# 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**

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

"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

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

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

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

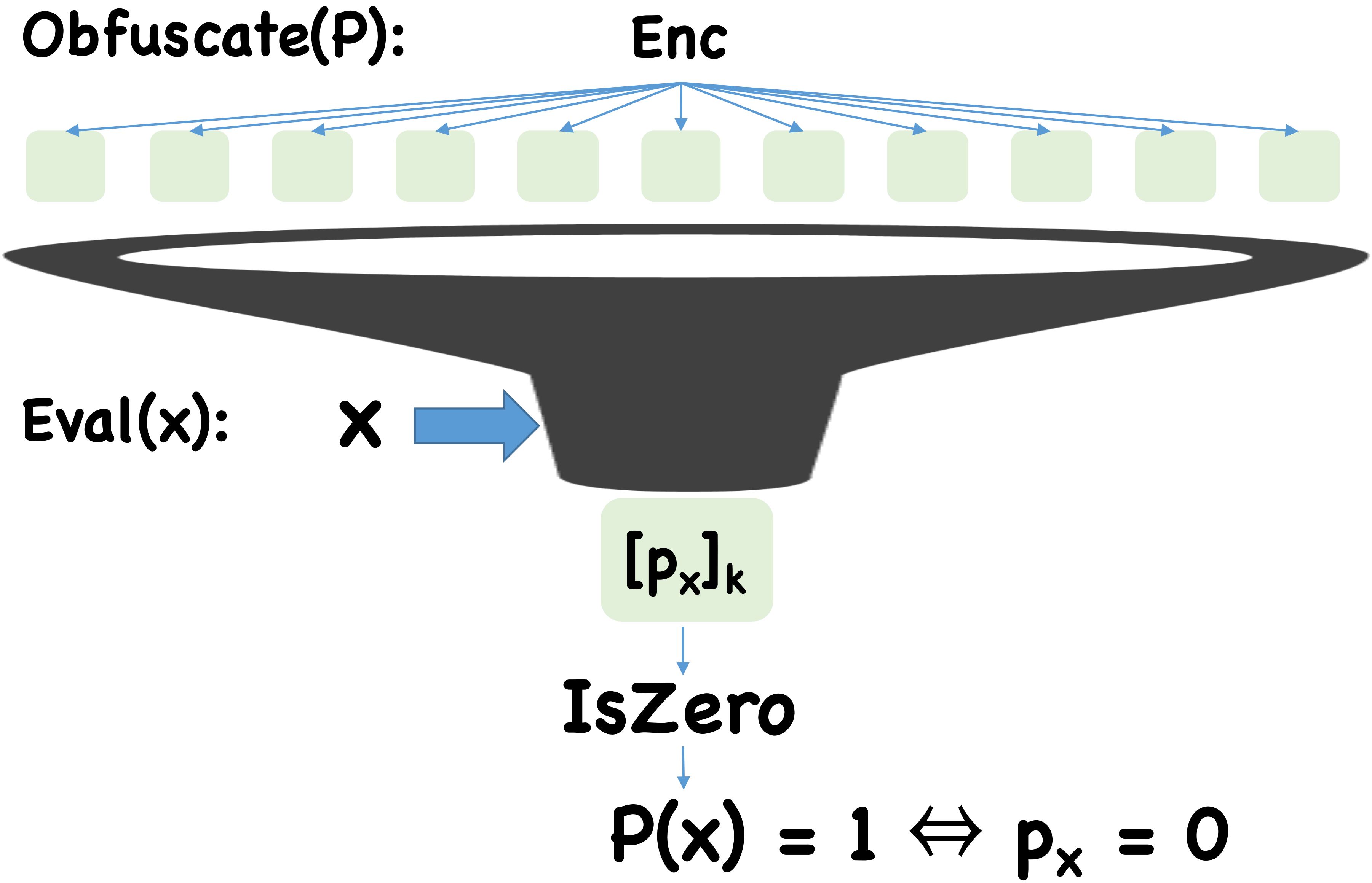
# Multilinear Maps (a.k.a. graded encodings)
[BS'03,GGH'13,CLT'13,GGH'15]

