New Negative Results on Differing-Inputs Obfuscation

May 12, 2016
EUROCRYPT 2016

Mihir Bellare
UC San Diego

Igors Stepanovs
UC San Diego

Brent Waters
The University of Texas at Austin

Our Main Result at a Glance

Differing-inputs obfuscation (Barak et al., 2001)

[GGHW14]: **Differing-inputs obfuscation is implausible**

… because it cannot coexist with another form of obfuscation that seems to be weaker.

This work: **Differing-inputs obfuscation is impossible**

… assuming sub-exponentially secure one-way functions.

Our Main Result at a Glance

Differing-inputs obfuscation (Barak et al., 2001)

for circuits

[GGHW14]: Differing-inputs obfuscation is implausible

… because it cannot coexist with another form of obfuscation that seems to be weaker.

sub-exp secure

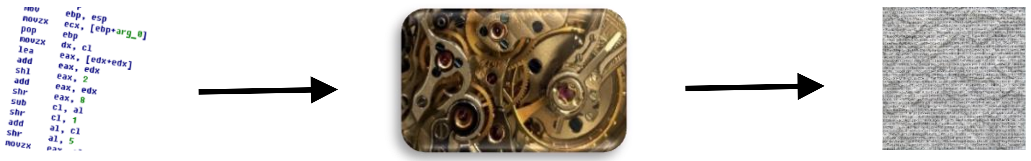
for TMs

This work: Differing-inputs obfuscation is impossible

… assuming sub-exponentially secure one-way functions.

Obfuscation

Bellare, Stepanovs, Waters - EUROCRYPT 2016

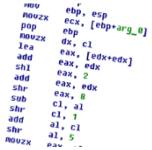


```
mov
movzx   ebp, esp
pop     ecx, [ebp+arg_0]
movzx   ebp
lea     dx, cl
add     eax, [edx+edx]
shl     eax, edx
add     eax, 2
shr     eax, edx
sub     eax, 8
shr     cl, al
add     cl, 1
shr     al, cl
movzx   al, 5
        eax
```

Program **P** Obfuscator Program $\mathbf{P}^*$

Circuits or Turing Machines

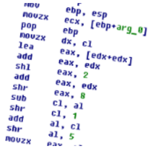
1. Correctness:

and  functionally equivalent,

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

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

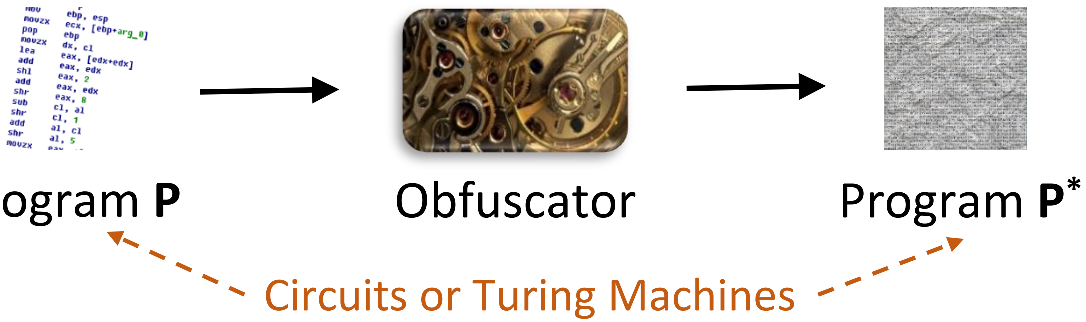
2. Security:

no more useful than an oracle for 

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

Obfuscation

Bellare, Stepanovs, Waters - EUROCRYPT 2016



```
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
```

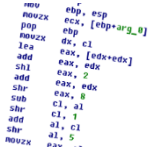
Program **P**

Obfuscator

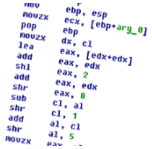
Program $\mathbf{P}^*$

Circuits or Turing Machines

1. Correctness:

and  functionally equivalent, i.e. $\mathbf{P}(x) = \mathbf{P}^*(x)$ for all x.

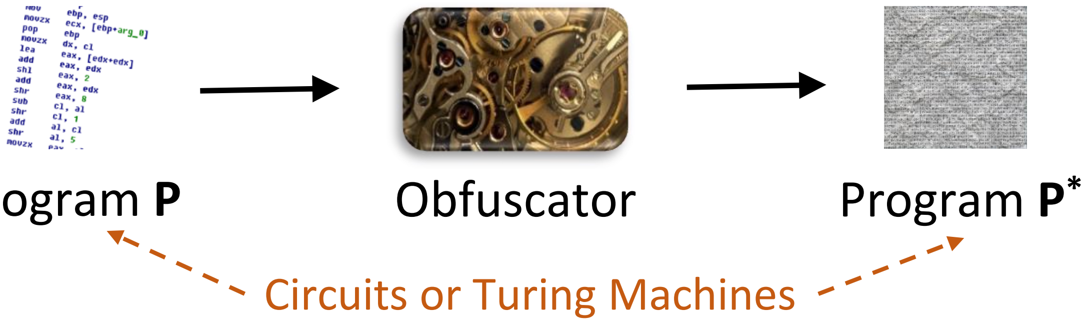
2. Security:

no more useful than an oracle for 

[BGIRSVY01]: Virtual Black Box Obfuscation is impossible!

Obfuscation

Bellare, Stepanovs, Waters - EUROCRYPT 2016



Program **P**

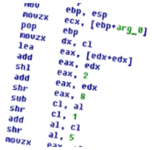
Obfuscator

Program **P***

Circuits or Turing Machines

1. Correctness:

and

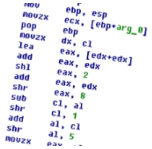


functionally equivalent,

i.e. **P**(x) = **P***(x) for all x.

2. Security:

no more useful
than an oracle for



[BGIRSVY01]: Virtual Black Box Obfuscation is impossible!

Are there **weaker forms of obfuscation** that are **achievable** and **useful**?

