How Fast Can We Obfuscate Using Ideal Graded Encoding Schemes

No Author Given

No Institute Given

Abstract. In this work, we present a new obfuscator using a Graded Encoding Scheme (GES) with a binary slot. We characterize a class of circuits taking locally keyed input (each input bit of the circuit is a keyed function over $c > 1$ bits of a binary-variable vector X of length n, where c is called the locality), called ideal functions, such that any function of algebraic degree d (called d-function) over them, can be obfuscated with multilinearity $\mu = (d+1)n/c$. Next we show that obfuscation of a general circuit C can be bootstrapped by $O(n)$-functions (the circuit (called RE) composing a garbled circuit (GC) with a pseudorandom function (PRF)), following an approach similar to that of Zimmerman and Applebaum et al. [35, 8], assuming PRF (or more precisely RE) exists among d-functions with constant d.

To instantiate the above scheme, we achieve the following:

- A concrete GC of algebraic degree 3 over its random bits, which has output size no more than $20\lambda|C|$ and random tape length about $10\lambda|C|$, where λ is the security parameter, $|C|$ denotes the number of gates of the circuit C.
- A candidate d-function construction, where we argue that $d = 1$ suffices to stop linear distinguishing attacks and $d = 2$ seems enough for fully secure PRF.
- Instantiation of the GES with a simplified version of the CLT multilinear map, and various techniques that further reduce μ of the core obfuscator cost-equivalently to $dn/(2c) + 1$ in cases of our interest.

If we replace the PRF with d-functions, then we get various heuristic obfuscation-friendly REs, and thus general obfuscators with explicit complexities. For the most optimistic choice, we have $\mu = 1.5n'/c + 2.5$, $n' \approx n + \log|C| + \log\lambda$, n is the number of input bits of C, and c is a selectable constant which result in a $2^c/c$ times increase of the key size of the RE.

Our general obfuscator is VBB secure assuming that our RE is secure and our simplified CLT map is a secure instantiation of our GES (defined relative to known attacks). We leave these assumptions with concrete parameter sets as open challenges.

We illustrate the efficiency of our methods with some examples:

- Our obfuscated AES ($c = 13$, $\mu = 20.5$) has code size $< 1.5 \times 10^{17}$ bits, whereas no implementable solution is known prior to this work.
- We can practically obfuscate conjunction functions for $n = 64$, while the latest implementation [20] can only handle $n = 32$ with comparable resources. We also verify the security against algebraic attacks in this example.

Keywords: Obfuscation, garbled circuits, pseudorandom functions, graded encoding scheme, CLT map

1 Introduction

1.1 Background and Motivation

Obfuscation of general circuits is a powerful functionality in cryptography [30] which was regarded infeasible [12] until the celebrated work [21]. But this great work only addressed the existence problem: the solution is infeasible for any meaningful examples. Significant improvement of efficiency was obtained in later constructions (e.g. [2, 35, 5, 31, 8, 25, 4, 22, 3, 14, 26]), but all these works only got that the complexity depends polynomially on the security parameter and circuit parameters: the complexity polynomial (even the degree) is not explicitly given. Here is the reason: known essential obfuscation methods, called core obfuscation, have to use a tool called Graded Encoding Scheme (GES) and can directly handle only NC^1 circuits; to obfuscate a more complex circuit, it is necessary to transform the task to obfuscating a simpler bootstrapping circuit which is NC^1; the latter needs a GES of huge (though polynomial) multilinearity μ. All known instantiations of GES need a multilinear map which is highly inefficient according to known constructions [21, 19]. A multilinear map noisily encodes integers in a huge ring. The key encoding complexity measure $\mathrm{GES}(\mu, \lambda)$ is the size of a single ring element which depends on μ and the security parameter λ (which means concrete security strength in this work: any successful attack has complexity no less than 2^λ). In current constructions, $\mathrm{GES}(\mu, \lambda)$ is proportional to μ^2. These amount to the status that all known methods for obfuscation of general circuits cannot be implemented in practice, in fact, the whole input-output table stays the best solution for not so large n, say $n \leq 128$, where n is the number of input bits.

Given that obfuscation is so wanted, it is worthwhile to investigate the "how fast" side of the obfuscation notion. In this respect, we do not insist on provable security in the standard model, as "hard to attack" is the second choice for security assurance when the first is not affordable. In practice, a "hard to attack" scheme is often obtained by instantiating a scheme secure in an ideal model (the most popular example is random oracle, later we will define what the instantiation of our GES is). Another general method to improve efficiency is to replace the low level primitives with those constructed with heuristic methods: instead of provable security, we try to defend known attacks. Our guideline to achieve such practical security is to begin with a sound construction in the ideal model and instantiate it heuristically in these ways. We think that this kind of compromise of security is inevitable in the way to get obfuscation to real world application because such a functionality seems unreachable at a first glance.

Our first task is to obtain a sound construction in the ideal model which is suitable for modifications as above. There are three kinds of methods to get general obfuscation. The first is based on obfuscating branching matrix programs:

μ is the length of the program, the translation from circuit to branching matrix programs will result in program length of polynomial of the circuit size, another major efficiency bottleneck besides the multilinear maps, making it hopeless to be practical. The second kind makes use of functional encryptions: it needs many steps each of which has to treat a recursively defined huge circuit where the circuit of the multilinear map tool itself is only a small unit, and no explicit complexity is available even when μ is reduced to 2 [25]. The third is the approach of Zimmerman and Applebaum et al. [35, 6, 8]: using the core obfuscators [35, 8] to obfuscate the Randomized Encodings (RE) of the circuit to be obfuscated. The existing (implicit) constructions of RE that work for obfuscation have μ of size only bounded by a (existential) polynomial in n, λ, but this seems the only bottleneck of efficiency. So we choose to follow the third approach.

1.2 The Basic Ideas

Since known bootstrapping NC^1 circuits cannot be handled by the Zimmerman and Applebaum et al. core obfuscators [35, 8], it is natural to firstly ask how they can be improved and what can be efficiently obfuscated directly. The core obfuscators [35, 8] work as follows: first transform the binary circuit to arithmetic one (which usually increases the size), and apply the GES encoding (in ElGamal form) on inputs and key bits: a GES encoding $[a, b]_v$ symbolically encodes two (or more) slots of values $[a, b]$ at a level v; such encodings can be operated as $[a_1, b_1]_{v_1}[a_2, b_2]_{v_2} = [a_1a_2, b_1b_2]_{v_1+v_2}$ and $[a_1, b_1]_v + [a_2, b_2]_v = [a_1 + a_2, b_1 + b_2]_v$; to enable addition of two encoded values at different levels, values are encoded in ElGamal form: $[a, b]$ is encoded as $([r, s]_v, [ra, sb]_v)$, where $[r, s]_v$ is called ElGamal scalar. Then do GES operations along the arithmetic circuit: at each gate, the multilinearity μ for the outwire is the sum of those of the inwires except for addition gate with inwires with the same ElGamal scalar, which is called free addition gate. The transformation from binary circuit to arithmetic has the effect that multiplications are extensively used and additions can hardly be made free. A small circuit can result in a huge μ in general. So our first choice is to use a GES with the first slot encodes binary values directly so that the binary to arithmetic transformation is avoided.

As μ must be added at an "and" gate, the best that can be expected is all "addition" gates are free. The primary subcircuit of such circuits can be characterized as follows: the level of each input gate x_i is indexed by the single index set $\{i\}$ (the level group contains the free abelian group with generator set $[n]$), the index set of an "and" gate is the disjoint union of those of the two inwires; and for "addition" gate, the index set of one inwire must be contained in that of the other (so that there is an ElGamal scalar to compensate the lower degree one to get the addition free); each output gate is indexed by the whole set $[n]$. We call sum of polynomially many of such primary circuits ideal function because it has the property that each output bit depends all input bits (a minimal requirement to replace the PRF of RE), and can be obfuscated with all addition gates free; moreover, all output bits share the same ElGamal scalar indexed by the whole input vector x, so that any function over it can be obfuscated in the ideal way that all "addition" gates are free. A d-function is an circuit of algebraic degree d over an ideal function. Obfuscation of d-functions with the core obfuscators [35, 8] will get $\mu = (d+1)n$ for IO or $\mu = dn + n^2$ for VBB security. Note that the notion of IO security makes no sense to our task (to hide the key of RE), the n^2 term in μ seems too expensive. Basically we modify the core obfuscators as follows: instead of encoding individual inputs and key bits, we encode the values of gates somewhere near the inputs: each such gate depends on no more than c bits (form a block) of the input. That is, we consider circuits taking locally keyed input, i.e. each input bit is an output bit $(f(k,x))_j$ of a keyed function $f(k,x)$ which depends on no more than c bits of the input x, where c is called the locality. To obfuscate such circuits, we encode each input bit with an ElGamal scalar indexed by the subvector of input x it depends on, and the level is also indexed by the block name. This simple technique reduces μ by a factor c, and enables blocks straddling to get VBB security for free, and also makes the obfuscator (when instantiated) more resilient to algebraic attacks.

Our first contribution is this new core obfuscator for d-functions specified in the ideal GES model. Ideal functions may be too complex to lie in NC^1 and Point functions and Conjunctions (the only meaningful family feasibly obfuscable in practice) are ideal functions, our core obfuscator remarkably augments the class of circuits directly obfuscable.

1.3 Bootstrapping in d-functions

Next we wish to realize RE in d-functions. Our RE has the form $gc(k, x, f(k', x))$, where gc is a garbled circuit, and its random tape is provided by a keyed function, so that RE becomes keyed function (unlike the meaning of the name in other literatures). There is a construction [9] of gc with algebraic degree 4 over its random bits assuming only quadratic minimal pseudorandom generators (PRG). If f is a pseudorandom function (PRF), then this RE is provably secure. One of our heuristics is that an f weaker than PRF may still keep RE secure.

