Post-Zeroizing Obfuscation

New Mathematical Tools and the Case of Evasive Circuits

Saikrishna Badrinarayanan – UCLA
Eric Miles – UCLA
Amit Sahai – UCLA
Mark Zhandry – MIT & Princeton
Obfuscation [BGIRSVY’01,GGHRSW’13]

Compiler: “scrambles” program, hiding implementation

“Industry accepted” security notion: indist. Obfuscation

\[ P_1(x) = P_2(x) \quad \forall (x) \quad \Rightarrow \quad iO(P_1) \cong_c iO(P_2) \]

“Crypto complete”:

![Diagram showing iO leading to "Most" Crypto]

[GGHRSW’13,SW’13,BZ’13,BST’13,GGHR’13,BP’14,HJKSWZ’14,CLTV’14,...]
Multilinear Maps (a.k.a. graded encodings) [BS’03, GGH’13, CLT’13, GGH’15]

Main tool for all constructions of obfuscation

Levels $1, \ldots, k$, Field/Ring $F$

<table>
<thead>
<tr>
<th>Secret</th>
<th>public</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \in F$, $i \in [k]$</td>
<td>$[a]_i \times [b]_j$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>+</th>
<th>$[a+b]_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[a]_i + [b]_i$</td>
<td>$[a+b]_i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\times$</th>
<th>$[ab]_{i+j}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[a]_i \times [b]_j$</td>
<td>$[ab]_{i+j}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IsZero</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[a]_k$</td>
<td>$\text{IsZero}$</td>
</tr>
</tbody>
</table>
**Multilinear Maps** (a.k.a. graded encodings) [BS’03, GGH’13, CLT’13, GGH’15]

**k** levels: compute arbitrary degree **k** polynomials

Asymmetric mmaps: additional restrictions
  • E.g. multilinear polynomials

Note: current mmaps not ideal
  • Non-unique encodings
  • Encodings may leak op’s that created them
    • Ex: \([a+b]_i \times [c]_j\) vs \([ac]_{i+j} + [bc]_{i+j}\)

• Solution: “re-randomize” by adding encodings of zero
Obfuscation From Multilinear Maps

Obfuscate(P):

\[ \text{Eval}(x): \]

\[ \text{Enc} \]

\[ [p_x]_k \]

\[ \text{IsZero} \]

\[ P(x) = 1 \iff p_x = 0 \]
Applications of Multilinear Maps
“Zeroizing” Attacks on MMaps
“Zeroizing” Attacks on MMaps

(Note: apps still possible using obfuscation)
Central Questions

Q1: Is obfuscation secure?

Q2: If so, how to show it?
## Flavors of Obfuscation

<table>
<thead>
<tr>
<th>Branching Program Obfuscation</th>
<th>NC¹ Obfuscation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GGHRSW’13, BR’14, BGKPS’14, PST’14, GLSW’14, AGIS’14, MSW’14,…]</td>
<td>[Zim’14, AB’15, BD’16]</td>
</tr>
<tr>
<td>• Better understood ✔</td>
<td>• Generally more efficient ✔</td>
</tr>
<tr>
<td>• Conceptually simpler? ✔</td>
<td>• Directly handle NC¹ circuits ✔</td>
</tr>
<tr>
<td>• Prime order mmaps ✔</td>
<td>• Composite-order mmaps ✗</td>
</tr>
<tr>
<td>• Need NC¹ → BP ✗</td>
<td></td>
</tr>
</tbody>
</table>

Boost to obfuscation for all circuits [GGHRSW’13, App’13,…]
This Work: BP Obfuscation

<table>
<thead>
<tr>
<th>Branching Program Obfuscation</th>
<th>NC¹ Obfuscation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GGHRSW’13,BR’14,BGKPS’14,PST ’14,GLSW’14,AGIS’14,MSW’14,...]</td>
<td>[Zim’14,AB’15,BD’16]</td>
</tr>
<tr>
<td>• Better understood</td>
<td>✓</td>
</tr>
<tr>
<td>• Conceptually simpler?</td>
<td>✔</td>
</tr>
<tr>
<td>• Prime order mmaps</td>
<td>✔</td>
</tr>
<tr>
<td>• Need NC¹ → BP</td>
<td>✘</td>
</tr>
<tr>
<td>• Generally more efficient</td>
<td>✓</td>
</tr>
<tr>
<td>• Directly handle NC¹ circuits</td>
<td>✔</td>
</tr>
<tr>
<td>• Composite-order mmaps</td>
<td>✘</td>
</tr>
</tbody>
</table>

Boost to obfuscation for all circuits [GGHRSW’13,App’13,...]
How To Argue Security of iO Candidates?

