Obfuscation and Weak Multilinear Maps

Mark Zhandry – Princeton University

Joint work with Saikrishna Badrinarayanan , Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan

Obfuscation [BGIRSVY’01,GGHRSW’13]

Compiler: “scrambles” program, hiding implementation

“Industry accepted” security notion: **indist. Obfuscation**

$$P_1(x) = P_2(x) \ \forall(x) \ \Rightarrow \ iO(P_1) \approx_c iO(P_2)$$

“Crypto complete”:

[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,…,k**, Field/Ring **F**

secret: $a \in F,\ i \in [k] \xrightarrow{\textbf{Enc}} [a]_i$

public:

$[a]_i + [b]_i \longrightarrow [a+b]_i$

$[a]_i \times [b]_j \longrightarrow [ab]_{i+j}$

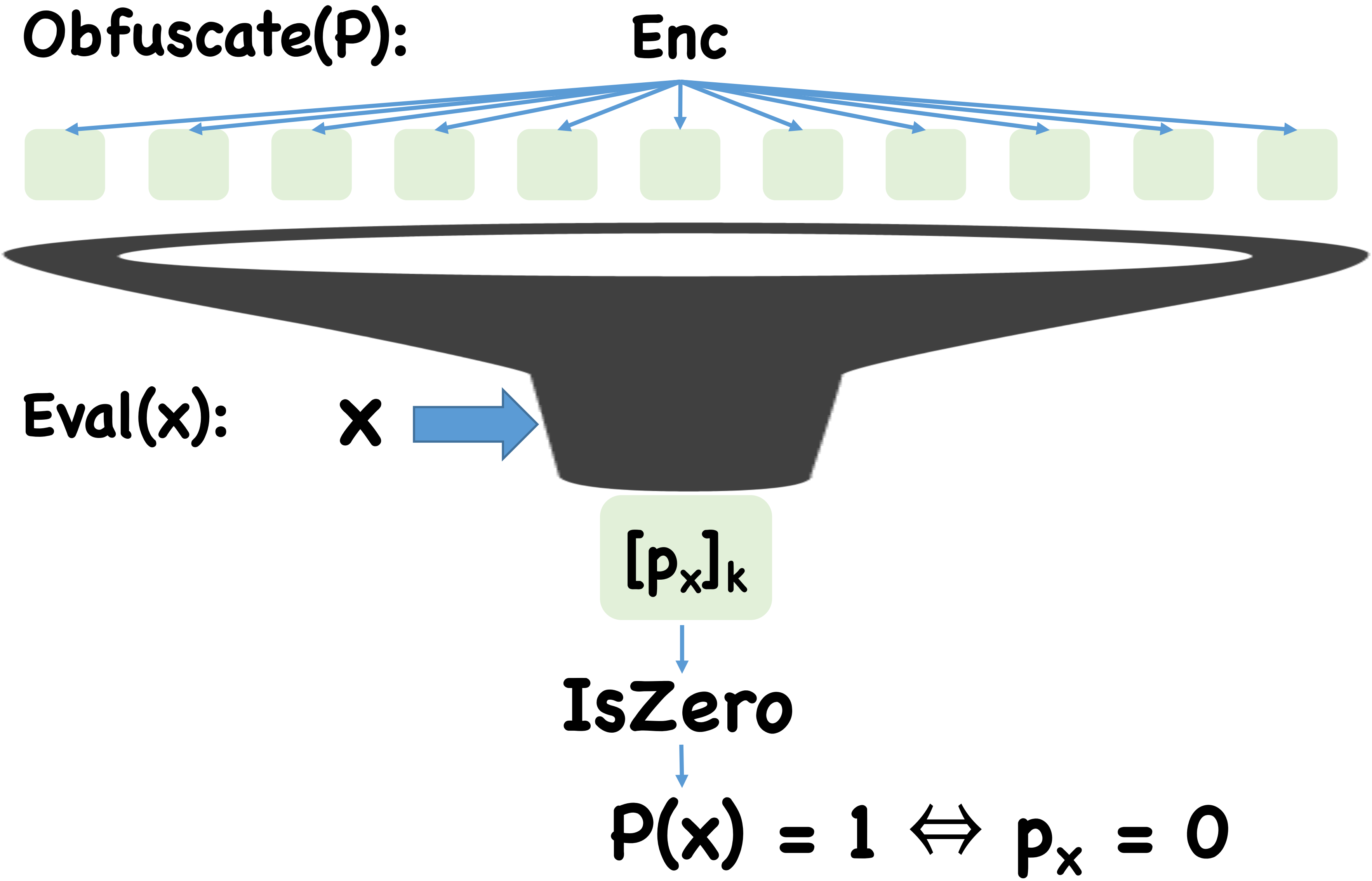
$[a]_k \xrightarrow{\textbf{IsZero}}$ **Yes/No**

