Preventing CLT Zeroizing Attacks on Obfuscation

Rex Fernando[*] Peter M. R. Rasmussen[*] Amit Sahai[*]

November 16, 2016

Abstract

We describe a defense against zeroizing attacks on indistinguishability obfuscation (iO) over the CLT13 multilinear map construction. This defense applies to the most recent extension of the attack by Coron et al. (ePrint 2016), under which a much larger class of branching programs is vulnerable. To accomplish this, we distill an essential common component of all previous zeroizing attacks on iO over CLT13. This leads to the notion of a function being "input partionable", meaning that the bits of the function's input can be partitioned into somewhat independent subsets. We find a way to thwart these attacks by requiring a signature structure to be added to the input of every function. The signature eliminates the possibility of finding independent subsets of the input that still leads to more than one valid input, and thus, stops this line of attack. Finally, two concrete instantiations of such signatures are suggested.

We can also apply our defense to a recent extension of attacks by Chen et al (ePrint 2016) on obfuscation in the context of GGH13 construction.

[*]UCLA and Center for Encrypted Functionalities. `{rex, rasmussen, sahai}@cs.ucla.edu`. Research supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

1 Introduction

Indistinguishability obfuscation (iO) has so far relied on multilinear maps for instantiation [GGH$^+$13b] and viable candidates for such are sparse. On top of that, the few that exist [GGH13a, CLT13, GGH15] have all been shown to suffer from significant vulnerabilities. However, not all attacks against these multilinear maps can be applied to iO. The very particular structure that most iO candidates induce puts numerous constraints on the way the encoded values can be combined, thus often not allowing the flexible treatment needed to mount an attack. That said, attacks on iO schemes have nonetheless slowly been found for obfuscation of increasingly general families of functions.

Current Attacks on iO over CLT13. In this paper we focus on the Coron-Lepoint-Tibouchi (CLT13) multilinear maps [CLT13]. The known attacks over CLT13, called *zeroizing attacks* [CHL$^+$14, CGH$^+$15, CLLT16], involve exploiting information which is leaked from a successful zero-test. Each of these attacks require sets of encodings that satisfy a certain structure. Namely, to attack a CLT instance of dimension n, an adversary needs three sets of encodings $\{A_i\}_{i\in[n]}$, $\{B_0, B_1\}$, and $\{C_j\}_{j\in[n]}$ such that for every $i, j \in [n]$ and $\sigma \in \{0,1\}$, $A_i B_\sigma C_j$ is a top-level encoding of zero. In other words, we must be able to vary the choice of encoding in each set independently of the other choices and always produce an encoding of zero. If an adversary is able to obtain such sets, then the adversary is able to factor the modulus of the ciphertext ring, completely breaking the CLT instance.

In the case of applying zeroizing attacks to iO constructions over CLT13, the above requirement leads to an interesting constraint on the behavior of the function being obfuscated. The intuition behind this is that iO schemes are designed so that the only way to achieve an encoding of zero is by performing an honest evaluation of the obfuscated program. The result of this is that the only way to obtain the products of the three sets of encodings described above is to vary the inputs to the obfuscated function. So for any function to be vulnerable to zeroizing attacks over CLT13, the input bit indices of the function must be divisible into three sets, corresponding to the three sets of encodings above. For each set of indices there must be a large set of possible assignments of bit values to those indices which evaluate to zero, and like before we must be able to vary the choices for each set independently of the others. (See the next section for a formal description of this requirement, and the appendix for an explanation of why this follows from the structure of iO constructions over CLT13.)

At first glance this requirement seems very restrictive, and in fact it is even more restrictive than described; the matrices of the obfuscated branching program must be organized in a specific way in relation to the three sets of inputs. Because of this, the first paper applying zeroizing attacks over CLT13 to iO only showed how to apply the attack to very simple branching programs [CGH$^+$15], and attacking more realistic targets seemed out of reach of this technique. However a very recent work by Coron et al. [CLLT16] introduced a simple method that can transform a much larger class of branching programs into ones that have this very specific structure. As such, zeroizing attacks appear much more threatening to the security of iO over CLT13 than previously thought.

