Obfuscating Conjunctions

Zvika Brakerski[1] and Guy N. Rothblum[2]

[1] Stanford University
[2] Microsoft Research

Abstract. We show how to securely obfuscate the class of *conjunction functions* (functions like $f(x_1, \ldots, x_n) = x_1 \wedge \neg x_4 \wedge \neg x_6 \wedge \cdots \wedge x_{n-2}$). Given any function in the class, we produce an obfuscated program which preserves the input-output functionality of the given function, but reveals nothing else.

Our construction is based on multilinear maps, and can be instantiated using the recent candidates proposed by Garg, Gentry and Halevi (EUROCRYPT 2013) and by Coron, Lepoint and Tibouchi (CRYPTO 2013). We show that the construction is secure when the conjunction is drawn from a distribution, under mild assumptions on the distribution. Security follows from multilinear entropic variants of the Diffie-Hellman assumption. We conjecture that our construction is secure for *any* conjunction, regardless of the distribution from which it is drawn. We offer supporting evidence for this conjecture, proving that our obfuscator is secure for any conjunction against *generic adversaries*.

1 Introduction

Code obfuscation is the problem of compiling a computer program so as to make it unintelligible to an adversary, or impossible to reverse-engineer, while preserving its input-output functionality. Obfuscation has been of long-standing interest to both the cryptography and security communities. However, despite the importance of the problem, and its many exciting applications, very few techniques or effective heuristics are known. In particular, the theoretical study of the problem (in the "virtual black-box model" [2]) led to a handful of known constructions, which apply to very limited classes of functions. These include the class of point functions, and extensions such as multi-point functions, "lockers" and constant-dimension hyperplanes.

In this work, we present an obfuscator for a new and different class: *conjunction functions*. These are functions that take n-bit strings as input and only accept if a subset of these bits are set to predefined values. Our construction relies on *(asymmetric) multilinear maps*, and is instantiated using the new candidate construction due to Garg, Gentry and Halevi [14].

Previous Results. The goal of an obfuscator is generating a program that preserves the functionality of the original program, but reveals nothing else. One commonly used formalization of this objective is "virtual black box" obfuscation,

due to Barak et al. [2]. Our work uses this formalization , as well as alternative formalizations from subsequent works (see below).

In their work, [2] also proved the impossibility of general-purpose obfuscators (i.e. ones that work for any functionality) in the virtual black box model. This impossibility result was extended in [15]. While these negative results show serious limitations on the possibility of general-purpose obfuscation, they focus on specific (often cryptographic or contrived) functionalities. Thus, they do not rule out that obfuscation may be possible for many programs of interest.

Positive results on obfuscation focus on specific, simple programs. One program family, which has received extensive attention, is that of "point functions": password checking programs that only accept a single input string, and reject all others. Starting with the work of Canetti [6], several works have shown obfuscators for this family under various assumptions [8, 20, 25], as well as extensions [7, 3]. Canetti, Rothblum and Varia [9] showed how to obfuscate a function that checks membership in a hyperplane of constant dimension (over a large finite field). Other works showed how to obfuscate cryptographic function classes under different definitions and formalizations. These function classes include checking proximity to a hidden point [12], vote mixing [1], and re-encryption [18]. Several works [6, 8, 17, 18] relaxed the security requirement so that obfuscation only holds for a random choice of a program from the family, we will also use this relaxation for one of our results. A different relaxation, known as "best-possible obfuscation", which allows the obfuscation to leak non black-box information was presented in [16].

This Work: Obfuscating Conjunctions. Our main contribution is a new obfuscator for *conjunctions*. A conjunction $C = (W, V)$ is a function on n bit inputs, specified by a set $W \subseteq [n]$ of "wildcard" entries, and a vector $V \in \{0,1\}^n$ of target values for non-wildcard entries. The conjunction accepts an input $\vec{x} \in \{0,1\}^n$ if for all $i \in ([n] \setminus W)$, $\vec{x}[i] = V[i]$, i.e. if for all non-wildcard entries in $\vec{x}$, their values equal those specified in V. We use the convention that if $W[i] = 1$ then $V[i] = 0$ (wildcard entries are ignored, so this does not effect the conjunction's functionality).

The class of conjunctions, while obviously quite limited, has a rich combinatorial and computational expressive power. They are studied in a multitude of settings throughout computer science (e.g. in learning theory [19]). One significant distinction from previous function classes for which obfuscators were known, is that a conjunction may ignore some of its input bits (the wildcard entries). An obfuscator for conjunctions needs to produce a program that hides which bits are ignored, and which ones are influential.

As an example of the applications of a conjunction obfuscator, consider the following setting. There are k passwords, each controlling access to a particular type of resource. Each individual knows some subset of the k passwords, which corresponds to the resources it is allowed to access. A gatekeeper wishes to check whether an individual has access to some combination of resources, i.e. whether the individual knows a particular subset $S \subset [k]$ of the passwords, without revealing to an observer which combination it is checking. A conjunction, which

takes as input k concatenated passwords, can check whether the passwords for resources in S are correct, while ignoring passwords for resources not in S. An obfuscation of this conjunction can be made public, and used to check whether an individual has access to that combination of resources, without revealing which resources are being checked (nor, of course, what any of the passwords are).

1.1 Our Construction and its Security

The main tool in our construction is multilinear maps. In particular, we utilize a recent candidate for graded encoding (a generalization of multilinear maps) due to [14].[3] We prove the security of our obfuscator when the conjunction is chosen from a distribution with sufficient entropy: namely, when sampling $C = (W, V)$ from the distribution, even given the wildcard locations W, there is sufficient (superlogarithmic) entropy in V. We stress that this *does not* imply that the attacker is allowed to learn W; on the contrary, we prove that if C is drawn from a distribution with the aforementioned property, the adversary cannot learn anything, wildcard locations included.[4] As noted above, here we follow several works [6, 8, 17, 18] which relax the security requirement to hold only when the circuit to be obfuscated is drawn from a distribution from a certain class (usually one with sufficient entropy).

We prove the above under two security assumptions on graded encodings schemes: The first is a translation of the SXDH assumption on bilinear groups to the setting of graded encoding schemes.[5] The second assumption is reminiscent of "Canetti's Assumption" [6] on Diffie-Hellman groups, which was introduced for the purpose of obfuscating point functions.

We conjecture that the construction is secure for *every* conjunction, but we were unable to produce a proof based on a well-established assumption (naturally, one can always take the security of the obfuscator as an assumption). As supportive evidence for the conjectured security, we prove that the obfuscator is secure against *generic adversaries*: Ones that only use the group structure and not the representation of the group elements. This is similar to the generic group model of [24, 21]. The proof of security against generic adversaries is non-trivial, and we view this as one of our main technical contributions. We note that previous works on obfuscation [20, 9] have also used the random oracle and generic group models to provide evidence for the security of constructions.

[3] We use the *asymmetric* variant of the encoding scheme, where there are several distinct "source groups".

[4] We remark that in this case nothing at all can be learned from black-box access to the function since it is infeasible to find an accepting input. We also remark that, for example, the conjunctions used for the k-resource application above naturally satisfy this condition, because of the entropy in each password.