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

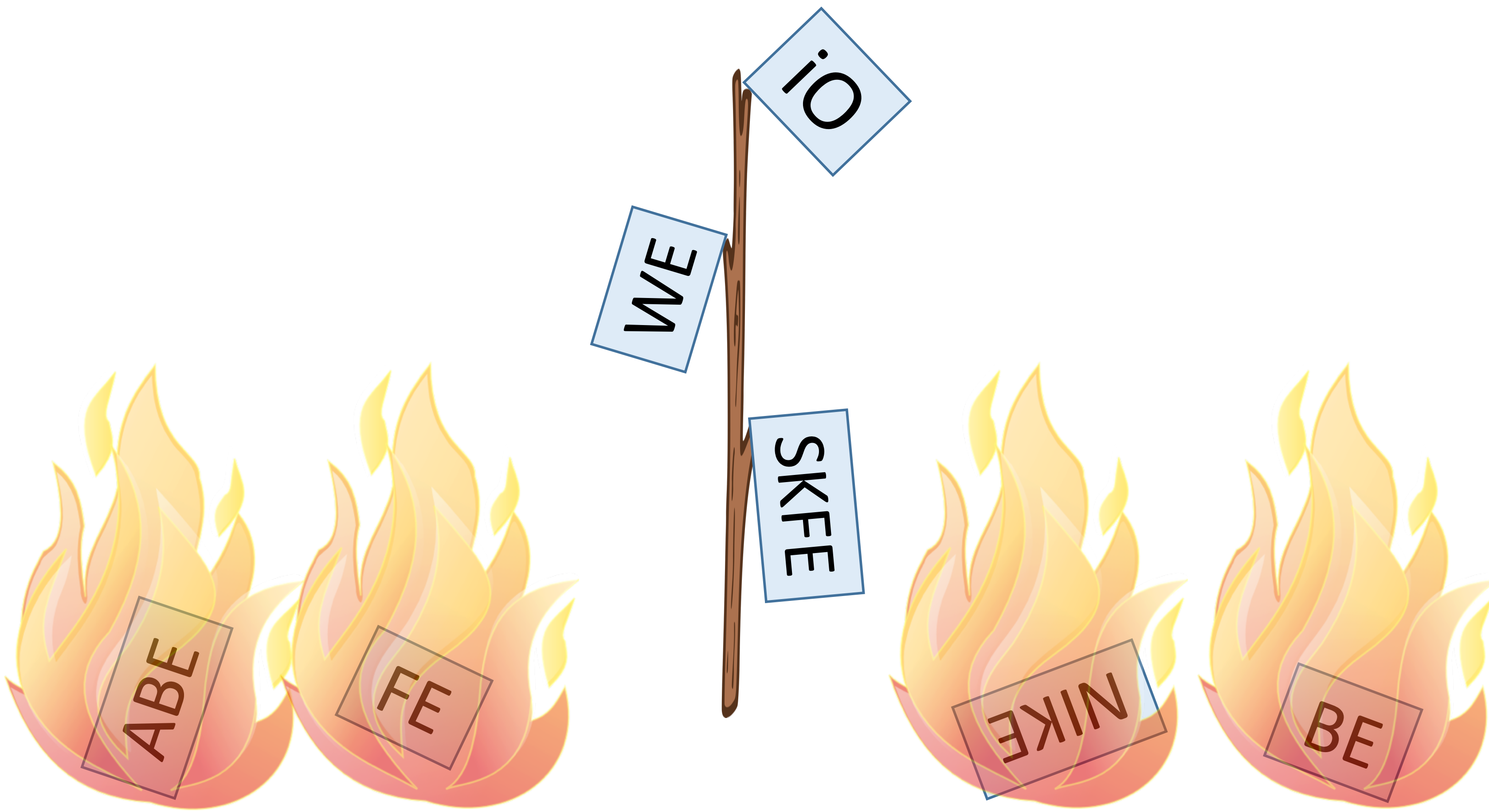
Obfuscation From Multilinear Maps



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?

