Towards Practical Obfuscation of General Circuits

Dingfeng Ye[1,2,3], Peng Liu[4], and Jun Xu[1,2,3]

[1] School of Cyber Security, University of Chinese Academy of Sciences, China
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[3] Data Assurance and Communications Security Research Center, Chinese Academy of Sciences, Beijing 100093, China
[4] Cyber Security Lab, College of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802, USA
ydf@is.ac.cn,pliu@ist.psu.edu,xujun@iie.ac.cn

Abstract. Known approaches for obfuscating a circuit are only feasible in theory: the complexity polynomially depends on the security parameter and circuit measures, but with too large polynomials and/or holds only with large enough security parameters, which leaves the methods not implementable for almost all applications at a required security level, say 128 bits. In this work, we initiate the task of exploiting ideas from theoretical constructions towards practical obfuscation. The starting concern is: how much do empirical methods help to improve efficiency? We followed the approach of Zimmerman and Applebaum et al.: obfuscating the randomized encodings (RE) with Graded Encoding Scheme (GES) over composites. We gave a new design of RE which is based on a new pseudorandom function and a new garbled circuit from a pseudorandom generator, whose obfuscation only needs GES of degree linear with n, the number of input variables. We also developed various techniques that further reduce the degree by a significant constant factor. These resulted a general obfuscator with code size $\left((28\lambda|C| + \frac{2^c}{c})10n\lambda\right)\mathrm{GES}(\frac{5n}{2c} + 6, \lambda)$, where $\mathrm{GES}(\mu, \lambda)$ denotes the size of a single ring element of the Graded Encoding Scheme with multilinearity μ and security level λ. Based on our implementation of the required GES with a simplified CLT multilinear map, we may assume $\mathrm{GES}(\mu, \lambda) \approx \mu^2\lambda$. When $n = 128$, we may get $\mu = 31$; for example, our obfuscated AES will have code size $< 10^{14}$ bits, whereas no implementable solution is known prior to this work. Our construction achieves VBB security if our pseudorandom function and pseudorandom generator and application of the CLT multilinear map are all secure.

Keywords: Obfuscation, randomized encoding, graded encoding scheme

1 Introduction

Obfuscation of general circuits is a powerful functionality of cryptography [26] which was regarded infeasible [10] until the celebrated work [17]. But this great work only addressed the existence problem: the solution is totally impractical for any instantiation of meaningful examples. Significant improvement of efficiency was obtained in later constructions (e.g. [1, 31, 4, 27, 7, 20, 3, 18, 2, 13, 21]), but all these works only got that the complexity polynomially depends on the security parameter and circuit measures: the polynomial (even the degree) is not explicitly given. Here is the reason: known essential obfuscation methods, called core obfuscator, can handle only in theory 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 later has huge size (though polynomial) of complexity measure μ (the multilinearity) as input to the complexity polynomial of the core obfuscator. All known core obfuscators use a mathematical tool called multilinear map which is highly inefficient according to known constructions [17, 16]. A multilinear map noisily encodes integers in a huge ring. The key complexity measure $\mathrm{GES}(\mu, \lambda)$ is the size of a single ring element which depends on multilinearity μ it supports and the security parameter λ. In current constructions, $\mathrm{GES}(\mu, \lambda)$ is at least $\mu^2\lambda$.

There are 3 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 though μ can be reduced to 2 [20]. The third is the approach of Zimmerman and Applebaum et al. [31, 5, 7]: obfuscating the Randomized Encodings (RE) with Graded Encoding Scheme over Composites: the existing (implicit) constructions of RE have μ of size polynomial in n, λ which is too huge to be considered, but this seems the only bottleneck of efficiency.

Given that obfuscation is so wanted and feasible only in theory with current theoretical approach, it is worthwhile to ask whether there are empirical methods that make it much more efficient and even practical for specific applications. A related question is: are there special cases which may be treated without using the general methods? The answer to the later question is relatively well understood: the simplest cases are Point Functions or Evasive Families where oracle access of functionality gives essentially no information, and even in this case some of the general methods have to be used and no practical solutions are known. As for the former question, there is no totally new method that seems to work, which makes us believe that the tools of current theoretical constructions can not be totally avoided, especially the common tool: multilinear map (abstracted as Graded Encoding Scheme). So the approach we would like to try is to replace provably secure components with empirical solutions in existing constructions. In this work, we follow the approach of Zimmerman and Applebaum et al. by giving a new design of RE to get around the efficiency bariers.

In this work, RE will mean the composition of a circuit garbling algorithm gc and a pseudorandom function (PF) f (whereas it means just the circuit garbling algorithm in some literatures). A circuit garbling algorithm gc satisfies: $D(gc(C,x,r)) = C(x)$ and $gc(C,x,r)$ leaks no information other than $C(x)$, if r is independently chosen for each x. Our RE will be the circuit that computes $gc(C,x,f(x))$ given the input x.

Known constructions of gc [29, 11] and PF f [24, 25, 19, 9, 30] will not work for an obfuscable RE. The core obfuscator [31, 7] usually works as follows: first transform the binary circuit to arithmetic (which usually increases the size), then apply the multilinear map on encoded inputs 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 of the same level, 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 a huge μ in general. So our first choice is a GES with a slot that encode binary values directly so that the binary to arithmetic transformation is avoided. Now even all addition gates were made free (which is far from the reality), the circuits of RE with known constructions of its components still have too enormous μ. The theoretical constructions of gc [12, 8] and PF f [24, 25, 19, 9] use extensive integer operations which have high algebraic degree when expressed in binary circuits. The empirical constructions in the traditional symmetric cryptographic world usually have many rounds of iterations. Known theoretical and emprical constructions of gc and PF will result in a μ at a size of a high degree polynomial of n. Thus, the design of obfuscation friendly gc and PF has not been addressed in previous work.

Our construction of gc is based on pseudorandom generators which can be heuristically realized by quadratic functions according to [23, 6]. To hide the value of inwires of a gate, some permutations on 2 or 4 elements are needed. This makes the total algebraic degree of our gc be 5, so the multiplicative factor it contributes to the multilinear degree μ is at least 5. Our construction of PF is inductive: a PF on $X \times Y$ is a quadratic function on PFs over X and Y. If we begin with spaces of c bits, then a PF on $\{0,1\}^n$ can be computed by an algebraic function of degree $\frac{n}{c}$.

The heuristic security assumptions behind our RE construction can be stated as follows. A function $pg : \{0,1\}^n \to \{0,1\}^m$ is called a (n,m) Pseudorandom Generator (PG), if $pg(K)$ is pseudorandom when K is random.

