Exotic & Powerful !!

# ADVANCED CRYPTOGRAPHIC PRIMITIVES

## PART 3: INTRO TO CODE OBFUSCATION

ACM INDIA SUMMER SCHOOL (12 – 06 – 2024)

*Venkata Koppula (IIT Delhi)*
*kvenkata@iitd.ac.in*

# PROGRAM OBFUSCATION

Make programs maximally unintelligible

$P$

```
function NewObject()
{
    this.SayHello=function(msg)
    {
        alert(msg);
    }
}
var obj=new NewObject();
obj.SayHello("Hello World.");
```

$P'$

```
var _0xfcad =
["\x53\x61\x79\x48\x65\x6C\x6C\x6F","\x48\x65\x6C
\x6C\x6F\x20\x57\x6F\x72\x6C\x64\x2E"];
function NewObject({this[_0xfcad[0]] =
function(_0xbbefx2){alert(_0xbbefx2)}}var obj= new
NewObject();obj.SayHello(_0xfcad[1])
```

https://javascriptobfuscator.com/Javascript-Obfuscator.aspx

# PROGRAM OBFUSCATION

Make programs maximally unintelligible

Obf : *compiler mapping programs to programs*

Function-Preserving

*Let* $P' = \mathsf{Obf}(P)$ . *For all inputs* $x$, $P(x) = P'(x)$

Efficiency

$|\mathsf{Obf}(P)| \leq \mathsf{poly}(|P|)$

Obf *must be efficient*

# PROGRAM OBFUSCATION

Make programs maximally unintelligible