[5] This assumption is actually known to be *false* for the construction and formulation of [14]. However, we show a more careful definition of the scheme and the assumption for which no attack is known. Also, no attack is known for the recent construction of Coron, Lepoint and Tibouchi [10].

We proceed with an overview of our construction and results. As we explained, the obfuscator uses the recent construction of multi-linear maps via graded encoding schemes [14]. We begin with a high-level overview on the properties of multilinear maps that will be used. We then proceed with an overview of our construction, and state our two main results.

Multilinear Maps and Graded Encoding Schemes: Background. We begin by recalling the notion of multilinear maps, due to Boneh and Silverberg [5]. Rothblum [23] considered the asymmetric case, where the groups may be different (this is crucial for our construction).

Definition 1.1 (Asymmetric Multilinear Map [5, 23]).
For $\tau + 1$ cyclic groups $G_1, \ldots, G_\tau$, G_T of the same order p, a τ-multilinear map $e : G_1 \times \ldots \times G_\tau \to G_T$ has the following properties:

1. *For elements $\{g_i \in G_i\}_{i=1,\ldots,\tau}$, index $i \in [\tau]$ and integer $\alpha \in \mathbb{Z}_p$, it holds that:*
$$e(g_1, \ldots, \alpha \cdot g_i, \ldots, g_\tau) = \alpha \cdot g(g_1, \ldots, g_\tau)$$
2. *The map e is non-degenerate: when its inputs are all generators of their respective groups $\{G_i\}$, then its output is a generator of the target group G_T.*

Recently, [14] suggested a candidate for graded encoding, a generalization of (symmetric or asymmetric) multilinear maps. See Section 2.2 for a more complete overview of these objects. For this introduction, we treat them as a generalization of asymmetric multilinear maps in the following way. For a τ-multilinear map e, for the group G_i of prime order p, we consider the ring $\mathbb{Z}_p$. For an element $\sigma \in \mathbb{Z}_p$, we can think of g_i^σ as an "encoding" of σ in G_i. We denote this by $\mathsf{enc}_i(\sigma)$. We note that this encoding is easy to compute, but (presumably) hard to invert. The multilinear map e lets us take τ encodings $\{\mathsf{enc}_i(\sigma_i)\}_{i\in[\tau],\sigma_i\in\mathbb{Z}_p}$, and compute the target group encoding $\mathsf{enc}_T(\prod_i \sigma_i)$. Graded encoding schemes afford a similar functionality, albeit with randomized and noisy encodings, and with a procedure for testing equality of encoded elements in the target group.

Our Construction. For a conjunction $C = (W, V)$ on n-bits inputs, the obfuscator uses the graded encoding scheme to obtain the above generalization to an $(n + 1)$-multilinear map. For each input entry $i \in [n]$, the obfuscator picks ring elements $(\rho_{i,0}, \rho_{i,1}, \alpha_{i,0}, \alpha_{i,1})$ distributed as follows: if $i \notin W$, namely the entry isn't a wildcard, then the ring elements are independent and uniformly random. If $i \in W$, namely the entry is a wildcard, then the ring elements are uniformly random under the constraint that $\alpha_{i,0} = \alpha_{i,1}$. After picking the ring elements, the obfuscator outputs two pairs of encodings for each $i \in [n]$:

$$\{(w_{i,b} = \mathsf{enc}_i(\rho_{i,b}), u_{i,b} = \mathsf{enc}_i(\rho_{i,b} \cdot \alpha_{i,b}))\}_{i\in[n],b\in\{0,1\}}$$

Note that if $i \in W$, then the ratio between the ring elements encoded in $u_{i,0}$ and $w_{i,0}$, is equal to the ratio between the ring elements encoded in $u_{i,1}$ and $w_{i,1}$

(these ratios are, respectively, $\alpha_{i,0}$ and $\alpha_{i,1}$, which are equal when $i \in W$). We remark that this part of the obfuscation depends only on the wildcards W, but not on the values V.

To complete the obfuscation, the obfuscator picks independent and uniformly random ring element ρ_{n+1}, and outputs a pair of encodings:

$$(w_{n+1} = \mathsf{enc}_{n+1}(\rho_{n+1}), u_{n+1} = \mathsf{enc}_{n+1}(\rho_{n+1} \cdot \underbrace{\prod_{i\in[n]} \alpha_{i,V[i]}}_{=\alpha_{n+1}}))$$

To evaluate the obfuscated program on an input $\vec{x} \in \{0,1\}^n$, we test equality between two multilinear products:[6]

$$e(\ldots, u_{i,\vec{x}[i]}, \ldots, w_{n+1}) \stackrel{?}{=} e(\ldots, w_{i,\vec{x}[i]}, \ldots, u_{n+1}) \tag{1}$$

The full construction is in Section 3.

Correctness. Examining the two multilinear products in Eq. (1), the element encoded in the left-hand side is $(\prod_{i\in[n]} \rho_{i,\vec{x}[i]} \cdot \alpha_{i,\vec{x}[i]}) \cdot \rho_{n+1}$. The element encoded in the right-hand side is $(\prod_{i\in[n]} \rho_{i,\vec{x}[i]} \cdot \alpha_{i,V[i]}) \cdot \rho_{n+1}$. Thus, Eq. (1) holds if and only if:

$$\prod_{i\in[n]} \alpha_{i,\vec{x}[i]} = \prod_{i\in[n]} \alpha_{i,V[i]} \tag{2}$$

For $i \in W$ we have $\alpha_{i,0} = \alpha_{i,1}$, the contributions from the i-th group to both products in Eq. (2) are identical. For $i \notin W$, the contribution from the i-th group in the left-hand side of Eq. (2) is $\alpha_{i,\vec{x}[i]}$. In the right-hand side, the contribution is $\alpha_{i,V[i]}$. Except for a negligible probability of error, Eq. (2) holds if and only if all these contributions are identical, i.e. if and only if $\forall i \notin W : \vec{x}[i] = V[i]$.

Security. Security is not as straightforward. A slightly misleading intuition for security, is that if a DDH-like assumption holds within each group G_i separately, then no observer can distinguish from that group's encodings whether $\alpha_{i,0} = \alpha_{i,1}$. This is true for each group in isolation, but it is insufficient because the obfuscation also includes encodings, in group G_{n+1}, of items that are correlated with the items encoded in group i. The multilinear map e might allow an adversary to distinguish whether the i-th entry is a wildcard.

For example, if in C all the entries are wildcards, the adversary can pick a random input, run the obfuscation, see that it accepts, and then by flipping the input bits one-by-one it can determine that all of the entries are wildcards. This attack clearly demonstrates that (for some conjunctions) an adversary can determine which entries are wildcards and which aren't. Note, however, that (for the specific example of a conjunction that is all-wildcards) this could also be accomplished using black-box access to the conjunction.

[6] For the candidate of [14], the encodings are randomized, but there is a procedure for testing equality between encoded elements in the target group.

Indeed, we prove the security of the obfuscator when the conjunction is drawn from a distribution, under mild assumptions on the distribution's entropy. We conjecture that the obfuscator is actually secure for *any* conjunction, and as supporting evidence we show that it is secure against generic adversaries. An overview on both results follows.