We give a reference construction of $d-$functions, where our cryptanalysis suggests that ideal functions may stop linear distinguishing attacks, and 2-functions may marginally resist known nonlinear distinguishing attacks. So it is reasonable to assume that there exists PRF in $d-$functions for some small constant d, which concludes that there exists general obfuscators under ideal GES model with $\mu = O(n)$.

We may use d-functions for $d = 1, 2, 3 \cdots$ for f to get heuristic REs with increasing assurance. To give RE candidates with concrete efficiency and security, we improve the GC in [9] to have degree 3 and also a smaller GC size by giving a candidate quadratic $\{0,1\}^{2\lambda} \rightarrow \{0,1\}^{(3+)\lambda}$ PRG. This leads to concrete RE examples in d-functions for $d = 3, 6, \cdots$, which may serve as targets for further cryptanalysis.

1.4 Instantiation of GES

We give a heuristic instantiation of the GES with a simplified version of the CLT multilinear map. We first show that the level group can be realized. Then additional optimizations of the core obfuscator are given, which further reduce μ of the core obfuscator to cost equivalently $dn/(2c)+1$ in cases of our interest. We also illustrate how to set the parameters of the CLT map in our general obfuscator, where $\mathrm{GES}(\mu,\lambda)/\eta^2$ is taken as the security level which is somewhat conservative. This leads to concrete general obfuscators (depending on concrete REs) with explicit complexity.

Then we investigate the security of our instantiation under known attacks. The first kind of attacks [19] on the original CLT map exploit the same weakness: multiple encodings lie in the same level, they are called exposed. These attacks work the same way with our modified version (the slot encoding binary value seems not harmful), and determine the choice of parameter sizes. The second kind attacks [18, 16, 17] (called algebraic attack, or zeroizing attack) make use of the leak of the zero-test: a known polynomial of the initial noises (called zero-test polynomial) over the field of rational numbers is returned by any acceptable zero-test. If one can partition the input set so that some set of zero-test polynomials vary with two input sets like a bilinear form, then CLT map is totally broken. In our core obfuscator, if the ElGamal scalar of the ideal function output can be obtained by different polynomials of the initial encodings, then this attack works. We can avoid this in our general obfuscator: we omit the ElGamal scalar of some blocks so that there is only one way to form the whole-vector ElGamal scalar, which means the ideal function is linear with respect to the blocks without ElGamal scalar. Now zero-test polynomials only come from honest computation, which factor through a PRF and thus are too complex to allow the input set partition. But it generally threats simple circuits such as conjunctions [18]. We use an explicit block straddling to defend our specific example against such attacks, and give a method to defend general circuit.

In general, we give a definition of "good instantiation" of GES which absorbs the "hard to verify" part of VBB definition in standard model, so that VBB security in ideal GES model transforms to VBB security in standard model under a Legal condition. We conjecture that the Legal condition for our CLT map is non-existence of the above-said partition of the initial encoding set relative to the set of testable symbolic polynomials, which excludes known algebraic attacks. Our obfuscators satisfy this Legal condition.

1.5 Examples

The complexity of our obfuscators only depends on $n, |C|, \lambda$. We give an estimation of the cost for $n = 128, |C| = 25000, \lambda = 128$, which supersedes that for obfuscating AES. The result shows that such task is implementable but far from practical. When our core obfuscator applied to the simplest kinds of ideal functions (the conjunction functions), the result is much more acceptable: we can practically handle the case $n = 64$.

1.6 Organization of the Paper

The rest of this paper is organized as follows: Section 2 is notations and definitions used in this work. section 3 introduces ideal functions and describes our core obfuscator. Section 4 gives our GC construction. Section 5 presents our candidate d-functions. Section 6 describes our general obfuscation scheme. Section 7 treats the instantiation of the GES. Section 8 gives examples: obfuscated AES, and obfuscation of conjunctions. Section 9 is some concluding remarks.

2 Preliminaries

2.1 Notations

We let $\mathbb{Z}_l$ denote the residue integer ring $\mathbb{Z}/l\mathbb{Z}$. $|A|$ denotes the number of elements of a set A. $[n]$ denotes the integer set $\{i : 1 \leq i \leq n\}$. If x is a vector of length n, x_A means the subvector corresponding to the subset $A \subset [n]$; $(x)_j$ is the jth component while x_j means a subvector. If S is a set, we like to use s (with subscripts) to represent its elements. By abuse of notations, X can refer to either the input space or the set of input variables, $X = Y \times Z$ means X is partitioned into two parts, i.e. the order of the variables is irrelevant. We use "$A + B = C$" to mean (A, B) is a partition of a set C. $A = A_1 + A_2 + \cdots$ is called a c-partition(A_i $c-$block), if $|A_i| \leq c$ $\forall i$. λ is the security parameter, and the security strength is in bits. All algorithms are efficient by default.

2.2 Circuits and Obfuscation

A circuit C in this work always means binary and may compute a keyed function $C : (K, X) \to \{0,1\}^l$, where K and X are called key space and input space respectively. For any $k \in K$, C_k denotes the circuit that computes the function $C(k, \cdot)$. In other words, the key is hardwired into definition of gates of C_k: each gate i of C_k is defined by a function $g_{i,k} : \{0,1\}^2 \to \{0,1\}$, and all C_ks have the same topology: each gate i receives inputs from the same gates i_0, i_1 for all k, where $-1, \cdots, -n$ index input gates, $i > i_0 > i_1$. We call this the keyed circuit for computing the keyed function $C(K, X)$. A keyed circuit can be obtained from a universal circuit, by first absorbing gates that do not depend on inputs, and then absorbing all single inwire gates into definition of their two-inwire receiving gates.

Obfuscation of a keyed function $C(K, X)$ is pair of PPT algorithms (E, Ob), such that there exists an evaluation algorithm $E(x, \mathrm{Ob}(k)) = C(k, x)$ $\forall k, x$. The input to the algorithm Ob is actually the keyed circuit C_k: n the number of input variables (corresponding to gates $-1, \cdots, -n$); $|C|$ the number of gates (labeled with integers $1 \leq i \leq |C|$); circuit topology $\{(i, i_0, i_1) : 1 \leq i \leq |C|\}$ where $i_0 < i_1 < i$; and gate definitions $g_{i,k} : \{0,1\}^2 \to \{0,1\}$.

There are two main security notions about obfuscation: IO and VBB. IO means $\mathrm{Ob}(k_1)$ is indistinguishable with $\mathrm{Ob}(k_2)$ if $C(k_1, \cdot) = C(k_2, \cdot)$. VBB means

Ob(k) reveals nothing beyond the oracle $C(k, \cdot)$, which cannot be defined meaningfully in the standard model: a simulator based definition would lead to nonexistence [12], whereas any other definition capturing this intuitive notion has an unprovable format (but can be easily falsified by attacks). VBB can be defined in an ideal model as follows. There is a simulator $\mathcal{S}$ such that: for any adversary $\mathcal{A}$, $(\text{ob}, \mathcal{O}) \leftarrow \text{Ob}(k)$; $\text{ob}' \leftarrow \mathcal{S}$;

$$\left|\Pr[\mathcal{A}^{\mathcal{O}}(\text{ob}) = 1] - \Pr[\mathcal{A}^{\mathcal{S}^{C(k,\cdot)}}(\text{ob}') = 1]\right| < \epsilon$$

where the oracle is outputted by the obfuscator means it is initialized there. The oracle we will use is GES defined later. VBB in standard model can be formulated as: for any predicate leak, any adversary $\mathcal{A}$, there exists an algorithm $\mathcal{S}$, such that

$$\left|\Pr[\mathcal{A}(\text{Ob}(k)) = \text{leak}(k)] - \Pr[\mathcal{S}^{C(k,\cdot)} = \text{leak}(k)]\right| < \epsilon.$$

This definition of obfuscation is not phrased in circuits but it is equivalent to the formulation in terms of circuit families.

Definition 1. *A Randomized Encoding (RE) of $C(k, X)$ is a keyed function $RE(\tilde{k}, k, X)$, such that there exist algorithms D, S such that $D(RE(\tilde{k}, k, x)) = C(k, x)$ $\forall x$ and for all adversary $\mathcal{A}$,*

$$\left|\Pr[\mathcal{A}^{RE(\tilde{k},k,\dot{)}} = 1] - \Pr[\mathcal{A}^{S(\tilde{k})^{C(k,\dot{)}}} = 1]\right| < \epsilon.$$

It is easily seen that VBB obfuscation of any RE of $C(k, X)$ gives a VBB obfuscation of $C(k, X)$.

Definition 2 ([13]). *A garbled circuit of $C(k, X)$ is a function $gc(k, x, r)$ such that there is an algorithm D: $\Pr[D(gc(k, x, r)) = C(k, x)] > 1 - \epsilon$ and there is a simulator S: for all adversary A,*

$$\left|\Pr[A(gc(k, x, r)) = 1] - \Pr[A(S(C(k, x)) = 1]\right| < \epsilon$$

Our RE will have the form $gc(k, x, f(\tilde{k}, x))$, where $gc(k, x, r)$ is a garbled circuit with a simulator $S(\cdot, r')$ such that r, r' share a substring r_0. When we say $f(\tilde{k}, x)$ is secure in this RE, we mean the simulator $S(\cdot, r')$, when r_0 is the corresponding substring of $f(\tilde{k}, x)$, works as the simulator of RE(it is called the composition of $S(\cdot, r')$ and $f(\tilde{k}, x)$). A PRF is obviously secure in RE, but a weaker $f(\tilde{k}, x)$ may still be secure.

