}[i] = 0$ and let $f_i = \boldsymbol{RGBF}[i] = s_i \neq 0$, otherwise.
- Compute L pairs $c_i = (r_i, r_i \cdot f_i)$ for each $i \in [L-1]$.
- Sample $\hat{r} \leftarrow\!\!\$\, \mathcal{M}$ and compute $c_{\tilde{\rho}} = (\hat{r}, \hat{r} \cdot \tilde{\rho})$.
- For $i \in [L-1]$, compute encodings

$$v_i = [c_i]_{i \to i+1} = (\mathsf{Encode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, r_i), \mathsf{Encode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, r_i \cdot f_i)),$$

  and

$$\zeta_i = [\eta_i]_{i \to i+1} = \mathsf{Encode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, \eta_i)$$

- Additionally, compute an encoding:

$$v_\kappa = [c_{\tilde{\rho}}]_{L \to L+1} = (\mathsf{Encode}(\mathsf{pk}_L, \mathsf{ek}_L, \mathsf{pk}_{L+1}, \hat{r}), \mathsf{Encode}(\mathsf{pk}_L, \mathsf{ek}_L, \mathsf{pk}_{L+1}, \hat{r} \cdot \tilde{\rho})).$$

- Output the obfuscated program as

$$\Pi_S = (\mathsf{pk}_0, \{\zeta_i\}_{i \in [L-1]}, \{v_i\}_{i \in [L-1]}, v_\kappa, (h_0, \ldots, h_{k-1})). \tag{10}$$

Fig. 4. Method for constructing the obfuscator Π_S.

sets I, J such that $I \cup J = [L-1]$ we compute:

$$\left(\prod_{i \in I} \zeta_i\right) \cdot \left(\boxplus_{j \in J} v_j\right)$$

The notation above defines a shorthand notation for performing operations over encodings defined along the paths they are indexed by. For example, if we want to compute $\zeta_0, v_1, \zeta_2, v_3, v_4, \zeta_5$ – indexed along the path from 0 to 6 sequentially – we denote $\zeta_0 \cdot v_1 \cdot \zeta_2 \boxplus v_3 \boxplus v_4 \cdot \zeta_5$ by defining $J = \{1, 3, 4\} \subset I = [5]$ and using the notation above.[10] In summary, we 'add' ($\boxplus$) sequential pairs of encodings v_0, v_1 and we multiply in all other cases. We refer back to Figure 1 for an example of how paths are traversed.

Let $S' \subset \mathcal{S}$ be a set of elements that is used for querying the obfuscated program. We detail the evaluation procedure in Figure 5. As above, we take J to be the set of k indices in the Bloom filter array that are chosen by the functions $h_i \leftarrow\!\!\$\, \mathcal{H}$; I represents all other indices.

$\mathcal{O}$: Evaluation of Π_S

- Parse Π_S as $(\mathsf{pk}_0, \{\zeta_i\}_{i \in [L-1]}, \{v_i\}_{i \in [L-1]}, v_\kappa, \{h_j\}_{j \in [k-1]})$.
- Let $y^* \in S'$ be a query element, and compute $J = \{h_j(y^*)\}$ for $0 \leq j \leq k-1$.
- Set $\mathrm{I} = [L-1] \setminus J$
- Compute the following interleaved product:

$$v_{y^*} = \left(\prod_{i \in I} \zeta_i\right) \cdot \left(\boxplus_{j \in J} v_j\right) \tag{11}$$

  where v_{y^*} takes the form

$$v_{y^*} = [c_{y^*}]_{0 \to L} = ([r_{y^*}]_{0 \to L}, [r_{y^*} \theta_{y^*}]_{0 \to L})$$

  for some value $\theta_{y^*} \in \mathcal{M}$.
- Finally, compute

$$x_{y^*} = v_\kappa \boxminus v_{y^*} = ([r_{y^*} \hat{r}]_{0 \to L+1}, [r_{y^*} \hat{r}(\tilde{\rho} - \theta_{y^*})]_{0 \to L+1}).$$

- Let $x_{y^*,1} = x_{y^*}[1]$ be the second component of x_{y^*} and output $b \leftarrow \mathsf{ZeroTest}(\mathsf{pk}_0, x_{y^*,1})$.

Fig. 5. Procedure for evaluating the obfuscated program Π_S on some input y^*.

[10] Note, that $(\zeta_0 \cdot v_1 \cdot \zeta_2)$ is a pair encoding and thus we compute $(\zeta_0 \cdot v_1 \cdot \zeta_2) \boxplus v_3$.

Correctness of evaluation. It is essential to show that the above evaluation procedure results in the evaluator receiving the correct answer in all but a negligible amount of cases. In particular, we need to show that:

$$\Pr[\Pi_S(y^*) = 1 \mid y^* \in S] = 1, \tag{12}$$

and, in addition:

$$\Pr[\Pi_S(y^*) = 0 \mid y^* \notin S] = 1 - \mathsf{negl}(\lambda), \tag{13}$$

for all inputs $y^* \in \mathcal{S}$.