Security for High Entropy. We prove the security of our scheme in the case where $C = (W, V)$ is drawn from a distribution where the entropy of V given W is superlogarithmic. We do so by resurrecting the flawed argument described above: We use the entropy to remove the dependence between the elements in G_{n+1} and those in the other groups, and then apply DDH in each group.

We start by noting that this dependence is due to the relation

$$\alpha_{n+1} = \prod_{i \in [n]} \alpha_{i,V[i]} \ , \tag{3}$$

and if we could replace α_{n+1} with a completely uniform variable, independent of the other α's, we'd be done. To this end, we notice that Eq. (3) describes an (almost) pairwise independent hash function, whose seed are the values $\alpha_{i,b}$ and whose input is V. We show that such a hash function is a good entropy condenser, so that almost all of the entropy in V is preserved in α_{n+1}. (It is important to notice that the distinguisher has side information which depends on W, and therefore we must require that the *conditional entropy* is high.)

Once we establish that α_{n+1} has superlogarithmic entropy, we use a "Canetti-like Assumption" [6]: we assume a high-entropy element in the exponent of a random group generator is indistinguishable from uniform.[7] We thus isolate α_{n+1} from the dependence on the other α's, which allows us to apply DDH in groups $G_1, \ldots, G_n$, and obtain the final result: that the obfuscated program comes from a distribution that can be efficiently simulated. The security proof is in Section 4.

Security in The Generic Model. We prove security against *generic adversaries*. A generic adversary is one that succeeds regardless of the representation of the encoding scheme. This is modeled by allowing it to only manipulate encodings in the graded encoding scheme via oracle access to an oracle for the operations that are available using the *evparams* parameters. We show that for any generic adversary $\mathcal{A}$, which takes as input an obfuscation and outputs a single bit, there exists a generic simulator $\mathcal{S}$ s.t. for any conjunction C, the adversary's output on an obfuscation of C is statistically close to the simulator's output given only black-box access to C. The distribution of the adversary is taken over the choice of a *random* graded encoding scheme oracle: an oracle that represents each encoding in each group using a (long enough) uniformly random string.

In this model, since each element's encoding is uniformly random, and the obfuscation contains the encodings of distinct ring elements, the obfuscation of any conjunction is simply a collection of uniformly random strings. Thus

[7] Wee [25] showed that these types of assumptions (hardness given only superlogarithmic entropy) are essential even for obfuscating point functions.

simulating the obfuscator's output is easy. The main challenge is that *the outputs to oracle calls* on the string in the obfuscation are highly dependant on the conjunction C. It is thus not clear how the simulator can simulate the oracle's outputs. For example, each accepting input $\vec{x}$ for C specifies two possible inputs to the oracle implementing the multilinear map, which should both yield the same encoding in the target group. Indeed, simulating the oracle call outputs proves challenging. Moreover, the more generalized notion of graded encoding schemes permits more general generic operations.

The simulator $\mathcal{S}$ operates as follows. It feeds the adversary $\mathcal{A}$ with a "dummy obfuscation" containing uniformly random strings. It then follows $\mathcal{A}$'s calls to the graded encoding scheme (GES) oracle, and tries to simulate the output. For each call made by $\mathcal{A}$, we show how $\mathcal{S}$ can (efficiently) identify a polynomial size set X of inputs, such that if $\forall \vec{x} \in X, C(\vec{x}) = 0$, then the oracle's output is essentially independent of C and can be simulated. On the other hand, if there exists $\vec{x} \in X$ s.t. $C(\vec{x}) = 1$, then the simulator can use its black-box access to C to identify this input,. Once an accepting input is identified, the simulator can further use its block-box access to C to retrieve the conjunction's explicit description (W, V) (see Claim 3 below). Once the simulator knows (W, V) it can (perfectly) simulate the adversary's behavior. We view this proof of security for generic adversaries as one of our main technical contributions.

The full specification and treatment of the generic GES model, as well as the proof of security for generic adversaries, are deferred to the full version due to space constraints.

2 Preliminaries

Notation. We use $\Delta(\cdot, \cdot)$ to indicate total variation distance (statistical distance). We use $\vec{1}$ (respectively $\vec{0}$) to denote the all-1 (all-0) vector (the dimension will be clear from the context).

2.1 Min-Entropy and Extraction

The following are information theoretic tools that will be required in our proof. The main notion of entropy used in this work is that of average min-entropy from [11], as well as its smooth version (see Definitions 2.1 and 2.2 below). We then show that applying a pairwise independent hash function with a large enough image on an average min-entropy source, roughly preserves the average min-entropy (that is, it is an entropy condenser). This is derived from the generalized "crooked" leftover hash lemma [13, 4].

We start by defining average min-entropy.

Definition 2.1 (average min-entropy [11]). *Let X, Z be (possibly dependent) random variables, the average min entropy of X conditioned on Z is:*

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

It follows from the definition that for every deterministic function f (that may depend on Z):

$$\widetilde{\mathbf{H}}_\infty(f(X)|Z) \leq \widetilde{\mathbf{H}}_\infty(X|Z) \ . \tag{4}$$

We also use a smooth variant introduced in [11, Appendix A] following [22].

Definition 2.2 (smooth average min-entropy [11]). *Let X, Z be as above and let $\epsilon > 0$, then*

$$\widetilde{\mathbf{H}}^\epsilon_\infty(X|Z) = \max_{(X',Z'):\Delta((X,Z),(X',Z'))\leq\epsilon} \widetilde{\mathbf{H}}_\infty(X'|Z') \ .$$

We will next show that pairwise independent functions condense average min-entropy in the following way.

Lemma 2.3. *Let X, Z be random variables, let $\mathcal{H}$ be a pairwise independent hash family with output length $\geq \left\lfloor \widetilde{\mathbf{H}}_\infty(X|Z) - 2\log(1/\epsilon) + 2 \right\rfloor$ (represented as binary string), for some $\epsilon > 0$. Then letting $h \leftarrow \mathcal{H}$ be a properly sampled function from this family, it holds that*

$$\widetilde{\mathbf{H}}^\epsilon_\infty(h(X)|Z,h) \geq \widetilde{\mathbf{H}}_\infty(X|Z) - 2\log(1/\epsilon) + 1 \ .$$

Proof. Let $X, Z, \mathcal{H}, h, \epsilon$ be as in the lemma statement. Our goal is to show that there exists a random variable Y such that

$$\Delta((h(X), Z, h), (Y, Z, h)) \leq \epsilon \ ,$$

and

$$\widetilde{\mathbf{H}}_\infty(Y|Z,h) \geq \widetilde{\mathbf{H}}_\infty(X|Z) - 2\log(1/\epsilon) + 1 \ .$$

Let f be the function that outputs the first k bits of its input, for

$$k = \left\lfloor \widetilde{\mathbf{H}}_\infty(X|Z) - 2\log(1/\epsilon) + 2 \right\rfloor \geq \widetilde{\mathbf{H}}_\infty(X|Z) - 2\log(1/\epsilon) + 1 \ ,$$

and note that $f(U)$ is uniform over $\{0,1\}^k$ (in fact, we can use any function that has this property).

