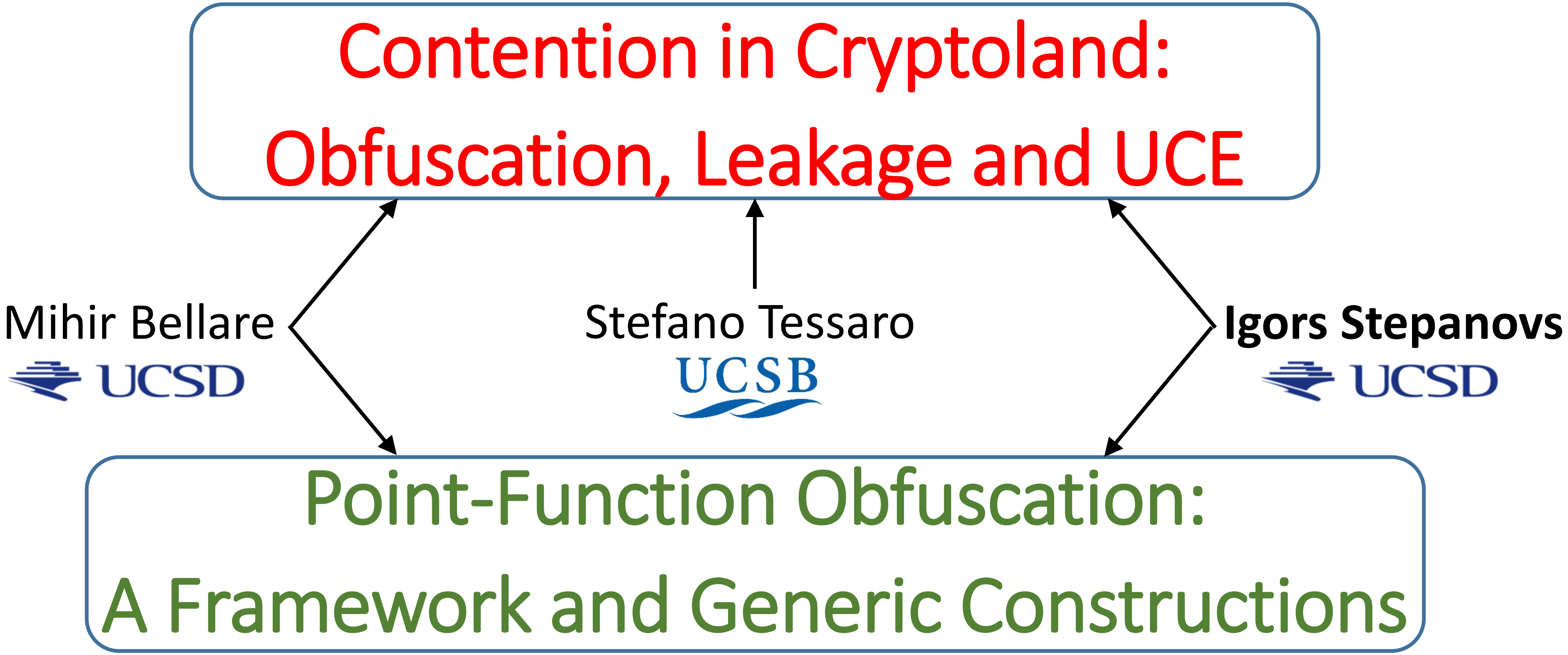


Contention in Cryptoland: Obfuscation, Leakage and UCE

Mihir Bellare
UCSD

Stefano Tessaro
UCSB

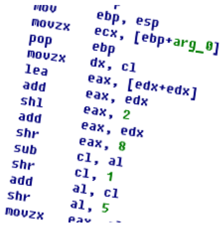
Igors Stepanovs
UCSD

Point-Function Obfuscation: A Framework and Generic Constructions

January 13, 2016

TCC 2016-A

Obfuscation



```
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax
```

Program **P**

Obfuscator

Program $\mathbf{P}^*$

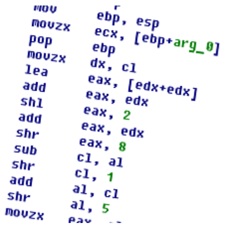
Correctness: $\mathbf{P} \equiv \mathbf{P}^*$

Functionally equivalent.

$\mathbf{P}(x) = \mathbf{P}^*(x)$ for all x.

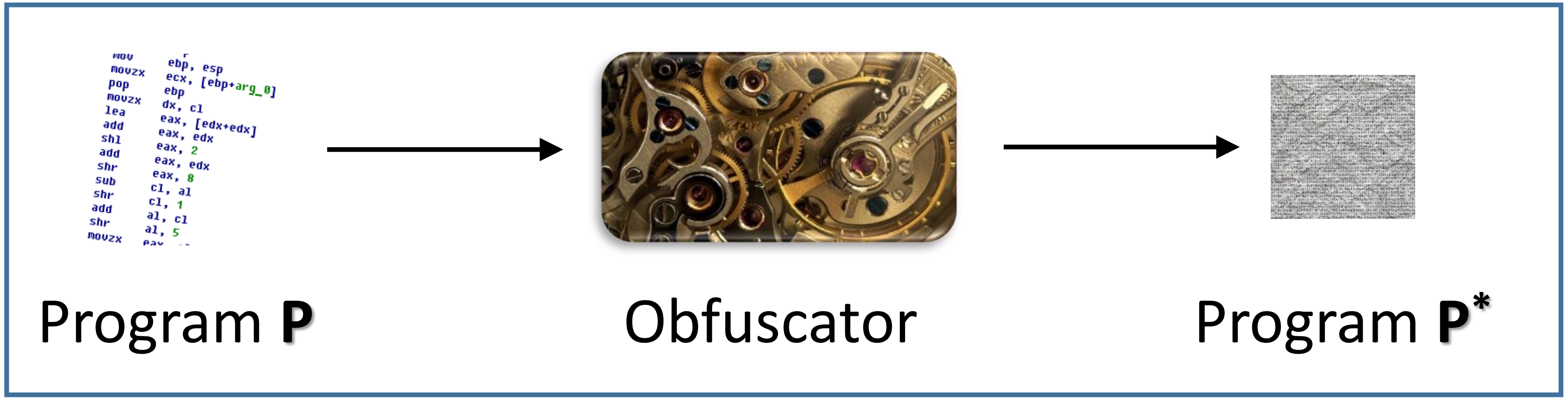
Security:

no more useful than an oracle for



```
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax
```

Obfuscation



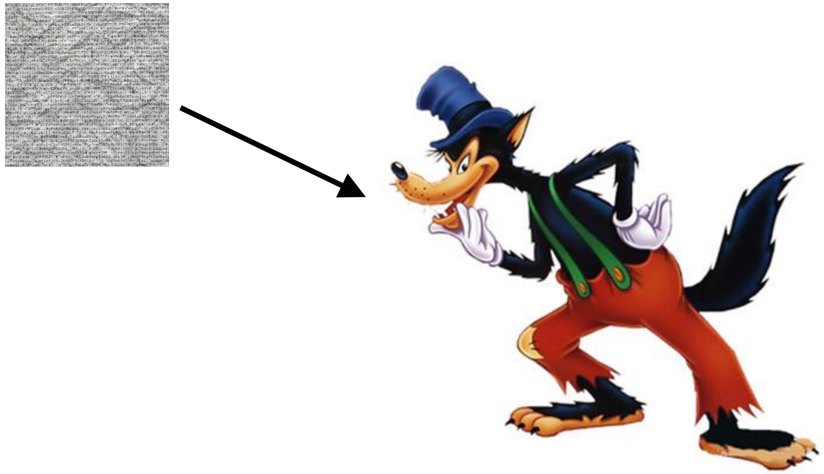
```
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax
```