2.3 The Ideal GES Model

The ideal GES in this work is similar as the MRG model of [8], it is used to encode values by the obfuscator and allow public operations of the encodings to manipulate the encoded values homomorphically and secretly in rules specified by levels and also to zero-test at a test-level. The encoded value lies in two slots:

the first slot for binary values (in $\mathbb{Z}_2$), and the second for a big ring $\mathcal{U}$ which is universal for an allowable (defined later) family $\mathcal{P}$ of polynomials in the sense: for any $p \in \mathcal{P}$ of m variables:

$$\Pr[p(\alpha_1, \cdots, \alpha_m) = 0 : \alpha_i \leftarrow_R \mathcal{U}] < 2^{-\lambda}$$

where $\leftarrow_R$ means "sampled uniformly randomly from". The boolean slot is used for circuit evaluation and the other one for authentication of the computing process. An encoded value will take the form $[b, \alpha]$. The levels that a value can be encoded are in an abelian group $\mathcal{V}$, that is, a set of generators $\{v_j\}$ and a set of relations $\{R_i = 0\}$ over $\{v_j\}$, where R_i is a linear combination of $\{v_j\}$ with integer coefficients. Each encoding has a degree $d > 0$ (to keep track of noise size). So a GES encoding takes the form $[b, \alpha]_v(d)$, where the content $[b, \alpha]$ is secret, but v, d is assumed to be known by the adversary. The adversary can only perform symbolic operations "$+$", "$-$" and "$\times$" to obtain new encodings:

$$\begin{aligned}
[b_1, \alpha_1]_v(d_1) + [b_2, \alpha_2]_v(d_2) &\sim [b_1 \oplus b_2, \alpha_1 + \alpha_2]_v(\max(d_1, d_2)),\\
[b_1, \alpha_1]_v(d_1) - [b_2, \alpha_2]_v(d_2) &\sim [b_1 \oplus b_2, \alpha_1 - \alpha_2]_v(\max(d_1, d_2)),\\
[b_1, \alpha_1]_{v_1}(d_1) \times [b_2, \alpha_2]_{v_2}(d_2) &\sim [b_1 \& b_2, \alpha_1 \times \alpha_2]_{v_1+v_2}(d_1 + d_2)
\end{aligned}$$

where $\sim$ stands for having the same content and level. The degree may be dropped if it is not stressed on. An ElGamal form for a bit b is a pair

$$\big([1, r]_v(d), [b, r\alpha]_v(d)\big),$$

where $[1, r]_v(d)$ is called the ElGamal scalar. In the ElGamal forms, any two encodings can be added,

$$\begin{aligned}
&\big([1, r_1]_{v_1}(d_1), [b_1, r_1\alpha_1]_{v_1}(d_1)\big) + \big([1, r_2]_{v_2}(d_2), [b_2, r_2\alpha_2]_{v_2}(d_2)\big)\\
&= \big([1, r_1 r_2]_{v_1+v_2}(d_1 + d_2), [b_1 + b_2, r_1 r_2 \alpha_1 \alpha_2]_{v_1+v_2}(d_1 + d_2)\big)
\end{aligned}$$

but the price is that degrees are also added. If scalars are the same, two ElGamal forms can be added without increasing the degree, this is called free addition. There is also a private interface

$$[b, \alpha]_v(d) \leftarrow \text{Encode}(b, \alpha, v, d)$$

and a public function $\text{ZeroTest}([b, \alpha]_v(d)) = b$ if $v = v_{zt}$, $\alpha = 0$, and $d \leq \mu$, otherwise it is rejected, where the multilinearity μ is the maximal degree d that is allowed in an encoding $[b, \alpha]_v(d)$; the test level v_{zt} is the only level at which the adversary can zero-test.

The formal definition of our GES is a state machine which works as follows. It maintains a list of elements of the form (h, h_1, h_2, o_h, E_h), where $h > h_1 > h_2$ are integers (called handles), o_h is one of the operators, E_h is the encoding pointed by handle h, and such a tuple means $E_h = E_{h_1} o_h E_{h_2}$. In the encoding stage, it responses to all encode requests and puts the results in a table $\{E_h, -1 \geq h \geq -l\}$, and is initialized with $h = 0$ and the list empty. Then it can response to the operator requests. Each request is of form (h_1, h_2, o): it first checks if the operation is allowed, then it checks if the result as a formal polynomial of the initial encodings coincides with some entry of the list: if yes, returns that handle; otherwise puts (h, h_1, h_2, o, E_h) on the list, and augments h by 1. Note that the list defines a symbolic circuit $\tilde{C}_h$ for each $h \geq 0$, where the initial encodings are input gates, so each h points to a symbolic polynomial $\tilde{P}_h$ over the initial encodings defined by $\tilde{C}_h$. To check if two such polynomials are identical, it only needs to compute their circuits over random inputs. This model means the encode function has different responses to requests of the same contents.

A GES can be used to obfuscate a circuit $C(k, x)$ by taking it as symbolic and encoding each input and key bit in initial encodings [35, 8]. We allow C to take locally keyed input, i.e. we can evaluate $C(f(k, x))$, where each output bit $(f(k, x))_j$ of $f(k, x)$ depends on no more than c bits $x_j \in X_j$ of x, where X_j denotes a space consisting of at most a constant c components of the input space X. Given a set of initial encodings $\{[b_i, r_i\alpha_i]_{v_i}(d_i) : 1 \leq i \leq l\}$, where the keyed input is encoded in $\{b_i\}$, and $\{\alpha_i\}$ is independent input, such that there is a symbolic circuit $\tilde{C}_j$ satisfying

$$\tilde{C}_j\big(\{[b_i, r_i\alpha_i]_{v_i}(d_i) : 1 \leq i \leq l\}|_x\big) \sim \big[(C(f(k, x)))_j, 0\big]_{v_{zt}}(\mu)$$

where $|_x$ means a selection (subset) indexed by x, and $(y)_j$ means the component of a vector y indexed by j. Now the zero test reveals $C(f(k, x))$. For such a $\tilde{C}$ to be constructed, the set of initial encodings should contain at least: the ElGamal forms

$$\Big\{\Big([1, r_{x_j}]_{v_{x_j}}, [g(k, x_j), r_{x_j}\alpha_j]_{v_{x_j}}\Big) : x_j \in X_j\Big\}$$

for each bit $(f(k, x))_j = g(k, x_j)$ of $f(k, x)$ that depends on X_j, and a check value for each output bit C_j of C. A check value for an output bit C_j of C is an encoding whose content is a multiple of $[0, C_j(\{\alpha_i\})]$, which is used to cancel the second slot of the content resulted from symbolic computing along C_j to form the testable symbolic circuit $\tilde{C}_j$. This generalization of the GES computing model [8] reduces the multilinear degree μ by a factor c when the circuit factors through a c-local input function.

Definition 3. *An obfuscator* Ob *in the* GES *model is an algorithm that works as follows: on getting the description of the circuit family $C(k, \cdot)$, it sets up the ring $\mathcal{U}$ and group $\mathcal{A}$, and makes public its strategy: a distribution of $K \times (\mathbb{Z}_2 \times \mathcal{U} \times \mathcal{A} \times \mathbb{Z})^l$; On getting k, it draws a sample of this distribution to get the* GES *initialized and output the l initial encodings as the obfuscated code* $\mathrm{Ob}(k)$.

- *Correctness: There is a symbolic circuit $\tilde{E}$, such that $\tilde{E}^{\mathrm{GES}}(\mathrm{Ob}(k), x) = C(k, x)$ for all x.*
- *VBB security: There is a simulator $\mathcal{S}$, such that for any adversary $\mathcal{A}$*

$$\Big|\Pr[\mathcal{A}^{\mathrm{GES}}(\mathrm{Ob}(k)) = 1] - \Pr[\mathcal{A}^{\mathcal{S}^{C(k,\cdot)}}(\mathrm{Ob}(0)) = 1]\Big| < \epsilon.$$

It is known from [8] that

Lemma 1 (Algebraic Security). *VBB security holds if*

- *(IO security) For any admissible (accepted by zerotest) symbolic circuit $\mathcal{C}$,*
$$\tilde{P}_{\mathcal{C}} \sim \sum_x c_x \tilde{P}_{\tilde{E}(\cdot,x)}$$
where $\sim$ means differing by a polynomial encoding 0 content, $\tilde{P}$ stands for symbolic polynomial of a symbolic circuit, and c_x is an integer, $\tilde{E}(\cdot,x)$ denotes the symbolic circuit of honest computation over input x.
- *There is an efficient algorithm to list all x such that $c_x \neq 0$.*

2.4 Symmetric Key Primitives

The bias of a boolean variable b is defined as $\varDelta(b) = |2\Pr[b = 1] - 1|$. The Piling lemma says $\varDelta(b_1 \oplus b_2) = \varDelta(b_1)\varDelta(b_2)$ if b_1, b_2 are independent. To distinguish b with uniform distribution for sure, the number of samples is about $1/\varDelta(b)^2$.

Definition 4. *A function $g : \{0,1\}^n \to \{0,1\}^m$ is called a (n,m) Pseudorandom Generator (PRG), where $m > n$, if for any adversary $\mathcal{A}$ runs in time t_A, and let*
$$\big|\Pr[\mathcal{A}(g(U_n)) = 1] - \Pr[\mathcal{A}(U_m) = 1]\big| = 1/2 + \epsilon$$
we have $t_A/\epsilon^2 \geq 2^\lambda$, where U denotes uniform distribution.

We need quadratic PRGs with this definition of concrete security. There are many constructions [27, 7] of this kind, though the security level is only bounded by linear distinguishing attacks, it is widely believed that such bounds also holds for other attacks. A linear distinguishing attack tries to find a linear combination of the outputs with largest bias ϵ. To keep $\epsilon < 2^{-\lambda/2}$, the key length is at least 2λ in these constructions.