Conjecture 1. *There exsits quadratic* $(2\lambda, 6\lambda)$ PG *(each output bit is a quadratic function of input bits).*

A Pseudorandom Function (PF) $F : X \times K \to \mathcal{R} : (x,k) \mapsto F(x,k) \in \mathcal{R}$ satisfies: when k is random, $F(x,k)$s are independently pseudorandom in the range $\mathcal{R}$ for different xs. When $\mathcal{R} = \{0,1\}^m$, we call such a PF a (X,m) PF.

Conjecture 2. *Let* $F = (f_1, f_2, \cdots, f_t)$, $G = (g_1, g_2, \cdots, g_t)$ *be* (X,t), (Y,t) PF*s respectively, then* $\sum_{1\le i\le t} f_i g_i$ *is* $(X \times Y, 1)$ PF *if* $t \ge 2\lambda$.

Conjecture 3. *Let* $F = (f_1, f_2, \cdots, f_t)$*. If any linear combination of* f_i*s is* $(X, 1)$ PF*, then* F *is* (X, t) PF*.*

Note that the circuit of RE we need to obfuscate has algebraic degree $\frac{5n}{c}$. We will give an GES encoding strategy which makes all additions free (this is remarkable and only possible for the special structure of our RE): it is easy to make all additions in the PF circuit free and the ElGamal scalars of PF outputs which enter the gc circuit are all same. Now it seems that the multilinear degree should be at least 2 times of the algebraic degree $\frac{5n}{c}$ because another piece of the same degree is needed to unify the zerotest level, according to the straddling techniques of current core obfuscators. We will explain the ideas which improves the result over this. To unify the final level, we let the level of PF outputs for different inputs differ only by a group element of exponent 5 (that is, our levels are in an abelian group), which makes the honest computations automatically result in the same level for all inputs. Another 2 times improvement on the multilinear degree μ comes from a special property of the CLT multilinear map which instantiates our GES. The Coppersmith type attack [22] seems decisive for choice of parameter sizes for the CLT multilinear map. Let $z_1 z_2^{-1}$ be a ring element the adversary can observe, such that z_i has smallest degree d, where z_is are encoded noise. Let $N = \prod_{0 \leq i \leq m} p_i$ be the CLT modulus. The Coppersmith type attack works when $m < \frac{\mu}{d}$, and correctness of GES functionality requires that bit length of each p_i is greater than $\frac{\mu}{d}\rho$, where ρ is the unit size of noise. That is, the equivalent (with respect to ring size) multilinear degree is $\frac{\mu}{d}$. In our obfuscator, we can achieve $d = 2$ by a simple straddling technique, or equivalently we allow some initial GES encodings have degree $\frac{1}{2}$. This explains the main part of the multilinear degree, the other 6 comes from the additional inputs of the PF for catering long output, and also from the secret of the circuit C.

Our construction can be proved to be VBB secure in the Ideal GES model under the heuristic assumptions that the PF and PG we used are secure. The proof follows similar procedures as in [31, 7]. The striking point is that our VBB security is obtained at no extra cost over the IO notion of security, this again benefits from the specific details of our RE.

In summary, our main contributions are:

- A novel design of RE which is based on a new garbled circuit and a pseudo-random function, which has low multilinearity for obfuscation.
- Techniques to improve the efficiency of the kind [7] of core obfuscators.
- A general obfuscation scheme with explicit complexity.

The rest of this paper is organized as follows: Section 2 is preliminaries and overview of our scheme. Section 3 gives our construction of a garbled circuit. Section 4 presents our PF. Section 5 describes our obfuscation scheme. Section 6 gives examples: obfuscated AES and a public random oracle. Section 7 is some concluding remarks.

2 Preliminaries and Overview of our approach

2.1 Circuits and Obfuscation

A circuit C in this work is always binary and may compute a keyed function $F(K, X)$, 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 $F(k, \cdot)$. In other words, the key information 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. We call this the keyed circuit for computing the keyed function $F(K, X)$. A keyed circuit can be obtained from a universal circuit by absorbing all single inwire gates into definition of their two-inwire receiving gates.

Obfuscation of a keyed function $F(K, X)$ is pair of PPT algorithms (E, Ob), such that $E(x, Ob(k)) = F(k, x)\ \forall k, x$. There are two main security notions about obfuscation: VBB means $Ob(k)$ can be simulated given only oracle access to $F(k, \cdot)$, whereas IO means $Ob(k_1)$ is indistinguishable with $Ob(k_2)$ if $F(k_1, \cdot) = F(k_2, \cdot)$. This definition of obfuscation is not phrased in circuits but seems sufficient for all known application of the notion in cryptography. It is known [10] that VBB secure obfuscation is impossible for general functions in standard model, but heuristic solutions in some ideal models are not excluded and would be useful in practice.

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 $1 \le i \le |C|$); circuit topology $\{(i, i_0, i_1) : 1 \le i \le |C|\}$ where $i_0 < i_1 < i$; and gate definitions $g_{i,k}$.

Known essential obfuscating algorithms, called core obfuscators, can only handle NC^1 circuits directly. To be able to obfuscate a general circuit C, it is necessary to transform it into NC^1, a highly paralleled form $\mathrm{RE}(C)$ of C, called Randomized Encoding (RE) of C: there is an evaluating algorithm D, such that $D(\mathrm{RE}(C)(x)) = C(x)$, and the oracle $\mathrm{RE}(C)()$ leaks no information other than the oracle $C()$. Now $OB(C) = Ob(\mathrm{RE}(C))$ is an obfuscation of C, and if Ob is VBB secure [31, 7], so is OB.

RE can be constructed using a circuit garbling algorithm gc and and a pseudorandom function (PF) f. A circuit garbling algorithm gc satisfies: $D(gc(C, x, r)) = C(x)$ and $gc(C, x, r)$ leaks no information other than $C(x)$, if the randomness r is independently chosen for each x. Our RE will be the circuit that computes $gc(C, x, f(x))$ where x is the input of the circuit C, that is, RE is the composition of a gc and a pseudorandom function f. Note that we need f supports long output to cater for long randomness needed by gc, whereas a pseudorandom generator is needed to stretch the output of the pseudorandom function as in the implicit RE construction of [5].

2.2 Graded Encoding Schemes

Obfuscation is so amazing that it is only achievable with the help of a magic tool called Graded Encoding Scheme (GES). The definition of GES in this work is almost the same as the MRG model of [7]. The differences are specified as follows. The values to be encoded are in a ring $\mathbb{Z}_2 \times \mathcal{G}$, where $\mathbb{Z}_2$ denotes the boolean ring (or called the binary field), $\mathcal{G}$ denotes a universal ring: for any nonzero polynomial p (formed by a polynomial sized circuit) of integer coefficients in m variables:

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

where $\leftarrow_R$ means "sampled uniformly randomly from". That is, there are two slots to encode values: the boolean one for circuit evaluation and the universal 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 $\mathcal{A}$, that is, a set of elements $\{v_j\}$ and a set of relations $\{R_i = 0\}$ over $\{v_j\}$ defines the abelian group $\mathcal{A}$, where R_i is a linear combination of $\{v_j\}$ with integer coefficients.