Program **P**

Obfuscator

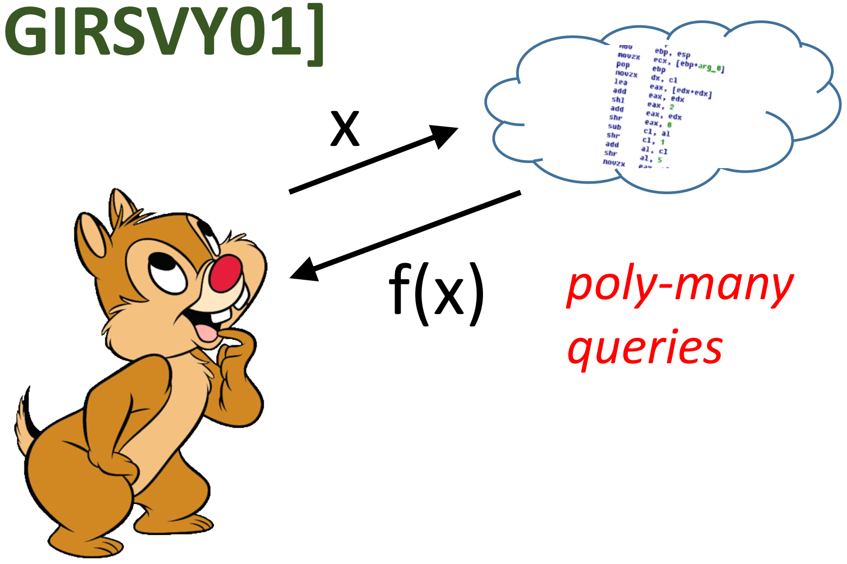
Program **P***

Virtual Black Box Security [BGIRSVY01]



Polynomial time adversary

vs.



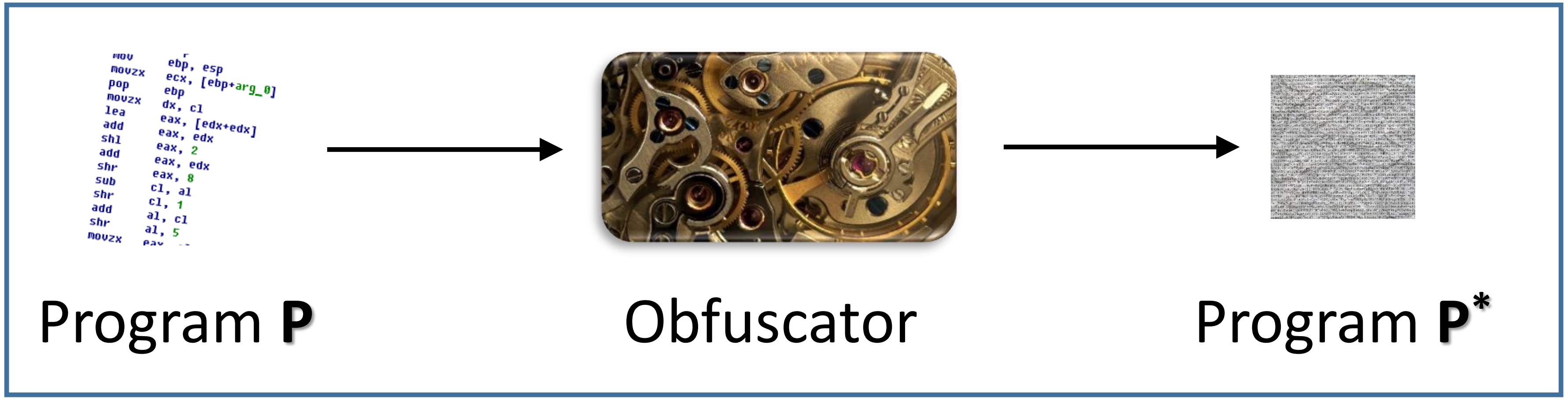
x

f(x)

poly-many queries

Polynomial time simulator

Obfuscation



```
mov     ebp, esp
movzx   ecx, [ebp+arg_0]
pop     ebp
movzx   dx, cl
lea     eax, [edx+edx]
add     eax, edx
shl     eax, 2
add     eax, edx
shr     eax, 8
sub     cl, al
shr     cl, 1
add     al, cl
shr     al, 5
movzx   eax
```

Program **P**

Obfuscator

Program **P***

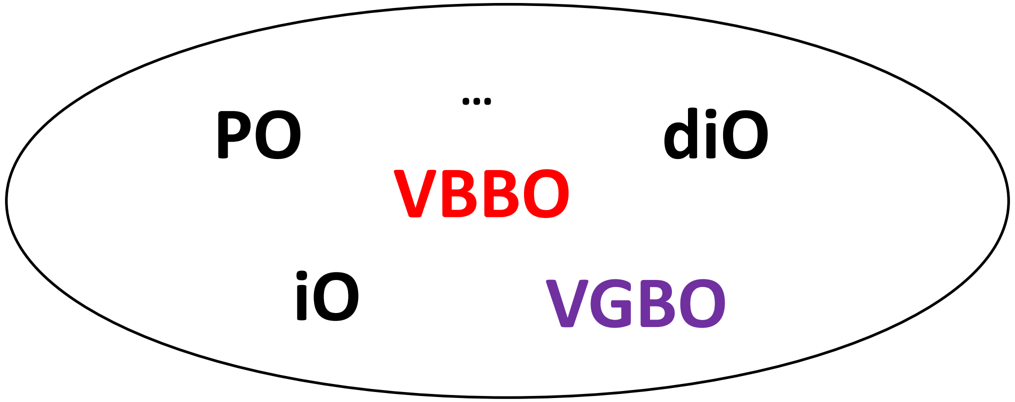
Virtual Black Box Security [BGIRSVY01]

... is not achievable!

Obfuscation

Are there special, weaker forms of obfuscation that are ...

- **achievable?**
- **interesting or useful?**



…

PO

diO

VBBO

iO

VGBO

PO – point-function obfuscation [C97, CMR98, LPS04, ...]