Firstly, take the case where $y^* \in S$. Then v_{y^*} from above is an encoding of the form:

$$v_{y^*} = ([r_{y^*}]_{0\to L}, [r_{y^*}\theta_{y^*}]_{0\to L})$$

where r_{y^*} is the accumulated product of randomness that is accrued during the interleaved product. Consequently, θ_{y^*} is the summation of all f_j terms associated with $j \in J = \{h_j(y^*)\}$. Since $y^* \in S$, by the construction of $\boldsymbol{RGBF}$, we have that $f_j = s_j$ for $j \in [k-1]$. In particular, by the method of sampling we have that

$$\sum_{j\in J} f_j = \sum_{j\in J} s_j = \tilde{\rho}$$

and thus when we compute x_{y^*} we get

$$x_{y^*} = ([r_{y^*}\hat{r}]_{0\to L+1}, [r_{y^*}\hat{r}(\tilde{\rho} - \theta_{y^*})]_{0\to L+1}) = ([r_{y^*}\hat{r}]_{0\to L+1}, [0]_{0\to L+1}).$$

Subsequently, computing $b \leftarrow \mathsf{ZeroTest}(\mathsf{pk}_0, x_{y^*,1})$ we get that $b = 1$, which is the correct output.

Secondly, take the case where $y^* \notin S$. Then v_{y^*} takes the same form as above, except that for $j \in J = \{h_j(y^*)\}$ we know $\exists$ a subset $N \subseteq J$ (with high probability) such that $f_n = \gamma_n$ for all $n \in [N]$ where γ_n is sampled fresh from $\mathcal{M}$. The probability of this not being the case is bounded by the negligible probability of false positives occurring in the original Bloom filter. Now, we have that

$$\sum_{j\in J} f_j \neq \tilde{\rho}$$

with high probability (by the Schwarz-Zippel lemma) due to the presence of independently sampled elements in the sum. As such, the output of $\mathsf{ZeroTest}(\mathsf{pk}_0, x_{y^*,1})$ is equal to 0 with high probability. This result still holds when an adversary chooses arbitrary J with $|J| = k' \neq k$, by the same argument.

5.3 VBB security

The final stage of our construction requires that we show that our obfuscator satisfies α-entropic security, as shown in Definition 6. Our obfuscator clearly meets the functionality and polynomial slowdown requirements of the definition.

Theorem 2. (α-entropic security) *Let $\mathcal{O}$ be the obfuscator from Section 5. Based on the hardness of the GXDH and α-LE security properties of our encoding scheme Γ, and the strongly universal hash family $\mathcal{H}$, $\mathcal{O}$ satisfies α-entropic security (Definition 8).*

Proof. Let $\mathsf{Sim}(1^\lambda, 1^\kappa)$ denote the simulator that we consider when proving the security of our obfuscator with $\kappa = L + 1$. First $\mathsf{Sim}(1^\lambda, 1^\kappa)$ chooses $(\mathsf{pk}_j, \mathsf{ek}_j) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ for $j \in [\kappa]$. For $i \in [L-1]$ it then samples

$$\{U_i\}_i \leftarrow_\$ \mathsf{REncode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, 1^\lambda),$$

$$\{U_i'\}_i \leftarrow_\$ (\mathsf{REncode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, 1^\lambda), \mathsf{REncode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, 1^\lambda)),$$

and

$$U_\kappa \leftarrow_\$ (\mathsf{REncode}(\mathsf{pk}_L, \mathsf{ek}_L, \mathsf{pk}_{L+1}, 1^\lambda), \mathsf{REncode}(\mathsf{pk}_L, \mathsf{ek}_L, \mathsf{pk}_{L+1}, 1^\lambda)).$$

The hash functions $h'_0, \ldots, h'_{k-1}$ are sampled freshly from $\mathcal{H}$ and so they still satisfy the strongly universal property. These are defined independently of the functions $h_0, \ldots, h_{k-1}$ used in the real-world execution. Finally $\mathsf{Sim}(1^\lambda, 1^\kappa)$ outputs the simulated program:

$$\tilde{\Pi} = (\mathsf{pk}_0, \{U_i\}_i, \{U'_i\}_i, U_\kappa, (h'_0, \ldots, h'_{k-1})), \tag{14}$$

where the ordering of the output respects the ordering imposed by the path of encodings.

A key facet of Sim is that it does not have access to the set S and thus cannot construct a Bloom filter for this set. Therefore, finding an element $y^* \in \mathcal{S}$ that satisfies $F^S_{\boldsymbol{BF}}$ using the real program leads to a distinguishing attack. By sampling S such that $F^S_{\boldsymbol{BF}}$ is evasive, an adversary has very low probability of being able to sample such a satisfying y^*. Hence our obfuscator only satisfies security in this setting.