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

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

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

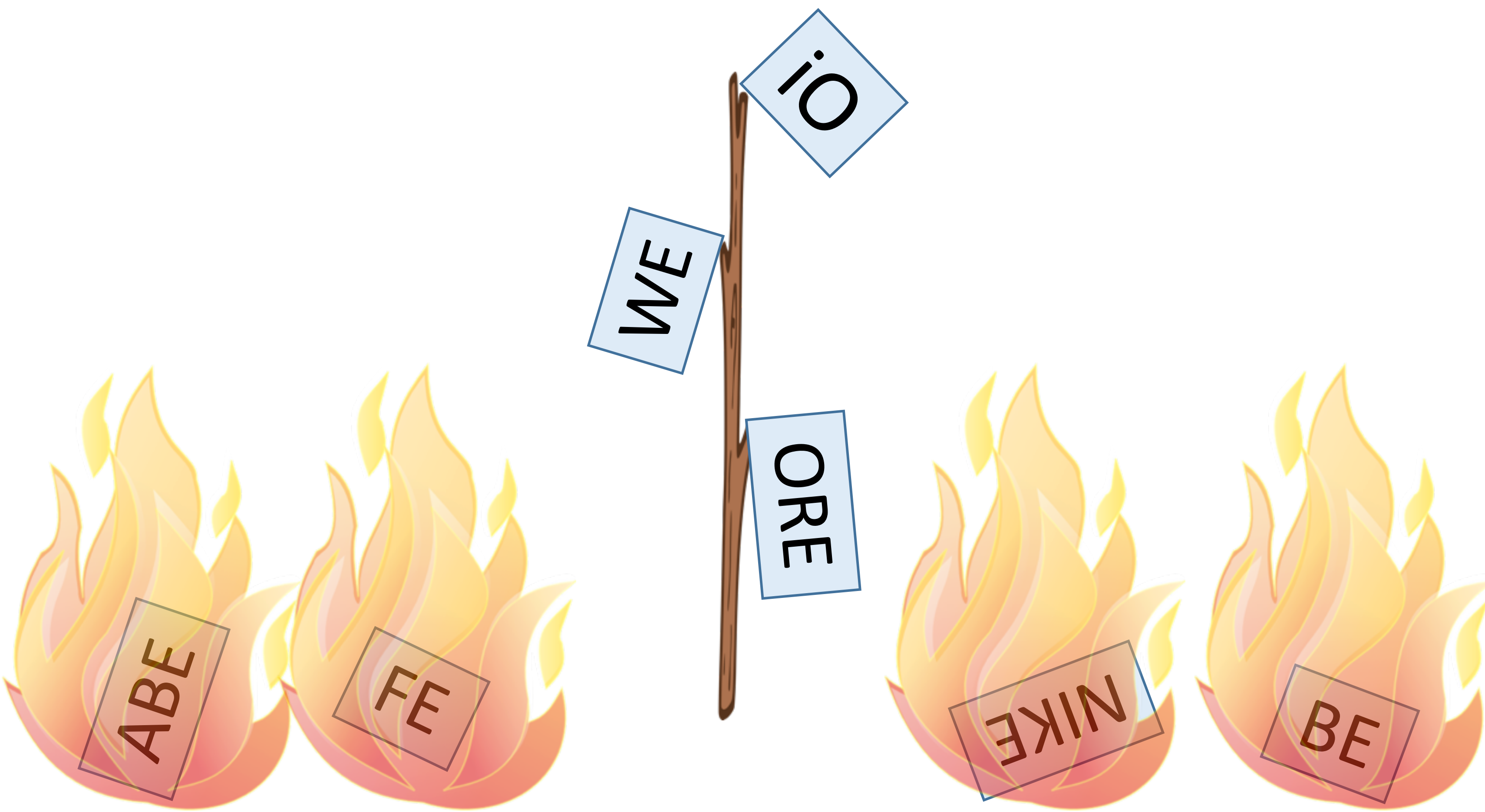
# Obfuscation From Multilinear Maps



# 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

| Branching Program Obfuscation | $NC^1$ Obfuscation |
|---|---|
| [GGHRSW'13,BR'14,BGKPS'14,PST'14,GLSW'14,AGIS'14,MSW'14,…] | [Zim'14,AB'15,BD'16] |
| • Better understood ✓<br>• Conceptually simpler?<br>• Prime order mmaps ✓<br>• Need $NC^1 \rightarrow$ BP ✗ | • Generally more efficient ✓<br>• Directly handle $NC^1$ circuits ✓<br>• Composite-order mmaps ✗ |

Boost to obfuscation for all circuits [GGHRSW'13,App'13,…]

# This Work: BP Obfuscation

| **Branching Program Obfuscation** | **NC$^1$ Obfuscation** |
|---|---|
| [GGHRSW'13,BR'14,BGKPS'14,PST'14,GLSW'14,AGIS'14,MSW'14,…] | [Zim'14,AB'15,BD'16] |
| • Better understood ✓<br>• Conceptually simpler?<br>• Prime order mmaps ✓<br>• Need NC$^1$ → BP ✗ | • Generally more efficient ✓<br>• Directly handle NC$^1$ circuits ✓<br>• Composite-order mmaps ✗ |

Boost to obfuscation for all circuits [GGHRSW'13,App'13,…]

# How To Argue Security of iO Candidates?

**Option 1: Reduction [illegible] "simp[illegible] assumption**

- Easier to analyze ass[illegible]than candidate
- Several ways to do th[illegible]obfuscation:
  - Directly: [GLSW'15]
  - Through FE: [AJ'1[illegible]15][illegible]HZ'16]
- Currently only kno[illegible]n for BP obfuscation