VBBO – virtual black box obfuscation [BGIRSVY01]

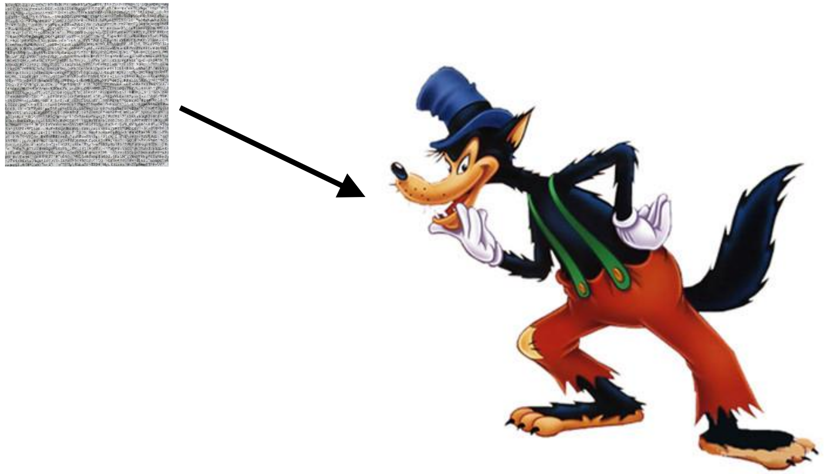
iO – indistinguishability obfuscation [BGIRSVY01, GGHRSW13, SW13, ...]

diO – differing-inputs obfuscation [BGIRSVY01, BCP13, ABGSZ13, ...]

VGBO – virtual grey box obfuscation [BC10, ...]

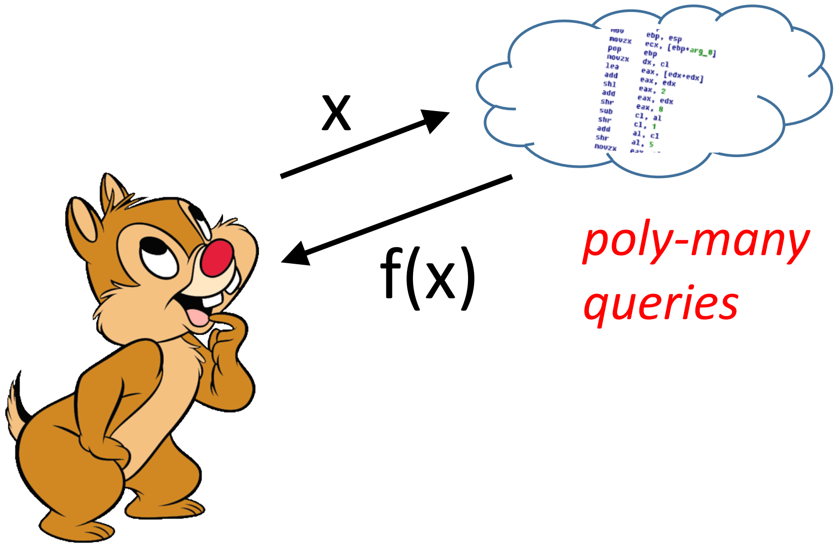
Virtual Grey Box Obfuscation (VGBO)

[Bitansky-Canetti-10]



Polynomial time adversary

vs.



x

f(x)

poly-many queries

Unbounded simulator

VGBO evades the negative results of [BGIRSVY01].

Is VGBO Achievable?

[BCKP14] "existing **candidate** indistinguishability **obfuscators** for all circuits may also be considered as **candidates for VGB obfuscation**"

How can we verify this conjecture?

Cryptoanalysis: analyze the used assumptions (multilinear maps, …)

Contentions: find an assumption or a primitive **X** s.t. **VGBO** $\not\longleftrightarrow$ **X**

We focus on ***contentions****.*

Directly reason about the **achievability of our goals**,

sidestepping an **involved analysis of assumptions**.

Past Work on Contentions

Contentions: find another assumption **X** such that **VGBO** $\not\longleftrightarrow$ **X**

Past work:

[BCPR14]: iO $\not\longleftrightarrow$ extractable one-way functions

[BM14]: iO $\not\longleftrightarrow$ multi-bit auxiliary-input PO

[GGHW14]: diO $\not\longleftrightarrow$ "special-purpose obfuscation"

Takeaways? Different feelings are possible…

Perception may evolve over time.

How plausible is iO?

Auxiliary-Input DH Inversion (AI-DHI) [Canetti ’97]

[BST16]

VGBO $\not\longleftrightarrow$ **AI-DHI**

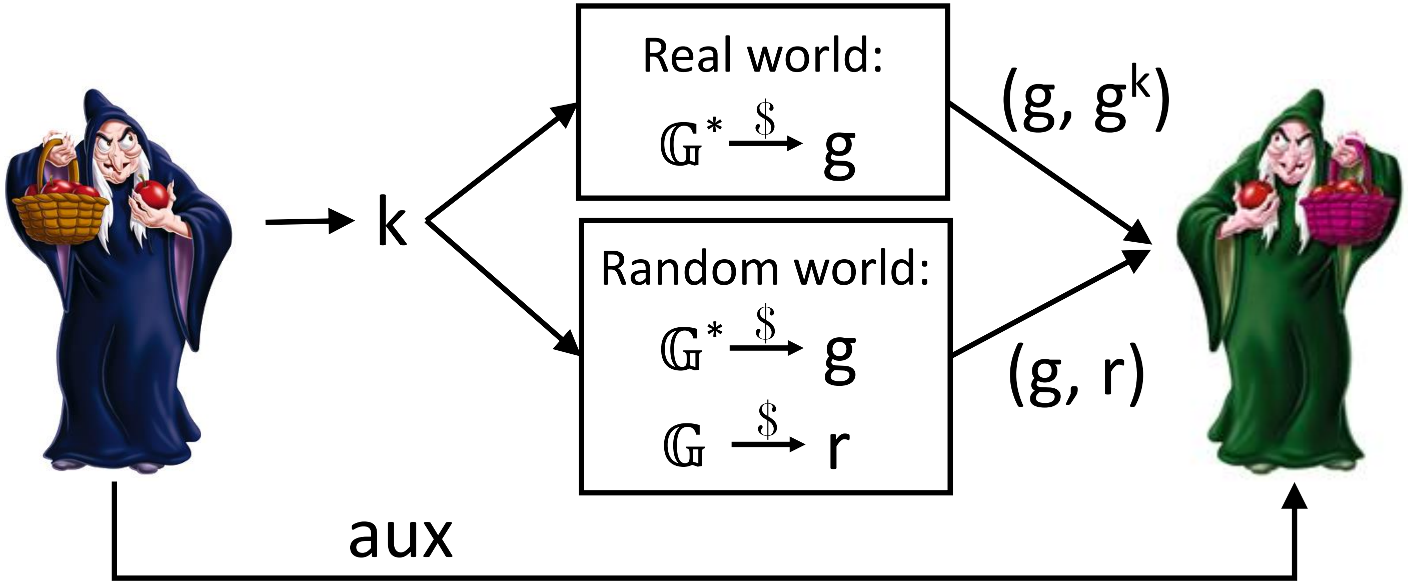
an assumption used to build point-function obfuscation (**PO**).

[Canetti ’97]

Introduced **AI-DHI** for oracle hashing.

Let $\mathbb{G}$ be a group of prime order.

Let $\mathbb{G}^*$ be the set of generators of $\mathbb{G}$.



