Obfuscation without Multilinear Maps

Dingfeng Ye
DCS Center
Institute of Information Engineering
Chinese Academy of Sciences
Beijing, China 100093
ydf@is.ac.cn

Peng Liu
Cyber Security Lab
College of Information Sciences and Technology
Pennsylvania State University
University Park, PA 16802
pliu@ist.psu.edu

Abstract

Known methods for obfuscating a circuit need to represent the circuit as a branching program and then use a multilinear map to encrypt the branching program. Multilinear maps are, however, too inefficient for encrypting the branching program. We found a dynamic encoding method which effectively singles out different inputs in the context of the matrix randomization technique of Kilian and Gentry et al., so that multilinear maps are no longer needed. To make the method work, we need the branching programs to be regular. For such branching programs, we also give most efficient constructions for NC1 circuits. This results in a much more efficient core obfuscator for NC1 circuits.

Index terms: Obfuscation, Matrix Branching Program, Dynamic Fencing

1 Introduction

Indistinguishable Obfuscation of general circuits is a powerful functionality of cryptography [6], which was regarded infeasible [2] until work [4] gives constructions for indistinguishable obfuscation. But this seminal work only addressed the existence problem: the solution is totally impractical for any meaningful circuit examples. Significant improvement of efficiency was obtained in later constructions (including [3, 1, 7, 8]), but these constructions are still far from practical use. The inefficiency lies in the fact that all these constructions rely on multilinear maps of large degree, which lack efficient instantiations.

Our work is inspired by [4, 7]. In these work, a circuit is first translated to a sequence of matrix pairs (the branching program), and circuit evaluation is represented as matrix multiplications. We observe that Kilian's randomization technique, denoted in this work as "fencing", already makes the branching program computation secure for single input, as long as the branching program is *regular* (roughly speaking, the rank function reveals nothing to the attacker). We also observe that the essential thing to make the whole computation secure is to separate computations for different inputs into different worlds, which may not need multilinear maps at all.

A natural way to do this is dynamic fencing: different inputs use independent fences, but this needs exponential storage. We approximate *dynamic fencing* with group actions: each fence is multiplied by a group element which is bound to an input and the branching variable at the place. This binding is achieved via a simple authentication method when working at higher dimensions.

Another contribution of this work is a new regular branching program for NC1 circuits. Our construction has similar efficiency as that of [7], but it is regular, so that Kilian's statistical simulation theorem still holds.

Finally, it should be noticed that the security of our scheme is not proved. The difficulty of a security proof is lack of suitable security model to validate our approximation of dynamic fencing. This also means that our scheme does not depend on any known assumptions; we treat attacking the scheme as a new concrete problem which seems to be hard.

2 Notations

Let $\mathbf{F_2}$ denote a binary field. Let $a+b$ and ab denote addition and multiplication of $a,b \in \mathbf{F_2}$, respectively. Let $M_{m,m'}(\mathbf{F_2})$ denote the set of $m \times m'$ matrices over $\mathbf{F_2}$.
Let $GL_d(\mathbf{F_2})$ denote the set of $d \times d$ invertible matrices. Let I denote the identity matrix of appropriate dimension. We often write a matrix without specifying its dimension, if it can be inferred from context; we also write a matrix of blocks without specifying dimensions of the blocks, which should make multiplication of matrices of blocks meaningful. By a matrix sequence $\{M_i : 1 \le i \le l\}$, we mean $\prod_i M_i$ is meaningful and specification of the length and dimensions are often omitted. Two matrix sequences $\{M_i\}$ and $\{N_j\}$ can be catenated if $(\prod_i M_i)(\prod_j N_j)$ is meaningful, and $\{M_i\} \vee \{N_j\}$ denote the matrix sequence of this catenation. For any matrix sequence $\{M_i\}$, $A\{M_i\}$ is the sequence $\{AM_1\} \vee \{M_i : i > 1\}$; similarly $\{M_i\}A$ means the sequence by multiplying on the right the last matrix with A.