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

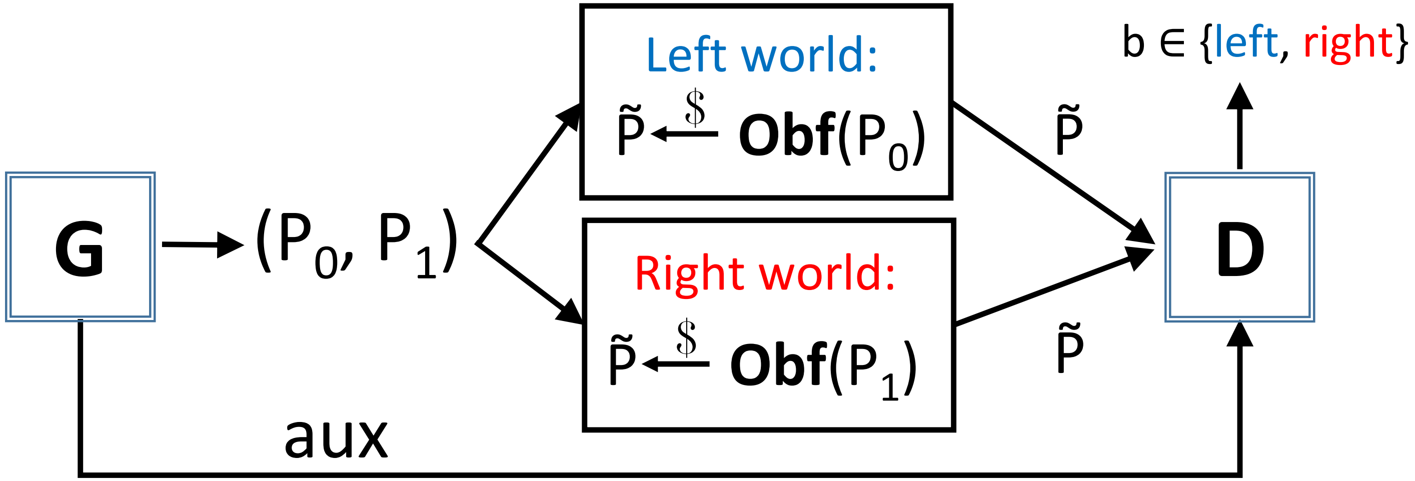
VGBO – virtual grey box obfuscation [BC10, ...]

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

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

Indistinguishability and Differing-Inputs Obfuscation

[BGIRSVY01]


G
(P_0, P_1)
Left world:
$\tilde{P} \xleftarrow{\$}$ **Obf**(P_0)
Right world:
$\tilde{P} \xleftarrow{\$}$ **Obf**(P_1)
$\tilde{P}$
$\tilde{P}$
D
b ∈ {left, right}
aux

Security of indistinguishability obfuscation (iO):

Obf is iO-secure if:

For all PT adversaries **G** that output (P_0, P_1) such that $P_0 \equiv P_1$

no PT adversary **D** can distinguish left from right.

computationally hard

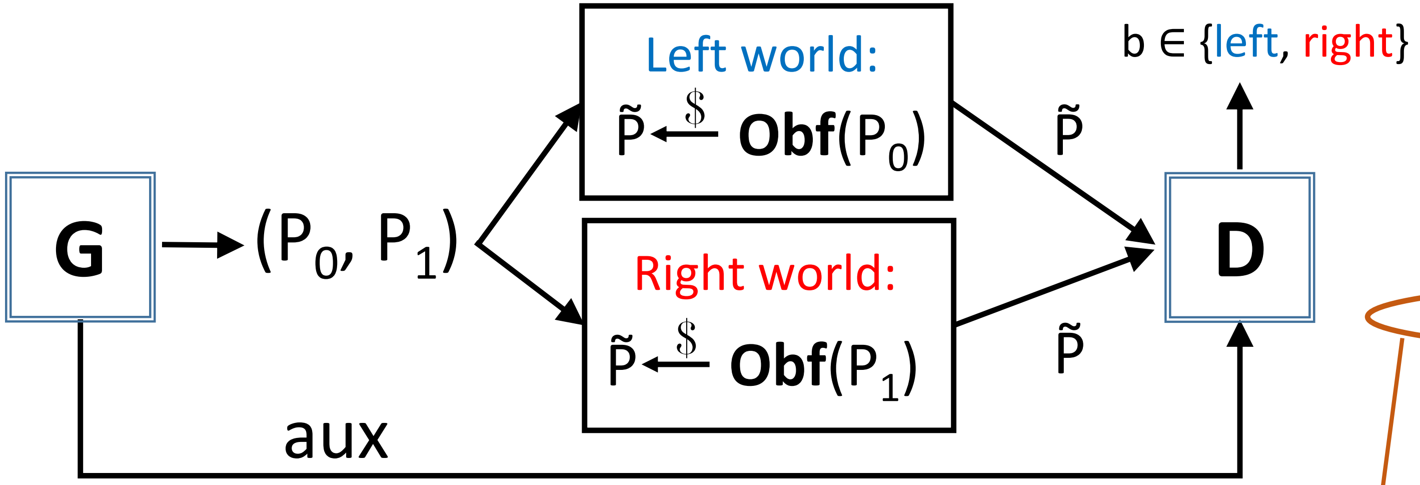
PT adversaries:

G – Generator;

D – Distinguisher;

Indistinguishability and Differing-Inputs Obfuscation

[BGIRSVY01]



b ∈ {left, right}

G

(P_0, P_1)

Left world:

$\tilde{P} \xleftarrow{\$} \textbf{Obf}(P_0)$

Right world:

$\tilde{P} \xleftarrow{\$} \textbf{Obf}(P_1)$

$\tilde{P}$

$\tilde{P}$

D

aux

Security of indistinguishability obfuscation (iO):

Obf is iO-secure if:

For all PT adversaries **G** that output (P_0, P_1) such that $P_0 \equiv P_1$ no PT adversary **D** can distinguish left from right.

Security of differing-inputs obfuscation (diO):

Obf is diO-secure if:

For all PT adversaries **G** that output (P_0, P_1) such that it is computationally hard to find x satisfying $P_0(x) \neq P_1(x)$ no PT adversary **D** can distinguish left from right.

PT adversaries:

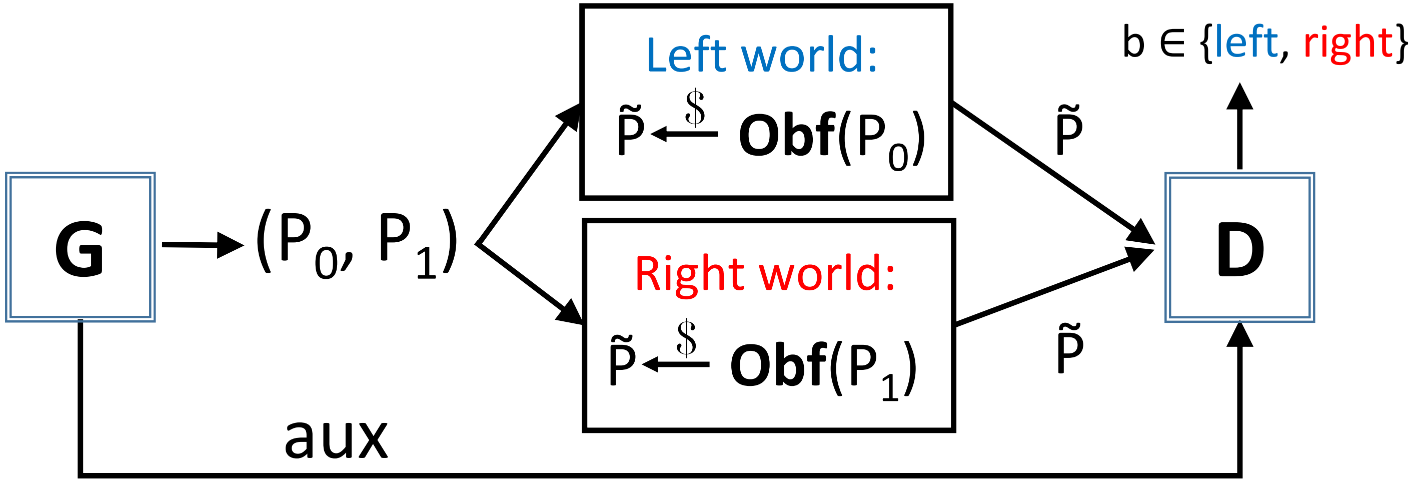
G – Generator;

D – Distinguisher;

Indistinguishability and Differing-Inputs Obfuscation

Bellare, Stepanovs, Waters - EUROCRYPT 2016

[BGIRSVY01]

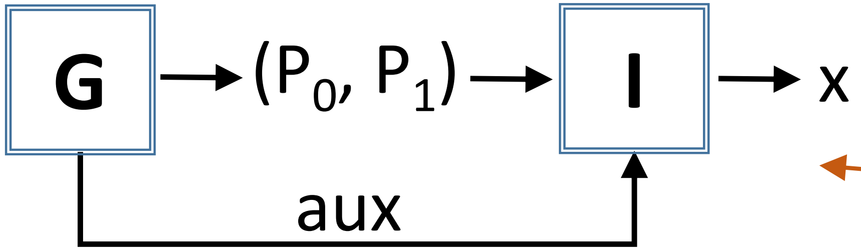

G
(P_0, P_1)
Left world:
$\tilde{P} \xleftarrow{\$} \mathbf{Obf}(P_0)$
Right world:
$\tilde{P} \xleftarrow{\$} \mathbf{Obf}(P_1)$
$\tilde{P}$
$\tilde{P}$
D
$b \in \{$left, right$\}$
aux

Security of indistinguishability obfuscation (iO):

Obf is iO-secure if:

For all PT adversaries **G** that output (P_0, P_1) such that $P_0 \equiv P_1$
no PT adversary **D** can distinguish left from right.

PT adversaries:
- **G – Generator;**
- **D – Distinguisher;**
- **I – Inverter.**



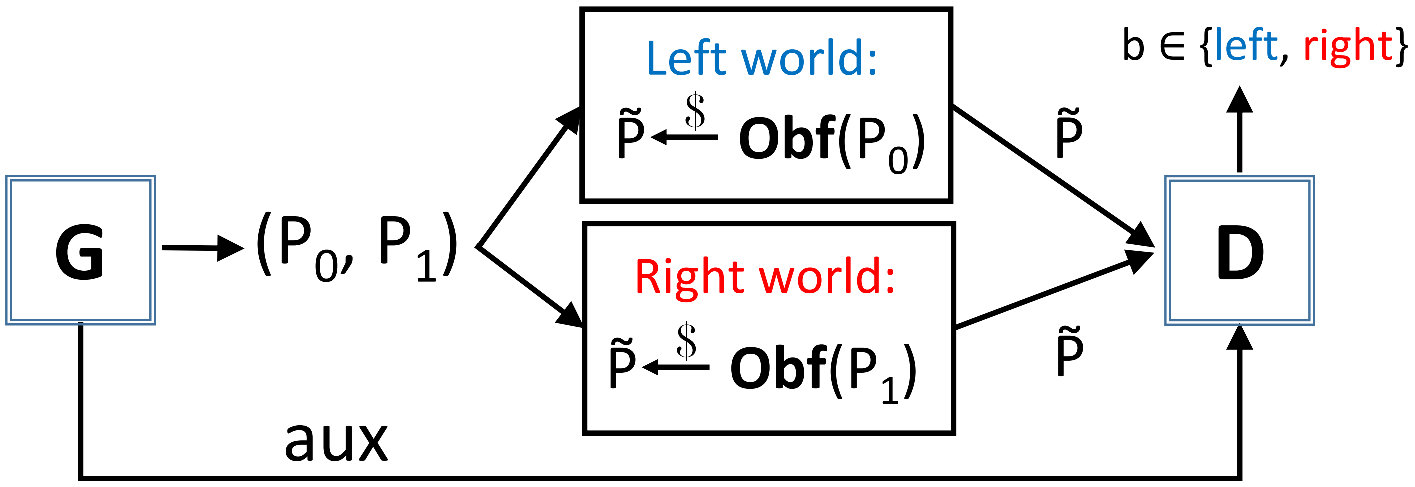
Security of differing-inputs obfuscation (diO):

Obf is diO-secure if:

For all PT adversaries **G** that output
(P_0, P_1) such that it is computationally hard to find x satisfying $P_0(x) \neq P_1(x)$
no PT adversary **D** can distinguish left from right.

Indistinguishability and Differing-Inputs Obfuscation

[BGIRSVY01]


G
(P_0, P_1)
Left world:
$\tilde{P} \xleftarrow{\$} \mathbf{Obf}(P_0)$
Right world:
$\tilde{P} \xleftarrow{\$} \mathbf{Obf}(P_1)$
$\tilde{P}$
$\tilde{P}$
D
$b \in \{\text{left}, \text{right}\}$
aux

Security of indistinguishability obfuscation (iO):

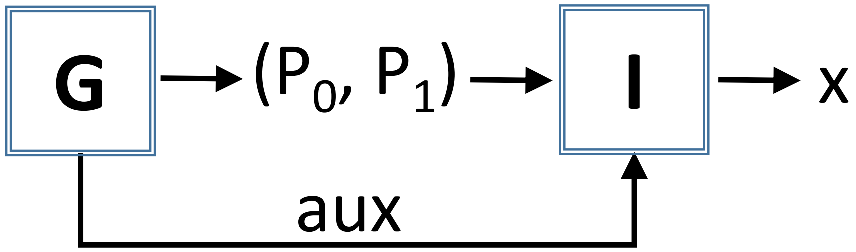
Obf is iO-secure if:

For all PT adversaries **G** that output (P_0, P_1) such that $P_0 \equiv P_1$

no PT adversary **D** can distinguish left from right.