Essentially all "simple" assumptions broken by zeroizing attacks

# How To Argue Security of iO Candidates?

**Option 2: Argue sec[illegible]y in id[illegible]mmap model**

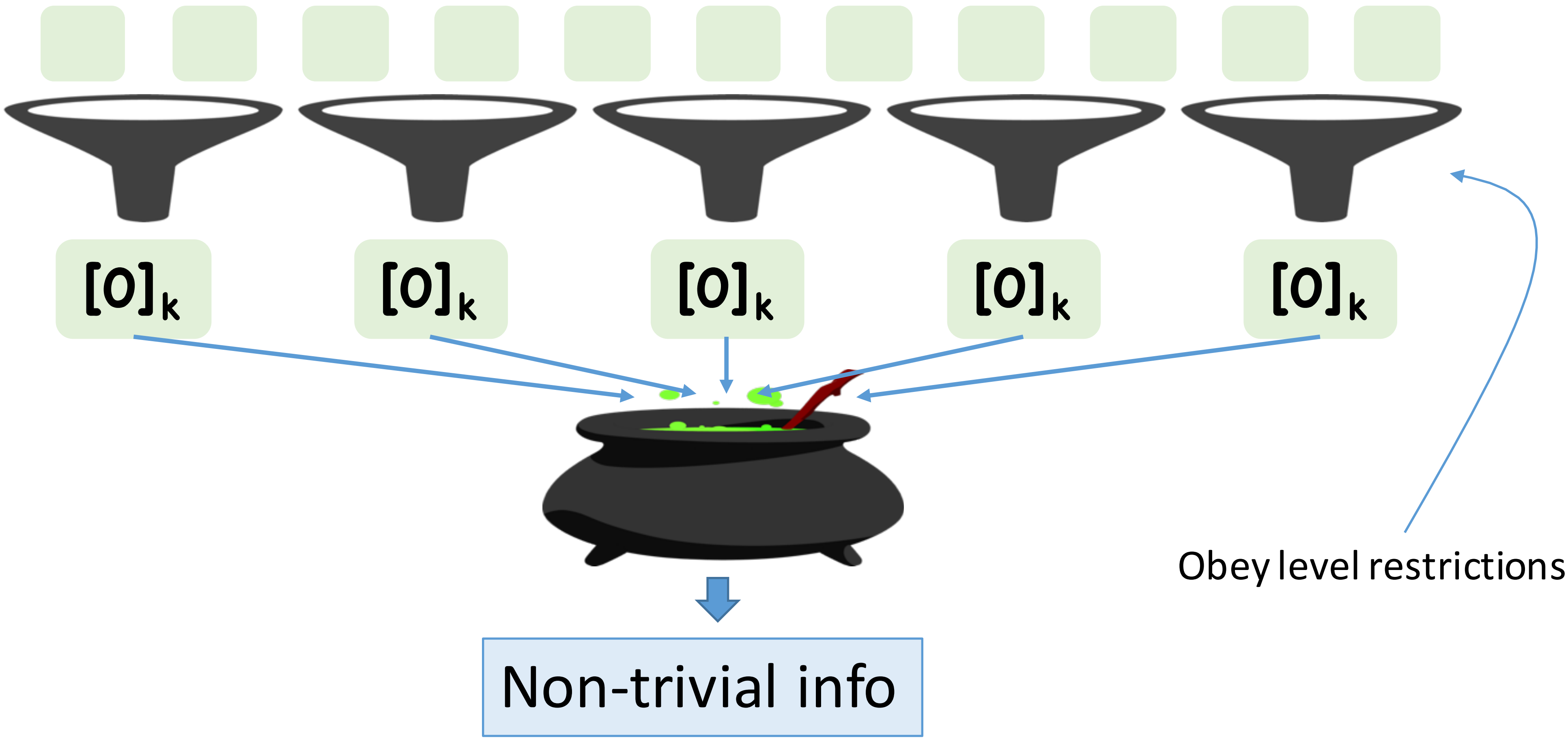
[BR'14,BGKPS'14,PS[illegible]14,Zim'14,AB'15,…]

- Prove that no "generi[illegible]acker exists (i.e. one that only interacts with [illegible]ough interfaces)
- May be reasonabl[illegible]o [illegible]generic attacks known
  - Ex: random oracle [illegible]odel, 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, CGHLMMRST'15,HJ'15,BGHLST1'5,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 [CGHLMMRST'15,MSZ'16a]
- Subject of follow-up works [MSZ'16a,GMS'16, MSZ'16b]

**<u>This work: Focus on 1, gives tools for 1 & 2</u>**

# Post-Zeroizing Security?

**Option 1: avoid zeros entirely**

- New ideal model: zero gives compl

Same as old model until successful zero test

**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]

**<u>This work: Focus on 1, gives tools for 1 & 2</u>**

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 $\mathbf{P}$

In particular, if $\mathbf{P_1(x)=P_2(x)}\,\forall\,\mathbf{(x)}$, 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 mmaps

- Construction very similar to prior works

Brand new analysis:

- Let $\mathbf{p_x}$ be element that is zero-tested when running $\mathbf{P(x)}$

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

Holds for any "level respecting" model

# Implications: Post-Zeroizing Security

Immediate corollary:

**Corollary:** If $\mathbf{P}$ 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 $NC^1 \rightarrow$ BP conversion
- Can't directly handle automata