Lemma 2. *[27] There exist $(2\lambda, 2\delta\lambda)$ generators with $\epsilon < 2^{-\lambda/2+\delta}$ for $\delta < \lambda/2$, and each output bit is a quadratic function of all input bits.*

But the one-wayness does not hold in this construction. We cut half of its output to get the following candidate PRG: $k \mapsto ((k^{1+2^i})^-)_{1\leq i\leq\delta}$, where $k \in \mathbb{GF}_{2^{2\lambda}}$, $()^-$ stands for taking half of the bits (to achieve one-wayness). For a small constant δ such as $\delta = 3$, the bias ϵ is bounded by above lemma, so the security of the PRG is not a big problem. We have a more optimistic conjecture: this candidate is secure for $\delta \leq \lambda$. We leave the cryptanalysis of this candidate PRG as an open problem.

Definition 5. *A pseudorandom function (PRF) $f : K \times X \to \mathcal{R} : (k,x) \mapsto f(k,x) \in \mathcal{R}$ satisfies: when k is randomly chosen, then for any adversary $\mathcal{A}$.*
$$\left|\Pr[\mathcal{A}^{f(k,\cdot)} = 1] - \Pr[\mathcal{A}^{\mathrm{RO}} = 1]\right| < \epsilon$$
where RO is a random oracle with range $\mathcal{R}$.

When $\mathcal{R} = \{0,1\}^l$, we call such a PRF a (X, l) PRF by omitting specification of the key space. Note that $(X \times \{0,1\}^m, l)$ PRF f implies $(X, l2^m)$ PRF f': $f'(x) = (f(x,t))_{t\in\{0,1\}^m}$, i.e. the position of output can be specified by m input variables.

2.5 Distinguishing Attacks

Traditional constructions of PRFs [28, 29, 24, 10, 34] are too complex to be used in this work. We give a new candidate using design principles in applied cryptography. This means that we aim to defend known attacks. Our PRF will have a wide and shallow circuit, most of the attacks making use of the simple structure of a round function do not apply here. The linear distinguishing attacks remain relevant. It seems that the most effective one with respect to our construction is the Cube Attack: summation over a cube; some algebraic structure of the resulted function remains handleable. Let $D_A f = \sum_{x_A \in X_A} f(x)$, where A is a subset of input variables, and x_A denotes the subvector of x. Let $x^A = \prod_{i \in A} (x)_i$, then $f = x^A D_A f + f_0$, where $\deg(f_0)$ over X_A is less than $|A|$. It is obvious that

Lemma 3.

$$D_A(fg) = \sum_{A=B+C} D_B f D_C g;$$

and if f does not depend on variables in B, and g does not depend on variables in A, there is

$$D_{A+B}(fg) = D_A f D_B g.$$

From an algebraic point of view, if a function has considerably less number of terms than a random one, it is obviously not a PRF. But this may not be detected by only black box access of f. One method to detect this is based on the following observation: given $m > n$ secret vectors $v_1, \cdots, v_m$ of a linear space of dimension n, then it is easy to distinguish the source $(L(v_1), \cdots, L(v_m))$ from a truly random source where L is a random linear function, given more than n samples. The distinguisher computes rank of a matrix $(L_i(v_j))$ of dimension $m \times m$ (so it is a nonlinear distinguishing attack). To distinguish a function $f(X)$ with a random one, one tries to write

$$X = Y \times Z \text{ and } f(y, z) = \sum_{1 \leq i \leq m} g_i(y) h_i(z),$$

i.e. f can be seen as a bilinear form over a space of dimension m. If $|Y|, |Z| > m$ and $m < 2^{\frac{\lambda}{3}}$, we may say the attack succeeds. We denote $\omega(f)$ as the minimal m for which such expression of f exists with respect to this partition. We do not define $\omega(f)$ to be smallest with respect to all partitions, because in practice we cannot find that partition but just try random partitions. In general, we fix a partition of the input variable set in form $[n] = A + B + A'$ with $|A| = |A'|$, where variables in B are taken as constants.

3 Ideal Functions and the Core Obfuscator

For a locally keyed circuit $C(f(k, X))$, each input gate of C is associated with a subset X_j(called an input block) of the variable set X on which $f_j(K, X)$ depends; where $|X_j| \leq c$, c is called the locality.

Definition 6. *A locally keyed circuit is called an ideal function, if it is a sum of polynomially many primary circuits. A locally keyed circuit $C(f(k, X))$ is called primary if: the set of input blocks form a partition of X, each gate of C is indexed by a subpartition and the following conditions are satisfied:*

1. *each input gate j of C is indexed by the set $\{X_j\}$.*
2. *for each "and" gate: the index sets of two inwires must be disjoint. The index set of this gate is the union of the two.*
3. *for "$\oplus$" gate: if the index sets of the two inwires are different, then one is the index set of some predating gate of the other. The index set of this gate is the larger one.*
4. *for each output gate, the index set is maximal: the whole partition of input blocks.*

Roughly speaking, the index set (if not maximal) of each gate of an ideal function is formed in a unique way, which is needed to keep uniqueness of ElGamal scalars. It is obvious that primary functions are ideal functions with only one input block partition, and a vector function is ideal if and only if so are the components.

Example of ideal functions: the conjunction function f, $f(k, x) = 1$ if and only if $x_0 = k$, where x_0 is a subvector of x at a secret index subset. As a keyed circuit, it is just a product of n bits. We can choose any c to make it a product of n/c bits by tabling the computation over the $c-$blocks of the input variables.

Definition 7. *A function is called d-function if it is of the form $g \circ C(f(k, X))$, where the degree $d = \deg(g)$ is over binary variables, and $C(f(k, X))$ is an ideal function.*

Next we will show how to obfuscate d-functions in the GES model by first giving a basic obfuscator and then some variations. We encode each primary component locally and independently but we need the following global variables:

$$\left\{r_{i,b}, v_{i,b} : 1 \le i \le n, b \in \{0, 1\}\right\}$$

where $r_{i,b} \in \mathcal{U}$,$v_{i,b} \in \mathcal{V}$ are independent random samples. Given input x, for any subvector x_j of index set X_j, we use the following notations:

$$r_{x_j} = \prod_{i \in X_j} r_{i,(x_j)_i} \text{ and } v_{x_j} = \sum_{i \in X_j} v_{i,(x_j)_i}$$

.

Within each primary component, there are some constants:

- for each input bit j: $\alpha_j \in \mathcal{U}$;
- for each input block j : $v_j \in \mathcal{V}$;

At each input bit j, the initial encodings are:

$$\left\{E_{j,x_j} = (S_{j,x_j}, T_{j,x_j}) = \left([1, r_{x_j}]_{v_j + v_{x_j}}, [f_j(k, x_j), r_{x_j}\alpha_j]_{v_j + v_{x_j}}\right) : x_j \in X_j\right\}$$

where j is also abused as the block name: S_{j,x_j} is the same for js in the same block.

Now we can evaluate a primary component over an input x as follows:

1. at each input gate j of C: the value of the gate is E_{j,x_j};
2. for each "and" gate: if the value of two inwires are $E_1 = (S_1, T_1), E_2 = (S_2, T_2)$, the value of this gate is $E_1E_2 = (S_1S_2, T_1T_2)$.
3. for "$\oplus$" gate: for two maximal inwires with values $E_1 = (S_1, T_1), E_2 = (S_2, T_2)$, the value of this gate is $(S_1, T_1 + T_2)$; otherwise, if S_1 is larger, there exists an S' such that $S_1 = S'S_2$, value of this gate is $(S_1, T_1 + S'T_2)$.
4. at each output gate i, its value (S_i, T_i) is if of the form (S_x, T_i), where $S_x = [1, r_x]_{v+v_x}$, $v = \sum_j v_j$.

Note that some ElGamal scalars may be missing and the value of the output gate is still reachable as long as in every addition gate the required patching S' exists. Now the level the output gates of all primary components must be same, which gives relations over the generators with block names.

Note that there might be many maximal ElGamal scalars equivalent to S_x. This does no harm in the ideal GES model, but makes the CLT map instantiation insecure because of the algebraic attacks. To make it work for the CLT map instantiation, there should be only one maximal ElGamal scalar. This gives a restriction on the structure of the ideal function circuit C: it is linear with respect to some blocks whose ElGamal scalars can be deleted from the initial encodings so that only one maximal ElGamal scalar can be obtained. This condition is enforced at our candidate ideal function structure for RE.

Now any $d-$function $g \circ C(f(k, X))$ can be evaluated as follows:

1. define the degree of the input gates of g to be 1;
2. for each "and" gate: if the value of two inwires are T_1, T_2, the value of this gate is T_1T_2, and the degree is the sum of those of the two inwires.
3. for "$\oplus$" gate: for two inwires of degree $d_1 \geq d_2$, the value of this gate is $T_1 + S_x^{d_1-d_2}T_2$, and the degree is d_1.
4. to add 1 to a value T of degree d', just do $T + S_x^{d'}$.

Now we get the encodings of the output bits $\{b_i\}$ of $g \circ C(f(k, X))$ of the form:

$$B_i = \left[b_i, p_i(\{\alpha_j\})r_x^d\right]_{d(v+v_x)}$$

where p_i is the polynomial defined by the output gate i.

To get to zero-test, we need a check value for each output gate to cancel the second slot, and also some encodings called patches to get to the unique zero-test level v_{zt}. The check values are given by

$$\left\{\mathrm{CHK}_i = [0, rp_i(\{\alpha_j\})]_w : i\right\}.$$

Choose a suitable (explained later) $c-$partition $X = \sum_i X_i$, the patches are:

$$\left\{P_{x_i} = [1, r_i]_{w_i - v_{x_i}} : x_i \in X_i, i\right\}$$

where $r = \prod_i r_i$, $w = \sum_i w_i$ and $w + dv = v_{zt}$. To complete the evaluation, we just zero-test

$$B_i \prod_j P_{x_j} - \mathrm{CHK}_i S_x^d$$

The output of our basic obfuscator consists of: the initial encodings, the check values and the patches. Assuming all these encodings have degree 1, then we get $\mu = (d+1)n/c$. This completes the description of the obfuscated code and its evaluation on a given input.