## STRONG VIRTUAL BLACK BOX (VBB) OBFUSCATION

Having obfuscated code ≈ having oracle access to code

∀ efficient $A$, ∃ efficient $S$ s.t. ∀ programs $P$,

$$A(\mathsf{Obf}(P)) \approx S^P$$

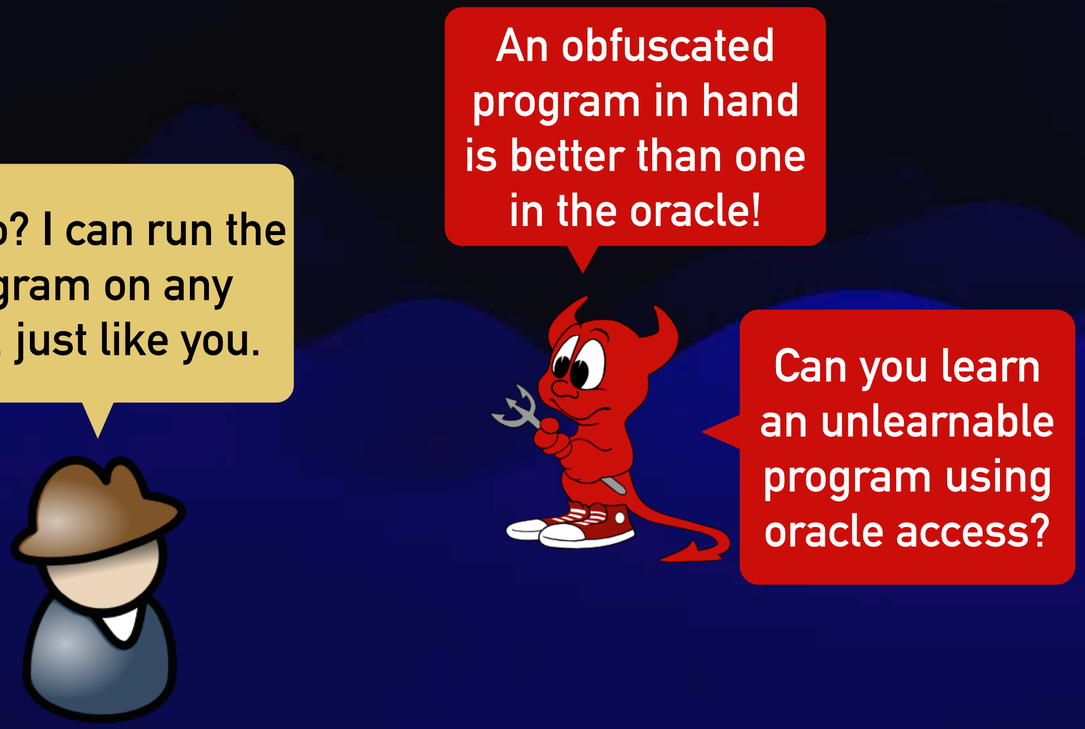
# STRONG VBB OBFUSCATION: TOO STRONG

Make programs maximally unintelligible

## STRONG VBB OBFUSCATION

$\forall$ efficient $A$, $\exists$ efficient $S$ s.t. $\forall$ programs $P$,

$$A(\mathsf{Obf}(P)) \approx S^P$$




*$A(\mathsf{Obf}(P))$ outputs $\mathsf{Obf}(P)$ .*

*Cannot be simulated if $P$ is unlearnable using oracle access .*

# VBB OBFUSCATION

Make programs maximally unintelligible

## VBB OBFUSCATION

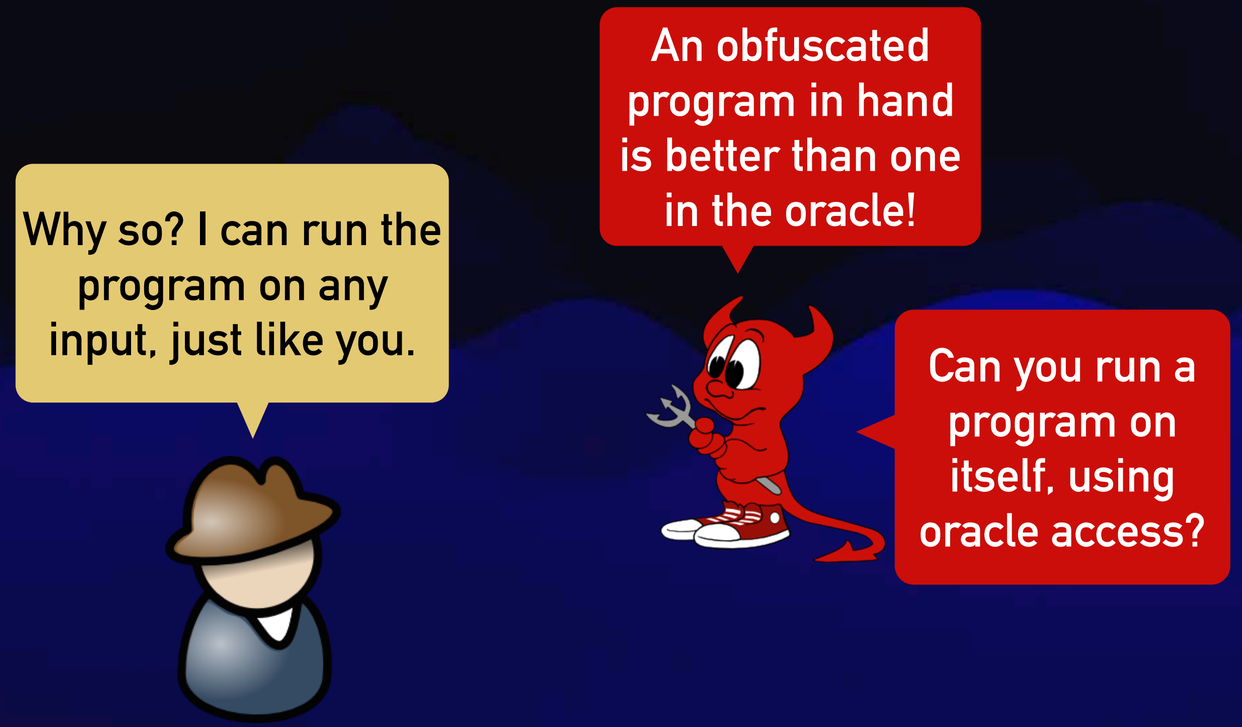
$\forall$ efficient $A$ with single-bit output ,

$\exists$ efficient $S$, s.t. $\forall$ programs $P$

$$A(\text{Obf}(P)) \approx S^P$$

**ON THE (IM)POSSIBILITY OF OBFUSCATING PROGRAMS**

[Barak-Goldreich-Impagliazzo-Rudich-Sahai-Vadhan-Yang 01]



# VBB OBFUSCATION: TOO STRONG

There exist programs that cannot be VBB obfuscated !