Let $\mathcal{D}_\alpha$ be the class of distribution ensembles $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$ sampling the pairs $(F^S_{\boldsymbol{BF}}, S) \leftarrow\!\!\$\, D_\lambda$ where S has min-entropy $\alpha(\lambda)$ given $F^S_{\boldsymbol{BF}}$. We want to show that the real program (Π_S, aux) is indistinguishable from the simulated program $(\tilde{\Pi}, \mathsf{aux})$ where $\Pi_S \leftarrow \mathcal{O}(1^\lambda, F^S_{\boldsymbol{BF}})$ and $\tilde{\Pi} \leftarrow \mathsf{Sim}(1^\lambda, 1^\kappa)$. We first prove the following lemma and then use this result to provide a proof for the overarching theorem.

Lemma 5. *A PPT adversary $\mathcal{A}$ attempting to distinguish between the distributions shown in Equation* (10) *and Equation* (14) *has advantage bounded by*

$$\mathsf{negl}(\lambda) + M \cdot \sqrt{\frac{2\pi k(L-k)}{L}} \cdot \left(\frac{k}{L-k}\right)^k \cdot \left(\frac{L-k}{L}\right)^L$$

Proof. **Hybrid** 0**:** This is the distribution as received directly from the obfuscated program Π_S as shown in Equation (10).

Hybrid 1**:** This is the same as the distribution in Hybrid 0 except that we now sample an independent set of hash functions $(h'_0, \ldots, h'_{k-1}) \leftarrow\!\!\$\, \mathcal{H}$, for the strongly universal hash family $\mathcal{H}$.

Notably, there is now a high probability that no $y \in \mathcal{S}$ will satisfy the requirement that $F^S_{\boldsymbol{BF}}(y) = 1$. However, since $F^S_{\boldsymbol{BF}}$ is evasive in the real setting (Hybrid 0) by the sampling of S then the probability of an adversary sampling a $y \in \mathcal{S}$ in Hybrid 0 that satisfies $F^S_{\boldsymbol{BF}}$ is bounded by $\mathsf{negl}(\lambda)$. Therefore, the advantage of an adversary distinguishing between these two hybrids is bounded by the same negligible probability.

Hybrid 2**:** This is the same distribution as in Hybrid 1, except that we replace v_κ with the randomly sampled encoding $U_\kappa \leftarrow (\mathsf{REncode}(\mathsf{pk}_L, \mathsf{ek}_L, \mathsf{pk}_{L+1}, 1^\lambda), \mathsf{REncode}(\mathsf{pk}_L, \mathsf{ek}_L, \mathsf{pk}_{L+1}, 1^\lambda))$. We argue that we can bound the advantage of an adversary attempting to distinguish between the two games by an adversary that is attempting to break the α-LE security of Γ. Let $\mathcal{A}'$ be an adversary in the α-LE game, let $\mathcal{A}_{1,2}$ be an adversary attempting to distinguish between Hybrid 1 and Hybrid 2. We show that $\mathcal{A}'$ is able to create two distributions for $\mathcal{A}_{1,2}$ that are indistinguishable from those in Hybrids 1 and 2.

Let D denote the distribution uniformly sampling Bloom filter functionalities $F^S_{\boldsymbol{BF}} = (\boldsymbol{BF}, (h_0, \ldots, h_{k-1}), S)$ from the class $\mathcal{C}^{\mathcal{S}}_{\boldsymbol{BF}}$. Let L_1 denote the 'hamming weight' of $\boldsymbol{BF}$. We write

$$(\mathsf{pk}, \mathsf{ek}, \mathsf{pk}', P, v_\kappa) \tag{15}$$

as the distribution received by $\mathcal{A}'$ in the α-LE game where P is the array of length L_1 containing elements $f_i \leftarrow\!\!\$\, \mathcal{M}$ for $i \in [L_1 - 1]$. It is clear that, either $v_\kappa \leftarrow \mathsf{Encode}(\mathsf{pk}, \mathsf{ek}, \mathsf{pk}', \tilde{t})$ (for $\tilde{t} \leftarrow\!\!\$\, \mathcal{M}$) or $v_\kappa \leftarrow \mathsf{REncode}(\mathsf{pk}, \mathsf{ek}, \mathsf{pk}', 1^\lambda)$.