This Work: Focus on GGH'13 Mmaps

Background…

High-level description GGH13

Level **i** encoding of **x**: $\dfrac{\mathbf{x} + \mathbf{g}\,\overbrace{\mathbf{s}}^{\text{“short”}}}{\mathbf{z}^{\mathbf{i}}} \pmod{\mathbf{q}}$

High-level description GGH13

Level **i** encoding of **x**: $\dfrac{x + g\underbrace{s}_{\text{“short”}}}{z^i} \pmod{q}$

- Add within levels

$$\frac{x_1+gs_1}{z^i} + \frac{x_2+gs_2}{z^i} = \frac{(x_1+x_2)+g(s_1+s_2)}{z^i}$$

High-level description GGH13

Level **i** encoding of **x**: $\frac{x + g\underbrace{s}_{\text{“short”}}}{z^i} \pmod{q}$

- Add within levels
- Multiplication makes levels add

$$\frac{x_1+gs_1}{z^i} \cdot \frac{x_2+gs_2}{z^j} = \frac{(x_1x_2)+g(s_1x_2+s_2x_1+gs_1s_2)}{z^{i+j}}$$

High-level description GGH13

Level **i** encoding of **x**: $\dfrac{x + g\underbrace{s}_{\text{“short”}}}{z^i} \pmod{q}$

- Add within levels
- Multiplication makes levels add
- Test for zero at “top level” **k**

Public parameter $p_{zt} = \dfrac{h\, z^k}{g}$ “not too big”

$p_{zt} \dfrac{gs}{z^k} = hs$ “not too big”

$p_{zt} \dfrac{x+gs}{z^k} = \dfrac{hx}{g} + hs$ “big”

High-level description GGH13

Level **i** encoding of **x**: $\frac{x + g\,\overbrace{s}^{\text{“short”}}}{z^i} \pmod{q}$

- Add within levels
- Multiplication makes levels add
- Test for zero at “top level” **k**

Notes:

- **z** must be secret (else can go down levels)
- **g** must be secret ([GGH’13] show attack otherwise)

Encodings of Zero – A Necessary Evil

Required for (Most) Applications

“Re-randomization”

- Needed for most (direct) applications
- Needed to use any “simple” assumption on mmaps

Add random subset of low-level zeros

Successful zero test $\Rightarrow$ top level zero

Encodings of Zero – A Necessary Evil

Required for (Most) Applications

Two low-level zeros:

$$\mathbf{e_1 = g\ s_1/z^i,\ e_2 = g\ s_2/z^{k-i}}$$

$$\Downarrow$$

Multiple of **g**

$$\mathbf{p_{zt}e_1e_2 \bmod q = hgs_1s_2}\ \text{(over } \mathbb{Z}\text{)}$$

Dangerous For Security

Encodings of Zero – A Necessary Evil

Required for (Most) Applications

Zeroizing attacks:

- GGH'13: "Source group" assumptions (e.g. DLin, Subgroup decision) are false
- CGHLMMRST'15: Immunizations don't work
- HJ'16: MDDH is false, multiparty NIKE broken
- Probably other assumptions broken too (MDHE, etc)

Dangerous For Security

Encodings of Zero – A Necessary Evil

Required for (Most) Applications

Dangerous For Security

What about Obf/WE/SKFE?

Good News:

No re-randomization needed in application

no low-level zeros (explicit or implicit)

Bad News:

Top level zeros may still be generated during use

Re-rand still needed for “simple” assumptions

Central Questions (Restated)

Q1: Can top-level zeros be used to attack iO?