## VBB OBFUSCATION

$\forall$ efficient $A$ with single-bit output ,

$\exists$ efficient $S$, s.t. $\forall$ programs $P$

$$A(\mathsf{Obf}(P)) \approx S^P$$

*Create a program $P$ s.t. $P(P)$ outputs some 'secret' that cannot be learnt using just oracle access.*

$$P_{\alpha,\beta,\gamma}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ \gamma & \text{if } x(\alpha) = \beta \\ \perp & \text{otherwise} \end{cases}$$

$$P_{\alpha,\beta,\gamma}\left(P_{\alpha,\beta,\gamma}\right) = \gamma$$

*If $\alpha, \beta \leftarrow \{0,1\}^n$,*
*cannot learn $\gamma$ with oracle access to $P_{\alpha,\beta,\gamma}$*

# VBB OBFUSCATION FOR CIRCUITS ?

What about obfuscating circuits?
Previous impossibility does not apply to circuits

## VBB OBFUSCATION

$\forall$ efficient $A$ with single-bit output ,

$\exists$ efficient $S$, s.t. $\forall$ programs $P$

$$A(\mathsf{Obf}(P)) \approx S^P$$

Qn: How to recover $\gamma$, given obfuscation of $C_{\alpha,\beta,\gamma,\mathsf{sk}}$ ?

Ans: Run obf. program on 0 → encryption of $\alpha$
Evaluate the obfuscated program on FHE ct →encryption of $\beta$
Feed the final ct to the circuit to learn $\gamma$

*Theorem: Assuming the existence of secure FHE, there exist circuit families that cannot be VBB obfuscated!*

$$C_{\alpha,\beta,\gamma,\mathsf{sk}}(x) = \begin{cases} \mathsf{FHE}.\mathsf{Enc}(\mathsf{sk},\alpha) & \text{if } x = 0 \\ \beta & \text{if } x = \alpha \\ \gamma & \text{if } \mathsf{FHE}.\mathsf{Dec}(\mathsf{sk},x) = \beta \\ \bot & \text{otherwise} \end{cases}$$

*Assuming the security of FHE, cannot learn $\gamma$ given oracle access to $C_{\alpha,\beta,\gamma,\mathsf{sk}}$*

# NEED SOMETHING MUCH WEAKER THAN VBB OBFUSCATION

An obfuscated program in hand is better than one in the oracle!

Can output the obfuscated program

Can run the obfuscated program on itself

Can use the obfuscated program for FHE evaluation

*VBB obfuscation is possible only for very small function classes*

# NEED SOMETHING MUCH WEAKER THAN VBB OBFUSCATION

## INDISTINGUISHABILITY OBFUSCATION (iO)

Obfuscations of functionally identical circuits are indistinguishable

$$If \;\; C_0(x) = C_1(x) \; for \; all \; x,$$

$$\mathsf{Obf}(C_0) \approx \mathsf{Obf}(C_1)$$

Function-Preserving

$$Let \; C' = \mathsf{Obf}(C) \; . \; For \; all \; inputs \; x, \; C(x) = C'(x)$$

Efficiency

$$|\mathsf{Obf}(C)| \leq \mathsf{poly}(|C|)$$

Obf *must be efficient*

Ans: If both efficiency requirements removed, then output the truth table. If Obf(C) must be poly(|C|), then output the smallest circuit that is functionally identical to C.

Qn: How to build iO if one/both efficiency requirements are removed?