The VBB security of the above obfuscator in the ideal GES model can be proved as follows.

Lemma 4. *The first condition (IO security) is satisfied.*

Proof. Any monomial (degree no more than μ) over the initial encodings which has level v_{zt} must have degree μ and indexed by unique (each patching block appears exactly once) x, and has a content of the form $[b, r_x^d r'_x \alpha *]$, where $\alpha *$ stands for something depending only on α-type variables. Now given a testable symbolic polynomial $\tilde{P} = \sum_x \tilde{P}_x$, we can write its $\mathcal{U}$ part as $\sum_x r_x^d \alpha *_x$ (which defines the allowable family). Since $\{r_x^d : x \in X\}$ is a linearly independent set over the polynomial ring of $\{\alpha_j\}$, we must have $\alpha *_x = 0 \ \forall x$. $\alpha *_x = 0$ means $\tilde{P}_x \sim c_x \tilde{E}_x$, where $\tilde{E}_x$ means the symbolic polynomial of the Evaluation algorithm on input x. □

To validate the second condition for VBB security, we need a more notion. We say that two partitions of a set are a straddling set [11] if no subset admits subpartition of the two partitions. There exists a straddling set consisting of 2-partition, so given any c-partition of a set X, there exists a c-partition to pair it into a straddling set for $c \geq 2$.

Choose the partition of the patching to be the partner of the partition of some primary component to form a straddling set. Choose an monomial of this primary with maximal degree. This allows us to prove the VBB security: To complete this monomial with any number of inputs, any symbolic circuit must use at least the same number of "addition" gates, and by inspecting these gates, all x with $c_x \neq 0$ can be identified. This leads to the first main theorem of this work:

Theorem 1. *The obfuscator above is VBB secure in the ideal GES model which obfuscates d-functions of the locality c with $\mu = (d+1)n/c$.*

Next we give some variations of the basic obfuscator.

- The case where g is keyed, and has degree 1 over the key bits: encodes each key bit in ElGamal form with a common ElGamal scalar $S_0 = [1, r]_w$, and in the patchings let $1 = \prod_i r_i$, $0 = \sum_i w_i$.
- For $d > 1$, no patchings: let the levels $\{v_{i,b} : i, b\}$ lie in a subgroup of exponent d, i.e. satisfying $dv_{i,b} = 0 \forall i, b$; the zero-test encoding changes to

$$B_i S_0 - \mathrm{CHK}_i S_x^d$$

if g is keyless, or

$$B_i - \text{CHK}_i S_x^d$$

if g is keyed as above.

- Twisting ElGamal form: in the ElGamal values of the initial encodings, v_j is replaced with v'_j, and again the relations on these generators are defined by the additions of encodings.
- The conjunction function case: no ElGamal scalars and only 1 output bit. Suppose the check value encode β in the second slot, choose a new $c-$partition $X = \sum_i Y_i$, the original ElGamal scalars and check value are replaced by

$$\left\{ S_{x_i} = [1, r_{y_i} \beta_i]_{v'_i + v_{y_i}} : y_i \in Y_i, i \right\}$$

where $\beta = \prod_i \beta_i$; and the zero-test encoding changes to

$$(\prod_j T_{x_j} - \prod_i T_{y_i}) \prod_i P_{x_i}$$

The security of these variations remains unchanged except the no-patching version. The security proof goes through if there is a pair of primary components whose partitions form a straddling set: choose the maximal monomial given by a product $\prod \alpha_j$ over all blocks in the straddling set that appears (divides some monomial) in the check values. In this case, we get $\mu = dn/c$.

4 The Garbled Circuit *gc*

A garbled circuit $gc(k, x, r)$ is an encryption of a circuit $C(k, \cdot)$ and input x that only reveals $C(k, x)$. Most constructions only concern the efficiency requirement that $gc(k, x, r)$ be a local function over (C, x), and thus leave $gc(k, x, r)$ a complex function over r. In this work we additionally require that it be of smallest possible algebraic degree d over the random bits r. There is a construction [9] which achieves $d = 4$ assuming existence of quadratic minimal PRG (stretching the input by only 1 bit), we will improve the result to $d = 3$ under the same assumption. Moreover using our candidate PRGs with linear stretch, we also reduce the output size and random tape length.

The only crypto-ingredient of the construction is one-time encryption of arbitrarily long message, where the security is defined as that the ciphertext is pseudorandom for only one encryption. This primitive $E_k()$ can be realized by any PRG g as follows:

$$E_k(m, \{r_i\}) = \big(g(k) \oplus r_1, g(r_1) \oplus r_2, \cdots, g(r_s) \oplus m\big)$$

that is, the key is repeatedly refreshed and stretched until it is enough to cover the long message. Note that the ciphertext is quadratic over key and random bits if so is g, and affine over m. Its size decreases with larger stretch of the PRG, can be the same as $|m|$ if g has enough stretch.

We also use four functions $\{\mathrm{CK}_{a,b} : a, b \in \{0,1\}\}$ over two strings:

$$\mathrm{CK}_{a,b}(W_0^0||W_0^1, W_1^0||W_1^1) = W_0^b \oplus W_1^a$$

where $||$ means string catenation, that is, each function choose parts of the input to add up. They satisfy the property: for 4 independent strings

$$\{W_{a,b} : a, b \in \{0,1\}\},\ \{\mathrm{CK}_{a,b}(W_{0,a}, W_{1,b}) : a, b \in \{0,1\}\}$$

are independent.

The construction of GC in [9] encrypts each gate independently but shares some random bits among some gates. The value v_i of a gate i is encrypted by a random bit e_i, and the ciphertext $b_i = v_i \oplus e_i$ is revealed along the computation. There are 4 strings $\{W_a^b : a, b \in \{0,1\}\}$, they stand for values of the two inwires. The encryption of gate i is a table

$$\big\{E_{\mathrm{CK}_{a\oplus e_0, b\oplus e_1}(W_0^{a\oplus e_0}, W_1^{b\oplus e_1})}(T_{g_{a,b}\oplus e_i}||g_{a,b} \oplus e_i) : a, b = 0, 1\big\}$$

where e_0, e_1 stand for the key bits of the two inwires, $g_{a,b}$ is the gate definition, and T_b is the catenation of the corresponding W^b of all next stop gates. At output gate i, $e_i = 0$; and an input gate is given by a T_b. Gate i can be evaluated as follows: suppose two inwire values are $(W_0, a), (W_1, b)$, we can use key $\mathrm{CK}_{a,b}(W_0, W_1)$ to decrypt the (a, b) entry of the table to provide the input needed by next stop gates.

Note that the encryption key $\mathrm{CK}_{a\oplus e_0, b\oplus e_1}(W_0^{a\oplus e_0}, W_1^{b\oplus e_1})$ is quadratic over the random bits W, e, this makes the GC have degree 4. We modify the table as

$$\big\{\mathrm{CK}_{a,b}(E_{W_0^a}(0), E_{W_1^b}(0)) \oplus (T_{g_{a,b}\oplus e_i}||\mathrm{MARK}) : a, b = 0, 1\big\}.$$

That is, we do not make use of $b_i = v_i \oplus e_i$ in decryption, and use a constant string MARK of length λ to allow correct decryption. At output gate, MARK is replaced by True or False to indicate the output value. Now the degree is given by the term $T_{g_{a,b}\oplus e_i}$, which is 3. Note that the output size and random tape length is also reduced compared to the original construction: the length of T_b is cut to half. Note that if a gate has large fan-out, then T_b is long and tabling 4 such elements is a waste. Additional improvement to address this issue is as follows: at gate i with fan-out > 2, two additional keys $R_b : b = 0, 1$ are included. In the table, T_b is replaced with R_b, and a table $\{E_{R_b}(T_{b+e_i}||\mathrm{MARK}) : b = 0, 1\}$ is added.

To see this GC is secure, we give a specific simulator S as follows. S has $b_i, W_0^{b_0}, W_1^{b_1}, (R_{b_i})$ as part of its random tape at gate i, we treat these as the same as in GC, where the gate values are defined by a random fixed key, and the simulator compute the part of the GC output defined by these quantities and leave the other part random. This simulator works as long as E_k is secure. We say $\{e_i, W_0^{b_0}, W_1^{b_1}(R_{b_i}) : i\}$ is the part of S's random tape shared with GC.

To get concrete instantiation, we set the key length of one-time encryption to 2λ, and choose the PRG to be our candidate PRG. For the most optimistic choice, we assume the one-time encryption does not need additional random bits (the stretch of the PRG is enough), then it is not hard to formulate that output size of GC is no more than $20\lambda|C|$ and random tape length is no more than $10\lambda|C|$.

5 The Pseudorandom Function

Traditional constructions of PRFs [28, 29, 24, 10, 34] are too complex to be obfuscated directly, so we want to realize PRFs in d-functions (constant d). Since natural arithmetic circuits are difficult to be expressed in d-functions, there seems little hope to base the security on well known problems. Instead we follow the empirical approach of applied cryptography: design and analysis. We give a candidate construction and apply the distinguishing attacks to it.

Note that the class of ideal functions is closed under $\oplus$ and under permutations on input variables, we consider the following huge key and parallel components strategy:

$$F_d(k, x) = g_d\big(\{f(k_i, \tau_i x) : 1 \le i \le td\}\big)$$

where f is a primary subcircuit, τ_is are known random permutations on components of input x, k_is are independent subkeys, and g_d is a fixed simple function over td variables: the sum of t terms each of which is a product of different d variables, seen as elements of $\mathbb{GF}_{2^l}$. Note that F_1 is an ideal function, and F_d is a d-function.