$\rightarrow$ k

Real world:

$\mathbb{G}^* \xrightarrow{\$} g$

Random world:

$\mathbb{G}^* \xrightarrow{\$} g$

$\mathbb{G} \xrightarrow{\$} r$

(g, g^k)

(g, r)

aux

Auxiliary-Input DH Inversion (AI-DHI) [Canetti ’97]

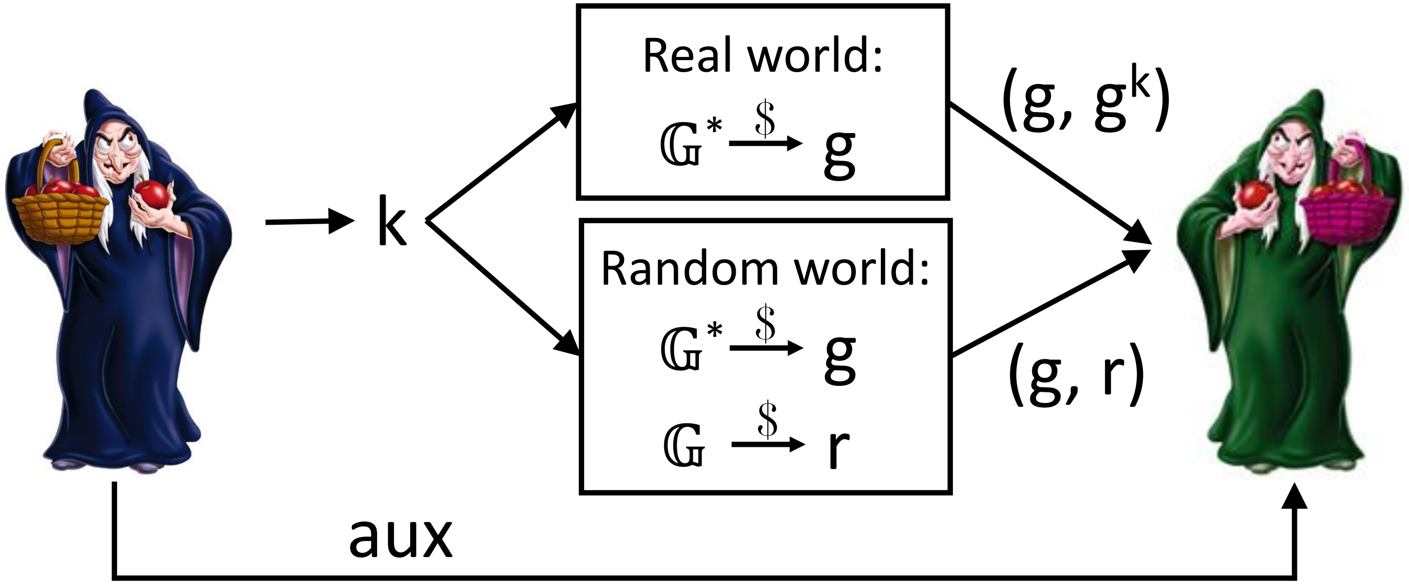
[BST16] **VGBO** $\not\longleftrightarrow$ **AI-DHI**

an assumption used to build point-function obfuscation (**PO**).

[Canetti ’97] Introduced **AI-DHI** for oracle hashing.

Let $\mathbb{G}$ be a group of prime order.

Let $\mathbb{G}^*$ be the set of generators of $\mathbb{G}$.



Real world:

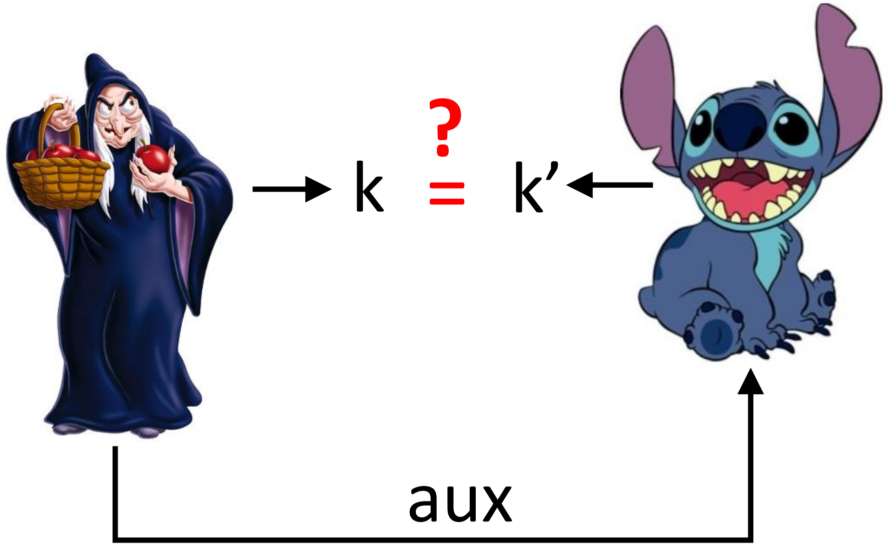
$\mathbb{G}^* \xrightarrow{\$} g$

Random world:

$\mathbb{G}^* \xrightarrow{\$} g$

$\mathbb{G} \xrightarrow{\$} r$

It should be hard to recover k from auxiliary information aux:



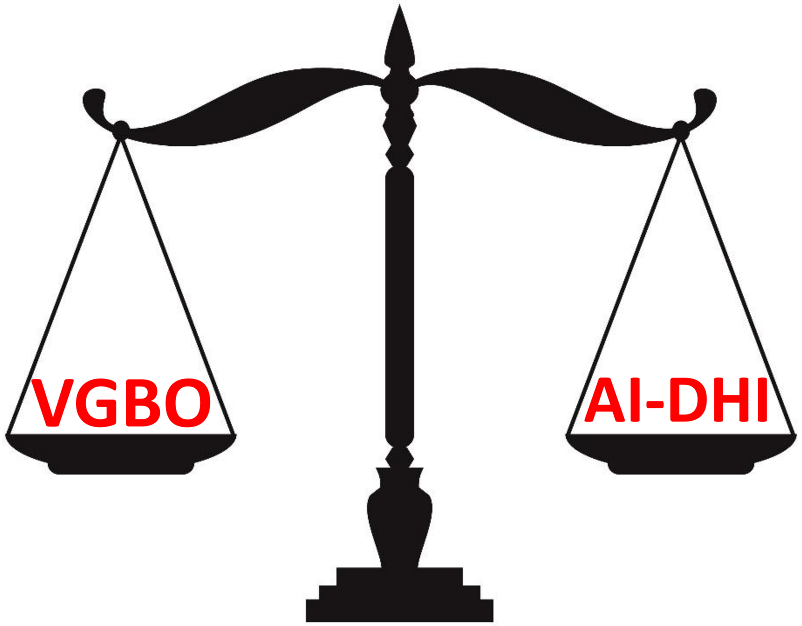
$k \overset{?}{=} k'$

VGBO vs. AI-DHI: Interpretation

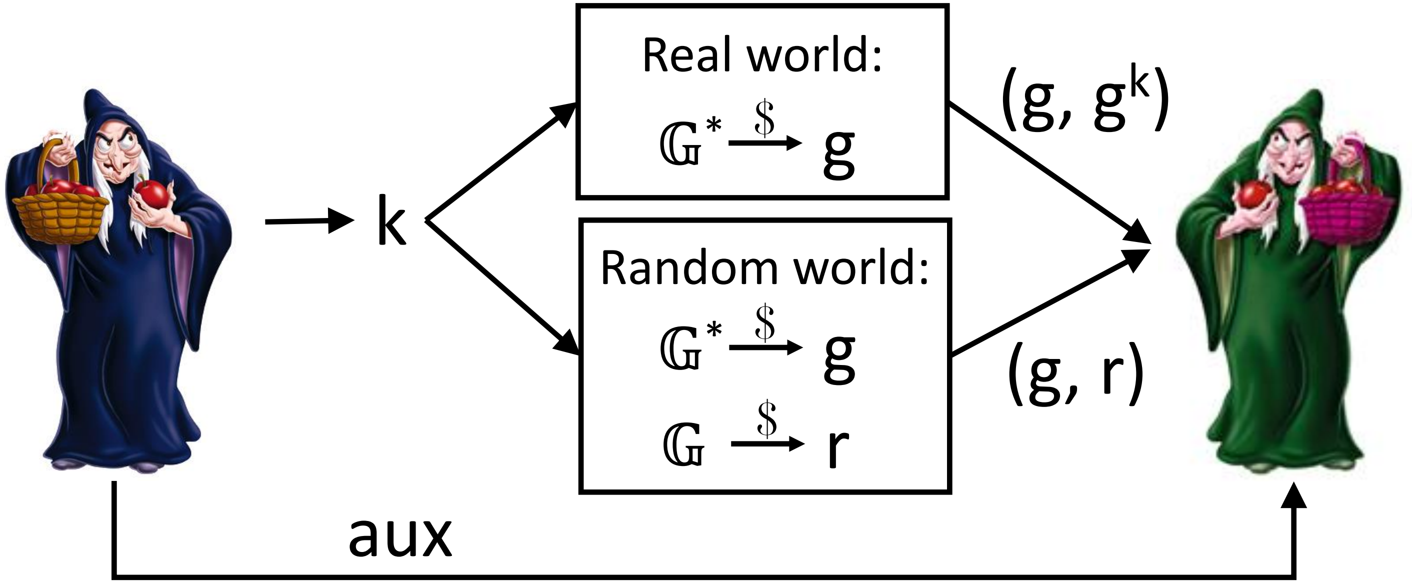
[BST16] **VGBO** ↮ **AI-DHI**

VGBO and **AI-DHI** cannot co-exist. <u>At least</u> **one does not exist**.

Which one is more plausible?... *Different feelings are possible…*



VGBO vs. AI-DHI: The Attack



k

Real world:

$\mathbb{G}^* \xrightarrow{\$} g$

Random world:

$\mathbb{G}^* \xrightarrow{\$} g$

$\mathbb{G} \xrightarrow{\$} r$

(g, g^k)

(g, r)

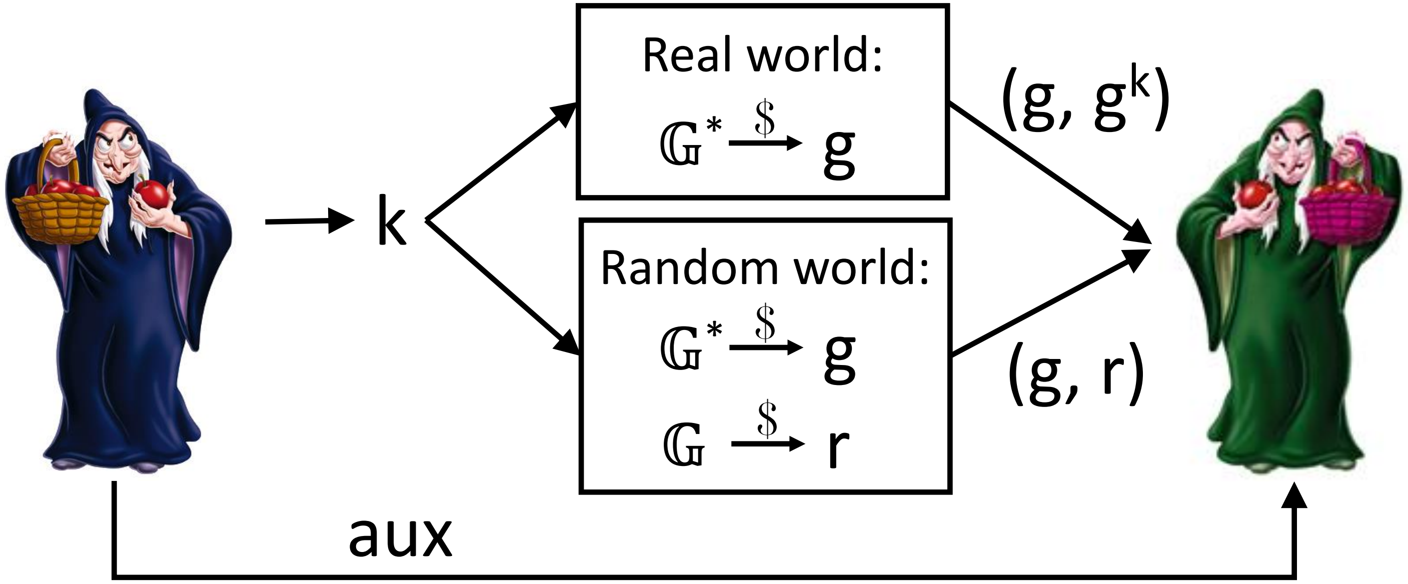
aux

Idea: use VGBO to break AI-DHI.

1. Sample k uniformly at random.
2. Set aux := $\mathrm{Obf}_{VGB}(C_k)$ for C_k defined as follows:

$$C_k(g, u) = 1 \quad \text{if } g^k = u$$
$$C_k(g, u) = 0 \quad \text{if } g^k \neq u$$

VGBO vs. AI-DHI: The Attack



$\to k$

Real world:

$\mathbb{G}^* \xrightarrow{\$} g$

Random world:

$\mathbb{G}^* \xrightarrow{\$} g$

$\mathbb{G} \xrightarrow{\$} r$

(g, g^k)

(g, r)

aux

Idea: use VGBO to break AI-DHI.

1. Sample k uniformly at random.
2. Set aux := $\mathrm{Obf}_{VGB}(C_k)$ for C_k defined as follows:

$$C_k(g, u) = 1 \quad \text{if } g^k = u$$
$$C_k(g, u) = 0 \quad \text{if } g^k \neq u$$

(1) Can distinguish between worlds:

Real world: $C_k(g, g^k) = 1$

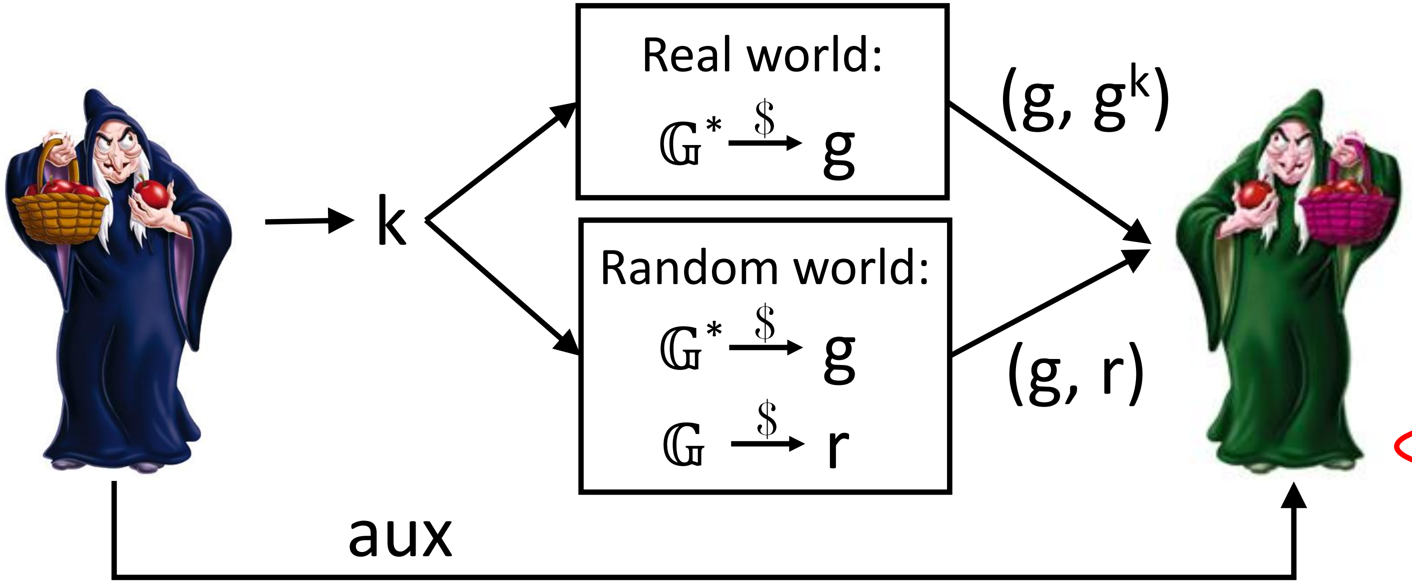
Random world: $C_k(g, r) = 0$ (w.h.p.)

(2) Hard to extract k from $\mathrm{Obf}_{VGB}(C_k)$:

We show that $\mathrm{Obf}_{VGB}(C_k)$ is indistinguishable from $\mathrm{Obf}_{VGB}(C^0)$ for

$$C^0(g, u) = 0$$

VGBO vs. AI-DHI: The Attack



$\rightarrow$ k

Real world:

$\mathbb{G}^* \xrightarrow{\$} g$

Random world:

$\mathbb{G}^* \xrightarrow{\$} g$

$\mathbb{G} \xrightarrow{\$} r$

(g, g^k)

(g, r)

aux

Idea: use VGBO to break AI-DHI.

1. Sample k uniformly at random.

2. Set aux := $\mathrm{Obf}_{VGB}(C_k)$ for C_k defined as follows:

$C_k(g, u) = 0 \quad \text{if } g \notin \mathbb{G}^* \text{ or } u \notin \mathbb{G}$

$C_k(g, u) = 1 \quad \text{if } g^k = u$

$C_k(g, u) = 0 \quad \text{if } g^k \neq u$

(1) Can distinguish between worlds:

Real world: $C_k(g, g^k) = 1$

Random world: $C_k(g, r) \;= 0$ (w.h.p.)

(2) Hard to extract k from $\mathrm{Obf}_{VGB}(C_k)$:

We show that $\mathrm{Obf}_{VGB}(C_k)$ is indistinguishable from $\mathrm{Obf}_{VGB}(C^0)$ for

$$C^0(g, u) = 0$$

VGBO vs. AI-DHI: The Attack

Claim: $\mathrm{Obf}_{\mathrm{VGB}}(C_k)$ is indistinguishable from $\mathrm{Obf}_{\mathrm{VGB}}(C^0)$

1. k is uniformly random.
2. C_k is defined as follows:

   $C_k(g, u) = 0$ if $g \notin \mathbb{G}^*$ or $u \notin \mathbb{G}$

   $C_k(g, u) = 1$ if $g^k = u$

   $C_k(g, u) = 0$ if $g^k \neq u$
3. C^0 is a zero-circuit:

   $C^0(g, u) = 0$

$\mathrm{Obf}_{\mathrm{VGB}}(C_k)$

Polynomial time adversary

Indistinguishable output distribution by the security of VGBO.

poly-many queries

g, u → C_k

$C_k(g, u)$

Unbounded simulator

Information-theoretically indistinguishable.

g, u → C^0

poly-many queries

$C^0(g, u)$

Unbounded simulator

VGBO vs. AI-DHI: Implications

AI-DHI is the main assumption used to construct auxiliary-input point-function obfuscation (**AIPO**).

$$\textbf{VGBO} \not\longleftrightarrow \textbf{AI-DHI} \xrightarrow{\text{[Canetti'97]}} \textbf{AIPO}$$

[BS16]

Can we recover constructions of point-function obfuscation from other assumptions?

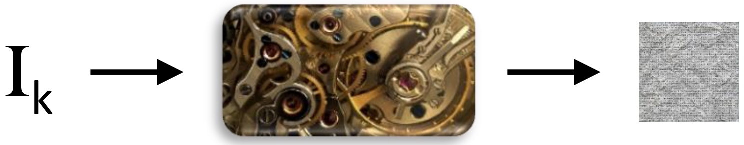
Point-Function Obfuscation (PO)

[Canetti’97, CMR98, LPS04, GK05, Wee’05, ...]

For any **target point** k, define a **point function** I_k:

$$\mathrm{I}_k(x) = 1 \quad \text{if } x = k$$
$$\mathrm{I}_k(x) = 0 \quad \text{if } x \neq k$$

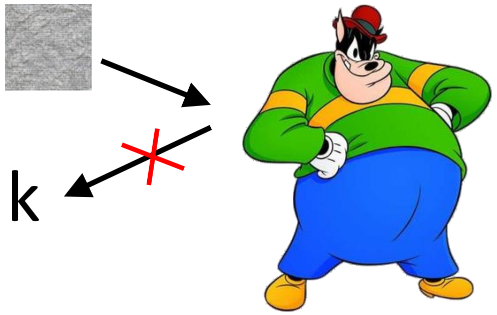
Obfuscation:



Correctness: same as before.

Security (informally):

*It should be **hard to extract any information about k**.*



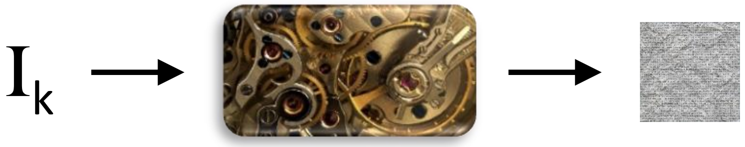
Point-Function Obfuscation (PO)

[Canetti'97, CMR98, LPS04, GK05, Wee'05, ...]

For any **target point** k, define a **point function** I_k:

$$I_k(x) = 1 \quad \text{if } x = k$$
$$I_k(x) = 0 \quad \text{if } x \neq k$$

Obfuscation:

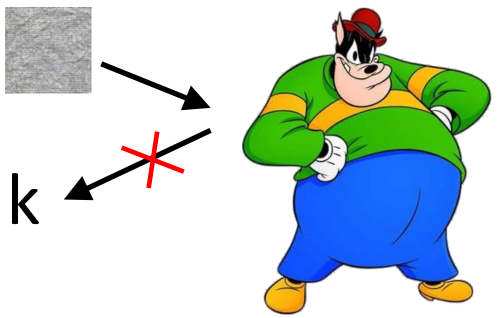


I_k

Correctness: same as before.