PT adversaries:

- **G – Generator;**
- **D – Distinguisher;**
- **I – Inverter.**



Security of differing-inputs obfuscation (diO):

Obf is diO-secure if:

For all PT adversaries **G** that output (P_0, P_1) such that it is computationally hard to find x satisfying $P_0(x) \neq P_1(x)$

no PT adversary **D** can distinguish left from right.

We consider two security levels:

(1) Polynomially diO-secure .... polynomially hard

(2) Sub-exponentially diO-secure .... sub-exponentially hard

Indistinguishability Obfuscation (iO)

Is iO achievable?

Why should I care?!

[GGHRSW13, ...]

Here is a candidate construction!

[SW13, ...]

We can build many crypto primitives from iO!

"iO as a central hub of cryptography"

Indistinguishability Obfuscation (iO)

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Is iO achievable?

Why should I care?!

[GGHRSW13, …]

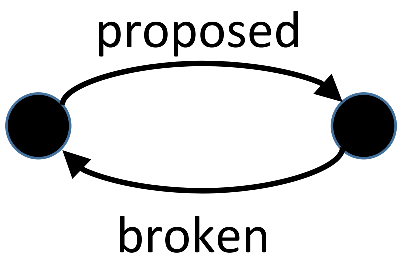
Here is a candidate construction!

[SW13, …]

We can build many crypto primitives from iO!

“iO as a central hub of cryptography”

Heavy, ad-hoc assumptions.
Constructions are getting broken.



Does iO exist?

Indistinguishability Obfuscation (iO)

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Is iO achievable?

Why should I care?!

[GGHRSW13, …]

Here is a candidate construction!

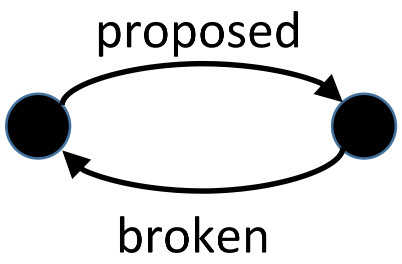
[SW13, …]

We can build many crypto primitives from iO!

“iO as a central hub of cryptography”

Heavy, ad-hoc assumptions.
Constructions are getting broken.

Candidate iO constructions conjectured to meet diO.
(Proven in idealized models by BR13, BGKPS13).



Does iO exist?

We make progress towards settling the existence of iO by providing negative results for diO.

Implausibility of Differing-Inputs Obfuscation

[GGHW14]

Theorem ([GGHW14]): **Polynomially secure diO for circuits does not exist** if:

there exists an existentially unforgeable **digital signature scheme** DS, and
there exists a collision-resistant **hash function** H, and
there exists a **special-purpose obfuscator** for H and DS.

A novel, ad-hoc assumption introduced by [GGHW14].
Is it more plausible than diO?

[GGHW14]

Differing-inputs obfuscation is implausible!

Our Results

Theorem A. Sub-exponentially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist.

← The proof uses iO!

Theorem B. Polynomially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist, and
sub-exponentially secure **indistinguishability obfuscation for circuits** exists.

Our Results

Theorem A. Sub-exponentially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist.

← The proof uses iO!

Theorem B. Polynomially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist, and
sub-exponentially secure **indistinguishability obfuscation for circuits** exists.

	Type of programs	**Assumptions**
[GGHW14] theorem	Circuits	Special-purpose obfuscation, …
Theorem A	Turing Machines	Sub-exponentially secure OWFs [and sub-exponentially secure iO]

Our Results

Theorem A. Sub-exponentially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist.

← The proof uses iO!

Theorem B. Polynomially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist, and
sub-exponentially secure **indistinguishability obfuscation for circuits** exists.

	Type of programs	**Assumptions**
[GGHW14] theorem	Circuits	Special-purpose obfuscation, …
Theorem A	Turing Machines	Sub-exponentially secure OWFs [and sub-exponentially secure iO]

Obtain a corollary for circuits from:

[ABGSZ13, BCP14]

FHE + diO for circuits + SNARKs → diO for TMs.

Our Results

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Theorem A. Sub-exponentially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist.

← The proof uses iO!

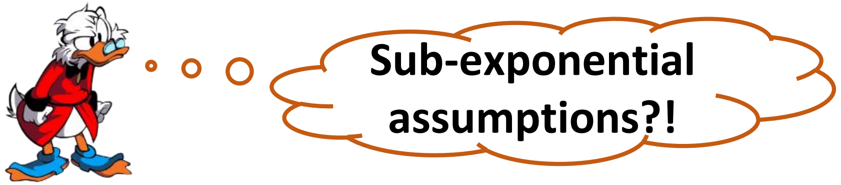
Theorem B. Polynomially secure diO for TMs does not exist if:
sub-exponentially secure **one-way functions** exist, and
sub-exponentially secure **indistinguishability obfuscation for circuits** exists.

	Type of programs	Assumptions
[GGHW14] theorem	Circuits	Special-purpose obfuscation, …
Theorem A	Turing Machines	Sub-exponentially secure OWFs [and sub-exponentially secure iO]

Obtain a corollary for circuits from:

[ABGSZ13, BCP14]

FHE + diO for circuits + SNARKs ⟶ diO for TMs.



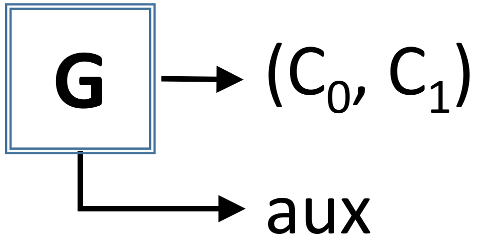
When natural problems are hard,
they appear to be sub-exponentially hard.

(Factoring, DLOG, LWE, SVP, ...).

[GGHW14] Attack

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Construct generator G using: digital signature scheme DS, “special-purpose obfuscator” spO, hash function H.



Let Obf be any obfuscator. It is not diO-secure if:

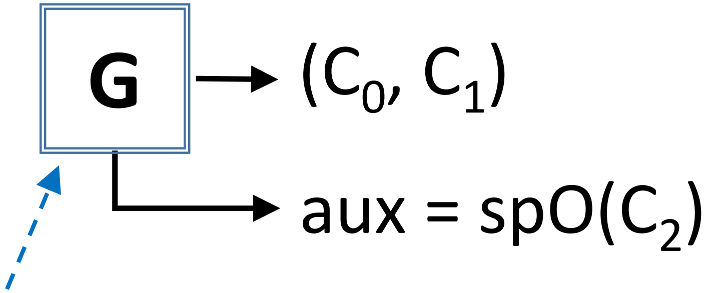
(1) It is easy to distinguish Obf(C_0) from Obf(C_1).