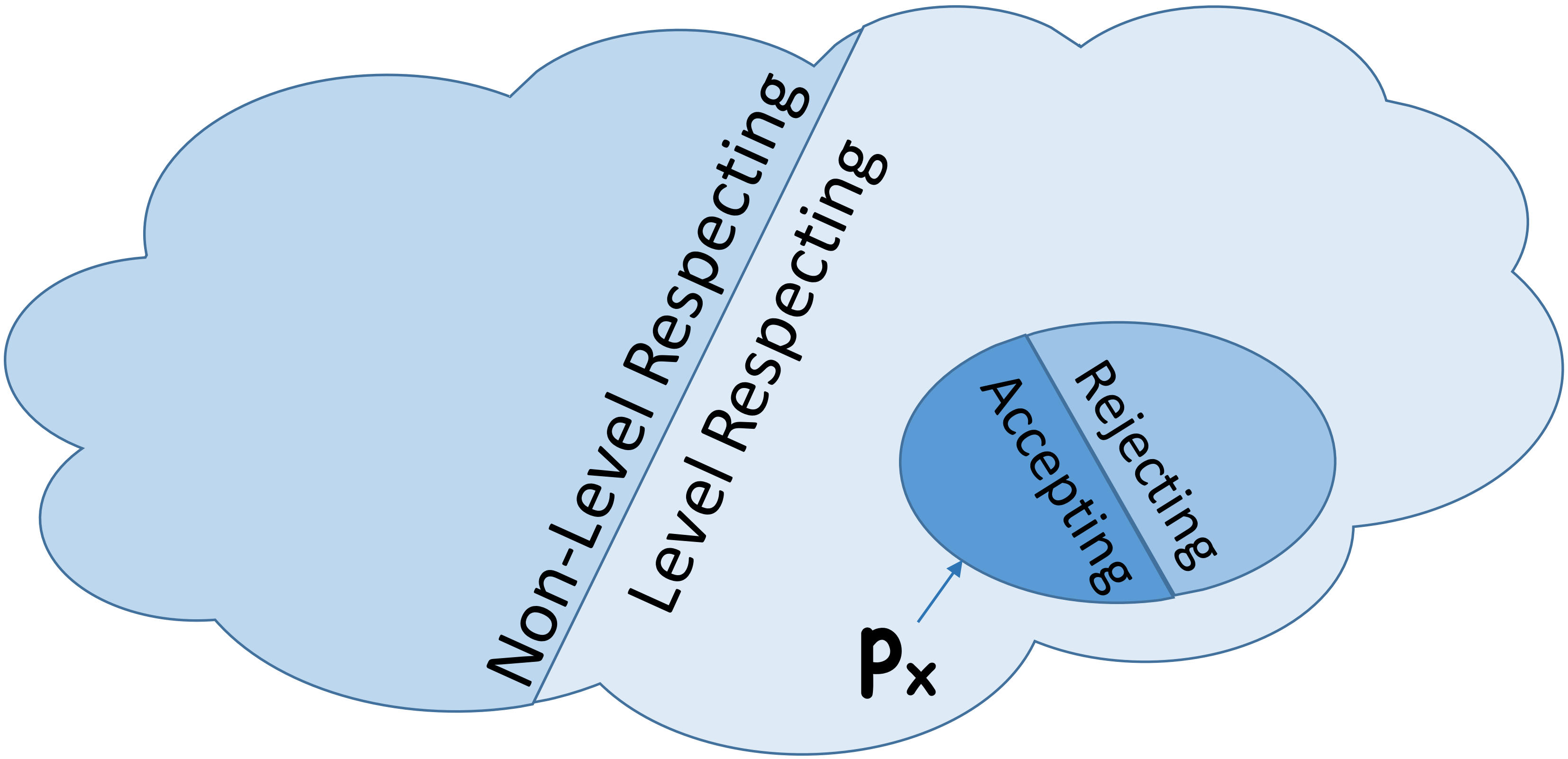
**Our Analysis:** security for "essentially all" BPs

- Allows for much more efficient $NC^1 \rightarrow$ BP conversion
  - Still not quite as efficient as direct $NC^1$ obfuscators
- Can directly handle automata
- Tools useful in other settings [BLRSZZ'15]