Our Contribution. Our aim in this paper is to provide a robust defense against the known classes of zeroizing attacks for iO over CLT13, and to some potential future extensions of these attacks. We do this by showing how to change the input structure of any function f to be obfuscated in such a way that it is impossible to divide its input bit indices into three sets such that the indices in each set can be varied independently. More specifically, given $f\colon \{0,1\}^n \to \{0,\perp\}$ we will construct a function $h\colon \{0,1\}^n \to \{0,1\}^m$ and a new program $f'\colon \{0,1\}^{n+m} \to \{0,\perp\}$ from f such that $f'(s)$ outputs the value of $f(x)$ if and only if the input is of the form $s = xh(x)$. We find a necessary and sufficient condition on h to guarantee input non-independence of f' and present and discuss various possible instantiations of this, i.e. different possibilities for h.

We remark that the attacks in [CLLT16] still do not apply to obfuscations of all branching programs. Specifically, if the branching program is too long compared to the input size then there is a blowup associated with the transformation in [CLLT16] which becomes infeasible. Also, there does not seem to be any straightforward way to apply the attack to dual-input branching programs, due to a similar blowup in complexity. Even if future work reduces the blowup to extend the attacks to these two cases, though, our defense should still hold against them.

It is noteworthy to contrast this line of work with the recent attacks on iO over the GGH13 [GGH13a] multilinear maps construction. In [MSZ16] Miles et al. implement the first known such attacks, which they call *annihilating attacks*. A follow-up paper [GMM$^+$16] give a weakened multilinear map model (by altering the ideal model given in [BGK$^+$14]) and an obfuscation construction in this model which is safe against all annihilating attacks. We stress that the attacks over CLT13 are not related to these annihilating attacks, which are not known to work over CLT13. However, a recent paper attacking obfuscation over GGH13 [CGH16], in which the authors extend annihilating attacks to the original GGHRSW construction of iO, does use an input partition as part of their attack. They do this as a first step in order order to recover a basis of the ideal which defines the plaintext space. Our defense applies to this step of their attack.

As a final note, our defense does not operate in a weak multilinear map model, in contrast to the one defined in [GMM$^+$16]. We leave it as an important open question to develop such a weak multilinear map model for CLT13.

2 Securing Functions against Partition Attacks

Throughout this paper, S_t will denote the set of permutations of $\{1, 2, \ldots, t\}$ and for a bit string $x = x_1 \ldots x_t \in \{0,1\}^t$ we will write $\sigma(x) = x_{\sigma(1)} \ldots x_{\sigma(t)}$ for $\sigma \in S_t$.

2.1 Attack Model

This section seeks to describe the exact attack model we wish to work from. Since we are interested in inputs to programs which result in an output of 0, we consider computable functions of the form $f\colon \{0,1\}^t \to \{0, \perp\}$, where $\perp$ represents any input other than 0. We first formally define the function behavior which is required in order to perform all known zeroizing attacks.

Definition 1 (Input Partition)**.** *Let $f\colon \{0,1\}^t \to \{0,\perp\}$ be a function. An* input partition *for f of degree k is a tuple*

$$\mathcal{I}_f^k = \left(\sigma \in S_t,\ \{a_i\}_{i\in[k]} \subseteq \{0,1\}^l,\ \{c_j\}_{j\in[k]} \subseteq \{0,1\}^m\right)$$

with $a_i \neq a_j$, $c_i \neq c_j$ for all $i, j \in [k]$ and $l + m = t$ such that for all $i, j \in [k]$,

$$f(\sigma(a_i c_j)) = 0.$$

As explained in the introduction, this definition requires that there are sets of input bit indices where the input bits corresponding to each set can be varied independently of the other bits always resulting in a zero output. The use of σ means that the sets of bit indices do not have to be sequential.

Here we use two sets of indices instead of the three required for the actual attack defined in [CGH$^+$15]. This is sufficient since preventing a partitioning of the input into two parts naturally also prevents a partitioning into three parts. We do this to simplify the exposition.

We now describe the attack we will prevent. All known zeroizing attacks on iO over CLT have this step, which exploits the function behavior described in the previous definition. By defending against this attack we defend against all such zeroizing attacks.