Sample a key pair $(\mathsf{pk}'', \mathsf{ek}'') \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ and elements $\hat{r} \leftarrow \mathcal{M}$. Compute $v_{\hat{r}} \leftarrow \mathsf{Encode}(\mathsf{pk}'', \mathsf{ek}'', \mathsf{pk}, \hat{r})$ and $v'_{\hat{r}} \leftarrow \mathsf{Encode}(\mathsf{pk}'', \mathsf{ek}'', \mathsf{pk}', \hat{r})$ then let:

$$v_\kappa = (v'_{\hat{r}}, v_{\hat{r}} \cdot v_\kappa) = ([\hat{r}]_{\mathsf{pk}'' \to \mathsf{pk}'}, [\hat{r}\tilde{t}]_{\mathsf{pk}'' \to \mathsf{pk}'}).$$

Secondly, sample L key pairs $\{(\mathsf{pk}_i, \mathsf{ek}_i)\}_{i \in [L-1]} \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$ and then let $(\mathsf{pk}_L, \mathsf{ek}_L) = (\mathsf{pk}'', \mathsf{ek}'')$ and $\mathsf{pk}_{L+1} = \mathsf{pk}'$. Sample a set of $L_0 = (L - L_1)$ elements $\S_0 = (f'_0, \ldots, f'_{L_0-1})$ from $\mathcal{M}$. Sample additional sets of values $T = (\eta_0, \ldots, \eta_{L-1})$ and $R = (r_0, \ldots, r_{L-1})$ from $\mathcal{M}$. Let $\S_1 = P$ and notice that $|\S_0 \cup \S_1| = L$.

Let J denote the indices where $\boldsymbol{BF}[j] = 0$ and let I denote the indices such that $\boldsymbol{BF}[i] = 1$ for all $i \in I$. Correspondingly, let $\boldsymbol{RGBF}[j] = f'_j$ for each $j \in J$, where f'_j is a randomly chosen element from $\S_0$. For $i \in I$, let $\boldsymbol{RGBF}[i] = f_i$ for randomly chosen $f_i \in \S_1$. Observe that $\boldsymbol{RGBF}[l] = f_l$ for $l \in [L-1]$ where we have $f_l \in \S_0$ or $f_l \in \S_1$.[11]

Compute the following encodings:

- $\zeta_l = [\eta_l]_{l \to l+1}$;
- $v_l = ([r_l]_{l \to l+1}, [r_l f_l]_{l \to l+1})$;

for $l \in [L-1]$. Recall, that v_κ is an encoding with respect to the path $L \to L+1$. Output the following distribution:

$$(\mathsf{pk}_0, \{\zeta_l\}_{l \in [L-1]}, \{v_l\}_{l \in [L-1]}, v_\kappa, (h'_0, \ldots, h'_{k-1})),$$

when v_κ, from Equation (15), is an encoding of the value $\tilde{t}$ the distribution is equivalent to the one in Hybrid 1. Alternatively, if v_κ is a randomly sampled encoding, then the distribution is equivalent to the one proposed in Hybrid 2. Thus, the distributions can be successfully constructed for $\mathcal{A}_{1,2}$ by an instantiation of $\mathcal{A}'$ and thus the bound is correct.

Hybrid 3.$[i]$**:** We define Hybrid 3.$[-1]$ to be the same as Hybrid 2 and then define Hybrid 3.$[i]$ for $i \in [L-1]$ such that Hybrid 3.$[i]$ is the same as Hybrid 3.$[i-1]$ except that we swap v_i for

$$U'_i \leftarrow (\mathsf{REncode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, 1^\lambda), \mathsf{REncode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, 1^\lambda)).$$

Let $\mathcal{A}_{2,3}$ be an adversary attempting to distinguish between Hybrids 3.$[i]$ and 3.$[i-1]$. Let $\mathcal{A}_{\mathsf{gxdh2}}$ be an adversary in the GXDH2 game, then we show that we can bound the advantage of $\mathcal{A}_{2,3}$ by the advantage of $\mathcal{A}_{\mathsf{gxdh2}}$.

Let $(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, u, v)$ be the distribution in GXDH2, where $r, s \leftarrow\!\!\$\, \mathcal{M}$ and $u \leftarrow \mathsf{Encode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, r)$, $v \leftarrow \mathsf{Encode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, rs)$; or $u, v \leftarrow \mathsf{REncode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, 1^\lambda)$. Note that each v_i in Hybrid 3.$[i-1]$ takes the form $([r_i]_{i \to i+1}, [r_i f_i]_{i \to i+1})$ and in Hybrid 3.$[i]$ it takes the form $(U'_{i,0}, U'_{i,1})$ for some encodings $U'_{i,0}, U'_{i,1} \leftarrow \mathsf{REncode}(\mathsf{pk}_0, \mathsf{ek}_0, \mathsf{pk}_1, 1^\lambda)$.

In particular, the distributions in Hybrids 3.$[i-1]$ and 3.$[i]$ are identical to the distributions seen by $\mathcal{A}_{\mathsf{gxdh2}}$. As such, the adversary $\mathcal{A}_{2,3}$ must have advantage bounded by the GXDH adversary and thus, bounded by $\mathsf{negl}(\lambda)$ based on the GXDH hardness property.