We recall that the generalized "crooked" leftover hash lemma [4, Lemma 7.1] implies that

$$\Delta((f(h(X)), Z, h), (f(U), Z, h)) \leq \epsilon \ .$$

Now, consider a 2-step process for sampling the joint distribution $(h(X), Z, h)$: first, sample $(f(h(X)), Z, h)$ from the appropriate marginal distribution; and then sample $h(X)$ conditioned on the previously sampled values.

We define Y using using the following process: First, sample a tuple according to the distribution $(f(U), Z, h)$, and then apply the second stage of the sampling process from above. The result will be the distribution (Y, Z, h). Clearly,

$$\Delta((h(X), Z, h), (Y, Z, h)) = \Delta((f(h(X)), Z, h), (\underbrace{f(Y)}_{=f(U)}, Z, h)) \leq \epsilon \ ,$$

where the first equality is since there is a deterministic mapping (f) from the left hand side to the right hand side, and a randomized mapping (the second step sampler) from the right hand side to the left hand side.

To conclude, we notice that

$$\widetilde{\mathbf{H}}_\infty(Y|Z,h) \geq \widetilde{\mathbf{H}}_\infty(f(Y)|Z,h) = \underbrace{\widetilde{\mathbf{H}}_\infty(\{0,1\}^k|Z,h)}_{=k} \geq \widetilde{\mathbf{H}}_\infty(X|Z) - 2\log(1/\epsilon) + 1 \,.$$

2.2 Graded Encoding Schemes and Assumptions

We begin with the definition of a graded encoding scheme, due to Garg, Gentry and Halevi [14]. While their construction is very general, for our purposes a more restricted setting is sufficient as defined below.

Definition 2.4 (τ-Graded Encoding Scheme [14]). *A τ-encoding scheme for a ring R is a collection of sets $\mathcal{S} = \{S_{\mathbf{v}}^{(\alpha)} \subset \{0,1\}^* : \mathbf{v} \in \{0,1\}^\tau, \alpha \in R\}$, with the following properties:*

1. *For every index $\mathbf{v} \in \{0,1\}^\tau$, the sets $\{S_{\mathbf{v}}^{(\alpha)} : \alpha \in R\}$ are disjoint, and so they are a partition of the indexed set $S_{\mathbf{v}} = \bigcup_{\alpha \in R} S_{\mathbf{v}}^{(\alpha)}$.*
2. *There are binary operations "$+$" and "$-$" such that for all $\mathbf{v} \in \{0,1\}^\tau$, $\alpha_1, \alpha_2 \in R$ and for all $u_1 \in S_{\mathbf{v}}^{(\alpha_1)}$, $u_2 \in S_{\mathbf{v}}^{(\alpha_2)}$:*

$$u_1 + u_2 \in S_{\mathbf{v}}^{(\alpha_1+\alpha_2)} \quad \text{and} \quad u_1 - u_2 \in S_{\mathbf{v}}^{(\alpha_1-\alpha_2)} \,,$$

   where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in R.
3. *There is an associative binary operation "$\times$" such that for all $\mathbf{v}_1, \mathbf{v}_2 \in \{0,1\}^\tau$ such that $\mathbf{v}_1 + \mathbf{v}_2 \in \{0,1\}^\tau$, for all $\alpha_1, \alpha_2 \in R$ and for all $u_1 \in S_{\mathbf{v}_1}^{(\alpha_1)}$, $u_2 \in S_{\mathbf{v}_2}^{(\alpha_2)}$, it holds that*

$$u_1 \times u_2 \in S_{\mathbf{v}_1+\mathbf{v}_2}^{(\alpha_1 \cdot \alpha_2)},$$

   where $\alpha_1 \cdot \alpha_2$ is multiplication in R.

In this work, the ring R will always be $\mathbb{Z}_p$ for a prime p.

To the reader who is familiar with the [14] work, we note that the above is the special case of the [14] construction in which we consider only binary index vectors (in the [14] notation, this corresponds to setting $\kappa = 1$), and we construct our encoding schemes to be *asymmetric* (as will become apparent below when we define our zero-text index $\mathbf{vzt} = \vec{1}$).

Definition 2.5 (Efficient Procedures for a τ-Graded Encoding Scheme [14]). *We consider τ-graded encoding schemes (see above) where the following procedures are efficiently computable.*

- *Instance Generation:* $\mathsf{InstGen}(1^\lambda, 1^\tau)$ *outputs the set of parameters* $params$*, a description of a* τ*-Graded Encoding Scheme. (Recall that we only consider Graded Encoding Schemes over the set indices* $\{0,1\}^\tau$*, with zero testing in the set* $S_{\vec{1}}$*). In addition, the procedure outputs a subset* $evparams \subset params$ *that is sufficient for computing addition, multiplication and zero testing*[8] *(but possibly insufficient for encoding or for randomization).*
- *Ring Sampler:* $\mathsf{samp}(params)$ *outputs a "level zero encoding"* $a \in S_0^{(\alpha)}$ *for a nearly uniform* $\alpha \in_R R$*.*
- *Encode and Re-Randomize:*[9] $\mathsf{encRand}(params, i, a)$ *takes as input an index* $i \in [\tau]$ *and* $a \in S_0^{(\alpha)}$*, and outputs an encoding* $u \in S_{\mathbf{e}_i}^{(\alpha)}$*, where the distribution of* u *is (statistically close to being) only dependent on* α *and not otherwise dependent of* a*.*
- *Addition and Negation:* $\mathsf{add}(evparams, u_1, u_2)$ *takes* $u_1 \in S_{\mathbf{v}}^{(\alpha_1)}, u_2 \in S_{\mathbf{v}}^{(\alpha_2)}$*, and outputs* $w \in S_{\mathbf{v}}^{(\alpha_1+\alpha_2)}$*. (If the two operands are not in the same indexed set, then* add *returns* $\perp$*). We often use the notation* $u_1 + u_2$ *to denote this operation when evparams is clear from the context. Similarly,* $\mathsf{negate}(evparams, u_1) \in S_{\mathbf{v}}^{(-\alpha_1)}$*.*
- *Multiplication:* $\mathsf{mult}(evparams, u_1, u_2)$ *takes* $u_1 \in S_{\mathbf{v}_1}^{(\alpha_1)}, u_2 \in S_{\mathbf{v}_2}^{(\alpha_2)}$*. If* $\mathbf{v}_1 + \mathbf{v}_2 \in \{0,1\}^\tau$ *(i.e. every coordinate in* $\mathbf{v}_1 + \mathbf{v}_2$ *is at most 1), then* mult *outputs* $w \in S_{\mathbf{v}_1+\mathbf{v}_2}^{(\alpha_1 \cdot \alpha_2)}$*. Otherwise,* mult *outputs* $\perp$*. We often use the notation* $u_1 \times u_2$ *to denote this operation when evparams is clear from the context.*
- *Zero Test:* $\mathsf{isZero}(evparams, u)$ *outputs 1 if* $u \in S_{\vec{1}}^{(0)}$*, and 0 otherwise.*

In the [14, 10] constructions, encodings are noisy *and the noise level increases with addition and multiplication operations, so one has to be careful not to go over a specified noise bound. However, the parameters can be set so as to support* $O(\tau)$ *operations, which are sufficient for our purposes. We therefore ignore noise management throughout this manuscript. An additional subtle issue is that with negligible probability the initial noise may be too big. However this can be avoided by adding rejection sampling to* samp *and therefore ignored throughout the manuscript as well.*

It is important to notice that our definition deviates from that of [14] as we define two sets of parameters *params* and *evparams*. While the former will be used by the obfuscator in our construction (and therefore will not be revealed to an external adversary), the latter will be used when evaluating an obfuscated program (and thus will be known to an adversary). When instantiating our definition, the guideline is to make *evparams* minimal so as to give the least amount of information to the adversary. In particular, in the known candidates [14, 10], *evparams* only needs to contain the zero-test parameter **pzt** (as well as the global modulus).