Improved security analysis $\Rightarrow$ improved efficiency

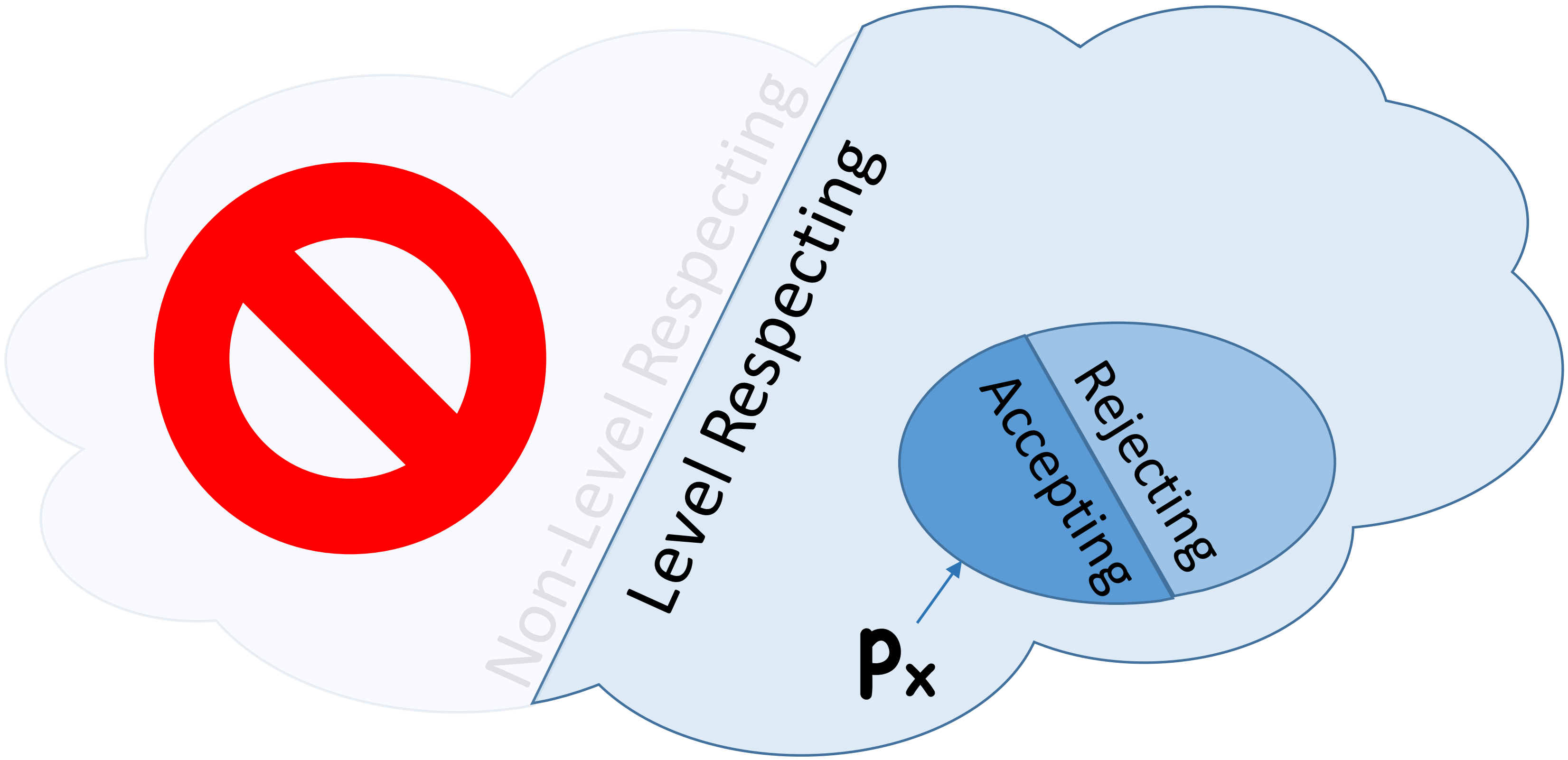
# Proof Overview

Consider arbitrary polynomial of encoded terms



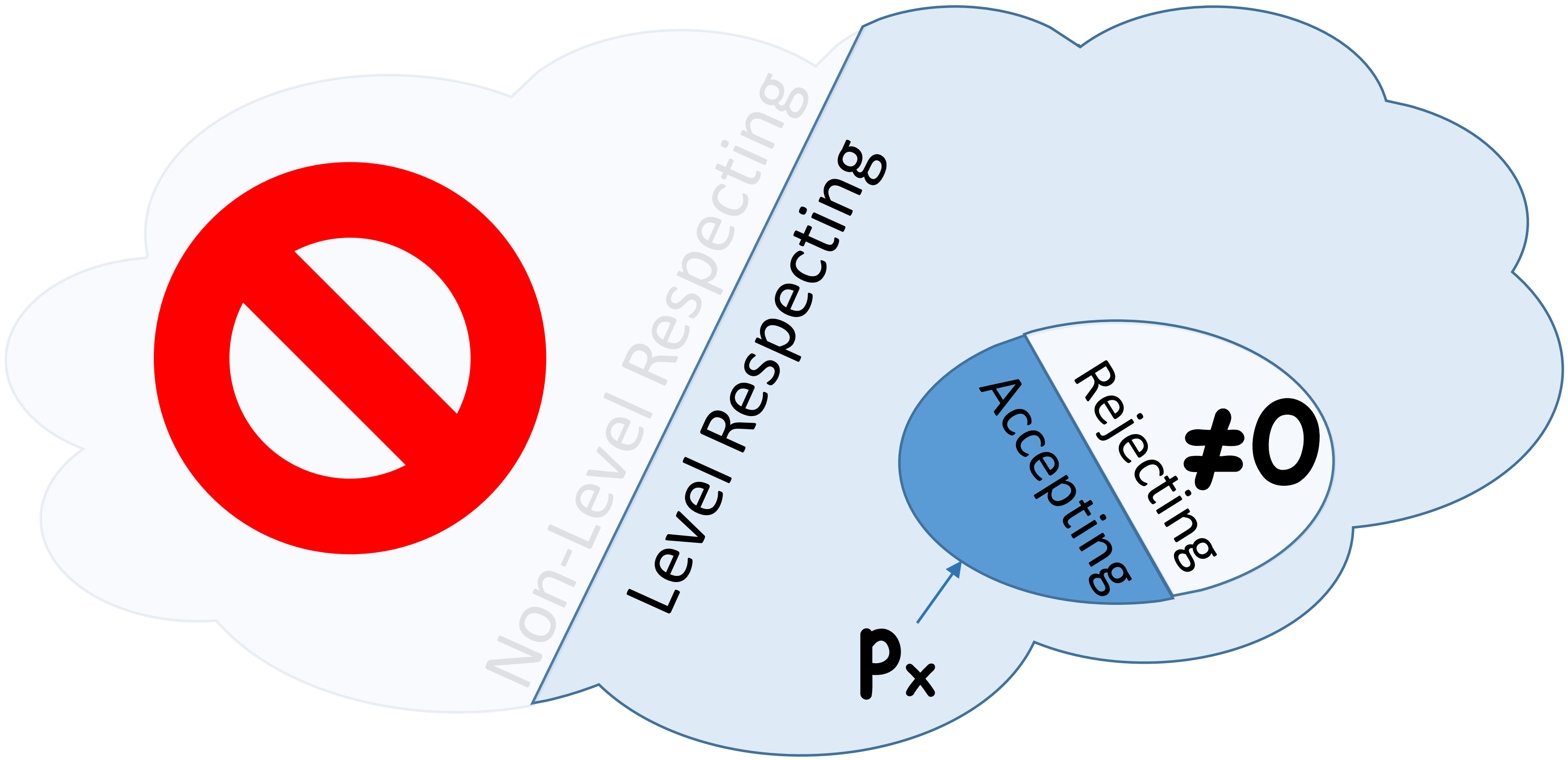