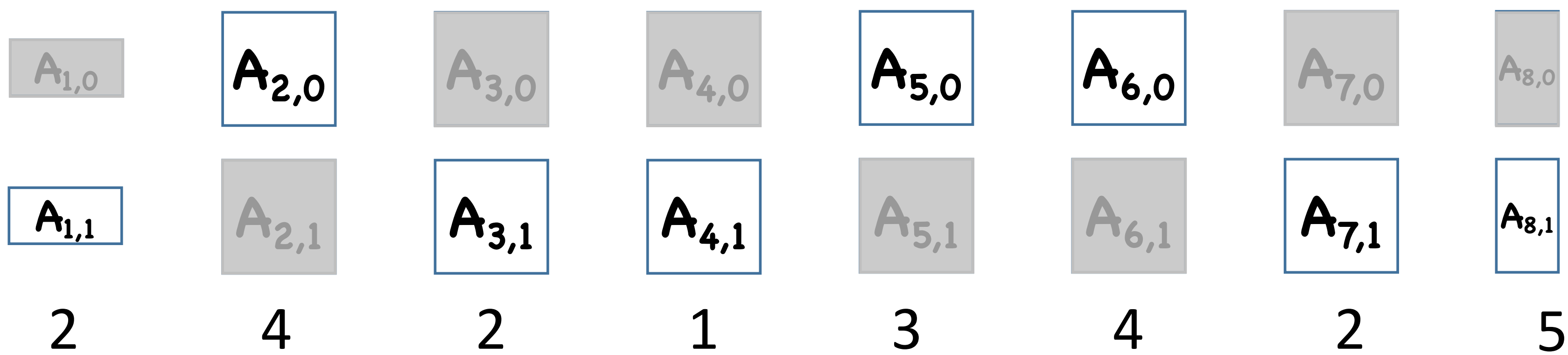
Q2: How to argue security against zeroizing attacks?

Q1: Affirmative!

Thm* [MSZ’16]: The branching program obfuscators in [BGKPS’14, PST’14, AGIS’14, BMSZ’16] over GGH’13 do not satisfy iO

*Small heuristic component

(Single input) Branching Programs



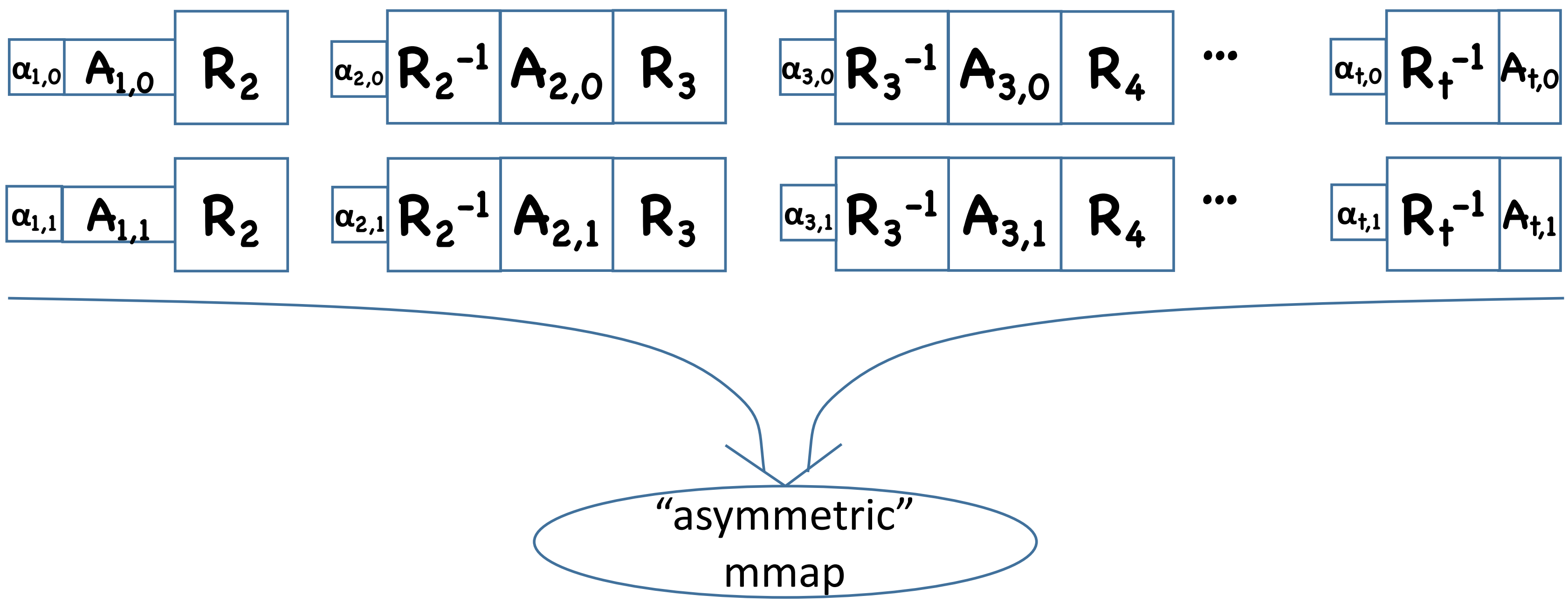
x = 11001:

$$\mathbf{IMP}_x(\{A_{i,b}\}) = A_{1,1}\,A_{2,0}\,A_{3,1}\,A_{4,1}\,A_{5,0}\,A_{6,0}\,A_{7,1}\,A_{8,1}$$

If $\mathbf{IMP}_x$ **= 0**, output **1**, otherwise output **0**

[BMSZ'16] Obfuscator

Building on [GGHRSW'13,BR'14,BGKPS'14,AGIS'14,...]



[BMSZ'16] Over GGH'13

Randomized Branching Program

$$\boxed{B_{i,b}} = \boxed{\alpha_{i,b}}\boxed{R_i^{-1}}\boxed{A_{i,b}}\boxed{R_{i+1}}$$

Encoding randomness

$$\boxed{S_{i,b}}$$

Obfuscation encodings

$$\boxed{C_{i,b}} = \frac{\boxed{B_{i,b}} + \boxed{g}\boxed{S_{i,b}}}{\boxed{z_{i,b}}} \bmod q$$

Evaluation:

$$T_x = \boxed{p_{zt}} \times IMP_x\left(\boxed{C_{i,b}}\right) \bmod q$$

test if "not too big"

Attack: Annihilating Polynomials

$$T_x = \boxed{p_{zt}} \times IMP_x\left(\boxed{c_{i,b}}\right) \bmod q$$

$$= \frac{\boxed{h}}{\boxed{g}} \times IMP_x\left(\boxed{B_{i,b}} + \boxed{g}\boxed{S_{i,b}}\right) \bmod q$$

$$= \frac{\boxed{h}}{\boxed{g}} \times IMP_x\left(\boxed{B_{i,b}}\right) + D_x\left(\boxed{\alpha_{i,b}}, \boxed{S_{i,b}}, \boxed{R_i}\right)$$

$$+ \boxed{g} \times E_x\left(\boxed{\alpha_{i,b}}, \boxed{S_{i,b}}, \boxed{R_i}\right) \bmod q$$

Attack: Annihilating Polynomials

Suppose $\mathbf{P(x) = 1} \Rightarrow \mathbf{IMP_x(\boxed{B_{i,b}}) = 0}$

$$T_x = \frac{\boxed{h}}{\boxed{g}} \times \cancel{IMP_x(\boxed{B_{i,b}})} \underbrace{+ D_x(\boxed{\alpha_{i,b}}, \boxed{S_{i,b}}, \boxed{R_i}) + \boxed{g} \times E_x(\boxed{\alpha_{i,b}}, \boxed{S_{i,b}}, \boxed{R_i})}_{\text{“not too big”, so holds over } \mathbb{Z}} \cancel{\bmod q}$$

Attack: Annihilating Polynomials

Suppose $\mathbf{P(x) = 1}$

$$T_x = D_x(\boxed{\alpha_{i,b}}, \boxed{S_{i,b}}, \boxed{R_i}) + \boxed{g} \times E_x(\boxed{\alpha_{i,b}}, \boxed{S_{i,b}}, \boxed{R_i})$$