# NEED SOMETHING MUCH WEAKER THAN VBB OBFUSCATION

## INDISTINGUISHABILITY OBFUSCATION (iO)

Obfuscations of functionally identical programs are indistinguishable

$$\textit{If } C_0(x) = C_1(x) \textit{ for all } x,$$

$$\mathsf{Obf}(C_0) \approx \mathsf{Obf}(C_1)$$

*Theorem: If* $\mathsf{P} = \mathsf{NP}$*, then indistinguishability obfuscation exists.*

*If* $\mathsf{P} = \mathsf{NP}$*, then* $\nexists$ *one-way functions.*

*Existence of ind. obfuscation* **does not** *imply existence of one-way functions!*

# IS INDISTINGUISHABILITY OBFUSCATION OF ANY USE ?

## CANDIDATE iO SCHEME

[Garg-Gentry-Halevi-Raykova-Sahai-Waters 13]

## HOW TO USE iO

[Sahai-Waters 14]

Witness Encryption

Public Key Encryption

Non interactive Zero Knowledge proofs

Functional Encryption

PPAD Hardness

Short Signatures

Homomorphic Encryption

2-round MPC

# iO + OWFs : where crypto dreams come true !

*Witness Encryption* ⟶ *PKE, IBE, ABE*

*Short Digital Signatures*

# WITNESS ENCRYPTION

Clay Mathematical Institute's Millennium Prize Problems

How to manage these awards?

A 3-step crypto solution

1. Put the prize money in a bank account
2. Encrypt the bank account info using the mathematical problem statement as 'public key'
3. Anyone with the correct solution/proof can use the proof as the 'secret key'

*Witness encryption* → 2

*Witness decryption* → 3

# WITNESS ENCRYPTION

NP language L with relation R

$$x \in \mathsf{L} \iff \exists w \; s.t. \; (x, w) \in \mathsf{R}$$


**CORRECTNESS**

$$If \; (x, w) \in R, \mathsf{ct} \leftarrow \mathsf{W}.\mathsf{Enc}(m, x), \; then \; \mathsf{W}.\mathsf{Dec}(\mathsf{ct}, w) = m$$

**SECURITY**

$$If \; x \notin L, \; then \; \mathsf{W}.\mathsf{Enc}(x,0) \approx \mathsf{W}.\mathsf{Enc}(x,1)$$

# WITNESS ENCRYPTION + PRG $\Longrightarrow$ PKE

Pseudorandom generator G : $\{0,1\}^n \rightarrow \{0,1\}^{2n}$

NP language L = $\left\{ x \in \{0,1\}^{2n} : \exists w \in \{0,1\}^n \text{ s.t. } G(w) = x \right\}$

Setup $() : Sample\ s \leftarrow \{0,1\}^n$

$\mathsf{pk} = G(s) \qquad \mathsf{sk} = s$

Witness encryption also implies IBE, ABE.

Qn: How to define encryption and decryption?

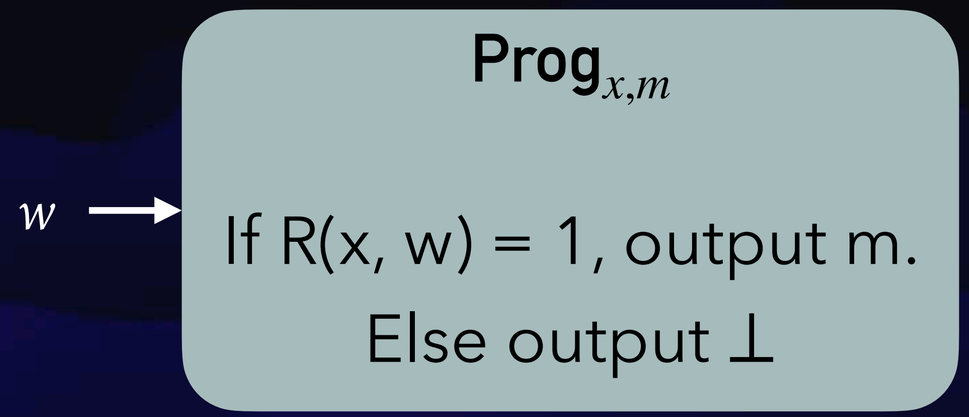
Qn(*): Prove security of the above construction, assuming G is secure PRG and security of witness encryption scheme.