The expression of the multilinearity is also different. 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:

$$[b_1, \alpha_1]_v(d_1) + [b_2, \alpha_2]_v(d_2) \sim [b_1 \oplus b_2, \alpha_1 + \alpha_2]_v(\max(d_1, d_2))$$

$$[b_1, \alpha_1]_v(d_1) - [b_2, \alpha_2]_v(d_2) \sim [b_1 \oplus b_2, \alpha_1 - \alpha_2]_v(\max(d_1, d_2))$$

$$[b_1, \alpha_1]_{v_1}(d_1) \times [b_2, \alpha_2]_{v_2}(d_2) \sim [b_1 \& b_2, \alpha_1 \times \alpha_2]_{v_1+v_2}(d_1 + d_2)$$

where $\sim$ stands for having the same content and level. A symbolic circuit is a circuit whose inputs are encodings and whose gates are symbolic operations. The multilinearity μ of the GES is the maximal degree d that is allowed in an encoding $[b, \alpha]_v(d)$. In addition, there is a test level v_{zt}, at which the adversary can query (zero test) if the content of an encoding is zero.

A GES can be used to evaluate a public circuit C where each input and key bit is encoded in initial encodings [31, 7]; we allow C taking locally keyed input (output of a keyed local function $f(k, x)$: each output bit of $f(k, x)$ depends on no more than c bits of x): given a set of initial encodings $\{[b_i, r_i\alpha_i]_{v_i}(d_i) : 1 \le i \le l\}$, where the keyed input is encoded in $\{b_i\}$, and $\{\alpha_i\}$ is input independent, such that there is a symbolic circuit $\tilde{C}_j$ satisfying

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

where $|_x$ means a selection (subset) indexed by x, and $(y)_j$ means the jth component of a vector y. 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: an ElGamal form for each bit $(f(k, x))_j$ of $f(k, x)$ and a check value for each output bit C_j of C. An ElGamal form for a bit b is a pair $([1, r]_v(d), [b, r\alpha]_v(d))$, where $[1, r]_v(d)$ is called the ElGamal scalar. In ElGamal form, any two encodings can be added, 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. 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 should be subtracted from 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 [7] reduces the multilinear degree μ by a factor of c in cases where the circuit factors through a c-local input function, as in our RE circuit.

The formal definition of our GES is composed of the following algorithms and oracles:

$$(sk, pk, \mathcal{G}, v_{zt}) \leftarrow \text{KeyGen}(1^\lambda, \mu, \mathcal{A})$$

where pk is the public parameter which enables anyone to access the public oracles " $+$ ", " $-$ " and " $\times$ ";

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

$\text{ZeroTest}([b, \alpha]_v(d)) = b$ if $v = v_{zt}$, $\alpha = 0$, and $d \le \mu$, otherwise it is undefined.

An obfuscator has sk and $\mathcal{G}$ and thus can use the Encode oracle to provide the set of initial encodings for the above circuit evaluation. We always let the set $\{\alpha_i\}$ be independently uniform samples of $\mathcal{G}$, and the set $\{r_i\}$ be obtained by a set of algebraic operations on a set of independently uniform samples of $\mathcal{G}$, this set of operations is known to the adversary, but the samples are secret. The security is defined as that no symbolic circuit $\tilde{C}'$ exists such that

$$\tilde{C}'(\{[b_i, r_i\alpha_i]_{v_i}(d_i) : 1 \le i \le l\}) \sim [b', 0]_{v_{zt}}(d)$$

and $d \le \mu$ and b' is not derivable from the oracle $C(f(k, \cdot))$. If this can be verified efficiently [7], the scheme is VBB secure; it is called IO otherwise.

It is easy to see that μ will be exponential in the depth of the circuit C given constant degrees of initial encodings if C is quite general. Known instantiations of GES has complexities proportional with μ^2, so obfuscation of general circuits can not directly apply a GES as above. Instead, GES is applied to the RE circuit which is NC^1 (having depth $O(\log \lambda)$) for several known constructions of gc and PF [9]. However, the μ of these constructions is a high degree polynomial of λ, n that is too huge to be considered even feasible in practice. We gives the first construction of RE in this work with $\mu = \frac{5n}{2c} + 6$, where the constant c can be chosen such that 2^c is not too huge. For example, for $n = 128$, we can let $c = 13$ which results $\mu = 31$. That is, all circuits with input boolean variables ≤ 128 can be VBB obfuscated using a GES of degree 31. The details of our RE construction and its obfuscation will be described in the following sections.

The only known instantiation of GES we need is obtained from the following simplified version of the CLT multilinear map of [16], where a symbolic encoding is just an element of a huge ring, and symbolic operations are just ring operations. The public key is $N = \prod_{0 \le i \le m} p_i$, where p_is are relatively prime integers. $\mathcal{G} = \prod_{1 \le i \le m} \mathbb{Z}_{g_i}$, g_is are different primes and we let $\mathbb{Z}_l$ denote the residue integer ring $\mathbb{Z}/l\mathbb{Z}$. A level v corresponds to an element $z_v \in \mathbb{Z}_N$. Let ρ be the unit size of noise. Then

$$\text{Encode}(sk, b, \alpha, v, d) = [b, \alpha]_v(d) = z_v^{-1}\text{CRT}_{(p_i)}(b+2s_0, a_1+s_1g_1, \cdots, a_m+s_mg_m)$$

where $\alpha = (a_1, \cdots, a_m)$, s_is are random integers such that $-2^{d\rho} < a_i + s_i g_i < 2^{d\rho}$, and $\mathrm{CRT}_{(p_i)}(z_i)$ 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

$$z_t = z_{v_{zt}} \mathrm{CRT}_{(p_i)}(2^{-1}\frac{N}{p_0}, g_1^{-1}\frac{N}{p_1}, \cdots, g_m^{-1}\frac{N}{p_m})$$

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

$$z_t[b, 0]_{v_{zt}}(d) = \sum_{0 \le i \le m} t_i \frac{N}{p_i} + b\frac{p_0 + 1}{2}\frac{N}{p_0}$$