(2) It is hard to find x such that $C_0(x) \neq C_1(x)$.

[GGHW14] Attack

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Construct generator G using: digital signature scheme DS, “special-purpose obfuscator” spO, hash function H.



Generates a key pair (vk,sk) for DS.

```
$C_0(m, \sigma)$:
    Return 0

$C_1(m, \sigma)$:
    $d \leftarrow$ DS.Verify(vk, m, $\sigma$)
    Return d

$C_2(\tilde{C})$:
    $m \leftarrow H(\tilde{C})$
    $\sigma \leftarrow$ DS.Sign(sk, m)
    $b \leftarrow \tilde{C}(m, \sigma)$
    Return b
```

Let Obf be any obfuscator. It is not diO-secure if:

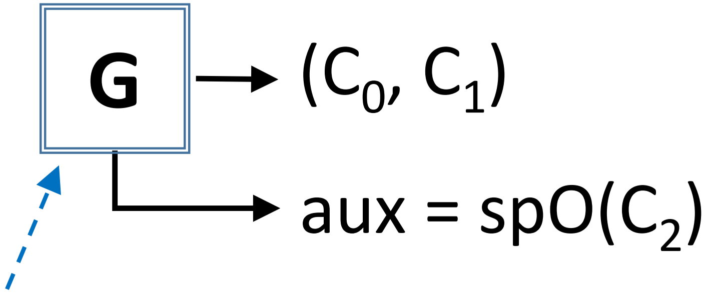
(1) It is easy to distinguish Obf(C_0) from Obf(C_1).

(2) It is hard to find x such that $C_0(x) \neq C_1(x)$.

[GGHW14] Attack

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Construct generator G using: digital signature scheme DS, “special-purpose obfuscator” spO, hash function H.



Generates a key pair (vk,sk) for DS.

$C_0(m, \sigma)$:
```
Return 0
```

$C_1(m, \sigma)$:
```
$d \leftarrow$ DS.Verify(vk, m, σ)
Return d
```

$C_2(\tilde{C})$:
```
$m \leftarrow H(\tilde{C})$
$\sigma \leftarrow$ DS.Sign(sk, m)
$b \leftarrow \tilde{C}(m, \sigma)$
Return b
```

Let Obf be any obfuscator. It is not diO-secure if:

(1) It is easy to distinguish $\text{Obf}(C_0)$ from $\text{Obf}(C_1)$. ☑

(2) It is hard to find x such that $C_0(x) \neq C_1(x)$.

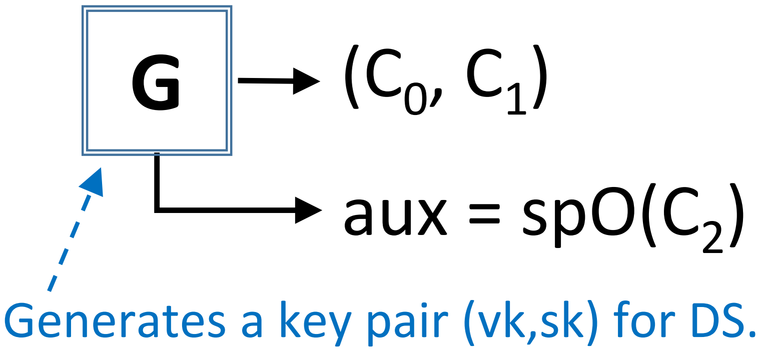
$$C_2(\tilde{C}) = \begin{cases} 0 & \text{if } \tilde{C} \text{ is } \text{Obf}(C_0) \\ 1 & \text{if } \tilde{C} \text{ is } \text{Obf}(C_1) \end{cases}$$

$D(\tilde{C}, \text{aux})$:
```
$b \leftarrow \text{aux}(\tilde{C})$
Return b
```

[GGHW14] Attack

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Construct generator G using: digital signature scheme DS, “special-purpose obfuscator” spO, hash function H.



G → (C_0, C_1)

aux = spO(C_2)

Generates a key pair (vk,sk) for DS.

```
C_0(m, σ):
    Return 0

C_1(m, σ):
    d ← DS.Verify(vk, m, σ)
    Return d

C_2(C̃):
    m ← H(C̃)
    σ ← DS.Sign(sk, m)
    b ← C̃(m, σ)
    Return b
```

Let Obf be any obfuscator. It is not diO-secure if:

(1) It is easy to distinguish Obf(C_0) from Obf(C_1). ☑

(2) It is hard to find x such that $C_0(x) \neq C_1(x)$. ☑

$$C_2(\tilde{C}) = \begin{cases} 0 & \text{if } \tilde{C} \text{ is Obf}(C_0) \\ 1 & \text{if } \tilde{C} \text{ is Obf}(C_1) \end{cases}$$

```
D(C̃, aux):
    b ← aux(C̃)
    Return b
```

Assume there exists spO that hides sk “sufficiently good”.

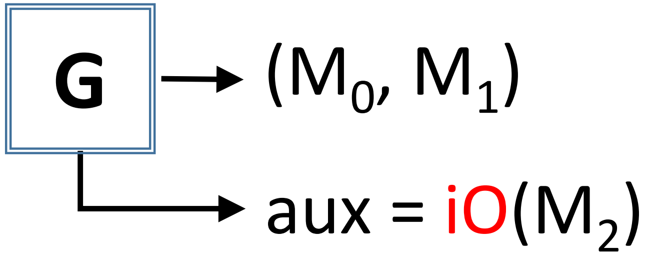
[GGHW14]

spO is more plausible than diO!

Our Attack

Bellare, Stepanovs, Waters - EUROCRYPT 2016

Construct generator G using: digital signature scheme DS, indistinguishability obfuscator iO.



$M_0(m, \sigma)$:

```
Return 0
```

$M_1(m, \sigma)$:

```
If $|m| \neq k$ then return 0
$d \leftarrow$ DS.Verify(vk, m, σ)
Return d
```

$M_2(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
$\sigma \leftarrow$ DS.Sign(sk, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

Let Obf be any obfuscator. It is not diO-secure if:

(1) It is easy to distinguish Obf(M_0) from Obf(M_1). ☑

(2) It is hard to find x such that $M_0(x) \neq M_1(x)$.