[8] The "zero testing" parameter **pzt** defined in [14] is a part of *evparams*.

[9] This functionality is not explicitly provided by [14], however it can be obtained by combining their encoding and re-randomization procedures.

Hardness Assumptions In this work, we will use two hardness assumptions over graded encoding schemes. The first, which we call "graded external DDH" (or GXDH, Assumption 2.6 below) is an analog of the symmetric external DH assumption (SXDH), instantiated for the multilinear case. The second assumption (GCAN Assumption 2.7) is an analog of Canetti's assumption [6], that taking a random generator to a high-entropy power results in a random-looking element. We note that we make these assumptions against non-uniform adversaries.

Assumption 2.6 (Graded External DH). *Letting* $(params, evparams) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\tau)$*, for all* $i = 1, \ldots, \tau$*, sample* $r_{i,0}, r_{i,1}, a_{i,0}, a_{i,1} \leftarrow \mathsf{samp}(params)$ *and consider the following values:*

$$\begin{aligned} w_{i,0} &\leftarrow \mathsf{encRand}(params, i, r_{i,0}) & w_{i,1} &\leftarrow \mathsf{encRand}(params, i, r_{i,1}) \\ u_{i,0} &\leftarrow \mathsf{encRand}(params, i, r_{i,0} \times a_{i,0}) & u_{i,1} &\leftarrow \mathsf{encRand}(params, i, r_{i,1} \times a_{i,1}) \\ & & u'_{i,1} &\leftarrow \mathsf{encRand}(params, i, r_{i,1} \times a_{i,0}) \end{aligned}$$

The GXDH assumption is that for every choice of $\tau \in \mathbb{N}$ *and* $i^* \in [\tau]$*, no ensemble of polynomial time adversaries can have have non-negligible advantage in distinguishing the distributions:*

$$\big(evparams, \{(w_{i,0}, u_{i,0}, w_{i,1}, u_{i,1}, u'_{i,1})\}_{i \neq i^*}, (w_{i^*,0}, u_{i^*,0}, w_{i^*,1}, u_{i^*,1})\big)$$

and

$$\big(evparams, \{(w_{i,0}, u_{i,0}, w_{i,1}, u_{i,1}, u'_{i,1})\}_{i \neq i^*}, (w_{i^*,0}, u_{i^*,0}, w_{i^*,1}, u'_{i^*,1})\big)$$

We note that a stronger version of this assumption, where the distinguisher is given access to *params* rather than *evparams*, was presented in the the early versions of [14]. It was later shown that this stronger assumption is false, see later versions of [14] for the attack. We emphasize that *no attacks are known if only evparams is given as above*. Furthermore, the new candidate of [10] is not known to be sensitive to such attacks even if *params* is given.

Since we only provide our distinguisher with *evparams*, it may not be able to generate DDH tuples by itself. We therefore provide it with correctly labeled DDH samples for all groups except i^*. This is the minimal assumption that is required for our construction, however we conjecture that a stronger variant where the adversary is allowed to receive an unbounded number of labeled samples at any group (including i^*) is also true.

For our next assumption, we introduce the following notation. Consider a distribution D over $S_{\mathbf{v}} = \cup_{\alpha \in R} S_{\mathbf{v}}^{(\alpha)}$. The distribution $\mathsf{enc}^{-1}(D)$ is defined by the following process: Sample $x \leftarrow D$, let α be such that $x \in S_{\mathbf{v}}^{(\alpha)}$, output α. We also recall the definition of smooth average min-entropy (see Definition 2.2 above).

Assumption 2.7 ("Graded Canetti"). *Let* $(params, \mathbf{pzt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\tau)$ *and let* $\{(D_\lambda, Z_\lambda)\}_{\lambda \in \mathbb{N}}$ *be a distribution ensemble over* $S_0 \times \{0,1\}^*$*, such that*

$$\widetilde{\mathbf{H}}_\infty^\epsilon(\mathsf{enc}^{-1}(D_\lambda) | Z_\lambda) \geq h(\lambda) \ ,$$

for some $\epsilon = negl(\lambda)$ and function $h(\lambda) = \omega(\log \lambda)$.

The GCAN assumption is that no ensemble of polynomial time adversaries and indices i can have non-negligible advantage in distinguishing the distributions

$$(params, evparams, w, u, z) \text{ and } (params, evparams, w, u', z) \ ,$$

where we let: $(params, evparams) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\tau)$, $r \leftarrow \mathsf{samp}(params)$, $w \leftarrow \mathsf{encRand}(params, i, r)$, $(x, z) \leftarrow (D_\lambda, Z_\lambda)$, $u \leftarrow \mathsf{encRand}(params, i, r \times x)$, $u' \leftarrow \mathsf{encRand}(params, i, \mathsf{samp}(params))$. *(In this definition, the distinguisher is given both params and evparams.)*

This assumption is consistent with out knowledge on candidate graded encoding schemes. However, if we want to make an even weaker assumption, we can set the minimal entropy requirement to be higher than just $\omega(\log \lambda)$. The constructions in this paper can trivially be adapted to such weaker variants (with the expected degradation in security).

2.3 Obfuscation

Definition 2.8 (Virtual Black-Box Obfuscator [2]).

Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of polynomial-size circuits, where $\mathcal{C}_n$ is a set of boolean circuits operating on inputs of length n. And let $\mathcal{O}$ be a PPTM algorithm, which takes as input an input length $n \in \mathbb{N}$, a circuit $C \in \mathcal{C}_n$, a security parameter $\lambda \in \mathbb{N}$, and outputs a boolean circuit $\mathcal{O}(C)$ (not necessarily in $\mathcal{C}$).