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

To enforce the correctness requirement at minimal cost, it is necessary to consider the minor extra noise caused by free additions. Let $\delta(\tilde{C})$ denote the bit length of this extra noise (In our scheme, we have $\delta(\tilde{C}) < \mu(\log \lambda + 2)$), and let g_i have bit length $\frac{\rho}{2} + 1$ for $1 \le i \le m$ (the largest to support the required algebraic relations among contents of the initial encodings), then we have

Lemma 1. *$\log p_i \ge (\mu - \frac{1}{2})\rho + \delta(\tilde{C}) + \log(m+1) + 1$ for $1 \le i \le m$ and $\log p_0 \ge \mu\rho + \delta(\tilde{C}) + \log(m+1) + 1$ is sufficient to fulfill the correctness requirement.*

Proof. Suppose $z_t[b, 0]_{v_{zt}}(\mu) = \sum_i t_i \frac{N}{p_i} + b\frac{p_0+1}{2}\frac{N}{p_0}$, we have $\log |t_i| \le (\mu - \frac{1}{2})\rho + \delta(\tilde{C}) - 1$ and $\log |t_0| \le \mu\rho + \delta(\tilde{C}) - 1$, that is, $|t_i \frac{N}{p_i}| < \frac{N}{4(m+1)}$ for $0 \le i \le m$. □

Since surely we have $\log(m+1) + 1 < \frac{\rho}{2}$, we will set $\log p_i = \mu\rho + \delta(\tilde{C})$ for $i \ge 1$, and $\log p_0 = \log p_1 + \frac{\rho}{2}$ in the following security analysis.

Let the exposed degree be scaled to 1, μ and ρ also correspondingly scaled. That is, the adversary can observe $\mathrm{CRT}_{(p_i)}(z_i z_i'^{-1})$, where $-2^\rho < z_i, z_i' < 2^\rho$, which seems to be the essential leak of secret information of the obfuscator. There are two kinds of attacks making use of this leakage: one is to guess z_i, z_i' by birthday paradox [15] and the other is the Coppersmith method [22]. The complexity of the birthday attack is about 2^ρ multiplications of modulus N, where $\log N \approx \mu^2\lambda$. We could let $\rho = \lambda - 2(\log \lambda + 2)$ to stop birthday attacks, assuming each multiplication of modulus N has complexity $\ge 16\lambda^2$. The Coppersmith method may recover p_is if $(m+2)(m+1)\rho < \log N$, or may recover some p_i if $\log(z_i z_i') < \frac{\log N}{(m+1)^2}$. Since $\log N \le (m+1)(\mu\rho + \delta(\tilde{C})) + \frac{\rho}{2}$, and $\delta(\tilde{C}) < \mu(\log \lambda + 2)$ in our scheme, we could let $m \ge \mu(1 + \frac{\log \lambda + 2}{\rho}) + \frac{1}{2(m+1)} - 2$, that is, m is just slightly bigger than μ is enough to invalidate the first condition. For the second, noting that by trying $\frac{2^l}{m}$ times, we would get a sample with $\log(z_i z_i') \approx 2\rho - l$. If the adversary could tolerate less than $\frac{2^\lambda}{m(4\lambda)^3}$ tries, we would assume $\log(z_i z_i') > 2\rho + 3(\log\lambda + 2) - \lambda = \rho + (\log\lambda + 2)$, which means $m \geq \mu$ is sufficient to avoid the attack based on the second condition. Hence, we can let $m = \lceil \mu(1 + \frac{\log\lambda+2}{\rho}) + \frac{1}{2(\mu+1)} - 2 \rceil$.

In summary, we will let $\rho = \lambda - 2(\log\lambda + 2)$, $\log g_i = \frac{\rho}{2} + 1$, $\log p_i = \mu(\rho + \log\lambda + 2)$ for $i > 0$, $\log p_0 = \log p_1 + \frac{\rho}{2}$, and $m = \lceil \mu(1 + \frac{\log\lambda+2}{\rho}) + \frac{1}{2(\mu+1)} \rceil - 2$ in our CLT multilinear map. In our AES example, $\mu = 31$, $\lambda = 128$; we get $m = 32$ and $\text{GES}(31, 128) = 33 \times 31 \times 119 + 55$. Similarly, we could have $\text{GES}(16, 128) = 17 \times 16 \times 119 + 55$.

3 The Garbled Circuit *gc*

A function $pg : \{0,1\}^n \to \{0,1\}^m$ is called a (n, m) Pseudorandom Generator (PG), where $m > n$, if $pg(k)$ is pseudorandom when k is random. The following conjecture is based on [23, 6], where we assume the so called "ϵ-biased" means secure strength $1/\epsilon^2$.

Conjecture 1. There exists quadratic $(2\lambda, 6\lambda)$ PG (each output bit is a quadratic function of input bits).

In fact, the construction of [23] subjects that quadratic $(2\lambda, 2e\lambda)$ PG exists for $e = o(\lambda)$; the above conjecture is sufficient for our *gc* construction though larger stretch PG would yield slightly more efficient *gc*.

The input to *gc* is x and 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 $g_{i,k}$. The definition of $g_{i,k}$ is given by 4 bits

$$\{e_i(a, b) = g_{i,k}(a, b) : a, b \in \{0, 1\}\}$$

(a stands for inwire from gate i_0). Let $i_{-1} < \cdots < i_{-l_i}$ be all gates next to i. The random r is composed of: for each gate i, l_i pairs of strings $\{(r_{i,j,0}, r_{i,j,1}) : 1 \leq j \leq l_i\}$ of length 2λ to represent value $0, 1$ respectively, and 4 bits $t_{i,0}, t_{i,1}, s_{i,0}, s_{i,1}$ for defining permutations.

A bit b defines a permutation τ_b on two elements (a_0, a_1) as:

$$\tau_b(a_0, a_1) = ((1 - b)a_0 + ba_1, ba_0 + (1 - b)a_1).$$

By composition, 2 bits can define a permutation on 4 elements:

$$\tau_{a,b}(a_1, a_2, a_3, a_4) = \tau_b(\tau_a(a_1, a_2), \tau_a(a_3, a_4)).$$

Note that if a, b is random, then a_i goes to the jth position with equal probability for any pair (i, j). We also use notation $(A, B)[0] = A$, $(A, B)[1] = B$. Let pg_i be a PG that maps strings of 2λ to those of length $2(2l_i + 1)\lambda$, which can be seen as two elements of length $(2l_i + 1)\lambda$ ($l_i = 0$ at output gate i). The output of our *gc* consists of:

– At each input gate i: $(r_{i,i_{-j},x_i})_{1 \leq j \leq l_i}$

– At gate i: 4 strings of length $(2l_i + 1)\lambda$:

$$\tau_{s_{i,0},s_{i,1}}((R_{0,0}||\text{Valid}) \oplus T_{0,0}, (R_{0,1}||\text{Valid}) \oplus T_{0,1}, \\ (R_{1,0}||\text{Valid}) \oplus T_{1,0}, (R_{1,1}||\text{Valid}) \oplus T_{1,1}))$$

where $R_{a,b} = (1 - e_i(a,b))(||_{1\le j\le l_i} r_{i,i_{-j},0}) + e_i(a,b)(||_{1\le j\le l_i} r_{i,i_{-j},1})$, **Valid** is a constant string of length λ for checking correctness, $||$ means string catenation; and

$$T_{a,b} = \tau_{t_{i,0}}(pg_i(r_{i_0,i,a}))[b] \oplus \tau_{t_{i,1}}(pg_i(r_{i_1,i,b}))[a]$$

for $a, b \in \{0,1\}$; which means the other part of the PG output is used when the other input changes.