Definition 2 (Input Partition Attack). *For each $t \in \mathbb{N}$ let $\mathcal{F}_t$ be a family of functions $f\colon \{0,1\}^t \to \{0,\perp\}$ and let $\mathcal{F} = \{\mathcal{F}_t\}_{t\in\mathbb{N}}$. We shall say that a PPT adversary $\mathcal{A}$ performs an* input partition attack *of degree k on $\mathcal{F}$ if for a non-negligible function ϵ,*

$$\Pr_{w,f\leftarrow\mathcal{F}_t}\left[\mathcal{A}(f) = \mathcal{I}_f^k \text{ such that } \mathcal{I}_f^k \text{ is an input partition of } f \text{ of degree } k\right] > \epsilon(t)$$

where the probability is taken over the randomness w of $\mathcal{A}$ and the choice of f.

Turning the above definition around, we can ensure security against the above type of attack, if the function we obfuscate satisfies the following.

Definition 3 (Input Partition Resistance). *For each $t \in \mathbb{N}$ let $\mathcal{F}_t$ be a family of functions $f\colon \{0,1\}^t \to \{0,\perp\}$ and let $\mathcal{F} = \{\mathcal{F}_t\}_{t\in\mathbb{N}}$. We say that $\mathcal{F}$ is input partition resistant for degree k if no PPT adversary $\mathcal{A}$ successfully performs an independent input attack on f of degree k.*

A stronger version of this is for a function to simply not admit any input partitions which would clearly also make attacks requiring a partition of the input impossible.

Definition 4 (Input Unpartitionable Function). *A function $f\colon \{0,1\}^n \to \{0,\perp\}$ is* input unpartitionable *for degree k if it does not admit an input partition of degree k. If f is input unpartitionable for degree 2, we simply say that it is* input unpartitionable.

2.2 Securing Functions

Having made precise the attack model, we proceed to determine the framework from which we will be working. The basic idea of countering the attacks of the model is to modify the functions being obfuscated such that no input partition attack can succed.

Definition 5 (Completely Securing a Function). *Let $f\colon \{0,1\}^n \to \{0,\perp\}$ and $h\colon \{0,1\}^n \to \{0,1\}^m$ be functions and construct a function $g\colon \{0,1\}^{n+m} \to \{0,\perp\}$ by*

$$g(ab) = \begin{cases} f(a), & h(a) = b \\ \perp, & h(a) \neq b. \end{cases}$$

We say that h completely secures *f if g is input unpartitionable.*

A slightly less strict definition is the following, which does not exclude the existence of partitions, but simply guarantees that such partitions are hard to find.

Definition 6 (Computationally Securing a Function Family). *Let $\mathcal{F} = \{f_t\}_{t\in\mathbb{N}}$ where $f_t\colon \{0,1\}^t \to \{0,\perp\}$ and let $\mathcal{H} = \{\mathcal{H}_t\}_{t\in\mathbb{N}}$ where for each $t \in \mathbb{N}$, $\mathcal{H}_t$ is a family of functions $h\colon \{0,1\}^t \to \{0,1\}^{\ell(t)}$ for a polynomially bounded function ℓ. Further, define for each $t \in \mathbb{N}$ a family of functions $\mathcal{G}_t = \{g_h\}_{h\in\mathcal{H}}$ where $g_h\colon \{0,1\}^{t+\ell(t)} \to \{0,\perp\}$ is the function satisfying*

$$g_h(ab) = \begin{cases} f(a), & h(a) = b \\ \perp, & h(a) \neq b. \end{cases}$$

We say that $\mathcal{H}$ computationally secures *$\mathcal{F}$ if $\mathcal{G}$ is input partition resistant for degree two.*

While we have chosen to work with rather strong definitions that exclude input partitions of degree two, the known attacks following the input partition attack model uses significantly more partitions. So although we only work with the two definitions above in this paper, it is worth defining the following notions for completeness.

Definition 7 (k-securing a function). *Let $f\colon \{0,1\}^n \to \{0,\perp\}$ and $h\colon \{0,1\}^n \to \{0,1\}^m$ be functions and construct a function $g\colon \{0,1\}^{n+m} \to \{0,\perp\}$ by*