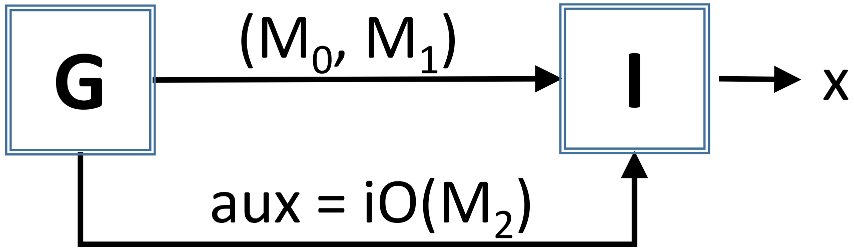
We change the construction of G as follows:

1. Replace spO with iO.
2. Replace circuits with TMs.
3. Require $|m| = k$ in M_1.
4. Remove hash function.
5. …

We now use a hybrid argument to prove (2).

Hybrid Argument

Bellare, Stepanovs, Waters - EUROCRYPT 2016



$M_0(m, \sigma)$:

```
Return 0
```

$M_1(m, \sigma)$:

```
If |m| ≠ k then return 0
d ⟵ DS.Verify(vk, m, σ)
Return d
```

$M_2(\tilde{M})$:

```
m ⟵ M̃
σ ⟵ DS.Sign(sk, m)
b ⟵ M̃(m, σ)
Return b
```

Adversary **I** wins if it outputs **x** such that…

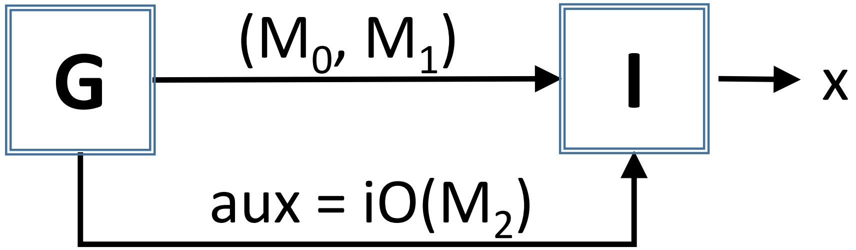
Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

String of length k.

Hybrid Argument

Bellare, Stepanovs, Waters - EUROCRYPT 2016



$M_0(m, \sigma)$:

```
Return 0
```

$M_1(m, \sigma)$:

```
If $|m| \neq k$ then return 0
$d \leftarrow$ DS.Verify(vk, m, $\sigma$)
Return d
```

$M_2(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
$\sigma \leftarrow$ DS.Sign(sk, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

Adversary **I** wins if it outputs **x** such that…

Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

String of length k.

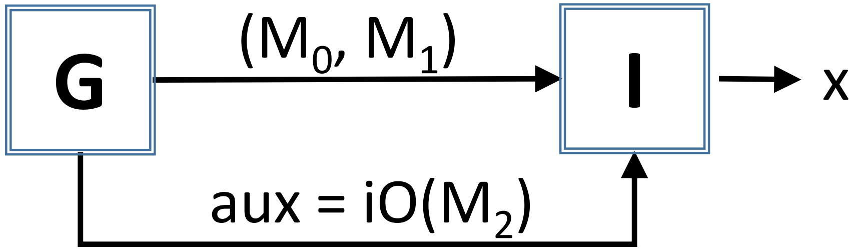
Hybrid game 2^k.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** > “11…11”.

Adversary cannot win.

Hybrid Argument

Bellare, Stepanovs, Waters - EUROCRYPT 2016



M_0(m, σ):

```
Return 0
```

M_1(m, σ):

```
If |m| ≠ k then return 0
d ← DS.Verify(vk, m, σ)
Return d
```

M_2($\tilde{M}$):

```
m ← $\tilde{M}$
σ ← DS.Sign(sk, m)
b ← $\tilde{M}$(m, σ)
Return b
```

Adversary **I** wins if it outputs **x** such that…

Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…00".

String of length k.

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…01".

…

Hybrid game 2^k-1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "11…11".

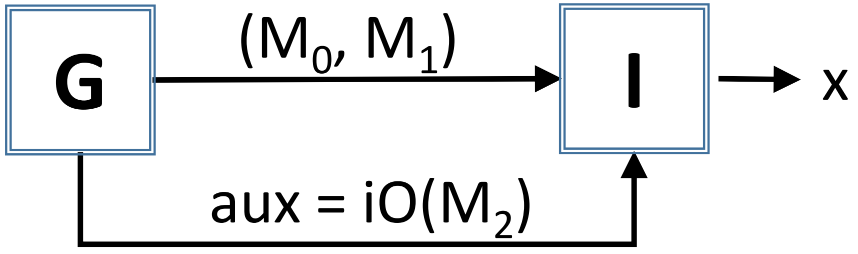
Hybrid game 2^k.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** > "11…11".

Adversary cannot win.

Hybrid Argument

Bellare, Stepanovs, Waters - EUROCRYPT 2016



```
$M_0(m, \sigma)$:
    Return 0
```

```
$M_1(m, \sigma)$:
    If $|m| \neq k$ then return 0
    $d \leftarrow$ DS.Verify(vk, m, σ)
    Return d
```

```
$M_2(\tilde{M})$:
    $m \leftarrow \tilde{M}$
    $\sigma \leftarrow$ DS.Sign(sk, m)
    $b \leftarrow \tilde{M}(m, \sigma)$
    Return b
```

Adversary **I** wins if it outputs **x** such that…

Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…00".

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…01".

…

Hybrid game 2^k-1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "11…11".

Hybrid game 2^k.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** > "11…11".

String of length k.

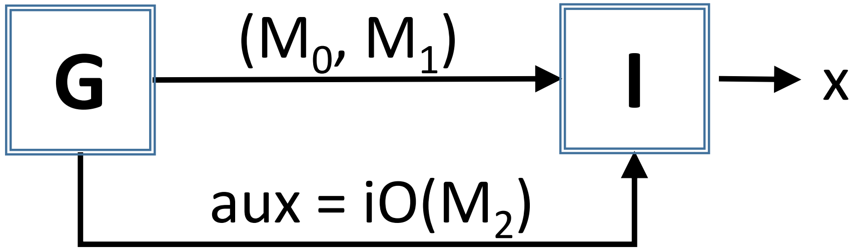
sub-exp small

sub-exp small

Adversary cannot win.

Hybrid Argument

Bellare, Stepanovs, Waters - EUROCRYPT 2016



$M_0(m, \sigma)$:

```
Return 0
```

$M_1(m, \sigma)$:

```
If $|m| \neq k$ then return 0
$d \leftarrow$ DS.Verify(vk, m, σ)
Return d
```

$M_2(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
$\sigma \leftarrow$ DS.Sign(sk, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

Adversary **I** wins if it outputs **x** such that…

Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…00".