Hybrid 4.$[i]$**:** Let Hybrid 4.$[-1]$ be the distribution in Hybrid 3.$[L-1]$, define Hybrid 4.$[i]$ to be the same as Hybrid 4.$[i-1]$ except that we swap ζ_i for $U_i \leftarrow \mathsf{REncode}(\mathsf{pk}_i, \mathsf{ek}_i, \mathsf{pk}_{i+1}, 1^\lambda)$ for $i \in [L-1]$. Let $\mathcal{A}_{3,4}$ be an adversary attempting to distinguish between Hybrids 4.$[i-1]$ and 4.$[i]$ and let $\mathcal{A}_{\mathsf{gxdh1}}$ be an adversary in the GXDH1 game. Using an identical argument as the previous hybrid we can show that the advantage of $\mathcal{A}_{3,4}$ is bounded by the advantage of $\mathcal{A}_{\mathsf{gxdh1}}$. Thus, by the hardness of the GXDH security property, we are done.

Note that the distribution in Hybrid 4.$[L-1]$ is of the form

$$(\mathsf{pk}_0, \{U_i\}_i, \{U'_i\}_i, U_\kappa, (h'_0, \ldots, h'_{k-1})),$$

which is clearly equivalent to the distribution shown in Equation (14). Notice that all proof hops require only a negligible loss in security so there exists a negligible function, negl', such that we can bound the sum total of losses by $\mathsf{negl}(\lambda)'$. However, notice that in Hybrid 0, the adversary knows that there exists M subsets of size k that satisfy the obfuscator. By the final distribution the adversary has lost this characteristic (with high probability), since all encodings are sampled randomly.

Consequently, the adversary has a probability of success equal to choosing any of M subsets of size k from a total set of size L. Importantly, since the hash family $\mathcal{H}$ is strongly universal, then the output of h_i is uniform across $[L-1]$ for all $i \in [k-1]$. As a result, the adversary's best chance of winning is by

[11] We do not worry about choosing locations using $h'_0, \ldots, h'_{k-1}$ since these hash functions are defined independently the choice of $\boldsymbol{BF}$ by Hybrid 1.

choosing k random entries and evaluating the obfuscator on these entries. Therefore, the total security loss in distinguishing between the hybrid games is actually equal to:

$$\mathsf{Adv}(\mathcal{A}) = \mathsf{negl}(\lambda)' + \frac{M}{\binom{L}{k}}$$

and so, by Stirling's approximation, we have

$$\mathsf{Adv}(\mathcal{A}) \approx \mathsf{negl}(\lambda)' + M \cdot \sqrt{\frac{2\pi k(L-k)}{L}} \cdot \left(\frac{k}{L-k}\right)^k \cdot \left(\frac{L-k}{L}\right)^L,$$

and thus the proof of Lemma 5 is concluded. □

Finally for proving Theorem 2, we have to prove that the bound provided by Lemma 5 is negligible for the choice of parameters implied by the use of a Bloom filter. By Equation (8) we choose $L \geq M \cdot \log_2 e \cdot \log_2 1/\epsilon$, where L is the length of the Bloom filter, $M = |S|$ and ϵ is the chance of a false-positive occurring. Recall, that we choose L large enough that we can apply Lemma 4 and so by Equation (7) we get that $k = L/M \ln 2$.[12] Firstly, it is clear that:

$$M \cdot \sqrt{\frac{2\pi k(L-k)}{L}} \cdot \left(\frac{k}{L-k}\right)^k \cdot \left(\frac{L-k}{L}\right)^L \leq M \cdot \sqrt{2\pi k} \cdot \left(\frac{k}{L-k}\right)^k \cdot \left(\frac{L-k}{L}\right)^L \tag{16}$$

and now, for appropriate choice of k and for increasing $L >> k$, then we have that $(k/L-k)^k \cdot (L-k/L)^L \to 0$ exponentially. As a consequence, it is possible to choose L and k such that the quantity in Equation 16 is negligible. For example, for $k = 60$, $L = 10000$ then $(k/L - k)^k << 2^{-80}$ and $(L - k/L)^L < 2^{-80}$ and M can still be chosen such that L satisfies Equation (8) and Lemma 4. As a result, we have that $\mathsf{Adv}(\mathcal{A}) < \mathsf{negl}^*(\lambda)$ for a negligible function negl^* and the proof is complete. □

Acknowledgements. This work was supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1). We would also like to thank Martin Albrecht, Carlos Cid and Zvika Brakerski for reading earlier drafts of this work and engaging in helpful discussions.