– At each output gate i : a vector of 4 strings of length λ:

$$\tau_{s_{i,0},s_{i,1}}(R_{0,0} \oplus T_{0,0}, R_{0,1} \oplus T_{0,1}, R_{1,0} \oplus T_{1,0}, R_{1,1} \oplus T_{1,1})$$

where $R_{a,b} = (1 - e_i(a,b))\text{False} + e_i(a,b)\text{True})$, and **True** and **False** are constant strings of length λ standing for the output value, and

$$T_{a,b} = \tau_{t_{i,0}}(pg_i(r_{i_0,i,a}))[b] \oplus \tau_{t_{i,1}}(pg_i(r_{i_1,i,b}))[a]$$

Lemma 2. *The above gc is secure.*

Proof. Given input x, $C(x)$ and topology of C, the above output of $gc(C, x, r)$ can be simulated as:

– At each input gate i: $(r_{i,i_{-j}})_{1\le j\le l_i}$.
– At gate i having input gates i_0, i_1 with $r_{i_0,i}, r_{i_1,i}$ defined: randomly choose 2 bits b_0, b_1; randomly choose $k \in [4]$; and for $1 \le j \le l_i$: choose $r_{i,i_{-j}} \leftarrow_R \{0,1\}^{4\lambda}$. Compute

$$((||_{1\le j\le l_i} r_{i,i_{-1}})||\text{Valid}) \oplus pg_i(r_{i_0,i})[b_0] \oplus pg_i(r_{i_1,i})[b_1]$$

and put it at the kth position of the vector; the other 3 elements of the vector are randomly chosen.
– At each output gate i with input gate i_0, i_1: randomly choose $k \in [4]$, and randomly choose 2 bits b_0, b_1; compute

$$C(x) \oplus pg_i(r_{i_0,i})[b_0] \oplus pg_i(r_{i_1,i})[b_1]$$

(where $C(x)$ is in form **True** or **False**) and put it at the kth position of the vector, the other 3 elements are randomly chosen.

It is a standard hybrid argument to prove that the two distributions are computationally indistinguishable.

Note that gc needs $(8\lambda + 4)|C|$ bits of randomness, and it has $20\lambda|C|$ output bits. If all pg_i are quadratic, then each output bit is a polynomial of degree ≤ 5 over the random bits. It is possible that some l_i is so large that no quadratic

$(2\lambda, 2(2l_i+1)\lambda)$ PG exists. In this case, we can insert some virtual single inwire gates to make enough output copies. For example, given only quadratic $(2\lambda, 6\lambda)$ PG, we can cope with any $l_i > 1$ as follows. Let $r_{i,b}$ denote the output of gate i, pg be a quadratic $(2\lambda, 6\lambda)$ PG, b_i be random in $\{0,1\}$. The first virtual gate after gate i will have two outwires holding values $\{r_{i,i_0,b} : i_0 = 1,2\}$, and the gc output at this virtual gate is

$$\tau_{b_i}((r_{i,1,0}||r_{i,2,0}||\text{Valid}) \oplus pg(r_{i,0})^-, (r_{i,1,1}||r_{i,2,1}||\text{Valid}) \oplus pg(r_{i,1})^-)$$

where T^- means T cut to suitable length. Each $r_{i,i_0,b}$ can further multiply into $\{r_{i,i_0,i_1,b} : i_1 = 1,2,3\}$ at virtual gate (i, i_0) whose gc output is:

$$\tau_{b_i}((r_{i,i_0,1,0}||r_{i,i_0,2,0}||r_{i,i_0,3,0}) \oplus pg(r_{i,i_0,0}), (r_{i,i_0,1,1}||r_{i,i_0,2,1}||r_{i,i_0,3,1}) \oplus pg(r_{i,i_0,1}))$$

The procedure can go on until enough copies are obtained. Note that the number of the extra (compared with the no virtual gates case) random bits is less than $4\lambda|C|$, and the number of extra output bits is also less than $4\lambda|C|$.

4 The Pseudorandom Function

A Pseudorandom Function (PF) $F : X \times K \to \mathcal{R} : (x,k) \mapsto F(x,k) \in \mathcal{R}$ satisfies: when k is random, $F(x,k)$s are independently pseudorandom in the range $\mathcal{R}$ for different xs. When $\mathcal{R} = \{0,1\}^m$, we call such a PF a (X, m) PF by omitting specification of the key space. Note that $(X \times \{0,1\}^{km})$ PF implies (X, km) PF: the position of output can be specified by k input variables.

Traditional constructions of PFs [24, 25, 19, 9, 30] are so complex to be obfuscated directly. We will give a novel concrete construction using empirical methods: i.e. we will not base security on well-known assumption, but only aim to prevent known attacks; which follows the paradigm in traditional design of symmetric cryptographic algorithms. For efficiency, we need the circuit be wide and shallow and have low algebraic degree. For security, we need the whole input bits be thoroughly mixed. The intuition is that huge key space may be helpful to achieve both goals simultaneously. We let the key space be PFs on constant size small spaces and use inductive method to construct the PF we need: $\{0,1\}^n \to \{0,1\}^m$, where $m = (16\lambda+4)|G|) \leq 2\lambda t$. Let T be a set of t elements which is just enough to index the outputs, what we want is a $(T \times \{0,1\}^n, 2\lambda)$ PF. Let c be a constant such that n can be written as sum of a d pairs of integers, each of which is c or $c-1$. The key of the target PF consists of: a $(T, 2\lambda)$ PF; $2d$ $(\{0,1\}^c, 10\lambda)$ or $(\{0,1\}^{c-1}, 10\lambda)$ PFs.