# Proof Overview

Consider arbitrary polynomial of encoded terms



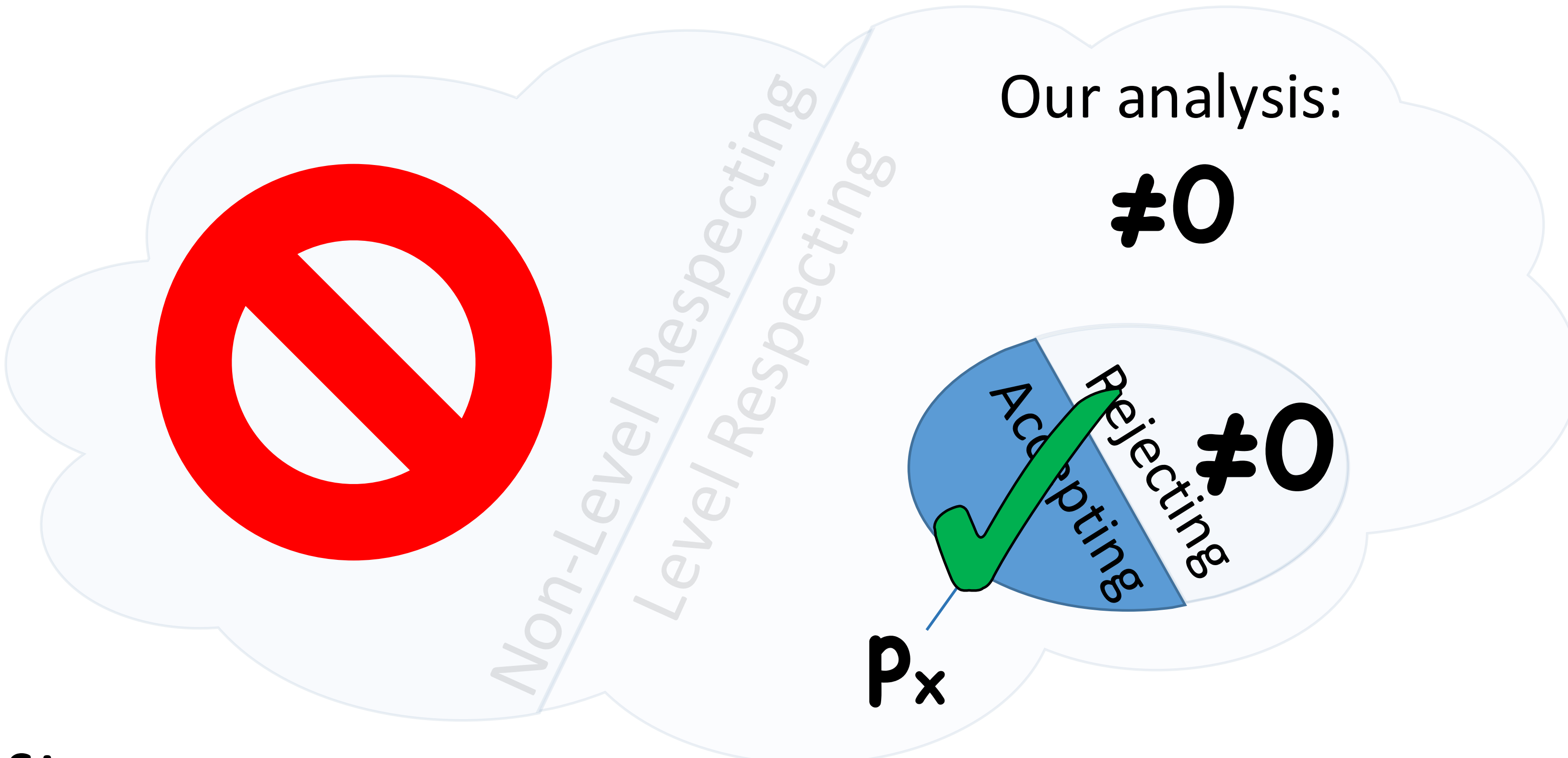
# Proof Overview

Consider arbitrary polynomial of encoded terms



# Proof Overview

Consider arbitrary polynomial of encoded terms




Tools:

- Prior characterization of level-respecting polys [BGKPS’14,MSW’14]
- Schwartz-Zippel $\Rightarrow$ anything but $\mathbf{p_x}$ 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!