References

1. M. Ajtai. *Generating Hard Instances of the Short Basis Problem*, pages 1–9. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.
2. D. Apon, N. Döttling, S. Garg, and P. Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over ggh13. Cryptology ePrint Archive, Report 2016/1003, 2016. http://eprint.iacr.org/2016/1003.
3. B. Barak, N. Bitansky, R. Canetti, Y. T. Kalai, O. Paneth, and A. Sahai. Obfuscation for evasive functions. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 26–51. Springer, Heidelberg, Feb. 2014.
4. B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014.
5. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, Aug. 2001.
6. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.
7. P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. H. M. Smid, and Y. Tang. On the false-positive rate of bloom filters. *Inf. Process. Lett.*, 108(4):210–213, 2008.
8. Z. Brakerski and G. N. Rothblum. Obfuscating conjunctions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 416–434. Springer, Heidelberg, Aug. 2013.
9. Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 505–524. Springer, Heidelberg, Aug. 2011.

[12] In effect, k is an additional security parameter for our scheme. We could use $k = \log 1/\epsilon$ from Equation (7) though this requires minimising L, which may have implications for the results of Lemma 4.

10. Z. Brakerski, V. Vaikuntanathan, H. Wee, and D. Wichs. Obfuscating conjunctions under entropic ring LWE. In M. Sudan, editor, *ITCS 2016*, pages 147–156. ACM, Jan. 2016.
11. C. Cachin and J. Camenisch, editors. *EUROCRYPT 2004*, volume 3027 of *LNCS*. Springer, Heidelberg, May 2004.
12. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 455–469. Springer, Heidelberg, Aug. 1997.
13. R. Canetti and Y. Chen. Constraint-hiding constrained prfs for nc1 from lwe. Cryptology ePrint Archive, Report 2017/143, 2017. http://eprint.iacr.org/2017/143.
14. R. Canetti and J. A. Garay, editors. *CRYPTO 2013, Part I*, volume 8042 of *LNCS*. Springer, Heidelberg, Aug. 2013.
15. R. Canetti, S. Goldwasser, and O. Poburinnaya. Adaptively secure two-party computation from indistinguishability obfuscation. In Dodis and Nielsen [24], pages 557–585.
16. R. Canetti, G. N. Rothblum, and M. Varia. Obfuscation of hyperplane membership. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 72–89. Springer, Heidelberg, Feb. 2010.
17. J. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.
18. J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Cryptanalysis of GGH15 multilinear maps. In Robshaw and Katz [53], pages 607–628.
19. J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Zeroizing attacks on indistinguishability obfuscation over clt13. Cryptology ePrint Archive, Report 2016/1011, 2016. http://eprint.iacr.org/2016/1011.
20. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In Canetti and Garay [14], pages 476–493.
21. E. D. Cristofaro, P. Gasti, and G. Tsudik. Fast and private computation of cardinality of set intersection and union. In J. Pieprzyk, A.-R. Sadeghi, and M. Manulis, editors, *CANS 12*, volume 7712 of *LNCS*, pages 218–231. Springer, Heidelberg, Dec. 2012.
22. A. Davidson and C. Cid. An efficient toolkit for computing private set operations. In J. Pieprzyk and S. Suriadi, editors, *Information Security and Privacy: 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II (Lecture Notes in Computer Science)*. Full version at: http://eprint.iacr.org/2016/108.
23. S. K. Debnath and R. Dutta. Secure and efficient private set intersection cardinality using bloom filter. In J. Lopez and C. J. Mitchell, editors, *ISC 2015*, volume 9290 of *LNCS*, pages 209–226. Springer, Heidelberg, Sept. 2015.
24. Y. Dodis and J. B. Nielsen, editors. *TCC 2015, Part II*, volume 9015 of *LNCS*. Springer, Heidelberg, Mar. 2015.
25. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Cachin and Camenisch [11], pages 523–540.
26. Y. Dodis and A. Smith. Correcting errors without leaking partial information. In Gabow and Fagin [29], pages 654–663.
27. C. Dong, L. Chen, and Z. Wen. When private set intersection meets big data: an efficient and scalable protocol. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 789–800. ACM Press, Nov. 2013.
28. M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In Cachin and Camenisch [11], pages 1–19.
29. H. N. Gabow and R. Fagin, editors. *37th ACM STOC*. ACM Press, May 2005.
30. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.
31. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, Oct. 2013.
32. S. Garg, E. Miles, P. Mukherjee, A. Sahai, A. Srinivasan, and M. Zhandry. Secure obfuscation in a weak multilinear map model. In M. Hirt and A. D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 241–268. Springer, Heidelberg, Oct. / Nov. 2016.
33. C. Gentry, S. Gorbunov, and S. Halevi. Graph-induced multilinear maps from lattices. In Dodis and Nielsen [24], pages 498–527.
34. C. Gentry, A. B. Lewko, A. Sahai, and B. Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In V. Guruswami, editor, *56th FOCS*, pages 151–170. IEEE Computer Society Press, Oct. 2015.
35. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.
36. S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Robustness of the learning with errors assumption. In *ICS*, pages 230–240. Tsinghua University Press, 2010.
37. R. Goyal, V. Koppula, and B. Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. Cryptology ePrint Archive, Report 2017/120, 2017. http://eprint.iacr.org/2017/120.
38. S. Hada. Zero-knowledge and code obfuscation. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 443–457. Springer, Heidelberg, Dec. 2000.