3 Kilian's Randomization

This section is generalization of Kilian's Randomization to regular matrix sequences. Let $\{M_i : 1 \le i \le l\}$ be a matrix sequence with dimensions $d_{i-1} \times d_i$, and let $\{U_i \in GL_{d_i}(\mathbf{F_2}) : 0 \le i \le l\}$ be a sequence of invertible matrices, $KR(\{M_i\},\{U_i\})$ will denote the sequence $\{U_{i-1}^{-1} M_i U_i : 1 \le i \le l\}$. Here, we call $\{U_i\}$ the *fences*. If the fences are uniformly random, $KR(\{M_i\},\{U_i\})$ will be called the Kilian's Randomization (denoted **KR**) of the matrix sequence $\{M_i\}$.

Lemma 3.1 KR *of the matrix sequence* $\{M_i\}$ *and* $\{M'_i\}$ *have the same distribution if there exists* $\{V_i\}$ *such that* $\{M'_i\} = KR(\{M_i\},\{V_i\})$.

Proof $KR(\{M_i\},\{V_iU_i\}) = KR(KR(\{M_i\},\{V_i\}),\{U_i\})$

If $\{M'_i\} = KR(\{M_i\},\{V_i\})$, we will write $\{M_i\} \rightsquigarrow_{(V_0,V_l)} \{M'_i\}$, and we will simply say that the two matrix sequences are **KR** *equivalent*. The following is obvious:

Lemma 3.2 $\{M_i\} \rightsquigarrow_{(U,V)} \{M'_i\} \Leftrightarrow \{M_i\} \rightsquigarrow_{(I,I)} U\{M'_i\}V^{-1}$.

Lemma 3.3 *If* $\{M_i\} \rightsquigarrow_{(U,V)} \{M'_i\}$ *and* $\{N_j\} \rightsquigarrow_{(V,V')} \{N'_j\}$, *then* $\{M_i\} \vee \{N_j\} \rightsquigarrow_{(U,V')} \{M'_i\} \vee \{N'_j\}$

Lemma 3.4 *If* $\{M_i\} \rightsquigarrow_{(U,V)} \{M'_i\}$ *and* $\{M'_i\} \rightsquigarrow_{(U',V')} \{N_i\}$, *then* $\{M_i\} \rightsquigarrow_{(UU',VV')} \{N_i\}$

Lemma 3.5 *[5] If* $\{M_i\}$ *and* $\{M'_i\}$ *are all invertible, then* $\{M_i\} \rightsquigarrow_{(V_0,I)} \{M'_i\}$ *and* $\{M_i\} \rightsquigarrow_{(I,V)} \{M'_i\}$ *where* $V = (\prod M_i)^{-1}(\prod M'_i)$ *and* $V_0 = (\prod M_i)(\prod M'_i)^{-1}$

Definition A matrix sequence $\{M_i\}$ is called *unit* if each M_i is of the form either I or $(I,0)$ or $(I,0)^T$ (T means transpose), and $\prod_i M_i = I$ (where each I may have different dimension). A matrix sequence is called *regular* if it is **KR** equivalent to a unit matrix.

.

Lemma 3.6 *If* $\{O_i\}$ *is a unit sequence, and* G *is invertible, then* $\{O_i\} \rightsquigarrow_{(G,G)} \{O_i\}$.

Proof Let $\{U_i\}$ be a fence of the form: $U_i = G$ or $U_i = \begin{pmatrix} G & \\ & I \end{pmatrix}$ according to dimension. Then $KR(\{O_i\}, \{U_i\}) = \{O_i\}$.

Lemma 3.7 *If* $\{M_i\}$ *is regular, and* G *is invertible, then both* $G\{M_i\}$ *and* $\{M_i\}G$ *are regular.*

Proof Obvious.

Lemma 3.8 *Let* $\{M_i\}$ *be regular and* $\{O_i\}$ *be the corresponding unit sequence, then* $\{M_i\} \rightsquigarrow_{(I,I)} V\{O_i\}$ *and* $\{M_i\} \rightsquigarrow_{(I,I)} \{O_i\}V$ *where* $V = \prod M_i$.