Security (informally):

It should be ***hard to extract any information about k.***



k

Definitional choices from prior work:

What is the distribution of k?

Is auxiliary information allowed? → Yes → How unpredictable is the target point, given aux? (comp., sub-exp., exp.)

Can use multiple, correlated target points?

…

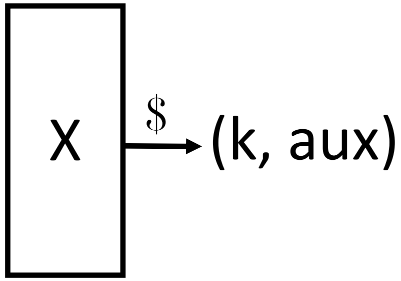
Framework for Point-Function Obfuscation

Propose **parameterized definitions** for point-function obfuscation (PO), and show how to get **generic constructions** from a number of assumptions.

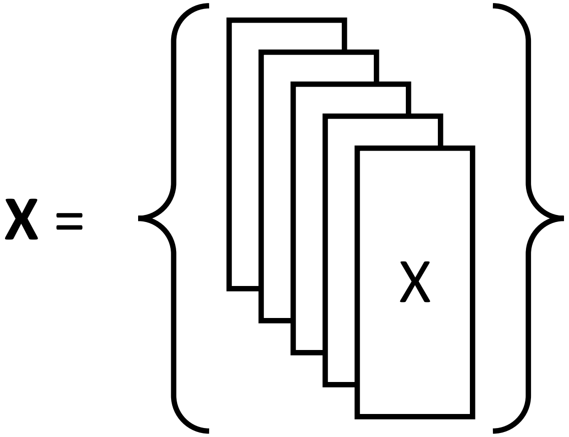
[BS16]

Similar to frameworks used for **UCE** [BHK13] and **(d)iO** [BST14].

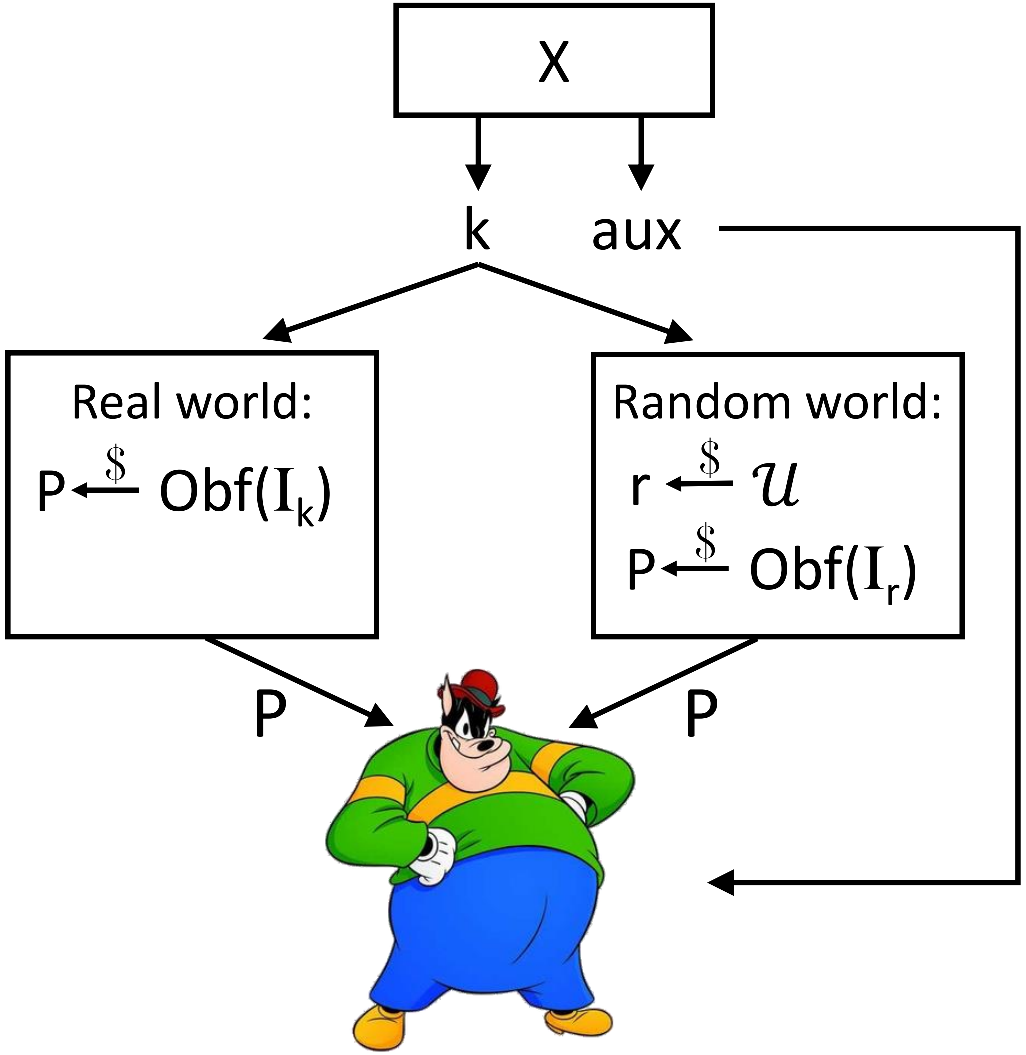
Target generator.



Class (set) of target generators.



IND[X]-secure Point-Function Obfuscators



X

k aux

Real world:

$P \xleftarrow{\$} \mathrm{Obf}(I_k)$

Random world:

$r \xleftarrow{\$} \mathcal{U}$

$P \xleftarrow{\$} \mathrm{Obf}(I_r)$

P

P

Obf is **IND[X]-secure** if no adversary can distinguish between the two worlds.

Some classes of target generators:

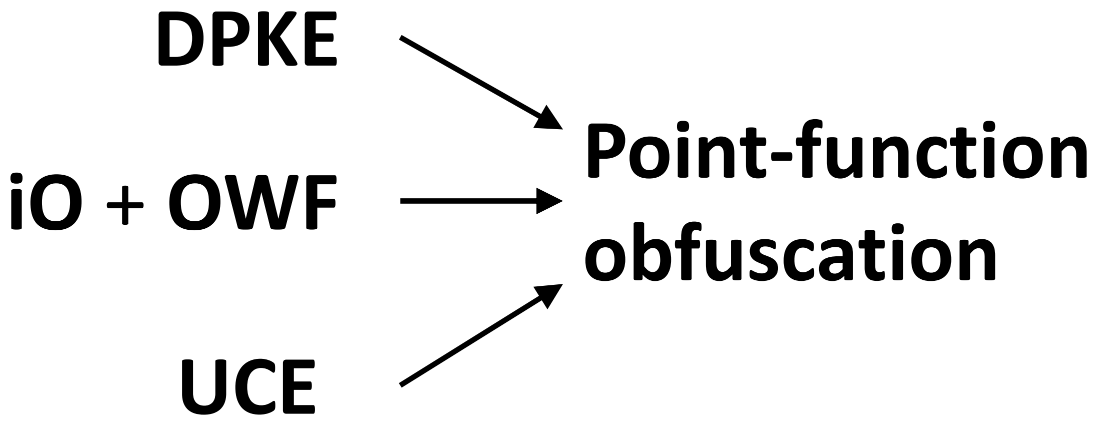
- $\mathbf{X}^{\varepsilon}$ – no auxiliary information
- $\mathbf{X}^{\mathrm{cup}}$ – computationally unpredictable
- $\mathbf{X}^{\mathrm{seup}}$ – sub-exponentially unpredictable
- $\mathbf{X}^{n}$ – n correlated target points

Some notions we recover:

- IND[$\mathbf{X}^{\mathrm{cup}} \cap \mathbf{X}^{1}$] – AIPO [Canetti'97, GK05, BP14, ...]
- IND[$\mathbf{X}^{\mathrm{cup}} \cap \mathbf{X}^{\varepsilon} \cap \mathbf{X}^{1}$] – basic PO [Canetti'97, ...]
- IND[$\mathbf{X}^{\mathrm{cup}}$] – composable AIPO [CD08, ...]