Efficiency:
Poly-many free vars

Exp-many inputs:
Pick larger poly set of D_x

Algebraic dependence:
$\exists$ poly Q: $Q(D_{x1}, D_{x2}, \ldots) = 0$

Attack: Annihilating Polynomials

Algebraic dependence:
$\exists$ poly $\mathbf{Q}$: $Q(D_{x1}, D_{x2}, \dots) = 0$

Annihilating polynomial

$$
\begin{aligned}
Q(T_{x1}, T_{x2}, \dots) &= Q(D_{x1}+gE_{x1}, D_{x2}+gE_{x2}, \dots) \\
&= Q(D_{x1}, D_{x2}, \dots) + gQ' + g^2Q'' + \dots \\
&= gQ' + g^2Q'' + \dots
\end{aligned}
$$

Multiple of **g**

Attack: Annihilating Polynomials

Goal: find **Q** that annihilates $\mathbf{P_1}$, but not $\mathbf{P_2}$

$\Downarrow$

Distinguishing Attack*

Extends to any "purely algebraic" obfuscator

Problem: in general, annihilation is hard

Thm ([Kay'09]): Unless PH collapses, there are dependent polys for which an annihilating polynomial requires super-polynomial sized circuits

Question: Can annihilating polys be found for particular obfuscators/programs?

* Additional work needed to test for multiple of **g**

Attack: Annihilating Polynomials

Consider "single-input" setting (used to prove iO)
Suppose "trivial" branching program: $\mathbf{A_{i,0}=A_{i,1}=A_i}$

Explicit annihilating polynomial for [BMSZ'16]:

$$\begin{aligned}\mathbf{q} = {} & (D_{000}D_{111})^2 + (D_{001}D_{110})^2 + (D_{010}D_{101})^2 + (D_{100}D_{011})^2 \\ & - 2D_{000}D_{111}D_{001}D_{110} - 2D_{000}D_{111}D_{010}D_{101} - 2D_{000}D_{111}D_{100}D_{011} \\ & - 2D_{001}D_{110}D_{010}D_{101} - 2D_{001}D_{110}D_{100}D_{011} - 2D_{010}D_{101}D_{100}D_{011} \\ & + 4D_{000}D_{011}D_{101}D_{111} + 4D_{111}D_{001}D_{010}D_{100}\end{aligned}$$

Computed by reducing problem to finite size, then brute-force search

Attack: Annihilating Polynomials

For dual input:

- First, reduce problem to finite size
- Brute-force annihilating poly in constant time
- Haven't found it yet, but still gives poly-time attack

Other obfuscators:

- [BR'14,BGKPS'14, PST'14, AGIS'14]: similar analysis

Also attack ORE (SKFE) [BLRSZZ'15] over GGH'13

Now What?

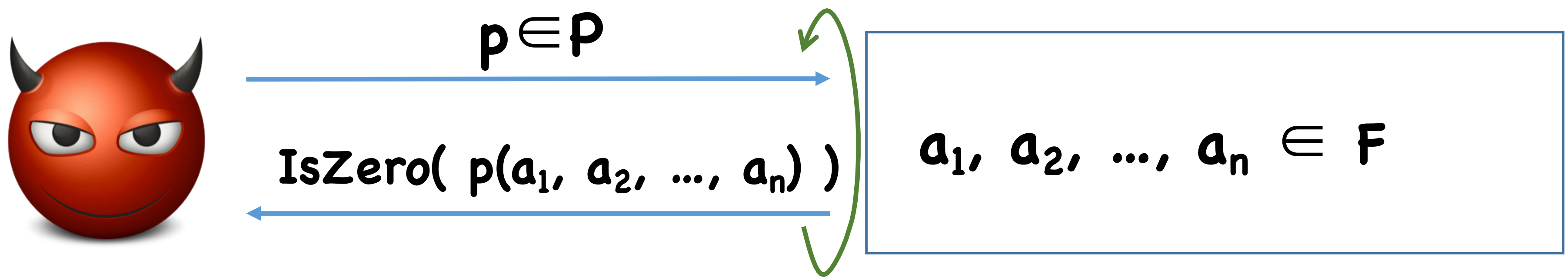
Goal: Argue security of other schemes

Problem: Cannot use “simple” assumptions

Solution: Argue security in abstract attack models

Restricted Black Box Fields

F = Field, **P** = class of polynomials on **n** variables



Generic Groups*: **P** = { Linear functions }
Black Box Fields*: **P** = { Polys with small algebraic circuits}
Symmetrix multilinear maps*: **P** = { Degree **k** polynomials}
Asymmetric multilinear maps*: **P** = { More complicated restrictions}