$$g(ab) = \begin{cases} f(a), & h(a) = b \\ \perp, & h(a) \neq b. \end{cases}$$

*We say that h k-*secures *f if g is input unpartitionable for degree k.*

Definition 8 (Computationally k-securing a function). *Let $\mathcal{F} = \{f_t\}_{t\in\mathbb{N}}$ where $f_t\colon \{0,1\}^t \to \{0,\perp\}$ and let $\mathcal{H} = \{\mathcal{H}_t\}_{t\in\mathbb{N}}$ where for each $t \in \mathbb{N}$, $\mathcal{H}_t$ is a family of functions $h\colon \{0,1\}^t \to \{0,1\}^{\ell(t)}$ for a polynomially bounded function ℓ. Further, define for each $t \in \mathbb{N}$ a family of functions $\mathcal{G}_t = \{g_h\}_{h\in\mathcal{H}}$ where $g_h\colon \{0,1\}^{t+\ell(t)} \to \{0,\perp\}$ is the function satisfying*

$$g_h(ab) = \begin{cases} f(a), & h(a) = b \\ \perp, & h(a) \neq b, \end{cases}$$

We say that $\mathcal{H}$ computationally k-secures *$\mathcal{F}$ if $\mathcal{G}$ is input partition resistant for degree k.*

2.3 Necessary and Sufficient Conditions

We proceed to find a necessary and sufficient condition on the function h of Definitions 5 and 6. In the following we consider bit strings $x \in \{0,1\}^n$ both as integers, in which case we simply write x, or as vectors over $\mathbb{Z}$, in which case we shall write $\overrightarrow{x} \in \mathbb{Z}^n$. When considered as vectors over $\mathbb{Z}$, addition and subtraction of the bit strings will be component-wise.

Now, for our necessary and sufficient condition, we have the following definitions.

Definition 9. *A function $h\colon \{0,1\}^n \to \{0,1\}^m$ is* safe *if for every $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ it is the case that if*

$$\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} = \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}},$$
$$\overrightarrow{h(x_{1,1})} - \overrightarrow{h(x_{2,1})} = \overrightarrow{h(x_{1,2})} - \overrightarrow{h(x_{2,2})},$$

then $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$.

Definition 10. *Let $\mathcal{H} = \{\mathcal{H}_t\}_{t\in\mathbb{N}}$ where for each $t \in \mathbb{N}$, $\mathcal{H}_t$ is a family of functions $h\colon \{0,1\}^t \to \{0,1\}^{\ell(t)}$ for a polynomially bounded function ℓ. We say that $\mathcal{H}$ is* computationally safe *if it satisfies the following. No PPT adversary $\mathcal{A}$ can with non-negligible probability as a function of t, for h sampled from $\mathcal{H}_t$, produce*

$$\mathcal{A}(h) = (x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}) \in (\{0,1\}^n)^4$$

with $x_{1,1} \neq x_{1,2}$ and $x_{1,1} \neq x_{2,1}$ such that

$$\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} = \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}},$$
$$\overrightarrow{h(x_{1,1})} - \overrightarrow{h(x_{2,1})} = \overrightarrow{h(x_{1,2})} - \overrightarrow{h(x_{2,2})}.$$

We shall soon enough prove that in fact safe and computationally safe functions secure and computationally secure every function according to Definition 5 and 6, respectively. First we prove two lemmas.

Lemma 1. *Let $\sigma \in S_{n+m}$, $a_1, a_2 \in \{0,1\}^n$, and $d_1, d_2 \in \{0,1\}^m$. Further, let*

$$\{t_1, \ldots, t_k\} = T \subseteq [n+m], t_1 < t_2 < \cdots < t_k$$

be a set of indices and $p_T(s) = s_{t_1}s_{t_2}\ldots s_{t_k}$ for a bit string $s \in \{0,1\}^{n+m}$. Then

$$\overrightarrow{p_T(\sigma(a_1d_1))} - \overrightarrow{p_T(\sigma(a_2d_1))} = \overrightarrow{p_T(\sigma(a_1d_2))} - \overrightarrow{p_T(\sigma(a_2d_2))}.$$

Proof. Fix $j \in \{1,2\}$. Writing $\overrightarrow{p_A(\sigma(a_1d_j))} = (e_1, \ldots, e_k)$ and $\overrightarrow{p_A(\sigma(a_2d_j))} = (f_1, \ldots, f_k)$ we get

$$\overrightarrow{y} := \overrightarrow{p_A(\sigma(a_1d_j))} - \overrightarrow{p_A(\sigma(a_2d_j))} = (e_1 - f_1, \ldots, e_k - f_k)$$