Proof Suppose $\{M_i\} \rightsquigarrow_{(U,U')} \{O_i\}$, Since $\{O_i\} \rightsquigarrow_{(U^{-1},U^{-1})} \{O_i\}$, we have $\{M_i\} \rightsquigarrow_{(I,U'\times U^{-1})} \{O_i\}$. By comparing the product, we must have $U' \times U^{-1} = V = \prod M_i$.

Lemma 3.9 *If* $\{M_i\}$ *and* $\{N_j\}$ *are regular, so is* $\{M_i\} \vee \{N_j\}$.

Proof Because catenation of unit sequences is also unit, suppose $\{M_i\} \rightsquigarrow_{(U,I)} \{O_i\}$ and $\{N_j\} \rightsquigarrow_{(I,V)} \{O_j\}$, then we have $\{M_i\} \vee \{N_j\} \rightsquigarrow_{(U,V)} \{O_i\} \vee \{O'_j\}$

4 Branching Programs of Circuits

A branching program is a pair (BP, inp), where $BP = \{B_{i,b} : 1 \le i \le l, b = 0,1\}$ is a sequence of matrix pairs, and $inp : [l] \to [n]$ is a map. A branching program can compute a function $C : {F_2}^n \to F_2$ as follows: $C(x_1, x_2, \cdots, x_n) = f(\prod_i B_{i,x_{inp(i)}})$ where f is some function from matrices to F_2. We often omit the specification of inp when it can be comprehended. The largest dimension of matrices in BP is called the width of the branching program. $BP = \{B_{i,b} : 1 \le i \le l, b = 0,1\}$ is called regular, if all matrix sequences $\{B_{i,b_i}\}$ are regular.

It is well known that branching programs can only efficiently compute NC1 circuits (circuits with 2-fan in gates and depth $d < O(\log(\lambda))$). The efficiency of a branching program is roughly measured by its width and length. Main results on branching programs for NC1 circuits are the following:

- The Barrington map: $width = 5$, $length \leq 4^d$, regular;
- [1]: $width \leq 2^d$, $length \leq 2^d$, regular;
- [7]: $width < d$, $length \leq 2^d$, not regular.

In the following, we will give a regular branching program for NC1 circuits with similar efficiency as [7]. To begin with, the operations on matrix sequences: KR, $\vee$, and $A\{M_i\},\{M_i\}A$ are straightforwardly applied to BPs. Another operation is Aug: $Aug(\{B_{i,b}\}) = \{{B_{i,b}}^+\}$, where ${B_{i,b}}^+ = \begin{pmatrix} 1 & \\ & B_{i,b} \end{pmatrix}$.

Let $C(x_1,\cdots,x_n)$ be a circuit with n input variables. We use $\begin{pmatrix} 1 & v \\ 0 & 1 \end{pmatrix}$ to represent the value v at a gate (and input). The following formula can be used to evaluate the circuit:

- not gate: $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & a+1 \\ 0 & 1 \end{pmatrix}$
- $\oplus$ gate: $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & a+b \\ 0 & 1 \end{pmatrix}$
- $\wedge$ gate: $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & & \\ & 1 & a \\ & & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & & \\ & 1 & b \\ & & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & ab \\ 0 & 1 \end{pmatrix}$

The branching program for computing $C(x_1,\cdots,x_n)$ is inductively constructed as follows:

- each input variable x_i is a BP of length 1: $\begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$, $inp(1) = i$.
- not gate: $(BP)\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$;
- $\oplus$ gate: $BP_1 \vee BP_2$;
- $\wedge$ gate: $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} Aug(BP_1) \vee \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} Aug(BP_2) \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$

Theorem 4.1 *The branching program constructed as above is regular.*

Proof Initially the inputs are regular, and each has product value of the type $\begin{pmatrix} 1 & v \\ 0 & 1 \end{pmatrix}$. By induction, this property is preserved at not gates and $\oplus$ gates. It only remains to verify this at $\wedge$ gates, which will be done by the following lemma.