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…01".

…

Hybrid game 2^k-1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "11…11".

Hybrid game 2^k.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** > "11…11".

String of length k.

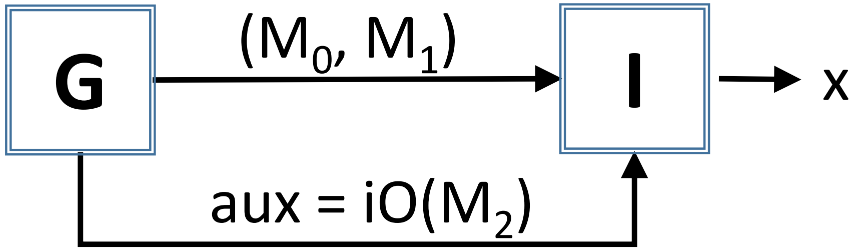
sub-exp small

sub-exp small

Adversary cannot win.

sub-exp small

Hybrid Argument: A Single Transition



```
M_0(m, σ):
    Return 0
```

```
M_1(m, σ):
    If $|m| \neq k$ then return 0
    $d \leftarrow$ DS.Verify(vk, m, σ)
    Return d
```

```
M_2(M̃):
    $m \leftarrow \tilde{M}$
    $\sigma \leftarrow$ DS.Sign(sk, m)
    $b \leftarrow \tilde{M}(m, \sigma)$
    Return b
```

Adversary **I** wins if it outputs **x** such that…

Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

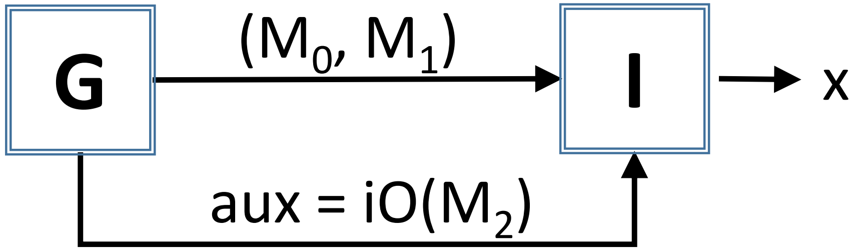
Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…01”.

sub-exp small

Hybrid Argument: A Single Transition

Bellare, Stepanovs, Waters - EUROCRYPT 2016



$M_0(m, \sigma)$:

```
Return 0
```

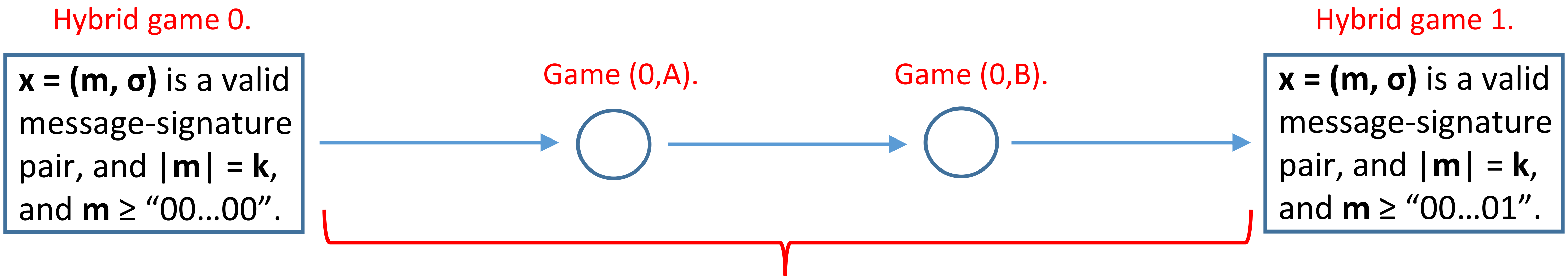
$M_1(m, \sigma)$:

```
If $|m| \neq k$ then return 0
$d \leftarrow DS.Verify(vk, m, \sigma)$
Return d
```

$M_2(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
$\sigma \leftarrow DS.Sign(sk, m)$
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

Adversary **I** wins if it outputs **x** such that…



Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

Game (0,A).

Game (0,B).

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…01”.

3 intermediate steps between every two hybrid games.

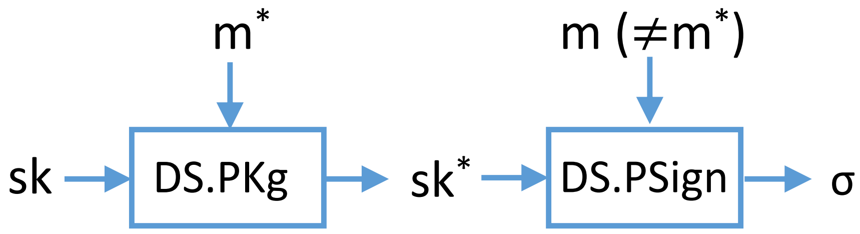
We use consistent puncturable signature schemes.
In the spirit of puncturable PRFs.

Consistent Puncturable Signature Schemes

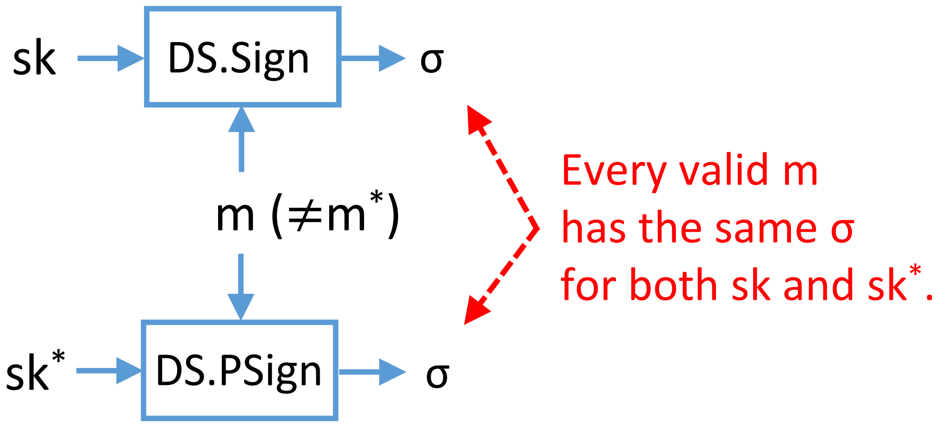
Bellare, Stepanovs, Waters - EUROCRYPT 2016

We define a signature scheme DS that is:

1. Puncturable.



2. Consistent.



Every valid m has the same σ for both sk and sk^*.