39. D. Hofheinz, J. Malone-Lee, and M. Stam. Obfuscation for cryptographic purposes. In Vadhan [55], pages 214–232.
40. S. Hohenberger, G. N. Rothblum, a. shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In Vadhan [55], pages 233–252.
41. Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, Feb. 2012.
42. S. Kamara, P. Mohassel, M. Raykova, and S. S. Sadeghian. Scaling private set intersection to billion-element sets. In N. Christin and R. Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 195–215. Springer, Heidelberg, Mar. 2014.
43. F. Kerschbaum. Outsourced private set intersection using homomorphic encryption. In H. Y. Youm and Y. Won, editors, *ASIACCS 12*, pages 85–86. ACM Press, May 2012.
44. V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu. Efficient batched oblivious PRF with applications to private set intersection. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 16*, pages 818–829. ACM Press, Oct. 2016.
45. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In Cachin and Camenisch [11], pages 20–39.
46. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May 2010.
47. D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, Apr. 2012.
48. D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In Canetti and Garay [14], pages 21–39.
49. D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th FOCS*, pages 372–381. IEEE Computer Society Press, Oct. 2004.
50. E. Miles, A. Sahai, and M. Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Robshaw and Katz [53], pages 629–658.
51. B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on OT extension. Cryptology ePrint Archive, Report 2014/447, 2014. `http://eprint.iacr.org/2014/447`.
52. B. Pinkas, T. Schneider, and M. Zohner. Scalable private set intersection based on OT extension. Cryptology ePrint Archive, Report 2016/930, 2016. `http://eprint.iacr.org/2016/930`.
53. M. Robshaw and J. Katz, editors. *CRYPTO 2016, Part II*, volume 9815 of *LNCS*. Springer, Heidelberg, Aug. 2016.
54. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In D. B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
55. S. P. Vadhan, editor. *TCC 2007*, volume 4392 of *LNCS*. Springer, Heidelberg, Feb. 2007.
56. H. Wee. On obfuscating point functions. In Gabow and Fagin [29], pages 523–532.
57. D. Wichs and G. Zirdelis. Obfuscating compute-and-compare programs under lwe. Cryptology ePrint Archive, Report 2017/276, 2017. `http://eprint.iacr.org/2017/276`.

A GGH15 instantiation of directed encodings

Here, we detail how we can instantiate the encoding scheme from Section 4 using the GGH15 MMAP with a security reduction to RLWE. We do not address the correctness of the scheme or the proof of security of the GXDH property since these are shown in [10]. The $\mathsf{ZeroTest}(\cdot)$ algorithm that we define is analogous to the $\mathsf{EqualTest}()$ algorithm used in this previous work. Furthermore, the correctness of additions over encodings follows trivially by the correctness of additions in the GGH15 scheme.

Let $\mathcal{R}$ be a degree-n number ring $\mathbb{Z}[x]/\langle\phi(x)\rangle$. Specifically we can take $\phi(x) = x^n + 1$, where n is a power-of-two. Let q be a prime number and define $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ as the associated quotient ring. Let $\sigma \in \mathbb{R}^+$ be the Gaussian standard deviation parameter. In the below, we will use $D_{\mathcal{R},\sigma}$ to denote sampling ring elements from $\mathsf{poly}(\chi^n)$, where $\chi = D_{\mathbb{Z},\sigma}$. We use the polynomial ring version of GGH15 so that we have a commutative plaintext space, which we require for some of our operations.

Encoding scheme. The description of the encoding scheme closely follows the description shown in [10] so a well acquainted reader can skip this.

- $\mathsf{Setup}(1^\lambda, 1^\kappa)$ runs $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \mathsf{TrapSamp}(1^\lambda)$ where

$$(\mathsf{pk}, \mathsf{ek}) = (\boldsymbol{A}, \boldsymbol{T}) \in \mathcal{R}_q^m \times \mathcal{R}^{m \times m}.$$

In the following, we set: $n = \Theta(\kappa\lambda\log(\kappa\lambda))$, $m = \Theta(n\kappa\log q)$, $q = (\kappa\lambda)^{\Theta(\kappa)}$ and $\sigma = \Theta(\sqrt{n\kappa\log q})$ (these are the same as the parameter settings in [10]).