We need the following assumptions:

Conjecture 2. Let $F = (f_1, f_2, \cdots, f_t)$, $G = (g_1, g_2, \cdots, g_t)$ be (X, t), (Y, t) PFs respectively, then $\sum\limits_{1 \leq i \leq t} f_i g_i$ is $(X \times Y, 1)$ PF if $t \geq 2\lambda$.

Conjecture 3. Let $F = (f_1, f_2, \cdots, f_t)$. If any linear combination of f_is is $(X, 1)$ PF, then F is (X, t) PF.

Known attacks on PFs are of linear type: use a linear combination of outputs over some distribution of inputs to form a distinguisher. The conjectures seem hold in this respect. The number 2λ is the birthday bound. If X, Y are small, $\lambda/2$ (the linear distinguishing bound) seems enough. Our inductive methods are:

Lemma 3. *Let $F = (f_1, f_2, \cdots, f_{10\lambda})$, $G = (g_1, g_2, \cdots, g_{10\lambda})$ be $(X, 10\lambda)$, $(Y, 10\lambda)$ PFs respectively, randomly choose 2λ linear functions $h_1, h_2, \cdots, h_{2\lambda}$ of weight 2λ over $\{0,1\}^{10\lambda}$, then $(h_1(f_1g_1, \cdots, f_{10\lambda}g_{10\lambda}), \cdots, h_{2\lambda}(f_1g_1, \cdots, f_{10\lambda}g_{10\lambda}))$ is $(X \times Y, 2\lambda)$ PF.*

Proof. According to coding theory [28], the minimal weight of any linear combination of $h_1, h_2, \cdots, h_{2\lambda}$ is about 2λ, the assertion follows Conjecture 3. $\square$

Lemma 4. *Let $F = (f_1, f_2, \cdots, f_{2\lambda})$, $G = (g_1, g_2, \cdots, g_{2\lambda})$ be $(X, 2\lambda)$, $(Y, 2\lambda)$ PFs respectively, $A_i : 1 \leq i \leq 2\lambda$ be matrices such that any linear combination of them is invertible, let $h_i = FA_iG^t$ (G^t means a column vector), then $(h_1, h_2, \cdots, h_{2\lambda})$ is $(X \times Y, 2\lambda)$ PF. To construct such $\{A_i\}$, consider any basis of $\mathbb{GF}_{2^{2\lambda}}$ over $\mathbb{Z}_2$, the set of matrices of the linear maps: multiplications by the basis elements, has the desired property. Note that we can make A_i very sparse: consider the polynomial basis where the irreducible polynomial has 3 or 5 nonzero terms.*

Starting with the key, the $2d$ PFs of output length 10λ are first combined into d PFs of output length 2λ. The reason behind this step is for applying the technique of halving μ. Then these d PFs are combined in arbitrary but fixed order to form a $(\{0,1\}^n, 2\lambda)$ PF f_0, which is finally combined with the T part key to get the desired $(T \times \{0,1\}^n, 2\lambda)$ PF.

5 The Obfuscator

Our obfuscator is a variation of the Simple Ob of [7], originated from [31]. The basic idea is evaluating the circuit on encrypted variables with two slots under a GES: one for functionality and the other for authentication. Obfuscation of a keyed function $C(K, X)$ is pair of PPT algorithms (E, Ob), such that $E(x, Ob(k)) = C(k, x)\ \forall k, x$.

The input to the algorithm Ob is actually the RE for computing $C(k, \cdot)$: the garbled circuit $gc(C(k, \cdot))$ composed with the $(T \times \{0,1\}^n, 2\lambda)$ PF f. The input of the RE is $x \in \{0,1\}^n$ which is divided into d pairs of length c or $c-1$ blocks whose spaces are denoted as $(X_j, Y_j) : 1 \leq j \leq d$; and the key variables include key of the PF f and all bits for defining gates of $C(k, \cdot)$. We will use a GES with a universal ring $\mathcal{G}$ for authentication and an abelian group $\mathcal{A}$ for levels.

We assume there is large subgroup U of $\mathcal{A}$ of exponent 5: $5u = 0\ \forall u \in U$. The output of Ob are initial encodings which include:

- Encoding of the $(T, 2\lambda)$ PF: $[1, r_T]_{v_T}(1)$ and
$$\{[b_{t,j}, r_T\alpha_{t,j}]_{v_T}(1) : t \in T, 1 \leq j \leq 2\lambda\}.$$

– Encoding of the $(X_j, 10\lambda)$ PF and $(Y_j, 10\lambda)$ PF: for $1 \leq j \leq d$,

$$\{[1, r_{x_j}]_{v_j+u_{x_j}}(1/2) : x_j \in X_j\} \text{ and } \{[1, r'_{y_j}]_{v'_j+u'_{y_j}}(1/2) : y_j \in Y_j\}$$

$$\{[b_{x_j,i}, r_{x_j}\alpha_{j,i}]_{w_{j,i}+u_{x_j}}(1/2) : x_j \in X_j, 1 \leq i \leq 10\lambda\}$$

and

$$\{[b'_{y_j,i}, r'_{y_j}\alpha'_{j,i}]_{w'_{j,i}+u'_{y_j}}(1/2) : y_j \in Y_j, 1 \leq i \leq 10\lambda\}$$

where $w_{j,i} + w'_{j,i} = w_j$ $\forall i$, $\sum_j(v_j + v'_j) = \sum_j w_j$, and all u, u's are in U. Let S_j, S'_j be the subsets of $[n]$ corresponding to X_j, Y_j, we set $r_{x_j} = \prod_{i \in S_j} r_{i,(x)_i}$, $r'_{y_j} = \prod_{i \in S'_j} r_{i,(x)_i}$, where $\{r_{i,b} : i \in [n], b = 0, 1\}$ is a set of independent samples; u_{x_j}s and u'_{y_j}s are independent random variables in U. Note that the ElGamal forms are at twisted levels, so that the exposed noise degree will be twice of their degree.

– Encoding of gate definitions: $[1, r]_v(1)$ and $\{[e_i(a, b), r\alpha_{i,a,b}]_v(1) : 1 \leq i \leq |G|, a, b \in \{0, 1\}\}$.