$\mathsf{Enc}\ (\mathsf{pk}, m) = \mathsf{W}\ .\ \mathsf{Enc}\ (\mathsf{pk}, m)$

$\mathsf{Dec}\ (\mathsf{sk}, m) = \mathsf{W}\ .\ \mathsf{Dec}\ (\mathsf{sk}, m)$

# iO $\Longrightarrow$ WITNESS ENCRYPTION

W . Enc( $x, m$ ) :

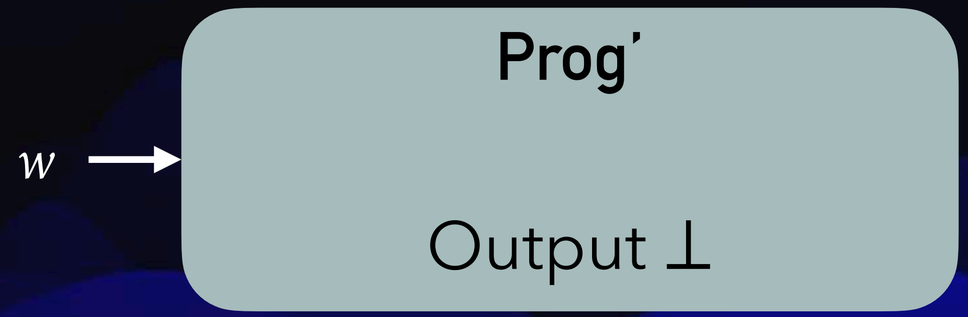



ct ← iO( $\text{Prog}_{x,m}$ )

W . Dec( sk, ct ) : *Output* ct( sk )

## SECURITY

*If* $x \notin L$, *then* W . Enc$(x,0) \approx$ W . Enc$(x,1)$




*Observation: If* $x \notin L$, *then* $\text{Prog}_{x,m}$ *and Prog' are functionally identical.*

$$\text{W . Enc}(x,0) = iO\left(\ \text{Prog}_{x,0}\ \right)$$

$$\approx iO\left(\ \text{Prog}'\ \right)$$

$$\text{W . Enc}(x,1) = iO\left(\ \text{Prog}_{x,1}\ \right)$$

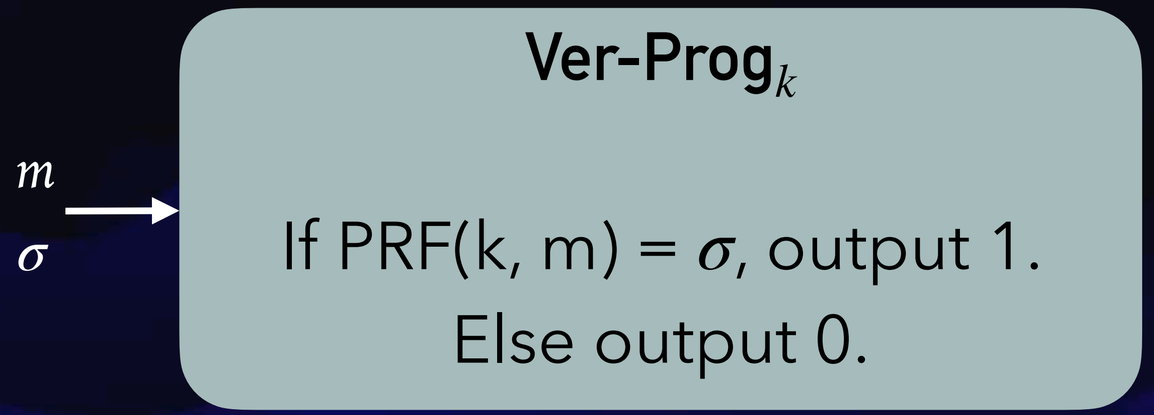
# iO + OWFs : where crypto dreams come true !

*Witness Encryption* ⟶ *PKE, IBE, ABE*

*Short digital signatures*

# iO + OWF $\Longrightarrow$ DIGITAL SIGNATURES

Setup $()$ : *Sample PRF key* $k$ . sk $= k$




vk $\leftarrow$ iO$\Big($ Ver-Prog$_k$ $\Big)$

*If we had VBB obfuscation instead of iO, security follows from PRF security!*

*How to use iO security? Not clear.*

Sign ( k, m ) : $\sigma = PRF(\ k, m\ )$

Verify ( vk, m, $\sigma$ ) : *Output* vk$(\ m, \sigma\ )$

# iO + OWF $\Longrightarrow$ DIGITAL SIGNATURES

Setup $()$ : *Sample PRF key* $k$ . sk $= k$

*f : One way function*

$m$
$\sigma$ →

**Ver-Prog$_k$**

If f( PRF(k, m) ) = f( $\sigma$ ), output 1.
Else output 0.