Let B be the set of indices i such that the bits e_i, f_i originally came from a_1 and a_2, respectively, and similarly let C be the set of indices i such that the bits e_i, f_i originally came from d_j. Then it is clear that for every $p \in C$, $y_p = e_p - f_p = 0$ since e_p and f_p represent the same bit of d_j. Thus, y is independent of j and our conclusion follows. □

Lemma 2. *Let $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ be given such that*

$$\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} = \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}}. \tag{1}$$

Then there exists

$$\sigma \in S_n,\ a_1, a_2 \in \{0,1\}^k,\ c_1, c_2 \in \{0,1\}^l$$

with $a_1 \neq a_2$, $c_1 \neq c_2$, and $k + l = n$ such that for $i, j \in \{1,2\}$,

$$\sigma(a_i c_j) = x_{i,j}.$$

Proof. In the following denote by $x_{i,j}^p$ the pth bit of $x_{i,j}$.

For $d \in \{-1,0,1\}$ denote by S_d the set of indices s such that the sth component of $\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}}$ is d. We have $\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} \in \{-1,0,1\}^n$ so clearly $S_{-1} \uplus S_0 \uplus S_1 = [n]$. Now, the following relations can be obtained from (1)

$$\begin{aligned}
\forall s \in S_1\colon\ & x_{1,1}^s = x_{1,2}^s = 1 \text{ and } x_{2,1}^s = x_{2,2}^s = 0,\\
\forall s \in S_0\colon\ & x_{1,1}^s = x_{2,1}^s \text{ and } x_{1,2}^s = x_{2,2}^s,\\
\forall s \in S_{-1}\colon\ & x_{1,1}^s = x_{1,2}^s = 0 \text{ and } x_{2,1}^s = x_{2,2}^s = 1.
\end{aligned}$$

Further, by rearranging terms we also have the relation

$$\overrightarrow{x_{1,1}} - \overrightarrow{x_{1,2}} = \overrightarrow{x_{2,1}} - \overrightarrow{x_{2,2}}.$$

and denoting by R_d the set of indices r such that the rth component of $\overrightarrow{x_{1,1}} - \overrightarrow{x_{1,2}}$ is d, we get the similar relations

$$\begin{aligned}
\forall r \in R_1\colon\ & x_{1,1}^r = x_{2,1}^r = 1 \text{ and } x_{1,2}^r = x_{2,2}^r = 0,\\
\forall r \in R_0\colon\ & x_{1,1}^r = x_{1,2}^r \text{ and } x_{2,1}^r = x_{2,2}^r,\\
\forall r \in R_{-1}\colon\ & x_{1,1}^r = x_{2,1}^r = 0 \text{ and } x_{1,2}^r = x_{2,2}^r = 1.
\end{aligned}$$

Clearly, $S_1 \cup S_{-1} \subseteq R_0$ and $R_1 \cup R_{-1} \subseteq S_0$. Write $R_0 = \{r_1, \ldots, r_k\}$ and $S_0 \setminus R_0 = \{s_1, \ldots, s_l\}$ where we note that $k + l = n$, define

$$\begin{aligned}
a_i &= x_{i,1}^{r_1} x_{i,1}^{r_2} \ldots x_{i,1}^{r_k}\\
b_j &= x_{1,j}^{s_1} x_{1,j}^{s_2} \ldots x_{1,j}^{s_l},
\end{aligned}$$

and let $\sigma \in S_n$ be the permutation such that $a_1 b_1 \mapsto x_{1,1}$. Then we must have

$$\sigma(a_2 b_1) = x_{2,1}$$

since $x_{1,j}^{s_1} x_{1,j}^{s_2} \ldots x_{1,j}^{s_l} = x_{2,j}^{s_1} x_{2,j}^{s_2} \ldots x_{2,j}^{s_l}$ by the relation defined by S_0. Similarly, by the relation defined by R_0, $\sigma(a_1 b_2) = x_{1,2}$ and finally, $\sigma(a_2 b_2) = x_{2,2}$ which completes the proof. □

Finally, it is time for the main theorem of the section.