Generic constructions for PO

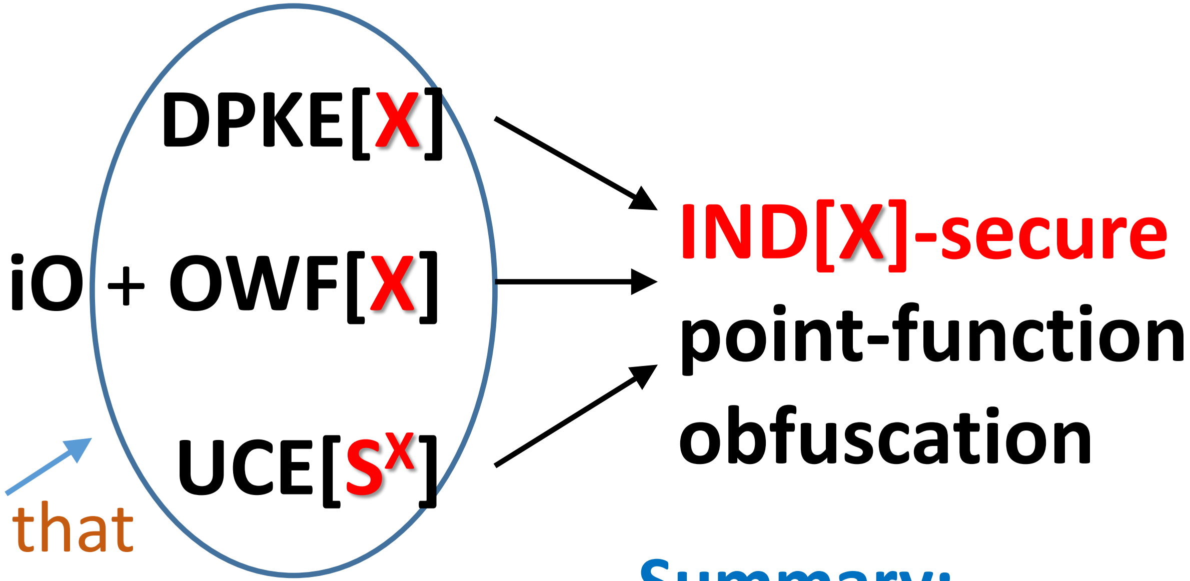
We provide three **generic constructions** of point-function obfuscation:



DPKE – Deterministic public-key encryption [BBO07, BFOR08, BS11, ...]
iO – Indistinguishability obfuscation [BGIRSVY01, GGHRSW13, SW13, ...]
OWF – One-way functions
UCE – Universal computational extractor [BHK13]

Generic constructions for PO

We provide three **generic constructions** of point-function obfuscation:



DPKE[X]

iO + OWF[X]

UCE[S^X]

IND[X]-secure point-function obfuscation

Extended definitions that are parameterized via **X**.

Brzuska-Mittelbach-15 concurrently showed a special case of our UCE construction.

Summary:

- We achieve new types of PO.
- We use standard assumptions in many cases.
- Negative results follow if IND[**X**] is known to be impossible (e.g. the case for IND[$\mathbf{X}^{\mathrm{cup}}$]).

More impossibility results for UCE

[BST16]: $\mathrm{iO} \not\leftrightarrow \mathrm{UCE}[\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}}]$

Brzuska-Mittelbach-15 obtained a similar but weaker contention regarding UCE[$\mathbf{S}^{\mathrm{s\text{-}cup}}$] in a **concurrent work**.

We **know no applications** of UCE[$\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}}$].

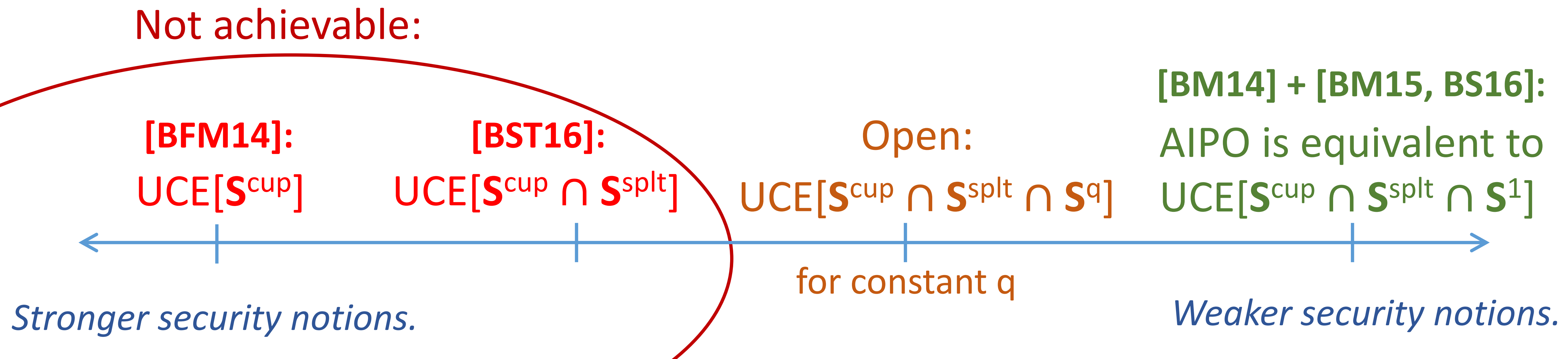
More impossibility results for UCE

[BST16]: $\mathrm{iO} \not\longleftrightarrow \mathrm{UCE}[\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}}]$

Brzuska-Mittelbach-15 obtained a similar but weaker contention regarding $\mathrm{UCE}[\mathbf{S}^{\mathrm{s\text{-}cup}}]$ in a **concurrent work**.

We **know no applications** of $\mathrm{UCE}[\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}}]$.

Current state of computationally unpredictable sources, assuming iO:



Not achievable:

[BFM14]: $\mathrm{UCE}[\mathbf{S}^{\mathrm{cup}}]$

[BST16]: $\mathrm{UCE}[\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}}]$

Open: $\mathrm{UCE}[\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}} \cap \mathbf{S}^{q}]$ for constant q

[BM14] + [BM15, BS16]: AIPO is equivalent to $\mathrm{UCE}[\mathbf{S}^{\mathrm{cup}} \cap \mathbf{S}^{\mathrm{splt}} \cap \mathbf{S}^{1}]$

Stronger security notions.

Weaker security notions.

Thank you!