*We need to use special PRFs called 'puncturable PRFs'*

vk $\leftarrow$ iO( Ver-Prog$_k$ )

Sign ( k, m ) : $\sigma = PRF(\ k, m\ )$

Verify ( vk, m, $\sigma$ ) : *Output* vk( $m, \sigma$ )

# iO + OWF $\Longrightarrow$ DIGITAL SIGNATURES

Qn: OWF → PRG → puncturable PRFs

How to use length-doubling PRG to construct puncturable PRFs?

Ans: Use GGM tree-based PRF construction
Punctured PRF key consists of n evaluations in the tree

*f : One way function*

*We need to use special PRFs called 'puncturable PRFs'*

*(F, Puncture, Eval): puncturable PRF*

$$Puncture(\text{ key } k, \text{ input } x\text{ }) \rightarrow k\{x\}$$

$$Eval(\text{ } k\{x\}, \text{ input } x'\text{ }) \rightarrow y$$

$$If\ x \neq x',\ Eval(\text{ } k\{x\}, \text{ input } x'\text{ }) = F(\text{ } k, x'\text{ })$$

$$Security: (\text{ } k\{x\}, F(k, x)\text{ }) \approx (\text{ } k\{x\}, random\text{ })$$

# iO + OWF $\Longrightarrow$ DIGITAL SIGNATURES

*f : One way function, (PRF, Puncture, Eval): puncturable PRF*

Setup () : *Sample PRF key* $k$ . sk $= k$

$m$
$\sigma$ $\longrightarrow$

**Ver-Prog$_k$**

If f( PRF(k, m) ) = f( $\sigma$ ), output 1.
Else output 0.

vk $\leftarrow$ iO( Ver-Prog$_k$ )

Sign ( k, m ) : $\sigma = PRF(\ k, m\ )$

Verify ( vk, m, $\sigma$ ) : *Output* vk( $m, \sigma$ )

*Security proof uses a new proof technique 'punctured programming' which is crucial for most iO based security proofs.*

*Theorem: Assuming f is a secure OWF, (PRF, Puncture, Eval) is a puncturable PRF, iO is a secure ind. obfuscation, the signature scheme is selectively secure.*

# CONCLUSIONS

- Various security definitions of code obfuscation
  - Strong VBB obfuscation : impossible for unlearnable functions - adversary can simply output the obfuscated program
  - VBB obfuscation for Turing machines: impossible for many functions - adversary can run the obfuscated program on itself
  - VBB obfuscation for circuits: impossible for many functions, assuming FHE exist - adversary can run FHE evaluation using obfuscated circuit

- Indistinguishability obfuscation : only guarantees that obfuscations of functionally identical programs are indistinguishable

- iO does not imply OWFs. But iO + OWFs → lot of cryptographic primitives
  - Witness encryption: an advanced crypto primitive with simple iO-based construction
  - For many advanced primitives, the only known constructions are using iO

# CONCLUSIONS

- Very active area of research over the last ten years
  - First candidate construction in 2013 by Garg-Gentry-Halevi-Raykova-Sahai-Waters
  - FE and iO are equivalent. One direction (iO $\Longrightarrow$ FE) was shown by Garg-Gentry-Halevi-Raykova-Sahai-Waters, while the other direction was shown by Ananth-Jain and Bitansky-Vaikuntanathan in 2015.
  - Attack → Fix → New attack → New Fix …
  - In 2020, Jain-Lin-Sahai gave construction using bilinear maps + LWE + low-depth PRGs —– well-studied cryptographic assumptions

- Several major questions are still open
  - Current constructions are terribly impractical. Improving efficiency?
  - Post-quantum construction? The current constructions use bilinear maps, and therefore not post-quantum secure

**THANK YOU!**