A reference candidate for f can be simply described as follows. The secret key for f is n/c PRFs (given as n/c tables each of 2^c entries) on subblocks $\{X_i : i\}$: $f_{X_i} : X_i \longrightarrow \{0,1\}^l$ where $X = \sum_i X_i$ is a c-partition. We inductively merge functions over blocks to form functions over larger blocks:

$$f_{U+V} = f_U f_V + 1,$$

where the multiplication is in the finite field $\mathbb{GF}_{2^l}$ and we index each function with its block name. Except for the last block U whereas it should be kept linear

$$f_{U+V} = f_U(f_V + 1)$$

at each round of merges. The inductive steps go in parallel rounds: each round gets (nearly) double-sized blocks and half number of blocks (the last block is not merged when the number of blocks is odd in that round). Finally we get $f = f_X$ defined on the whole X. If we let $l = \lambda$, it is seen that ideal functions may not be in NC^1.

5.1 Security Analysis

Next is some cryptanalysis of F_d, for some parameter choice of t, l, d. Unlike the weak PRF of [1], F_d is too complex for linear approximations:$t = \lambda/2$ seems to make F_1 immune even for $l = 1$. Similar is the case for learning attacks: F_d

has exponentially many terms. By intuition we suppose that t acts as the round number of block ciphers, l stops linear distinguishing attacks, and d enhance the nonlinearity. We only consider distinguishing attacks.

First consider the case $d = 1$. The best linear distinguishing attack seems to be the Cube Attack. Observe that

$$D_{A+B}(f_U f_V) = (D_A f_U)(D_B f_V)$$

for $A \subset U$, $B \subset V$, so

$$D_A f = h \prod_{A_i \neq \emptyset} D_{A_i} f_{X_i}$$

for some h, where $A_i = X_i \cap A$, which is 0 with probability about $k/2^l$ ($k << \lambda$ is number of terms in the product), when $k << 2^l$, and translates to that the bias of each bit is about $(\ln k)/l < (\ln \lambda)/l$, which suggests that the F_1 stops linear undistinguishing attack when $l \geq 2 \log \lambda$ and $t = \lambda/2$.

Now we consider the nonlinear distinguishing attack. If $t = 1$, we partition $X = A + B + A'$ as $A = U, A' = V$ if $f = f_X$ is the merge of f_U, f_V, and we get $\omega(f) = 1$. This can be generalized to $t = o(\log n)$ with a partition $X = A+B+A'$ where $|A| = |A'| > \log t$ and $\omega(F_1) = t$. When $t \geq \lambda/2 >> \log n$, we guess that any partition strategy would make $\omega(F_1)$ a sum of some $\omega(f)$s with respect to random partitions. So it is reasonable to estimate $\omega(f)$ with respect to a random partition. Note that

$$\omega(f_U) \leq 2^{\min(|U \cap A|, |U \cap A'|)}$$

for any U. Also note that $\omega(gh) \leq \omega(g)\omega(h)$ for any function g, h. We see that

$$\log \omega(D_V f) \leq |A| - \frac{\varDelta}{2}$$

where $V \subset B$ is some set for Cube attack, where $\varDelta$ is the sum of differences $|(|X_i \cap A| - |X_i \cap A'|)|$ such that $V \cap X_i \neq \emptyset$; which is in general too big to be compensated by the number of twisted copies and indicates a distinguishing attack. Note that such a complex distinguishing attack has little hope to be exploited in attacking RE.

Now we consider the case $d = 2$. Let us fix a $g = \sum_{1 \leq i \leq \tau/2} f_i g_i$, where f_i, g_i are independent copies of f. First we show that Cube attack works much harder now. Let $h = f_0 g_0$ be a product of two bits of two copies of f. Let D denote a D_V with $|V| = 1$. We should ensure that $\Pr[D(h) \neq 0] \geq 1/4$. Note that Df_0 is a product of $\log(n/c)$ random variables in $\mathbb{GF}_{2^l}$. We assume each random variable in $\mathbb{GF}_{2^l}$ vanishes with probability 2^{-l}. So we have

$$\rho = \Pr[Df \neq 0] = \left(1 - 2^{-l}\right)^{\log(n/c)} \approx 1 - \log(n/c)/2^l$$

and Df_0 as a component of Df does not vanish with probability $1 - (1/\rho)^{1/l} \approx 1 - (\log(n/c))^{1/l}/2$. By the following

Lemma 5. *Let a, b be two independent random boolean variable with $\Pr[a = 1] = \Pr[b = 1] = p\delta$, then $\Pr[a \oplus b = 1] \geq p$ if $p \leq \frac{1}{4}$, $\delta \geq 2 - 2^{\frac{1}{2}}$*

Proof. $\Pr[a \oplus b = 1] = 2(p\delta)(1-p\delta) \geq (2\delta)(1-(1/4)\delta)p$, and $2\delta(1-(1/4)\delta) \geq 1$ when $\delta \geq 2 - 2^{\frac{1}{2}}$. □

Assuming that $D(f_0 g_0) = f_0 D g_0 + g_0 D f_0$ is a sum of two independent variables, to ensure $\Pr[D(h) = 1] \geq \frac{1}{4}$, it is enough that

$$2 - (\log(n/c))^{1/l} \geq 2 - 2^{\frac{1}{2}}$$

that is $l \geq \log\log(n/c)$. For $|V| > 1$, D_V can be composed by D and a lower $|V|$, and the above arguments go through. This suggests that Cube attack can hardly go through when $l \geq \log\log(n/c)$.

Now we consider the nonlinear distinguishing attack when $d = 2$. Since the operation D_V is not useful in this case (exponentially increases the number of terms), we would directly estimate $\omega(g_0 f_0)$s as $\omega(f)^2$. We expect that $\log \omega(f) \geq \frac{|A|}{2}$ for a random partition to have a marginal security. We assume $\omega(f_{X_i}) = 2^{\min(|X_i \cap A|, |X_i \cap A'|)}$. We first estimate

$$\varDelta = \sum_i \big|(|X_i \cap A| - |X_i \cap A'|)\big|.$$

Let $h = |A|$, z be a random variable with $\Pr[z = 1] = \frac{1}{d'}$, $\Pr[z = 0] = 1 - \frac{1}{d'}$, where $d' = n/c$ is the number of blocks. We can model $|X_i \cap A|$, $|X_i \cap A'|$ as sum of h independent copies of z. Then the expected value of $(|X_i \cap A| - |X_i \cap A'|)^2$ is

$$2\left(\frac{h^2 - h}{d'^2} + \frac{h}{d'} - \frac{h^2}{d'^2}\right) = \frac{2(d'-1)h}{d'^2},$$

so the expected value of $\varDelta$ is

$$d'\big(\frac{2(d'-1)h}{d'^2}\big)^{\frac{1}{2}} = (2(d'-1)h)^{\frac{1}{2}},$$

which is $\leq h$ when $h \geq 2(d'-1)$. This means that one cannot apply the nonlinear distinguishing attack with $h \geq 2(d'-1)$. For very small d', this is already enough. For $h < 2(d'-1)$, each $|X_i \cap A|$ is expected to be < 2, and the lower terms of the merge operations give a significant additional number of terms compared to the simple product, we don't know how to estimate $\omega(f)$. But we guess that the nonlinear distinguishing attack cannot be mounted in this case either. There are much more complex 2-functions than our example, so we guess that "hard to attack" PRFs exists among 2-functions.

In summary, we have given a candidate d-function which seems suitable to replace the PRF in RE for $d = 1, 2$. A more conservative choice would be d-functions of larger d. Observe that a weaker PRF does not mean an insecure RE, so we believe that our 2−function PRF is secure in our application in RE. A more aggressive candidate RE is the composition of our candidate GC gc and F_1 (with $l = \log n$ and $t = \lambda/2$). We define a simulator for this RE to be the composition of the simulator of the gc and F_1, and leave it as a challenge to invalidate this simulator.

6 The Obfuscator for General Circuits

Our general obfuscator is an application of our core obfuscator (with no-patching, twisting ElGamal form and keyed g) to the RE circuit $gc(C(k,X),X,\mathrm{PRF}'(X)$ of the circuit $C(k,X)$ (to be obfuscated). Recall that the garbled circuit gc has a long random tape, which should be provided by a PRF' on input space X. Our PRF has output length l which is small. We need a set $T=\{0,1\}^m$ and a PRF f on $T\times X$ such that $l|T|$ is no less than the length of gc 's random tape, and PRF' is formed by evaluating $f(t,x)$ for all $t\in S\subset T$, where S is a specified set: each element $t\in S$ specifies a position where $f(t,x)$ lies in $\mathrm{PRF}'(x)$. The main problems and solutions (modifications of the core obfuscator) are as follows (assume f is a d-function, and $\deg(gc)=d'$):

- To guarantee no computations other than honest evaluation can be done: Each block of f has the form $T_j\times X_j$, the α type variable should depend on $t_j\in T_j$. Therefore the encodings of this block are

  $$\Big\{\big([1,r_{x_j}r_{t_j}]_{v_j+v_{t_j}+v_{x_j}},[f_j(k,t_j,x_j),r_{x_j}r_{t_j}\alpha_{j,t_j}]_{v'_j+v_{t_j}+v_{x_j}}\big):t_j\in T_j,x_j\in X_j\Big\}$$

  At (except for one) the last blocks of the twisted copies of the primary subcircuit, the ElGamal Scalars are not given. Therefore there is a unique $S_{t,x}$ for any (t,x).
- To make all output bits of PRF' have the same ElGamal form: there is a new block T with the following encodings: $\{P_t=[1,r_t^{-d}]_{w_T-v_t}:t\in S\}$, and the ElGamal encodings of the output bits of $f(t,x)$ should be multiplied by P_t before going into gc.
- To block illegal inputs at some output bits: we should prevent $(x)_i=0$ at the output bits for input of the garbled circuit that refers to $(x)_i=1$. This is solved by a straddling of the corresponding Check value and Patching value: the Check value at this position is replaced by $\mathrm{CHK}=[0,r^{-d'd}_{i,(x)_i rp^{d'}(\{\alpha\})}]_w$, and a new Patching $P'_t=[1,r^{-d}_{i,(x)_i}r_t^{-d}]_{w_T-v_t}$ is added. The zero-test encoding at this position is

  $$(P'_tB_i)^{d'}-(S^d_{t,x}P_t)^{d'}(\mathrm{CHK})$$

  where B_i stands for , where the gate values are defined by a random fixed key the encoding of the output value. If the input changes $(x)_i$, then the Check value is useless, and the computation gets nothing.