Theorem 1. *The function $h\colon \{0,1\}^n \to \{0,1\}^m$ completely secures every function $f\colon \{0,1\}^n \to \{0,\perp\}$ if and only if it is safe.*

Proof. First, suppose that h completely secures every function f and assume for contradiction that there exists $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ with $x_{1,1} \neq x_{1,2}$ and $x_{1,1} \neq x_{2,1}$ such that

$$\begin{aligned}
\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} &= \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}},\\
\overrightarrow{h(x_{1,1})} - \overrightarrow{h(x_{2,1})} &= \overrightarrow{h(x_{1,2})} - \overrightarrow{h(x_{2,2})}.
\end{aligned}$$

Let f be the function satisfying $f(x) = 0$ for every $x \in \{0,1\}^n$ and consider the function

$$g(ab) = \begin{cases} f(a), & h(a) = b \\ \perp, & h(a) \neq b. \end{cases}$$

Then we clearly have

$$\overrightarrow{x_{1,1}h(x_{1,1})} - \overrightarrow{x_{2,1}h(x_{2,1})} = \overrightarrow{x_{1,2}h(x_{2,1})} - \overrightarrow{x_{2,2}h(x_{2,2})}$$

and it follows by Lemma 2 that there exist

$$\sigma \in S_{n+m},\ a_1, a_2 \in \{0,1\}^k,\ c_1, c_2 \in \{0,1\}^l$$

with $a_1 \neq a_2$, $c_1 \neq c_2$, and $k + l = n + m$ such that for every $i, j \in \{1,2\}$,

$$\sigma(a_i c_j) = x_{i,j}h(x_{i,j}).$$

However, then $g(\sigma(a_i c_j)) = 0$ for every $i, j \in \{1,2\}$ which is a contradiction since there should not exist an input partition of g of degree two if h completely secures f.

Second, suppose that h is safe, such that for all $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ satisfying the equations

$$\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} = \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}}$$
$$\overrightarrow{h(x_{1,1})} - \overrightarrow{h(x_{2,1})} = \overrightarrow{h(x_{2,1})} - \overrightarrow{h(x_{2,2})}$$

either $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$. Now, let $f\colon \{0,1\}^n \to \{0,\perp\}$ be any function, define

$$g(ab) = \begin{cases} f(a), & h(a) = b \\ \perp, & h(a) \neq b, \end{cases}$$

and assume for contradiction that there exists an input partition for g of degree two

$$\mathcal{I}_g^2 = (\sigma \in S_{n+m},\ a_1, a_2 \in \{0,1\}^k,\ c_1, c_2 \in \{0,1\}^l).$$

Write $\sigma(a_i c_j) = x_{i,j}y_{i,j}$ with $|x_{i,j}| = n$ and $|y_{i,j}| = m$ and observe that then $h(x_{i,j}) = y_{i,j}$ for every choice of i, j. Furthermore, we have the equations

$$\overrightarrow{y_{1,1}} - \overrightarrow{y_{2,1}} = \overrightarrow{y_{1,2}} - \overrightarrow{y_{2,2}}$$
$$\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} = \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}}$$

by Lemma 1. Since $h(x_{i,j}) = y_{i,j}$ it follows directly from the condition on h that either $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$. The two cases are symmetric, so assume without loss of generality that $x_{1,1} = x_{1,2}$. Then $y_{1,1} = y_{1,2}$ and we get $\sigma(a_1c_1) = \sigma(a_1c_2)$. A contradiction. $\square$

The computational parallel of the above theorem follows similarly.

Theorem 2. *Let $\mathcal{H} = \{\mathcal{H}_t\}_{t\in\mathbb{N}}$ where for each $t \in \mathbb{N}$, $\mathcal{H}_t$ is a family of functions $h\colon \{0,1\}^t \to \{0,1\}^{\ell(t)}$ for a polynomially bounded function ℓ. The class $\mathcal{H}$ computationally secures every function family $\mathcal{F} = \{f_t\colon \{0,1\}^t \to \{0,\perp\}\}_{t\in\mathbb{N}}$ if and only if it is computationally safe.*

Proof. The proof is similar to the proof of Theorem 1. $\square$

2.4 Instantiations

In terms of actual instantiations of functions h that secure every function f, we present two number theoretical functions – most likely many others exist – and discuss computational instantiations in terms of hash functions.

2.4.1 Number Theoretical Functions

By the necessary and sufficient condition of Theorem 1 and the definition of a safe function, it seems that a function will secure every other function if it is somwhat non-linear everywhere. This is captured in the following corollary, letting us work with functions over the integers.