– Encoding of check values for gc outputs on gates:

$$\{\text{CHK}_{t,j} = [0, rp_{t,j}(\alpha*)]_v(1) : t \in T, 1 \leq j \leq 2\lambda\}$$

where $\alpha*$ means all α type variables, and $p_{t,j}$ is the polynomial defined by that output position: Evaluating the initial encodings along the RE circuit to the output bit of gc indexed by (t, j) would result in:

$$[gc_{t,j}(x), r(r_T \prod_i (r_{x_i} r'_{y_i}))^5 p_{t,j}(\alpha*)]_{v_{zt}}(\mu).$$

– Encoding of check values for gc output for input variables:

$$\{\text{CHK}_{t,j} = \{[0, r(r_{i,(x)_i})^{-1} f_{t,j}(\alpha*)]_v(1) : t \in T, 1 \leq j \leq 2\lambda\}.$$

Here the index (t, j) indicates that the ith input variable takes value $(x)_i$. For $t_b \in T$ pointing to $(x)_i = b$, set $\alpha_{t_b,j} = (r_{i,b})^{-1}\alpha'_{t*,j}$. $f_{t,j}$ is the polynomial defined by $f(t, x)$'s jth output bit. Then only if we choose the right pairs $t_b, (x)_i = b$, we can make use of the check values at gc outputs for the input variable $(x)_i$. This is the mechanism in our scheme to prevent double-use of the gc randomness.

The test level is $v_{zt} = v + 5(v_T + \sum_j(v_j + v'_j))$, and the multilinearity of the scheme is $\mu = 5d + 6$.

Evaluation of the obfuscated circuit on input x can be done as:

– Computing the encodings of $f_0(x)$: They have a common ElGamal scalar $E_x = [1, r_x]_{v_x}(d)$, and look like:

$$[(f_0(x))_i, r_x p_i(\{\alpha_{j,i}\alpha'_{j,i}\})]_{v_x}(d), 1 \leq i \leq 2\lambda,$$

where $r_x = \prod_j r_{x_j} r_{y_j}$, $v_x = \sum_j w_j + \sum_j(u_{x_j} + u'_{y_j})$, and p_i is a polynomial.

– Computing encodings of outputs of the PF: choose ts in T standing for input variables, and compute the encodings of $f(t,x)$:

$$F_{t,j} = [(f(t,x))_j, r_T r_x q_j(\{p_i, \alpha_{t,i}\})]_{v_x+v_T}(d+1).$$

– Computing gc outputs for input variables: choose ts in T standing for input variables, then

$$([1, r_x r_T]_{v_x+v_T}(d+1))^4[1,r]_v(1)F_{t,j} - ([1, r_x r_T]_{v_x+v_T}(d+1))^5\mathrm{CHK}_{t,j} \\ \sim [f(t,x)_j, 0]_{v_{zt}}(\mu),$$

so $(f_{t,x})_j$ can be obtained by zero test.

– Computing gc outputs for gates: Denote the jth bit of gc output at gate i as $gc_{i,j}$, which is a polynomial $p_{t,j}$ over some $(f(t,x))_j$s and the gate definition bits. Evaluating $p_{t,j}$ over corresponding encoding of its inputs will get to: $\mathrm{GC}_{i,j} = [gc_{i,j}, r(r_T r_x)^5 q_{t,j}(\alpha*)]_{v_{zt}}(\mu)$, where $q_{t,j}$ is some polynomial; and

$$\mathrm{GC}_{i,j} - [1, (r_T r_x)^5]_{5v_T+5v_x}(5d+5)\mathrm{CHK}_{t,j} \sim [gc_{i,j}, 0]_{v_{zt}}(\mu),$$

where adding a constant 1 to an ElGamal form $([1,r]_v(d), [b, r\alpha]_v(d))$ is done as $([1,r]_v(d), [b, r\alpha]_v(d) + [1,r]_v(d))$; adding of two ElGamal forms always use the "least common multiple" of their ElGamal scalar.

– Evaluating gc to get $C(k,x)$.

Lemma 5. *The above scheme has perfect correctness in the ideal GES model.*

Lemma 6. *There are no two different available encodings at the same level such that one of them has degree $\frac{1}{2}$.*

Theorem 1. *The above scheme is VBB secure in the ideal GES model.*

Proof. According to [7], it is enough to give an efficient algorithm for verifying if evaluating a given symbolic circuit $\tilde{C}$ or polynomial p on the initial encodings gets to a testable encoding. This verification is done as follows:

– Classifying monomials: it is easy to see that the level straddling guarantees that no two different subscripts x_j, x'_j or y_j, y'_j can appear as factors of a monomial of level v_{zt}. This means each legal monomial is indexed by a unique x.

– If no check values are used, then no nonzero testable polynomial exists: grouping monomials by xs results $p = \sum_x p_x$, where p_x has content at the second slot $r(r_T r_x)^5 q_x(\alpha*)$. Since r_xs are linearly independent (with respect to α-type variables) and $\alpha*$ is a set of algebraic independent variables, p evaluates to 0 at the second slot would force all $q_x = 0$ and thus $p = 0$. This fact can be checked by evaluating p on random evaluations of the symbolic variables in a universal ring, say $\mathbb{Z}_l$ for a large prime l [31].

– The general case can be reduced to the "no check value" case with the help of the oracle RE(): write p as a sum of two parts: one is a linear combination of check values and the other is free of check values. The legal multiple of a check value is of the form: $\sum_x [1, (r_T r_x)^5]_{v_{zt}-v}(5d+5)$; where each x must appear in the "no check value" part whose size bounds the number of such xs (this needs a little detail of the circuit). Now by subtracting the corresponding honest computation on these xs from the two parts of an testable polynomial would make the two parts evaluate to 0 which can be efficiently tested as above. Then oracle RE() can be used to decide the output of the zero test oracle.

It is easy to count the number of initial encodings when no virtual gates in gc are needed:

$$(28\lambda + 4)|C| + 2^c d \times 10\lambda \approx 28\lambda|C| + 10 \times 2^{c-1}\frac{n}{c}\lambda.$$

Suppose $\text{GES}(\mu, \lambda) \approx \mu^2\lambda$, the optimal choice for c is the one which minimize $(28|C| + 10\frac{2^{c-1}}{c})(\frac{n}{2c} + 6)^2$ which is irrelevant to λ. The computational cost (in number of GES multiplications) is also easy to estimate, the main parts are: $d \times 10\lambda + 2d \times (2\lambda)^2$ for computing $f_0(x)$; $8\lambda|C| \times 2\lambda$ for computing the PF outputs; $4|C| \times 2\lambda^2$ for computing gc outputs; all these sum to $\approx (32|C| + 8d)\lambda^2$ multiplications of GES encodings. When virtual gates are introduced, the complexity multiplies by a factor no larger than $\frac{9}{7}$.

The GES we required can be instantiated with our simplified CLT map, where the level subgroup U can be realized by choosing p_i as product of primes congruent to 1 mod 5 $\forall i$. If the total number of prime factors of N is l, then $|U| = 5^l$. Since some mismatched-inputs computation can be conducted with probability $\frac{1}{|U|}$, we should choose l such that $|U| > 2^\lambda$. On the other hand, samller divisors would make factoring N easier, we should ensure that each prime factor is large enough so that factoring N is hard. For example, when $m = 32, \lambda = 128$, we can let each p_i be a product of two primes so that the minimal prime factor has size over 1800 bits which is enough to prevent factoring N.