$\mathcal{O}$ is an obfuscator *for the circuit family $\mathcal{C}$ if it satisfies:*

1. Preserving Functionality: *For every $n \in \mathbb{N}$, and every $C \in \mathcal{C}_n$, and every $\vec{x} \in \{0,1\}^n$, with all but $negl(\lambda)$ probability over the coins of $\mathcal{O}$:*
$$(\mathcal{O}(C, 1^n, \lambda))(\vec{x}) = C(\vec{x})$$
2. Polynomial Slowdown: *For every $n, \lambda \in \mathbb{N}$ and $C \in \mathcal{C}$, the circuit $\mathcal{O}(C, 1^n, 1^\lambda)$ is of size at most $poly(|C|, n, \lambda)$.*
3. Virtual Black-Box: *For every (non-uniform) polynomial size adversary $\mathcal{A}$, there exists a (non-uniform) polynomial size simulator $\mathcal{S}$, such that for every $n \in \mathbb{N}$ and* for every *$C \in \mathcal{C}_n$:*
$$\left| \Pr_{\mathcal{O},\mathcal{A}}[\mathcal{A}(\mathcal{O}(C, 1^n, 1^\lambda)) = 1] - \Pr_{\mathcal{S}}[\mathcal{S}^C(1^{|C|}, 1^n, 1^\lambda) = 1] \right| = negl(\lambda)$$

Remark 2.9. A stronger notion of functionality, which also appears in the literature, requires that with overwhelming probability the obfuscated circuit is correct on *every input simultaneously*. We use the relaxed requirement that for every input (individually) the obfuscated circuit is correct with overwhelming probability (in both cases the probability is only over the obfuscator's coins). We note that our construction can be modified to achieve the stronger functionality property (by using a ring of sufficiently large size and the union bound).

Definition 2.10 (*Average-Case* Secure Virtual Black-Box).

Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of circuits and $\mathcal{O}$ a PPTM as in Definition 2.8. Let $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ be an ensemble of distribution families $\mathcal{D}_n$, where each $D \in \mathcal{D}_n$ is a distribution over $\mathcal{C}_n$.

$\mathcal{O}$ is an obfuscator *for the distribution class $\mathcal{D}$ over the circuit family $\mathcal{C}$, if it satisfies the functionality and polynomial slowdown properties of Definition 2.8 with respect to $\mathcal{C}$, but the virtual black-box property is replaced with:*

3. *Distributional* Virtual Black-Box: *For every (non-uniform) polynomial size adversary $\mathcal{A}$, there exists a (non-uniform) polynomial size simulator $\mathcal{S}$, such that for every $n \in \mathbb{N}$, every* distribution *$D \in \mathcal{D}_n$ (a distribution over $\mathcal{C}_n$), and every predicate $P : C_n \to \{0,1\}$:*

$$\left| \Pr_{\substack{C\sim D_n,\\ \mathcal{O},\mathcal{A}}}[\mathcal{A}(\mathcal{O}(C,1^n,1^\lambda)) = P(C)] - \Pr_{\substack{C\sim D_n,\\ \mathcal{S}}}[\mathcal{S}^C(1^{|C|},1^n,1^\lambda) = P(C)] \right| = negl(\lambda)$$

Remark 2.11. Our proof of average-case security for the conjunction obfuscator (Theorem 4.2) is in fact stronger. We show a simulator $\mathcal{S}$ that does not even require black-box access to the circuit C. Rather, for a circuit C drawn from a distribution in $\mathcal{D}$, the probability of predicting $P(C)$ from an obfuscation of C, is the same as the probability of predicting $P(C)$ from a "dummy obfuscation" that is independent of C. See the proof for further details.

3 Obfuscating Conjunctions

In this section we present our obfuscator for conjunctions ConjObf (Figure 1). We provide a proof of security for functions that are not determined by the locations of the wildcards in Section 4. In the full version, we provide evidence of the security of our construction for *any* conjunction, by proving that it is secure against generic adversaries that do not use the representation of the specific graded encoding scheme.

We start by defining the class of conjunctions, and a useful property thereof.

Definition 3.1 (n-bit Conjunction).

For an input length n, a conjunction $C = (W, V) : \{0,1\}^n \to \{0,1\}$ is a predicate on n-bit inputs, which is defined by two vectors $W, V \in \{0,1\}^n$. For every input $\vec{x} \in \{0,1\}^n$, $C(\vec{x}) = 1$ iff for all $i \in [n]$, $W[i] = 1$ or $V[i] = \vec{x}[i]$. For the sake of unity of representation, we require that whenever $W[i] = 1$, it holds that $V[i] = 0$.

We often alternate between treating W as an index vector and treating it as a subset of $[n]$. If $i \in W$ then we say that i is a wildcard location*.*

Definition 3.2 (Conjunction Ensemble).

A conjunction ensemble $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$ is a collection of conjunctions $C_n : \{0,1\}^n \to \{0,1\}$, one for each input length.

Claim. There exists an efficient algorithm $\mathcal{B}$, such that for any conjunction $C = (W, V)$, and any accepting input $\vec{x}$ of C, $\mathcal{B}$ can recover (W, V):

$$\forall C = (W, V), \forall \vec{x} : C(\vec{x}) = 1, \quad \mathcal{B}^C(\vec{x}) = (W, V)$$

Proof. Take $n = |\vec{x}|$, the algorithm $\mathcal{B}$ enumerates over the bits of $\vec{x}$. For each bit i, it flips the i-th bit of $\vec{x}$: $\vec{x}^{(i)} = \vec{x} \oplus \mathbf{e}_i$, and checks whether $C(\vec{x}^{(i)}) = 1$. If so, then i must be a wildcard: $W[i] = 1$ and $V[i] = 0$. Otherwise, i is not a wildcard: $W[i] = 0$ and $V[i] = \vec{x}[i]$.

Obfuscator $\mathsf{ConjObf}$**, on input** $(1^\lambda, 1^n, C = (W, V))$

1. generate $(params, evparams) \leftarrow \mathsf{InstGen}(1^\lambda, 1^{n+1})$
2. for $i \in [n]$:
   if $i \in W$, then: $a_{i,0} = a_{i,1} \leftarrow \mathsf{samp}(params) \in S_0^{(\alpha_{i,0})} = S_0^{(\alpha_{i,1})}$
   if $i \notin W$, then: $a_{i,0} \leftarrow \mathsf{samp}(params) \in S_0^{(\alpha_{i,0})}$, $a_{i,1} \leftarrow \mathsf{samp}(params) \in S_0^{(\alpha_{i,1})}$
3. for $i \in [n]$:
   $r_{i,0} \leftarrow \mathsf{samp}(params) \in S_0^{(\rho_{i,0})}$, $r_{i,1} \leftarrow \mathsf{samp}(params) \in S_0^{(\rho_{i,0})}$
   $s_{i,0} \leftarrow r_{i,0} \times a_{i,0} \in S_0^{(\rho_{i,0} \cdot \alpha_{i,0})}$, $s_{i,1} \leftarrow r_{i,1} \times a_{i,1} \in S_0^{(\rho_{i,1} \cdot \alpha_{i,1})}$
   $w_{i,0} \leftarrow \mathsf{encRand}(params, i, r_{i,0}) \in S_{\mathbf{e}_i}^{(\rho_{i,0})}$
   $w_{i,1} \leftarrow \mathsf{encRand}(params, i, r_{i,1}) \in S_{\mathbf{e}_i}^{(\rho_{i,1})}$
   $u_{i,0} \leftarrow \mathsf{encRand}(params, i, s_{i,0}) \in S_{\mathbf{e}_i}^{(\rho_{i,0} \cdot \alpha_{i,0})}$
   $u_{i,1} \leftarrow \mathsf{encRand}(params, i, s_{i,1}) \in S_{\mathbf{e}_i}^{(\rho_{i,1} \cdot \alpha_{i,1})}$
4. $a_{n+1} \leftarrow \left(\Pi_{i \in [n]} a_{i,V[i]}\right) \in S_0^{(\Pi_{i \in [n]} \alpha_{i,V[i]})}$
   $r_{n+1} \leftarrow \mathsf{samp}(params) \in S_0^{(\rho_{n+1})}$
   $s_{n+1} \leftarrow r_{n+1} \times a_{n+1} \in S_0^{(\rho_{n+1} \cdot \Pi_{i \in [n]} \alpha_{i,V[i]})}$
   $w_{n+1} \leftarrow \mathsf{encRand}(params, n+1, r_{n+1}) \in S_{\mathbf{e}_{n+1}}^{(\rho_{n+1})}$
   $u_{n+1} \leftarrow \mathsf{encRand}(params, n+1, s_{n+1}) \in S_{\mathbf{e}_{n+1}}^{(\rho_{n+1} \cdot \Pi_{i \in [n]} \alpha_{i,V[i]})}$
5. output the obfuscation:

$$(evparams, \{(w_{i,0}, u_{i,0}), (w_{i,1}, u_{i,1})\}_{i \in [n]}, (w_{n+1}, u_{n+1}))$$

Evaluation, on input $\vec{x} \in \{0,1\}^n$

1. $\mathbf{t} \leftarrow \left(w_{n+1} \times \Pi_{i \in [n]} u_{i,\vec{x}[i]}\right) \in S_{(1,1,\ldots,1)}^{\left(\rho_{n+1} \cdot (\Pi_{i \in [n]} \rho_{i,\vec{x}[i]} \cdot \alpha_{i,\vec{x}[i]})\right)}$
2. $\mathbf{t}' \leftarrow \left(u_{n+1} \times \Pi_{i \in [n]} w_i\right) \in S_{(1,1,\ldots,1)}^{\left(\rho_{n+1} \cdot (\Pi_{i \in [n]} \alpha_{i,V[i]}) \cdot (\Pi_{i \in [n]} \rho_{i,\vec{x}[i]})\right)}$
3. output the bit: $\mathsf{isZero}(evparams, (\mathbf{t} - \mathbf{t}'))$.

Fig. 1. Obfuscator for Conjunctions.

Our obfuscator for the class of conjunctions is presented in Figure 1. Correctness follows in a straightforward manner as described in the following lemma

(the proof is omitted). We note that the error is one sided, it is always the case that if $C(\vec{x}) = 1$ then for the obfuscated program $\mathcal{O}_C(\vec{x}) = 1$ as well.

Lemma 3.3 (Obfuscator Functionality). *Let C be an n-variable conjunction and consider its obfuscation $\mathcal{O}_C = \mathsf{ConjObf}(C, 1^n, 1^\lambda)$. Then for all $\vec{x}$,*

$$\Pr[\mathcal{O}_C(\vec{x}) \neq C(\vec{x})] \leq poly(n)/p \ ,$$

where $p = 2^{\Omega(\lambda)}$ is the order of the group in the graded encoding scheme, and the probability is taken over the randomness of ConjObf*.*

As a concluding remark, we note that if our graded encoding scheme has the property that $p \gg 2^n$ (which is indeed achievable in the candidate of [14]), then a stronger correctness guarantee, as mentioned in Remark 2.9, can be achieved by using the union bound. In this parameter range, the proof of security also becomes somewhat simpler (see Section 4). However, we want to present our scheme in the most generic way so as to be compatible with possible choices of the security parameter and with future graded encoding schemes.

4 Security from GXDH and GCAN

In this section we prove that ConjObf is a secure distributional black box obfuscator for any distribution over the conjunctions family for which the function is hard to determine (i.e. has super-logarithmic entropy) even if the locations of all the wildcards are known. Namely, there is sufficient min-entropy in V even given W (recall that $V[i] = 0$ wherever $W[i] = 1$).

Definition 4.1 (equivocality given wildcards). *Let $\mathcal{C}$ be the class of conjunctions, and let $\mathcal{D} = \{\mathcal{D}_\lambda\}$ be an ensemble of families of distributions. We say that $\mathcal{D}$ is* equivocal given the wildcards *if there exists $h(\lambda) = \omega(\log \lambda)$ such that for all $D \in \mathcal{D}_\lambda$, if $(V, W) \leftarrow D$ then*

$$\widetilde{\mathbf{H}}_\infty(V|W) \geq h(\lambda) \ .$$

We will prove the security of ConjObf for such functions under the GXDH and GCAN assumptions (see Section 2.2).

Theorem 4.2. *Based on the GXDH and GCAN assumptions, the algorithm* ConjObf *is an average-case black-box obfuscator for ensembles of distribution families that are equivocal given the wildcards.*

Proof. We start by stating a claim that will be used later on in the proof.

Claim. Let p be prime and let $k \in \mathbb{N}$ be integer. Consider the hash family $\mathcal{H} \subseteq \{0,1\}^k \to \mathbb{Z}_p^*$, where each function in $\mathcal{H}$ is defined by a sequence $a_0, a_1, \ldots, a_k \in \mathbb{Z}_p^*$ and

$$\mathcal{H}_{a_0,a_1,\ldots,a_k}(x_1, \ldots, x_k) = a_0 \cdot \prod_{i \in [k]} a_i^{x_i} \ ,$$

then $\mathcal{H}$ is pairwise independent.

This claim follows in a straightforward manner since $\mathcal{H}$ (defined therein) is a random linear function "in the exponent".

Consider a function $C = (W, V)$ drawn from a distribution $\mathcal{D}_\lambda$, and consider the distribution of a properly obfuscated program $\mathsf{ConjObf}(C)$. We will show, using a sequence of hybrids, that this distribution is computationally indistinguishable from one that does not depend on C, even for a distinguisher who knows the value of the predicate $P(C)$. This will immediately imply a simulator. We note that our proof works even for $P(C)$ with multiple-bit output, so long as $h(\lambda) - |P(C)| = \omega(\log \lambda)$.

1. In this hybrid, we use $\mathsf{ConjObf}$ as prescribed:

$$\mathcal{O}_C = \mathsf{ConjObf}(C) = \left(params, \mathbf{pzt}, \{(w_{i,b}, u_{i,b})\}_{i\in[n],b\in\{0,1\}}, (w_{n+1}, u_{n+1})\right)$$