Option 1: Reduction to “simple” assumption
- Easier to analyze assumption than candidate
- Several ways to do this obfuscation:
  - Directly: [GLSW’15]
  - Through FE: [AJ’15, BV’15], [GGHZ’16]
- Currently only known for BP obfuscation

Essentially all “simple” assumptions broken by zeroizing attacks
How To Argue Security of iO Candidates?

Option 2: Argue security in ideal mmap model

- [BR’14,BGKPS’14,PST’14,Zim’14,AB’15,...]
- Prove that no “generic” attacker exists (i.e. one that only interacts with system through interfaces)
- May be reasonable if no non-generic attacks known
  - Ex: random oracle model, generic group model for ECC

Zeroizing attacks are non-generic, so ideal mmap model no longer compelling
Zeroizing Attacks (for \[GGH’13, CLT’13\])

\[GGH’13, CHLRS’15, BWZ’14, CGHLMRST’15, HJ’15, BGHST1’5, Hal’15, CLR’15, MF’15, MSZ’16a\]

Implications:
- “Simple” assumptions & most direct applications broken
- Notable exceptions: some iO, WE, ORE candidates
Post-Zeroizing Security?

**Option 1: avoid zeros entirely**
- New ideal model: zero gives complete break

**Option 2: Analyze structure of zeros obtained**
- More refined ideal model [CGHLMRST’15, MSZ’16a]
- Subject of follow-up works [MSZ’16a, GMS’16, MSZ’16b]

**This work: Focus on 1, gives tools for 1 & 2**
Post-Zeroizing Security?

Option 1: avoid zeros entirely
- New ideal model: zero gives complete break

Option 2: Analyze structure of zeros obtained
- More refined ideal model [CGHLMMRST’15, MSZ’16]
- Subject of follow-up works [MSZ’16a, GMS’16, MSZ’16b]

This work: Focus on 1, gives tools for 1 & 2

Going forward, must figure out when adversary can get zeros, what the zeros “look like”
Limitations of Prior Security Arguments

Prior works prove following theorem:

**Thm ([BR’14,BGPKS’14,AGIS’14]):** View of generic adversary (in old model) can be simulated with black box access to $P$

In particular, if $P_1(x) = P_2(x) \forall (x)$, views are the same in old model (actually get VBB obfuscation in old model)

**Problem:** analysis gives no indication of when an adversary can find zeros, what the zeros look like
Our Main Result

We give a new obfuscator from mmmaps
• Construction very similar to prior works

Brand new analysis:
• Let $p_x$ be element that is zero-tested when running $P(x)$

**Thm** (This work, informal): Only zeros adversary can obtain are $[p_x]_k$ for known accepting $x$

Holds for any “level respecting” model
Implications: Post-Zeroizing Security

Immediate corollary:

**Corollary:** If $P$ is evasive (hard to find accepting input), can never find a zero $\Rightarrow$ (VBB) security in zero-avoiding model

Fist compelling post-zeroizing security argument for evasive function obfuscation
- Subsequent work: similar result for NC$^1$ obfuscation [BD’16]

Also crucially used in [GMS’16,MSZ’16b]: Obfuscator for all programs secure in refined ideal model
- Captures all known attacks
Implications: Efficiency Improvements

Prior analysis: security for “full rank” BP’s only
- Puts constraints on NC¹ → BP conversion
- Can’t directly handle automata

Our Analysis: security for “essentially all” BPs
- Allows for much more efficient NC¹ → BP conversion
  - Still not quite as efficient as direct NC¹ obfuscators
- Can directly handle automata
- Tools useful in other settings [BLRSZZ’15]

Improved security analysis ⇒ improved efficiency
Proof Overview

Consider arbitrary polynomial of encoded terms
Proof Overview

Consider arbitrary polynomial of encoded terms
Proof Overview

Consider arbitrary polynomial of encoded terms
Proof Overview

Consider arbitrary polynomial of encoded terms

Our analysis:
\[ p_x \neq 0 \]

Tools:
- Prior characterization of level-respecting polys \([\text{BGKPS'14,MSW'14}]\)
- Schwartz-Zippel \(\Rightarrow\) anything but \(p_x\), gives non-zero whp
Summary

New tools for analyzing obfuscation

• First obfuscation for evasive functions with compelling “post-zeroizing” security arguments

• Improved efficiency of BP obfuscation

• Basis for subsequent results

Thanks!