To enforce the correctness requirement at minimal cost, it is necessary to estimate the minor extra noise caused by free additions. For a homogeneous addition where all summands have the same level of noise, if l is the number of summands, then the extra noise produced by the addition is $\log l$ for perfect correctness, and $\frac{1}{2}\log l$ for probabilistic correctness. We use the later in this work. The extra noise for computing f_0 expands to $5d\frac{\log\lambda+1}{2} + 5(d-1)\frac{\log\lambda+2}{2}$ at gc outputs: the second term is an estimation of the extra noise when combining the d $(\cdot, 2\lambda)$ PFs; combining a pair with homogeneous components will result a PF with inhomogeneous components with respect to the extra noise; to make it more even, the heavy components will be placed in light positions at next round of merging; this will result the heuristic estimation. Computing $f(t,x)$ contributes another $\frac{5(\log\lambda+1)}{2}$. The gc circuit produces another $\log\lambda + 2$. All of these amounts no larger than $\mu(\lambda + 2)$.

6 Examples

The AES encryption algorithm may serve as a testbed for practicality of obfuscation. The cost of our scheme only depends on the number of input variables n and the circuit size $|C|$. In fact, our circuit size 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 [14]. We did not conduct this optimization because it is not so crucial to the result: this work only shows that obfuscation is implementable, but still far from practical use. So a safe estimation of $|C| < 25000$ is used. Virtual gates are not needed. We choose $c = 13$, and get $\mu = 31$. The GES ring size is $< 33 \times 31 \times 119 + 55 \approx 1.2 \times 10^5$ bits. The code size of the obfuscated AES is thus $\approx (28 \times 25000 \times 128 + 2^{13} \times 9 \times 1280) \times 1.2 \times 10^5 < 2.2 \times 10^{13}$ bits. To evaluate the obfuscated AES once would cost about 1.3×10^{10} modular multiplications with 1.2×10^5 bits modulus, which is still far from practical.

Note that just obfuscating the PF is more efficient, which can be used as a public random oracle. In this case we need to unify to the zero test level which would make μ doubled, and the group U is not needed. To get minimal code size, the μ halving technique does not profit. So we would have $\mu = \frac{2n}{c}$. The code size of the obfuscated PF will be $2^c \times \frac{n}{c} \times 2\lambda \times (\frac{2n}{c})^2 \lambda = \frac{2^{c+2}}{c^3} n^3 \lambda^2$, the optimal choice for c is 4. To implement a public random oracle for query of any length would require $n = 2\lambda$, so the minimal code size is about $8 \times \lambda^5$, and in which case the computation cost for one query is about $8\lambda^3$ modular multiplications of about λ^3 bits modulus. In this example, reducing the computational cost is more desirable. So we would exploit the μ halving technique and use a bigger c. When $\lambda = 128$, $c = 16$, we have $\mu = \frac{n}{c} = 16$, $m = 17$, and $\mathrm{GES}(16, 128) \leq 17 \times 16 \times 119 + 55 < 3.3 \times 10^4$. The code size will be under $16 \times 2^{16} \times 10 \times 128 \times \mathrm{GES}(16, 128) \approx 4.2 \times 10^{13}$ bits, which may be put on a data center; but the computation cost will reduce to $8 \times 10 \times 128 + 7 \times 4 \times 128^2 < 5 \times 10^5$ modular multiplications of $< 3.3 \times 10^4$ bits modulus, which is handleable on personal computers.

7 Concluding Remarks

We have made general obfuscation implementable, but still not practical. Now the degree of the multilinear map is not the main barrier, the huge number of modular multiplications of big modulus is more prohibitive. It is possible to relax requirements on PF and gc to get more efficient schemes. We expect that more empirical methods will take obfuscation closer to practical use.

References

1. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. IACR Cryptology ePrint Archive **2013** (2013) 689

2. Ananth, P., Jain, A., Sahai, A.: Robust transforming combiners from indistinguishability obfuscation to functional encryption. Cryptology ePrint Archive, Report 2017/127 (2017) `http://eprint.iacr.org/2017/127`.
3. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. Cryptology ePrint Archive, Report 2016/1097 (2016) `http://eprint.iacr.org/2016/1097`.
4. 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
5. 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
6. Applebaum, B., Bogdanov, A., Rosen, A.: A dichotomy for local small-bias generators. J. Cryptology **29**(3) (2016) 577–596
7. 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
8. Ball, M., Malkin, T., Rosulek, M.: Garbling gadgets for boolean and arithmetic circuits. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM (2016) 565–577
9. 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
10. 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
11. 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
12. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2014) 533–556
13. Boneh, D., Ishai, Y., Sahai, A., Wu, D.J.: Lattice-based snargs and their application to more efficient obfuscation. Cryptology ePrint Archive, Report 2017/240 (2017) `http://eprint.iacr.org/2017/240`.
14. 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
15. Chen, Y., Nguyen, P.Q.: Faster algorithms for approximate common divisors: Breaking fully-homomorphic-encryption challenges over the integers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2012) 502–519

16. 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
17. 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
18. Halevi, S., Halevi, T., Shoup, V., Stephens-Davidowitz, N.: Implementing bp-obfuscation using graph-induced encoding. Cryptology ePrint Archive, Report 2017/104 (2017) `http://eprint.iacr.org/2017/104`.
19. 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
20. Lin, H.: Indistinguishability obfuscation from ddh on 5-linear maps and locality-5 prgs. Cryptology ePrint Archive, Report 2016/1096 (2016) `http://eprint.iacr.org/2016/1096`.
21. Lin, H., Tessaro, S.: Indistinguishability obfuscation from bilinear maps and block-wise local prgs. Cryptology ePrint Archive, Report 2017/250 (2017) `http://eprint.iacr.org/2017/250`.
22. Lu, Y., Zhang, R., Peng, L., Lin, D.: Solving linear equations modulo unknown divisors: Revisited. In: Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part I. (2015) 189–213
23. 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
24. 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
25. Naor, M., Reingold, O., Rosen, A.: Pseudo-random functions and factoring. Electronic Colloquium on Computational Complexity (ECCC) **8**(064) (2001)
26. 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
27. Sahai, A., Zhandry, M.: Obfuscating low-rank matrix branching programs. IACR Cryptology ePrint Archive **2014** (2014) 773
28. Van Lint, J.H.: Introduction to coding theory. Volume 86. Springer Science & Business Media (2012)
29. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982. (1982) 160–164
30. 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
31. 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