2. We change the algorithm so that $a_{i,b} \notin S_0^{(0)}$. This is implemented efficiently by rejection sampling, using the zero-test procedure. In this hybrid, therefore, $\alpha_{i,b}$ is uniform in $\mathbb{Z}_p^*$.
   This hybrid only incurs a negligible $poly(\lambda)/p$ statistical distance in the distribution of $\mathcal{O}_C$ compared to the previous hybrid.
3. We change step 4 of the obfuscator. In particular, we will now sample $a_{n+1} \leftarrow \mathsf{samp}(params)$ (conditioned on it not being zero, as above). This means that $a_{n+1} \in S_0^{(\alpha_{n+1})}$ for a random $\alpha_{n+1} \in \mathbb{Z}_p^*$.
   We will now show that the resulting $\mathcal{O}_C$ distribution is computationally indistinguishable from the previous hybrid under the GCAN assumption (Assumption 2.7), even when the distinguisher knows $P(C)$. Namely, we will show that for some negligible ϵ, the distributions in the previous hybrid are such that

$$\widetilde{\mathbf{H}}_\infty^\epsilon(\alpha_{n+1} | \{(w_{i,b}, u_{i,b})\}_{i\in[n],b\in\{0,1\}}, P(C)) = \omega(\log \lambda) \ , \tag{5}$$

   which will allow us to apply GCAN and conclude that α_{n+1} can be replaced by a uniform variable.
   To show that Eq. (5) holds, we present a slightly different way to generate the variables $\alpha_{i,b}$ (note that from this point and on, we are a completely information-theoretic setting, so we will not worry about computational aspects). We will first sample $\{\hat{\alpha}_{i,b}\}_{i\in[n],b\in\{0,1\}}$ completely uniformly in $\mathbb{Z}_p^*$, and then set $\alpha_{i,b}$ as follows. If $W[i] = 0$ then $\alpha_{i,0} = \hat{\alpha}_{i,0}$, $\alpha_{i,1} = \hat{\alpha}_{i,1}$; and if $W[i] = 1$ then $\alpha_{i,0} = \alpha_{i,1} = \hat{\alpha}_{i,0}$. Note that the resulting distribution of the α's is exactly as prescribed. Further notice that

$$\alpha_{n+1} = \prod_{i\in[n]} \alpha_{i,0}^{1-V[i]} \alpha_{i,1}^{V[i]} = \prod_{i\in[n]} \hat{\alpha}_{i,0}^{1-V[i]} \hat{\alpha}_{i,1}^{V[i]} = \prod_{i\in[n]} \hat{\alpha}_{i,0} \cdot \prod_{i\in[n]} (\hat{\alpha}_{i,1}/\hat{\alpha}_{i,0})^{V[i]} \tag{6}$$

   where the second equality is since α and $\hat{\alpha}$ only differ where $W[i] = V[i] = 0$. By Claim 4 it follows, therefore, that α_{n+1} is the output of a pairwise-independent hash function applied to V.

We proceed to apply Lemma 2.3. Note that $\widetilde{\mathbf{H}}_\infty(V|W,P(C)) \geq \widetilde{\mathbf{H}}_\infty(V|W) - |P(C)| \geq h(\lambda) - 1$. Therefore there must exist $h'(\lambda) = \omega(\log\lambda)$ such that $h'(\lambda) \leq h(\lambda) - 1$, and in addition the length of α_{n+1} is at least $h'(\lambda)/3 + 2$. We can thus apply Lemma 2.3 with $\epsilon = 2^{-h'(\lambda)/3} = \mathit{negl}(\lambda)$ to argue that

$$\widetilde{\mathbf{H}}^{\epsilon}_\infty(\alpha_{n+1}|W,P(C),\{\hat{\alpha}_{i,b}\}) \geq h'(\lambda)/3 = \omega(\log\lambda)\ . \tag{7}$$

Finally, Eq. (5) follows by noticing that there is an invertible mapping between $W, \{\hat{\alpha}_{i,b}\}$ and $\{(w_{i,b}, u_{i,b})\}_{i\in[n],b\in\{0,1\}}$.

It is interesting to note that this hybrid (and therefore our entire argument) works not only for predicates. In fact, ℓ-bit functions of the circuit C can be used, so long as $h(\lambda) - \ell = \omega(\log\lambda)$.

At this point, $\mathcal{O}_C$ does not depend on V anymore, however it still depends on W via step 2 of $\mathsf{ConjObf}$.

4. We again allow $a_{i,b}$ to be zero. The statistical difference is $\mathit{poly}(\lambda)/p = \mathit{negl}(\lambda)$, as above.
5. We change step 2 of the obfuscator to always act as if $i \notin W$, namely $\alpha_{i,0}$ and $\alpha_{i,1}$ are uniform and independent.
   A sequence of n hybrids will show that any adversary distinguishing this distribution from the previous one, can be used to break GXDH with only a factor n loss in the advantage. This implies that the hybrids are computationally indistinguishable assuming GXDH. Note that knowledge of $P(C)$ (or even of C in its entirety) is useless for the distinguisher at this point.

After the last hybrid, we are at a case where all $a_{i,b}$, $r_{i,b}$, a_{n+1}, r_{n+1} are completely independent of each other, and are sampled in the same way regardless of (V,W). It follows that our final distribution is independent of C, but produces $\mathcal{O}_C$ indistinguishable from $\mathsf{ConjObf}(C)$ (even given $P(C)$). Since this distribution is efficiently sampleable (via the process we describe in the proof), the theorem follows.

Acknowledgments. We thank the reviewers of CRYPTO 2013 for their comments. We thank Masayuki Abe for pointing us to the zeroing attack on [14], and the authors of [10] for discussing its (in)applicability to their scheme. The first author wishes to thank the Simons Foundation and DARPA for their support.

References

1. B. Adida and D. Wikström. How to shuffle in public. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 2007.
2. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012. Preliminary version in CRYPTO 2001.
3. N. Bitansky and R. Canetti. On strong simulation and composable point obfuscation. In *CRYPTO*, pages 520–537, 2010.

4. A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 335–359. Springer, 2008.
5. D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002.
6. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, pages 455–469, 1997.
7. R. Canetti and R. R. Dakdouk. Obfuscating point functions with multibit output. In *EUROCRYPT*, pages 489–508, 2008.
8. R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *STOC*, pages 131–140, 1998.
9. R. Canetti, G. N. Rothblum, and M. Varia. Obfuscation of hyperplane membership. In *TCC*, pages 72–89, 2010.
10. J.-S. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. Cryptology ePrint Archive, Report 2013/183. To appear in CRYPTO 2013.
11. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. Preliminary version in Eurocrypt 2004.
12. Y. Dodis and A. Smith. Correcting errors without leaking partial information. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 654–663. ACM, 2005.
13. Y. Dodis and A. Smith. Correcting errors without leaking partial information. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 654–663. ACM, 2005.
14. S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013. See also Cryptology ePrint Archive, Report 2012/610.
15. S. Goldwasser and Y. T. Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.
16. S. Goldwasser and G. N. Rothblum. On best-possible obfuscation. In S. P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2007.
17. D. Hofheinz, J. Malone-Lee, and M. Stam. Obfuscation for cryptographic purposes. *J. Cryptology*, 23(1):121–168, 2010.
18. S. Hohenberger, G. N. Rothblum, A. Shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. *J. Cryptology*, 24(4):694–719, 2011.
19. M. J. Kearns and U. V. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, USA, 1994.
20. B. Lynn, M. Prabhakaran, and A. Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT*, pages 20–39, 2004.
21. U. M. Maurer. Abstract models of computation in cryptography. In *IMA Int. Conf.*, pages 1–12, 2005.
22. R. Renner and S. Wolf. Smooth renyi entropy and applications. In *IEEE International Symposium on Information Theory*, IEEE International Symposium on Information Theory, page 233, 2004.
23. R. Rothblum. On the circular security of bit-encryption. In *TCC*, pages 579–598, 2013.
24. V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.
25. H. Wee. On obfuscating point functions. In H. N. Gabow and R. Fagin, editors, *STOC*, pages 523–532. ACM, 2005.