Corollary 1. *Let $h\colon \{0,1\}^n \to \{0,1\}^m$ be given such that for all $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ satisfying the equations*

$$x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2}$$
$$h(x_{1,1}) - h(x_{2,1}) = h(x_{1,2}) - h(x_{2,2})$$

either $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$. Then h completely secures every function $f\colon \{0,1\}^n \to \{0,\perp\}$.

Proof. This follows immediately from Theorem 1 since $\overrightarrow{x_{1,1}} - \overrightarrow{x_{2,1}} = \overrightarrow{x_{1,2}} - \overrightarrow{x_{2,2}}$ implies $x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2}$ and $\overrightarrow{h(x_{1,1})} - \overrightarrow{h(x_{2,1})} = \overrightarrow{h(x_{2,1})} - \overrightarrow{h(x_{2,2})}$ implies $h(x_{1,1}) - h(x_{2,1}) = h(x_{1,2}) - h(x_{2,2})$. □

Intuitively, many functions we know and love would satisfy this as long as they have sufficient non-linearity. Here we list two examples.

Proposition 1. *Let p be a prime satisfying $2^n < p < 2^{n+1}$. The function $h\colon \{0,1\}^n \to \{0,1\}^{n+1}$ given by $h(x) = [x^2]_p$ completely secures every function $f\colon \{0,1\}^n \to \{0,\perp\}$.*

Proof. Let $x_{1,1}, x_{x_{1},2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ be given satisfying $x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2}$ and $h(x_{1,1}) - h(x_{2,1}) = h(x_{1,2}) - h(x_{2,2})$. We will show that $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$, concluding the proof by Corollary 1.

Directly from the conditions on the $x_{i,j}$, we get

$$\begin{aligned}(x_{1,1} + x_{2,1})(x_{1,1} - x_{2,1}) &\equiv h(x_{1,1}) - h(x_{2,1}) \\ &= h(x_{1,2}) - h(x_{2,2}) \\ &\equiv (x_{1,2} + x_{2,2})(x_{1,2} - x_{2,2}) \pmod{p}.\end{aligned}$$

This yields two cases. If $x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2} = 0$ then $x_{1,1} = x_{2,1}$ and we are done. Otherwise $x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2}$ is invertible modulo p since $p > 2^n$ and we get

$$x_{1,1} + x_{2,1} \equiv x_{1,2} + x_{2,2} \pmod{p}.$$

Adding $x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2}$ to both sides yields $2x_{1,1} \equiv 2x_{1,2} \pmod{p}$ which is equivalent to $x_{1,1} \equiv x_{1,2} \pmod{p}$. Hence, $x_{1,1} = x_{1,2}$ since $p > 2^n$ and we are done. □

Proposition 2. *Let p be a prime satisfying $2^n < p < 2^{n+1}$ with primitive root r. The function $h\colon \{0,1\}^n \to \{0,1\}^{n+1}$ given by $h(x) = [r^x]_p$ completely secures every function $f\colon \{0,1\}^n \to \{0,\perp\}$.*

Proof. Let $x_{1,1}, x_{x_{1},2}, x_{2,1}, x_{2,2} \in \{0,1\}^n$ be given satisfying $x_{1,1} - x_{2,1} = x_{1,2} - x_{2,2}$ and $h(x_{1,1}) - h(x_{2,1}) = h(x_{1,2}) - h(x_{2,2})$. We will show that $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$, concluding the proof by Corollary 1.

Directly from the conditions on the $x_{i,j}$, we get

$$\begin{aligned}r^{x_{1,1}}(1 - r^{x_{2,1} - x_{1,1}}) &\equiv h(x_{1,1}) - h(x_{2,1}) \\ &= h(x_{1,2}) - h(x_{2,2}) \\ &\equiv r^{x_{1,2}}(1 - r^{x_{2,2} - x_{1,2}}) \pmod{p}.\end{aligned}$$