* Often need greater functionality requirements for protocols. This model suffices for our discussion

Obfuscation in Restricted BBFs

(model used by [BR'14,BGKPS'14,AGIS'14,Z'15,AB'15,BMSZ'16])

Obfuscate(C):

BBF with restricted polynomial class **P**

$p_x \in P$

$a_1, a_2, \ldots, a_n \leftarrow D_C$

Eval(x): IsZero($p(a_1, a_2, \ldots, a_n)$)

- If **IsZero** gives "True", output 1
- If **IsZero** gives "False", output 0

Our Attack: Model is false for GGH'13

A Conservative Model [BMSZ'16]

$p \in \mathbf{P}$

BBF with restricted polynomial class $\mathbf{P}$

$a_1, a_2, \ldots, a_n \leftarrow D_C$

If **IsZero**($p(a_1, a_2, \ldots, a_n)$),

$\Downarrow$

ADVERSARY WINS

Obfuscation for evasive functions [BMSZ'16]

Honest executions always give non-zero

Thm([BMSZ'16]): Only way for "level respecting" adversary to get zero is through honest program executions

Impossible to find zeros anywhere for evasive funcs

Compare to prior "abstract model" theorems:

Thm([BR'14, BGKPS'14, …]): For "level respecting" adversary, can guess output of **IsZero** just by knowing **P(x)**

Doesn't say if/when finding a zero is possible

A Conservative Model [BMSZ'16]

$p \in \mathbf{P}$

BBF with restricted polynomial class $\mathbf{P}$

$a_1, a_2, \ldots, a_n \leftarrow D_C$

If **IsZero**($p(a_1, a_2, \ldots, a_n)$),

$\Downarrow$

ADVERSARY WINS

Model useless in "non-evasive" settings, e.g. iO, SKFE

Need model that allows for zeros to occur

Characterizing Attacks

<u>All Known Classical Attacks</u>

Compute polynomials obeying level restrictions

Several top level zero encodings

Attack

Characterizing Attacks

<u>All Known Classical Attacks</u>

Compute polynomials obeying level restrictions

Several top level zero encodings

Polynomial in the zeros

Multiple of **g**

Attack

Refined Abstract Model for Mmap attacks

$p \in P$

$a_1, a_2, \ldots, a_n \in F$

$s_1, s_2, \ldots, s_n \leftarrow\$ F$

indeterminant

Write $p(a_1+gs_1, \ldots, a_n+gs_n)$
$= c + dg + \ldots$

If $c \neq 0$, output "False"

If $c = 0$, output "True", d

Efficient polys*

$q \in Q$

Unrestricted BBF

$d_1\ d_2\ d_3 \ldots$

If $q(d_1, d_2, \ldots) = 0$,
adversary wins

* Also need to assume degree $<< |F|$

Refined Abstract Model for Mmap attacks

- Seems to capture intuition behind attacks

Proof in refined model → Heuristic evidence of security against current attacks

But keep in mind that:

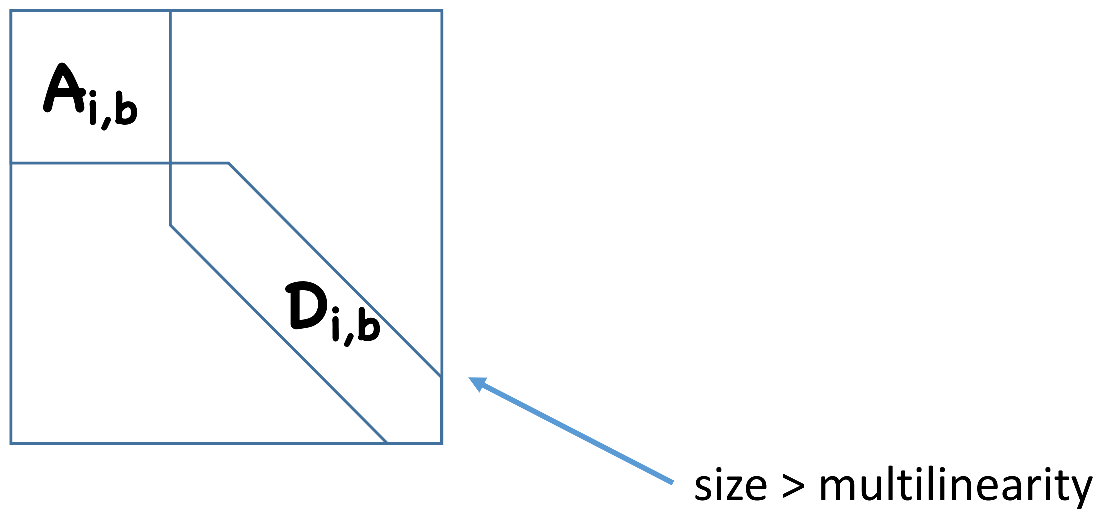
Attack in refined model ~~→~~ Attack on actual protocol

Not trivially

Blocking Attacks [GMMSSZ'16]

Notably absent from attacked schemes: [GGHRSW'13]

$A'_{i,b}$:



Random diagonal converts even "trivial" branching programs into non-trivial ones

Blocking Attacks [GMMSSZ'16]

Our fix: append random block matrix

$$A'_{i,b}: \quad \left[\begin{array}{c|c} A_{i,b} & \\ \hline & E_{i,b} \end{array}\right]$$

Potentially as small as 2×2

Blocking Attacks [GMMSSZ'16]

Let $\mathbf{BP_E}$ be branching program defined by $\mathbf{E}$ matrices
Let $\mathbf{E_x}$ be evaluation of $\mathbf{BP_E}$ on input $\mathbf{x}$

Thm: If polynomial $\mathbf{Q}$ annihilates $\{\mathbf{D_x}\}$*,
then it annihilates $\{\mathbf{E_x}\}$ as well

Let $\mathbf{BP_F}$ be any $\mathbf{BP}$, $\mathbf{F_x}$ evaluation of $\mathbf{BP_F}$

Thm: If polynomial $\mathbf{Q}$ annihilates $\{\mathbf{E_x}\}$*,
then it annihilates $\{\mathbf{F_x}\}$ as well

* = with noticeable probability

Example Proof Sketch

Thm: If polynomial **Q** annihilates $\{E_x\}$*, then it annihilates $\{F_x\}$ as well

Let $E_{i,b} = F_{i,b} + r\ E'_{i,b}$

random

$\Rightarrow E_x = F_x + r\ F'_x + r^2\ F''_x + \ldots$

$\Rightarrow Q(\ \{E_x\}\ ) = Q(\ \{F_x\}\ ) + r\ Q' + r^2\ Q'' + \ldots$

By Schwartz-Zippel, if $\Pr[Q = 0] = \text{non-negl}$,
Then **Q** must be identically **0**

$\Rightarrow Q(\ \{F_x\}\ ) = 0$

Branching Program Unannihilateability

Assumption: For any efficient polynomial **Q***, there is a branching program not annihilated by **Q**

"Easy" fact: PRFs in NC^1 give unannihilateable branching programs

Corollary: Assuming BPUA (or NC^1 PRFs), our obfuscator is secure in the weak mmap model for [GGH'13]

* of not too high degree

Future Directions

- Substantiate BPUA (P $\neq$ NP, general OWF, etc)
- Attack GGH'13 without annihilating polys
- Extend to obfuscation for circuits
  - Mostly solved: [DGGMM'16] assuming NC^1 PRFs
- Extend attacks to CLT'13, GGH'15
  - Partial progress: [CLLT'16] for single-input iO over CLT'13
- Useful abstract attack model for CLT'13, GGH'15

Thanks!