- $\mathcal{M} = \{s \in \mathcal{R} \ : \ \|s\|_\infty \leq m\}$
- $v \leftarrow \mathsf{Encode}(\boldsymbol{A}_0, T_0, \boldsymbol{A}_1, s)$:
  - Compute $b_1 = s\boldsymbol{A}_1 + e_1 \in \mathcal{R}_q^m$, where $e_1 \leftarrow\!\!\$\, D_{\mathcal{R}^m,\sigma}$.
  - Output a matrix $\boldsymbol{R}_{0\to 1} \leftarrow \mathsf{GaussSamp}(\boldsymbol{A}_0, \boldsymbol{T}_0, b_1, \sigma)$.[13]
- $\mathsf{REncode}(\boldsymbol{A}_0, \boldsymbol{A}_1, 1^\lambda)$ is the public encoding procedure that simply samples a matrix $\boldsymbol{R}_{0\to 1} \leftarrow D_{\mathcal{R},\sigma}^{m\times m}$.
- $\mathsf{Add}(\boldsymbol{R}, \boldsymbol{R}') = \boldsymbol{R} + \boldsymbol{R}'$ with addition over $\mathcal{R}_q$
- $\mathsf{Mult}(\boldsymbol{R}, \boldsymbol{R}') = \boldsymbol{R}\boldsymbol{R}'$ with multiplication also done over $\mathcal{R}_q$
- $b \leftarrow \mathsf{ZeroTest}(\boldsymbol{A}_0, \boldsymbol{R}, \boldsymbol{R}')$ where $b = 1$ if

$$\|\boldsymbol{A}_0(\boldsymbol{R} - \boldsymbol{R}')\|_\infty \leq q/8$$

and $b = 0$ otherwise.

Security. The instantiation above is shown to be secure with respect to the GXDH security property in [10] – the introduction of additions do not change the proof. As such, we will focus on proving that our scheme satisfies the α-LE security property shown in Definition 14.

Lemma 6. *Let n be a power of two, and let $\phi(x) = x^n + 1$ and subsequently $\mathcal{R} = \mathbb{Z}/\langle\phi(x)\rangle$. Let $q = 2^{\omega(\log\lambda)}$ such that $q \equiv 1 \mod 2n$ and define $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let $m \in \mathbb{N}$, and let $\chi = D_{\mathbb{Z},\sigma}$. Then, for every $k = k(\lambda), M = M(\lambda)$ the encoding scheme above satisfies α-LE (Definition 14) given the hardness of the $\mathsf{FLERLWE}_{n,m,q,\chi}^{M,k}$ problem.*

Proof. The proof for α-LE follows almost immediately from the hardness of the $\mathsf{FLERLWE}_{n,m,q,\chi}^{M,k}$ problem.

Let P be the array in the $\mathsf{FLERLWE}_{n,m,q,\chi}^{M,k}$ problem of length L and containing elements $s_i \leftarrow\!\!\$\, \mathsf{poly}(\chi^n)$. Observe that the following two distributions are computationally indistinguishable as they follow the exact format of the problem:

$$(\boldsymbol{A}_1, P, \tilde{s}\cdot\boldsymbol{A}_1 + e),$$

and

$$(\boldsymbol{A}_1, P, u)$$

where $\boldsymbol{A}_1, u$ are uniformly chosen from $\mathcal{R}_q$ and there are M strings $x^{(\beta)} \in \{0,1\}^L$ such that $\tilde{s} = \sum_{i=0}^{L-1} x_i^{(\beta)} \cdot s_i$. The indistinguishability of these distributions immediately implies the indistinguishability of the following:

$$\begin{aligned}
&(\boldsymbol{A}_0, \boldsymbol{T}_0, \boldsymbol{A}_1, P, \tilde{c} \leftarrow \mathsf{Encode}(\boldsymbol{A}_0, \boldsymbol{T}_0, \boldsymbol{A}_1, \tilde{s})) \\
=\ &(\boldsymbol{A}_0, \boldsymbol{T}_0, \boldsymbol{A}_1, P, \tilde{c} \leftarrow \mathsf{GaussSamp}(\boldsymbol{A}_0, \boldsymbol{T}_0, \tilde{s}\boldsymbol{A}_1 + e, \sigma)) \\
\overset{c}{\approx}\ &(\boldsymbol{A}_0, \boldsymbol{T}_0, \boldsymbol{A}_1, P, \tilde{c} \leftarrow \mathsf{GaussSamp}(\boldsymbol{A}_0, \boldsymbol{T}_0, u, \sigma)) \\
\overset{c}{\approx}\ &(\boldsymbol{A}_0, \boldsymbol{T}_0, \boldsymbol{A}_1, P, \tilde{c} \leftarrow \mathsf{REncode}(\boldsymbol{A}_0, \boldsymbol{T}_0, \boldsymbol{A}_1, 1^\lambda))
\end{aligned}$$

and thus the proof is complete. □

[13] By definition $\boldsymbol{R}_{0\to 1} \in \mathcal{R}^{m\times m}$ and $\boldsymbol{A}_0\boldsymbol{R}_{0\to 1} = b_1 = s\boldsymbol{A}_1 + e_1 \in \mathcal{R}_q^m$.