Now we have two cases. First, if $1 - r^{x_{2,1} - x_{1,1}} = 1 - r^{x_{2,2} - x_{1,2}}$ is invertible modulo p then $r^{x_{1,1}} \equiv r^{x_{1,2}} \pmod{p}$, yielding $x_{1,1} = x_{1,2}$ since r has order $p - 1 \geq 2^n$ modulo p. Second, if $1 - r^{x_{2,1} - x_{1,1}} = 1 - r^{x_{2,2} - x_{1,2}}$ is not invertible modulo p then clearly $1 - r^{x_{2,1} - x_{1,1}} \equiv 0 \pmod{p}$ and thus, $x_{2,1} - x_{1,1} = 0$ since the order of r is $\geq 2^n$. It follows that either $x_{1,1} = x_{1,2}$ or $x_{1,1} = x_{2,1}$. □

2.4.2 Hash Functions

The requirements for a function h which k-secures a computation f are reminiscent to that of a hash function. Indeed, to break such a function h it is sufficient to find two separate collisions where the inputs differ by the same bits. It is quite plausible that a wide variety of efficiently computable cryptographic hash functions would satisfy our condition (which is certainly satisfied with overwhelming probability by a random oracle). We leave for future work the study of concrete families of hash functions that are efficiently computable by matrix branching programs and satisfy our condition.

References

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 221–238. Springer, 2014.

[CGH$^+$15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancréde Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New mmap attacks and their limitations. *LNCS*, 9215:247–266, 2015.

[CGH16] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. Cryptology ePrint Archive, Report 2016/998, 2016. `http://eprint.iacr.org/2016/998`.

[CHL$^+$14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehle. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. `http://eprint.iacr.org/2014/906`.

[CLLT16] Jean-Sébastien Coron, Moon Sung Lee, Tancréde Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. *IACR Cryptology ePrint Archive*, 2016.

[CLT13] Jean-Sébastien Coron, Tancréde Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. *LNCS*, 8042:476–493, 2013.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–17. Springer, 2013.

[GGH$^+$13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 40–49. IEEE, 2013.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography Conference*, pages 498–527. Springer, 2015.

[GMM$^+$16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In *Theory of Cryptography Conference*, pages 241–268. Springer, 2016.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 629–658, 2016.

A Branching Program Structure Required for Zeroizing Attacks

Here we describe the form which the attack in [CGH+15] requires obfuscations of matrix branching programs to satisfy in order to work. Recall that a matrix branching program is a sequence of matrices

$$\{A_{i,0}, A_{i,1}\}_{i=1}^{r}$$

along with "bookend" vectors A_0 and A_{r+1} and an input selection function $inp(\cdot)$. For an input x, if x_i is the ith bit of x, then the evaluation of the branching program at that point is the product

$$A_0 \times \prod_{i=1}^{r} A_{i,x_{inp(i)}} \times A_{r+1}.$$

Almost all branching program obfuscators work by randomizing each individual matrix and then encoding them in the multilinear map.

Let

$$M(x) = \widehat{M}_0 \times \prod_{i=1}^{r} \widehat{M}_{i,x_{\mathrm{inp}(i)}} \times \widehat{M}_{r+1}, x \in \{0,1\}^t$$

be an obfuscation of a matrix branching program carried out in this way and partition the input bits as $\mathcal{A} \cup \mathcal{B} \cup \mathcal{C} = [t]$. Suppose one can write $A_x = \widehat{M}_0 \times \prod_{i=1}^{a} \widehat{M}_{i,x_{\mathrm{inp}(i)}}$, $B_x = \prod_{i=a+1}^{b} \widehat{M}_{i,x_{\mathrm{inp}(i)}}$, and $C_x = \prod_{i=b+1}^{r} \widehat{M}_{i,x_{\mathrm{inp}(i)}} \times M_{i,r+1}$ such that the value of A_x, B_x, and C_x rely only on $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, respectively. Further, suppose one can find different $\{x_{i,j}^{\sigma}\}_{\sigma \in \{0,1\} i,j \in [n]} \subseteq M^{-1}(0)$ such that $A_{x_{i,j}^{\sigma}}$ is invariant only of σ and j; $B_{x_{i,j}^{\sigma}}$ is invariant only of i and j; and $C_{x_{i,j}^{\sigma}}$ is invariant only of σ and i. Then the set

$$\left\{ A_{x_{i,j}^{\sigma}} B_{x_{i,j}^{\sigma}} C_{x_{i,j}^{\sigma}} \mid \sigma \in \{0,1\}, i,j \in [n] \right\}$$

is a set of top-level encodings of zero that can be used in the attack as described in the introduction.