The proofs of correctness and security of the core obfuscator can be routinely repeated on this general obfuscator. This results in the main theorem of this work:

Theorem 2. *Assume that there exist d-function PRFs for constant d, we have a general obfuscator with VBB security using an ideal GES of degree $\mu=O(n)$, where n is number of input bits of the circuit to be obfuscated.*

The remarkable side of this theorem is that it is easy to be instantiated: the building blocks PRG and PRF are the most elementary primitives and have potential to be optimized; instantiation of the ideal GES seems reasonable with known tools. We will first define what is an instantiation of an ideal GES, just called GES.

A real world GES has the same interfaces as those of an ideal GES, and should be stateless, and the public oracles of the ideal GES should be realized with public algorithms. The encodings are no longer symbolic, they are samples from predetermined distributions. These distributions are indexed in the same way as in the ideal case. The correctness requirements are: encodings with different contents at the same level have disjoint support, the operators map given samples to samples of distributions specified by the ideal GES. The security of GES is not so meaningful to define: on one side, the security guarantee of ideal GES is impossible to totally achieve, on the other side, the objective (VBB security) is hard to define in a easy-to-prove way. So we just put the easy-to-invalidate definition of VBB into that of GES.

Definition 8 (Security of GES). *Given any Legal set of initial encodings $ob = \{[b_i, u_i]_{v_i} : 1 \leq i \leq l\}$, where $u_i = U_i(\{\beta\})$ is obtained by evaluating polynomial U_i on the set (common to all i) of independent variables, and for any predicate p on $\{b_i\}$, there exist a set $\{\tilde{P}_j\}$ of polynomially many admissible ($P_j(\{U_i\}) = 0$, and level-respect) symbolic polynomials, and simulator S, such that for any adversary A:*

$$\Pr[A(ob) = p] \leq \Pr[S(\{\tilde{P}_j(\{b_i\})\}) = p].$$

Lemma 6. *Our obfuscators (core and general) are VBB secure when the ideal GES is replaced by a secure GES if the obfuscated code is Legal.*

Proof. The only admissible polynomials are honest evaluations. This means any adversary gets information no more than the black box. □

What is Legal is specific to the GES tool. For our simplified CLT map, known attacks only show that this concept depends on the set $\mathcal{P} = \{P_j : j\}$ of admissible symbolic polynomials ($P_j(\{U_i\}) = 0$, and level-respect) over initial encodings, where Legal requires the set of admissible symbolic polynomials over the set of initial encodings does not admit any t-form for $t \geq m$, where m is a parameter of the CLT map. A t-form of the set $\mathcal{P} = \{P_j : j\}$ is matrix $\{P^b_{i,j} \in \mathcal{P} : i, j \in [tm], b = 0, 1\}$, and there exist $\{Q^b_k : k \in [t], b = 0, 1\}$, $\{S_{i,k} : i, k\}$, $\{T_{k,j} : k, j\}$; such that

$$P^b_{i,j} = \sum_k S_{i,k} Q^b_k T_{k,j}.$$

Note that in our obfuscators, the simplest admissible polynomial is associated with a single input x, the fixed polynomials $\{Q_k : k \in [t]\}$ means part of the input variables are fixed. So a $[t]$ form gives a partition of X. See [18] for details of the attack given such a t-form.

We conjecture that excluding of t-forms (for polynomially large t) in the set of admissible symbolic polynomials is also sufficient for the Legal definition.

7 Instantiations

7.1 The Simplified CLT Map

The only known instantiation of GES that we need is obtained from the following simplified version of the CLT multilinear map of [19], where a symbolic encoding is just an element of a huge ring, and symbolic operations are just ring operations. Let $N = \prod_{1\leq i\leq m} p_i$, where p_is are prime integers of η bits. $\mathcal{G} = \prod_{1<i\leq m} \mathbb{Z}_{g_i}$, g_is are random small primes of $2\log(\lambda+\mu)$ bits (a small number to make $\mathcal{G}$ universal for polynomials of degree no larger than μ, and so some g_is may be the same). A level v corresponds to an element $z_v \in \mathbb{Z}_N$. Let ρ be the unit size of noise. Then

$$\text{Encode}(sk, b, \alpha, v, d) = [b, \alpha]_v(d) = z_v^{-1}\text{CRT}_{(p_i)}(b{+}2s_1, a_2{+}s_2g_2, \cdots, a_m{+}s_mg_m)$$

where $\alpha = (a_2, \cdots, a_m)$, s_is are random integers such that $-2^{d\rho} < a_i + s_ig_i < 2^{d\rho}$, and $\text{CRT}_{(p_i)}(z_i)$ denotes the number $z \in \mathbb{Z}_N$ satisfying $z = z_i \pmod{p_i}$ for all i. The map $v \mapsto z_v$ is a random injective homomorphism from $\mathcal{A}$ to the multiplicative group of $\mathbb{Z}_N$. Finally the test number

$$z_t = z_{v_{zt}}\text{CRT}_{(p_i)}\left(2^{-1}\frac{N}{p_1}, g_2^{-1}\frac{N}{p_2}, \cdots, g_m^{-1}\frac{N}{p_m}\right)$$

is used to recover the encoded boolean value of an encoding $[b, 0]_{v_{zt}}(d)$:

$$z_t[b, 0]_{v_{zt}}(d) = \sum_{1\leq i\leq m} t_i\frac{N}{p_i} + b\frac{p_1+1}{2}\frac{N}{p_1}$$

which should have disjoint distributions (easily distinguishable) between the case $b = 0$ and $b = 1$ when $d \leq \mu$. A sufficient condition for this is $(\ln\lambda)^{1/2}|\frac{t_i}{p_i}| < 1/(4m)^{1/2})$ $\forall i$. We will call this the correctness requirement.

First we we will show that

Lemma 7. *Any level group $\mathcal{V}$ can be realized in some $\mathbb{Z}_N$.*

Proof. Given any finite set $\{v_i\}$ of generators and a set of $\{\sum_i a_{i,j}v_i = 0 : j\}$ relations, by an elementary theorem [23] on structure of abelian groups, there is an efficient procedure to find another set $\{e_i : i\}$, and integers $d_1 \mid d_2 \mid \cdots \mid d_l$ such that

- $v_i = \sum_j b_{i,j}e_j$
- the relation set $\{d_ie_i = 0 : 1 \leq 1 \leq l\}$ defines the same group.

So we implement the group in an $\mathbb{Z}_p$ as follows: choose prime p such that $d_l|(p{-}1)$, for each $i \in [l]$, randomly choose $r \in \mathbb{Z}_p$, let $z_{e_i} = r^{(p-1)/d_i}$; for each $i > l$, randomly choose $r \in \mathbb{Z}_p$, let $z_{e_i} = r$. Finally, let $Z_{v_i} = \prod_j z_{e_j}^{b_{i,j}}$. Do this for enough number of such p to form N, and use CRT to get z_{v_i} s in $\mathbb{Z}_N$. □

7.2 Security Analysis

The first kind of attacks [19] on the original CLT map is related to parameter sizes, which go the same way despite our modifications. The attacks exploit the same weakness: multiple encodings lie in the same level, they are called exposed. The complexity of these attacks depends on the noise size ρ, prime factor size η, and modulus size $\gamma = m\eta$. The brute force attack (guess noise) has complexity 2^{ρ}. The second attack (solving the related Approximate Common Divisor Problem [32, 33, 19]) requires that $m \geq \eta\lambda$, the detailed analysis of which is very similar to [32, Appendix B.1]) and [19, Section 5.1]. The third (to find the level factor) [19] is also stopped by the parameter choice of the second attack. None of these attacks is related to choice of $\mathcal{G}$ and our $\mathbb{Z}_2$ slot.

The second kind of attacks [18, 16, 17] (called algebraic attack, or zeroizing attack) makes use of the leak of the zero-test: a known polynomial of the initial noises (called zero-test polynomial) over the field of rational numbers is returned by any acceptable zero-test. If one can partition the input set to get an m-form of set of the zero-test polynomials, then CLT map is totally broken. In our core obfuscator, if the ElGamal scalar of the ideal function output can be obtained by different polynomials of the initial encodings, then this attack works. We can avoid this in our general obfuscator by the explicit structure of our ideal function: there is only one copy of f that has the whole-vector ElGamal scalar. Now zero-test polynomials only come from honest computation, and the polynomials factor through a PRF which make the input set not partitionable. But it generally threats simple circuits such as conjuncts [18], but it also works harder when the locality c gets larger. We will give an explicit block straddling to defend our example of conjunct ($n = 64$,$c = 8$), where with a little overhead, we also give the general method to stop algebraic attacks for any circuit.

7.3 More Optimizing

The first improvement is about the testing. Since the test vector plays the same roles as a level, we can treat it as a block level, and use the definition of the level group to absorb it in, so that the resulted final level zero-test automatically. Then we do not need a test vector any more. The second improvement only concerns the general obfuscator: let the primary subcircuit of the PRF be defined as follows. The first round of merges of the blocks does not use field multiplications, uses component wise “and not” instead. The resulted initial encodings of the PRF key bits have twisted ElGamal form, so they can have half size noises. This reduce μ by half cost-equivalently.

7.4 Reference Instantiations

We choose gc to be our candidate GC which is of degree 3. There are 2 choices for the PRF:

- F_2 with $l = \log\log n'$ and $t = \lambda/2$, which results in $\mu = 3n'/c + 2.5$.