Lemma 4.2 *Let* $\{M_i\}$ *and* $\{N_j\}$ *be regular with product values* $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$ *and* $\begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$ *respectively, then* $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} Aug(\{M_i\}) \vee \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} Aug(\{N_j\}) \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$ *is regular with product value* $\begin{pmatrix} 1 & ab \\ 0 & 1 \end{pmatrix}$.

Proof Obviously *Aug* preserves regularity, so the stated matrix sequence is $\rightsquigarrow_{(I,I)}$ reduced to

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \{O_i\} \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} =$$
$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \{O_i\} \begin{pmatrix} 0 & 1 \\ 1 & ab \\ 0 & b \end{pmatrix} \rightsquigarrow (I,0)\{O_i\} \begin{pmatrix} 1 & ab \\ 0 & 1 \\ 0 & b \end{pmatrix}.$$

Let $\alpha = (0,b) \begin{pmatrix} 1 & ab \\ 0 & 1 \end{pmatrix}^{-1}$.

Let $G = \begin{pmatrix} I & 0 \\ \alpha & 1 \end{pmatrix}$,

then $(I,0)\{O_i\} \begin{pmatrix} 1 & ab \\ 0 & 1 \\ 0 & b \end{pmatrix} = (I,0)\{O_i\}G^{-1} \begin{pmatrix} 1 & ab \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$

$\rightsquigarrow (I,0)G^{-1}\{O_i\} \begin{pmatrix} I \\ 0 \end{pmatrix} \begin{pmatrix} 1 & ab \\ 0 & 1 \end{pmatrix}$

$= (I,0)\{O_i\} \begin{pmatrix} I \\ 0 \end{pmatrix} \begin{pmatrix} 1 & ab \\ 0 & 1 \end{pmatrix}$.

5 Dynamic Fencing

Let $(\{B_{i,b} : 1 \le i \le l\}, inp)$ be a branching program, where $B_{i,b}$ is of dimension $d_{i-1} \times d_i$; $\{U_i\}$ be a fixed random fence. Let $\underline{x} = (x_1, \cdots, x_n)$ denote the input. We wish using a fence $\{G_{i,\underline{x}} U_i\}$ to evaluate the circuit with this input, where $G_{i,\underline{x}}$ is independently random relative to its use in circuit evaluation for different inputs. In the following we will give a method to approximate this.

Let $\{G_{i,j,b} : 0 \le i \le l, 1 \le j \le n, b = 0,1\}$ be a set of private random invertible matrices of suitable dimensions, our intended $G_{i,\underline{x}}$ will be $\prod_j G_{i,j,x_j}$. We need the following additonal random data:

- $\{A_{i,b} \in GL_{d_i}(F_2) : 1 \le i \le l, b = 0,1\}$
- $\{A_{i,j,1,b} \in GL_{d_i}(F_2) : 1 \le i \le l, inp(i) < j \le n, b = 0,1\}$

- $\{A_{i,j,2,b} \in GL_{d_{i-1}}(F_2) : 1 \le i \le l, 1 \le j \le n, b = 0,1\}$
- $\{B_i \in M_{d_{i-1},d_i}(F_2) : 1 \le i \le l\}$
- $\{U_{i,j} \in GL_{2d_{i-1}+d_i}(F_2) : 1 \le i \le l, -n \le j < 0\}$
- $\{U_{i,j} \in GL_{d_{i-1}+2d_i}(F_2) : 1 \le i \le l, 0 \le j < n\}$

The dynamic fencing of $(\{B_{i,b}\}, inp)$ will be $DF(\{B_{i,b}\}, inp) = \vee_{1 \le i \le l} \mathcal{B}_i$, where $\mathcal{B}_i = \{B_{i,j,b} : -n \le j \le n, b = 0,1\}$:

- $B_{i,-n,b} = U_{i-1}^{-1}(B_i, I, I) \begin{pmatrix} A_{i,n,1,b}^{-1} & & \\ & A_{i,n,2,b}^{-1} & \\ & & G_{i-1,n,b}^{-1} \end{pmatrix} U_{i,-n}$
  if $inp(i) \neq n$; otherwise $B_{i,-n,b} = U_{i-1}^{-1}(B_i, I, I)$
  $\begin{pmatrix} A_{i,n,1,b}^{-1} A_{i,b}^{-1} & & \\ & A_{i,n,2,b}^{-1} & \\ & & G_{i-1,n,b}^{-1} \end{pmatrix} U_{i,-n}$
- for $inp(i) < j < n$, $B_{i,-j,b} = U_{i,-j-1}^{-1} \begin{pmatrix} A_{i,j,1,b}^{-1} & & \\ & A_{i,j,2,b}^{-1} & \\ & & G_{i-1,j,b}^{-1} \end{pmatrix} U_{i,-j}$
- for $j = inp(i)$, $B_{i,-j,b} = U_{i,-j-1}^{-1} \begin{pmatrix} A_{i,j,1,b}^{-1} A_{i,b}^{-1} & & \\ & A_{i,j,2,b}^{-1} & \\ & & G_{i-1,j,b}^{-1} \end{pmatrix} U_{i,-j}$
- for $0 < j < inp(i)$, $B_{i,-j,b} = U_{i,-j-1}^{-1} \begin{pmatrix} I & & \\ & A_{i,j,2,b}^{-1} & \\ & & G_{i-1,j,b}^{-1} \end{pmatrix} U_{i,-j}$
- $B_{i,0,b} = U_{i,-1}^{-1} \begin{pmatrix} A_{i,b} & & \\ & I & \\ & & B_{i,b} \end{pmatrix} U_{i,0}$
- for $0 < j < inp(i)$, $B_{i,j,b} = U_{i,j-1}^{-1} \begin{pmatrix} I & & \\ & A_{i,j,2,b} & \\ & & G_{i,j,b} \end{pmatrix} U_{i,j}$
- for $inp(i) \le j < n$, $B_{i,j,b} = U_{i,j-1}^{-1} \begin{pmatrix} A_{i,j,1,b} & & \\ & A_{i,j,2,b} & \\ & & G_{i,j,b} \end{pmatrix} U_{i,j}$
- $B_{i,n,b} = U_{i,n-1}^{-1} \begin{pmatrix} A_{i,n,1,b} & & \\ & A_{i,n,2,b} & \\ & & G_{i,n,b} \end{pmatrix} \begin{pmatrix} I \\ B_i \\ I \end{pmatrix} U_i$

Lemma 5.1 *$\mathcal{B}_i$ can be used to obtain $(G_{i-1,\underline{x}}U_{i-1})^{-1}B_{i,x_{inp(i)}}G_{i,\underline{x}}U_i$*

Proof The wanted is just

$$(\prod_{1\le j\le n} B_{i,-j,x_j})B_{i,0,x_{inp(i)}}(\prod_{1\le j\le n} B_{i,j,x_j}) = U_{i-1}^{-1}(B_i,I,I)\begin{pmatrix} I & & \\ & I & \\ & & (G_{i-1,\underline{x}})^{-1}B_{i,x_{inp(i)}}G_{i,\underline{x}} \end{pmatrix} \times \begin{pmatrix} I \\ B_i \\ I \end{pmatrix} U_i$$

It can be seen from this computation that, if the adversary deviates from the computation as the lemma, he will get a messed encoding at the middle term: $G_{i-1,\underline{x}}^{-1}B_{i,b_i}G_{i,\underline{x}'} + B_iJ_1 + J_2B_i$, where J_1, J_2 are not all identity. This kind of messed information seems to be hard to exploit for meaningful computation. This means that the encoding of $B_{i,b}$ is bound to the input variable $x_{inp(i)}$, which means that any kind of interleaved computation will result inconsistent encodings, thus hard to derive useful information. These observations seem to support the following conjecture:

Conjecture 5.2 *$DF(\{B_{i,b}\}, inp)$ is computationally equivalent to dynamic fencing $\{KR(\{B_{i,x_{inp(i)}}\}, \{U_{i,\underline{x}}\}) : \underline{x} \in F_2^n\}$ as long as $\min\{d_i\} \ge O(\lambda)$ (i.e. All groups $GL_{d_i}(F_2)$ are complex enough.)*

6 The Obfuscation Scheme

Let $C(x_1, x_2, \cdots, x_n)$ be an NC1 circuit of n input variables. Let (BP, inp) be the regular branching program of C constructed as previous. Let $\tilde{BP} = Aug^{\lambda}(BP)$. Let α, β be a random pair of vectors such that

$$\alpha \begin{pmatrix} I & & \\ & \begin{pmatrix} 1 & v \\ 0 & 1 \end{pmatrix} & \\ & & \end{pmatrix} \beta = v$$

. Then the obfuscation of C is just $DF(\alpha\tilde{BP}\beta, inp)$.

Lemma 6.1 *The scheme has perfect completeness.*

Proof Given $DF(\alpha\tilde{BP}\beta, inp)$, let $\alpha\tilde{BP}\beta = \{\tilde{B_{i,b}}\}$. For input $\underline{x}$ we can compute $KR(\alpha\tilde{B_{i,x_{inp(i)}}}\beta.$ $G_{i,\underline{x}}, U_i)$ whose product value is same as that of $\alpha\tilde{B_{i,x_{inp(i)}}}\beta$, which is $C(x_1, x_2, \cdots, x_n)$.

Lemma 6.2 *Assuming the conjecture, the scheme achieves VBB security.*

Proof Assuming the conjecture, it only needs to prove that the circuit evaluation is simulatable for single input $\underline{x} = (x_1, \cdots, x_n)$. Let $v = C(\underline{x})$,

then KR of $\alpha\{\tilde{B_{i,x_{inp(i)}}}\}\beta$ can be simulated as KR of $\alpha \begin{pmatrix} I & & \\ & \begin{pmatrix} 1 & v \\ 0 & 1 \end{pmatrix} & \\ & & \end{pmatrix} \{O_i\}\beta$.

Even the conjecture is false, it is still possible that the scheme achieves indistinguishability security, we conjecture that this is the case.

For keyed circuits, i.e. the first k input variables are key bits, we can make the branching program slightly shorter. On constructing the BP for all variables, we place the input variables left-first; then we multiply the matrices belonging to key bits to the left. This results in a branching program where each matrix contains information about key bits. When doing dynamic fencing, it should be noted that $rank(B_{i,0} - B_{i,1})$ may disclose key information. This can be avoided by choosing $A_{i,b}$ such that $rank(B_{i,0} - B_{i,1}) + rank(A_{i,0} - A_{i,1})$ is independent of key.

Finally, the complexity of the scheme is obviously bounded by $O(n(d+\lambda)^2|C|)$, where $|C|$ is the circuit size, d is the algebraic degree of the circuit.

7 Conclusion

We provide the construction of an efficient obfuscation scheme whose complexity is bounded by $O(n(d+\lambda)^2|C|)$. A unique characteristic of the scheme is that it does not use multilinear maps. Nevertheless, the scheme relies on a new conjecture to achieve indistinguishability security. How to prove or disprove the conjecture is an open problem yet to be investigated.

References

[1] Prabhanjan Aanauth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. Cryptology ePrint archive, 2014. http://eprint.iacr.org/.

[2] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, A. S. Salil, P. vadahan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[3] Dan Boneh, Sanjam Grag, Amit Sahai, and Mark Zhandry. Differing inputs obfuscation and applications. Cryptology ePrint archive, 2013. http://eprint.iacr.org/, 2013/689.

[4] Sanjam Gary, Craig Gentry, Shai Halevi, Marianna Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[5] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, 1988.

[6] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *STOC*, 2014.

[7] Amit Sahai and Mark Zhandry. Obfuscating low rank matrix branching programs. Cryptology ePrint archive, 2014. http://eprint.iacr.org/.

[8] Joe Zimmerman. How to obfuscate programs directly. In *STOC*, 2014.