- F_1 with $l = \log n'$ and $t = \lambda/2$, which results in $\mu = 1.5n'/c + 2.5$.

where $n' = n + \log\lambda + \log|C|$. The number of GES encodings is about $\big(20|C| + n'(l+1)2^c/2c\big)\lambda$. The second choice will be used in our AES example, where the number of GES multiplications in one evaluation is about $(20 + 5l)\lambda^2|C|$ (by tabling intermediate results, each PRF bit evaluation costs about $l\lambda/2$ multiplications).

8 Examples

8.1 Obfuscating AES

For AES, $|C|$ is smaller than usual measure: firstly all key operations are absorbed, so our obfuscation of AES is not burdened by the key scheduling part. Secondly, our gate can be any binary function, so it should be smaller than known counts after optimized by some automatic tool [15]. We did not conduct this optimization because it is not so crucial to the result: this work only shows that general obfuscation is implementable now, but still far from practical use. So a safe estimation of $|C| < 25000$ is used: each S-box costs less than 110 gates, the linear layer costs less than 600 gates, so the total is less than $10\times(16\times110+600)$. We choose $c = 13$, and get $\mu = 20.5$. We choose $l = 8$.

The remaining part is to decide the concrete parameters for the CLT map. To get a smaller η, we need give a good estimate of the noise size. The final noise is modeled as a sum of $2^{2\delta}$ products, each has length $\mu\rho$ (much smaller in fact). δ is about the bit length of this extra noise caused by addition. Therefore $\eta \geq \mu\rho + \delta + \log m + 2$ is sufficient to fulfill the correctness requirement. In our example, we have $\mu = 20.5$, $\delta < 100$, $\lambda = 128$; and we let $\rho = 114$, and $\eta \approx 2440$. The GES ring size is

$$\eta^2 \times 128 \approx 7.8 \times 10^8.$$

The code size of the obfuscated AES is thus about

$$(20 \times 25000 + 2^{13} \times 12 \times 9) \times 128 \times 7.8 \times 10^8 < 1.5 \times 10^{17} \text{ bits}.$$

To evaluate the obfuscated AES once would cost about

$$(20 + 5 \times 8) \times 25000 \times 128^2 \approx 2.5 \times 10^{10}$$

modular multiplications with 7.8×10^8 bits modulus, which is still far from practical.

8.2 Conjunction

We have $n = \lambda$ and $\mu = n/c$. For $n = 64, c = 8$, the GES ring size is about $(58 \times 8)^2 \times 64 < 1.5 \times 10^7$. The obfuscated code size is $3 \times 2^8 \times 8 \approx 6000$ GES encodings or about 9×10^{10} bits. The evaluation only costs 22 multiplications under 1.5×10^7 bits modulus. This can be regarded practical, while the latest implementation [20] only made it for $n = 32$, but its security is based on standard assumptions.

We will illustrate with this example how to defend against algebraic attacks. Recall the CLT map entities: $\{g_i, p_i : i\}$. A noise is of the form $z = \{z_i : 1 \leq i \leq m\}$, seen as a vector. The initial encodings encode the following sets of noises:

- At ElGamal scalars, $[64] = A_1 + A_2 + \cdots + A_8$ is an 8-partition: $\{z_{A_j, x_{A_j}} : x_{A_j} \in \{0,1\}^8\}$ for $1 \leq j \leq 8$.
- At ElGamal values, $[64] = A'_1 + A'_2 + \cdots + A'_8$ is an 8-partition: $\{z_{A'_j, x_{A'_j}} : x_{A'_j} \in \{0,1\}^8\}$ for $1 \leq j \leq 8$.
- At Patchings, $[64] = B_1 + B_2 + \cdots + B_8$ is an 8-partition: $\{z_{B_j, x_{B_j}} : x_{B_j} \in \{0,1\}^8\}$ for $1 \leq j \leq 8$.

The zero test on input x will return a polynomial over the secret noises:

$$\left(\left(\prod_j z_{A_j, x_{A_j}} - \prod_j z_{A'_j, x_{A'_j}} \right) \prod_j z_{B_j, x_{B_j}} \right) \cdot \left(\left\{ \frac{1}{g_i p_i} : i \right\} \right)$$

where multiplication is component wise, and "$\cdot$" is inner product. The algebraic attack is to find a partition $[64] = V + V' + W$, such that $|V| = |V'| \geq \log 2m \geq 13$, and for each of the partitions $A, , A', B$, no block intersect V, V' at the same time. If this can be done, then the zero-test polynomial can be seen as a bilinear form of dimension $2m$ with respect to quantities varying with variable in V, V' respectively, so that the attack of [18] works. The following block straddling make this impossible: $[64] \sim \mathbb{Z}_8^2 = \{(a, b, c) : a + b + c = 0\}$, the partitions $A, , A', B$ correspond the coordinates a, b, c respectively (a partition corresponding a coordinate means elements of a block have the same coordinate value). The verification can be done by excluding the cases for $t = 1, 2, 3$ respectively, where t is the minimal number of blocks of a partition that intersects V. If V intersects each coordinate in 4, then the only solution for V's coordinate sets are $(e, e, e), (o, o, e)$ where e, o stands for even or odd respectively. But the solution for V' does not exist in these cases.

In general, algebraic attacks can be stopped by sharing the Check value in l additive shares, by straddling these l partitions (randomly) it is easy to defeat any input set partitions if l is large enough.

9 Concluding Remarks

We have made general obfuscation implementable, but still far from practical. Now the degree of the multilinear map is not the main barrier, the huge number of modular multiplications (resulted from the RE circuit) of big modulus (caused by the CLT map) is more prohibitive. The lack of efficient GES instantiation is the main obstacle for making obfuscation practical.

9.1 Acknowledgements

We thank anonymous referees for pointing to us the nonlinear distinguishing attack, the work of [9] and valuable suggestions on presentations.

References

1. Akavia, A., Bogdanov, A., Guo, S., Kamath, A., Rosen, A.: Candidate weak pseudorandom functions in AC0 o MOD2. Electronic Colloquium on Computational Complexity (ECCC) **21** (2014) 33
2. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive **2013** (2013) 689
3. Ananth, P., Jain, A., Sahai, A.: Robust transforming combiners from indistinguishability obfuscation to functional encryption. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. (2017) 91–121
4. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. (2017) 152–181
5. Ananth, P.V., Gupta, D., Ishai, Y., Sahai, A.: Optimizing obfuscation: Avoiding barrington's theorem. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014. (2014) 646–658
6. Applebaum, B.: Bootstrapping obfuscators via fast pseudorandom functions. In: Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. (2014) 162–172
7. Applebaum, B., Bogdanov, A., Rosen, A.: A dichotomy for local small-bias generators. J. Cryptology **29**(3) (2016) 577–596
8. Applebaum, B., Brakerski, Z.: Obfuscating circuits via composite-order graded encoding. In: Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II. (2015) 528–556
9. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. In: 20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA. (2005) 260–274
10. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. (2012) 719–737
11. Barak, B., Garg, S., Kalai, Y.T., Paneth, O., Sahai, A.: Protecting obfuscation against algebraic attacks. In: Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings. (2014) 221–238

12. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. (2001) 1–18
13. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the 2012 ACM conference on Computer and communications security, ACM (2012) 784–796
14. Boneh, D., Ishai, Y., Sahai, A., Wu, D.J.: Lattice-based SNARGs and their application to more efficient obfuscation. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III. (2017) 247–277
15. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings. (2010) 178–189
16. Cheon, J.H., Fouque, P., Lee, C., Minaud, B., Ryu, H.: Cryptanalysis of the new CLT multilinear map over the integers. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. (2016) 509–536
17. Cheon, J.H., Han, K., Lee, C., Ryu, H., Stehlé, D.: Cryptanalysis of the multilinear map over the integers. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. (2015) 3–12
18. Coron, J., Lee, M.S., Lepoint, T., Tibouchi, M.: Zeroizing attacks on indistinguishability obfuscation over CLT13. In: Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I. (2017) 41–58
19. Coron, J., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. (2013) 476–493
20. Cousins, D.B., Crescenzo, G.D., Gür, K.D., King, K., Polyakov, Y., Rohloff, K., Ryan, G.W., Savaş, E.: Implementing conjunction obfuscation under entropic ring LWE. Cryptology ePrint Archive, Report 2017/844 (2017) `http://eprint.iacr.org/2017/844`.
21. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. (2013) 1–17
22. Halevi, S., Halevi, T., Shoup, V., Stephens-Davidowitz, N.: Implementing BP-obfuscation using graph-induced encoding. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017. (2017) 783–798
23. Lang, S.: Algebra (3. ed.). Addison-Wesley (1993)
24. Lewko, A.B., Waters, B.: Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009. (2009) 112–120

25. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 prgs. In: Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. (2017) 599–629
26. Lin, H., Tessaro, S.: Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In: Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. (2017) 630–660
27. Mossel, E., Shpilka, A., Trevisan, L.: On ϵ-biased generators in NC0. In: 44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings. (2003) 136–145
28. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997. (1997) 458–467
29. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring. Electronic Colloquium on Computational Complexity (ECCC) **8**(064) (2001)
30. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. In: Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing. STOC '14, New York, NY, USA, ACM (2014) 475–484
31. Sahai, A., Zhandry, M.: Obfuscating low-rank matrix branching programs. IACR Cryptology ePrint Archive **2014** (2014) 773
32. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2009/616 (2009) `http://eprint.iacr.org/2009/616`.
33. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. (2010) 24–43
34. Yu, Y., Steinberger, J.P.: Pseudorandom functions in almost constant depth from low-noise LPN. In: Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II. (2016) 154–183
35. Zimmerman, J.: How to obfuscate programs directly. In: Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II. (2015) 439–467