We require selective puncturable unforgeability:

PT adversary **A**:

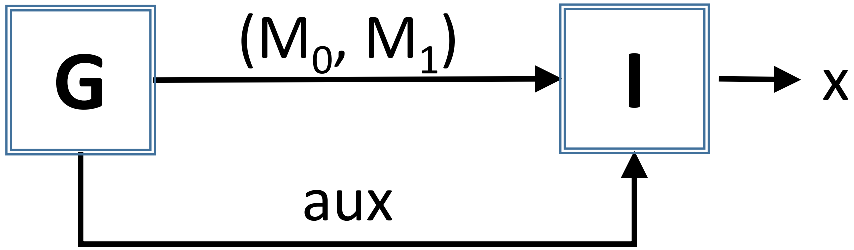
1. Chooses a challenge message m^*.
2. Receives (vk, sk^*), where sk^* is punctured at m^*.
3. Is asked to forge a valid signature for m^*.

We build a consistent puncturable signature scheme from iO and PPRF.

Our construction follows Sahai-Waters signatures [SW13].

Hybrid Argument: A Single Transition

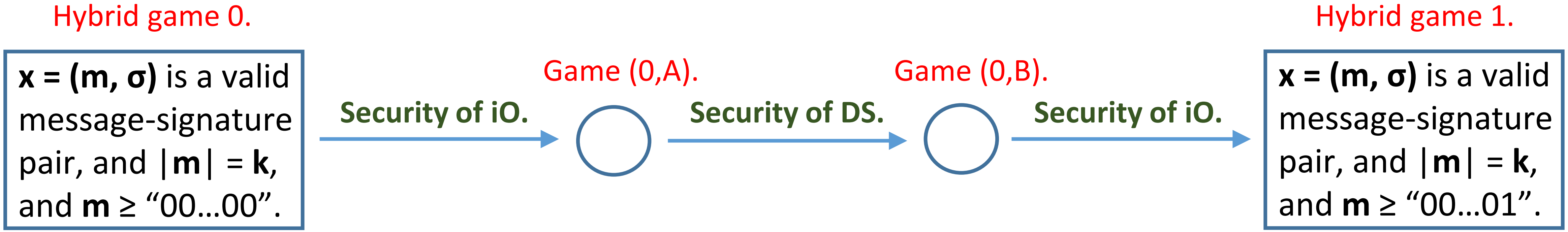
Bellare, Stepanovs, Waters - EUROCRYPT 2016



$M_2(\tilde{M})$:

```
m ← M̃
σ ← DS.Sign(sk, m)
b ← M̃(m, σ)
Return b
```

Adversary **I** wins if it outputs **x** such that…



Hybrid game 0.

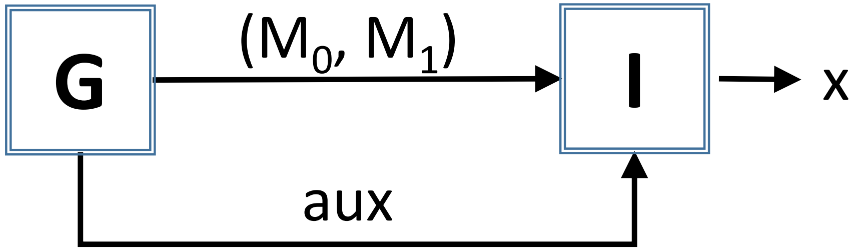
x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…01”.

Hybrid Argument: A Single Transition

Bellare, Stepanovs, Waters - EUROCRYPT 2016



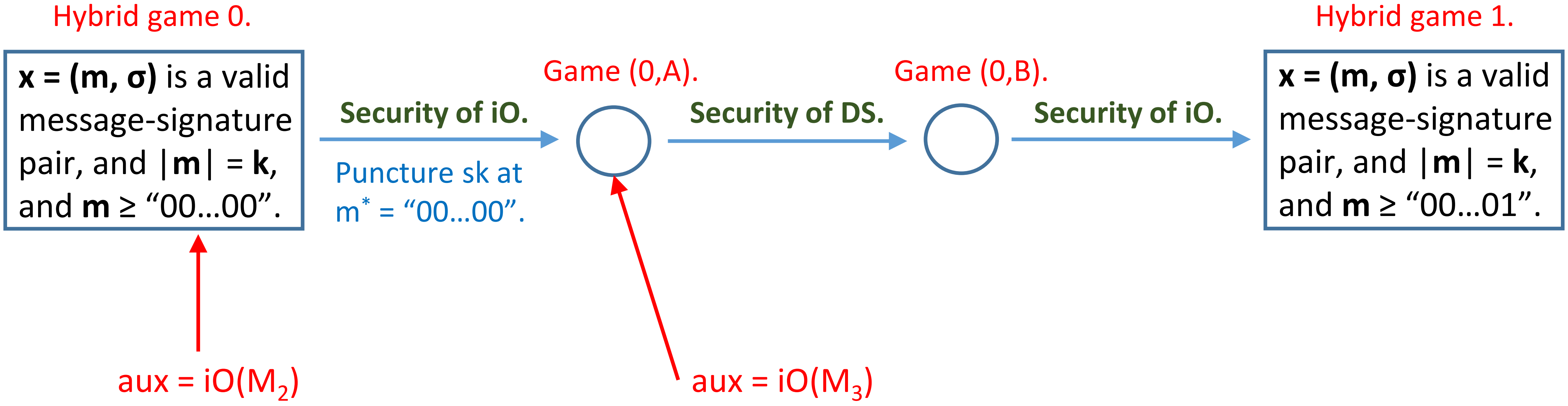
$M_2(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
$\sigma \leftarrow$ DS.Sign(sk, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

$M_3(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
If $(m = m^*)$ then return $b^*$
$\sigma \leftarrow$ DS.Sign($sk^*$, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

Adversary **I** wins if it outputs **x** such that…



Hybrid game 0.

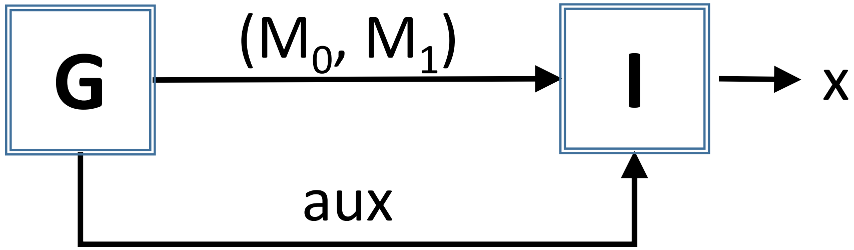
x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…01”.

Hybrid Argument: A Single Transition

Bellare, Stepanovs, Waters - EUROCRYPT 2016



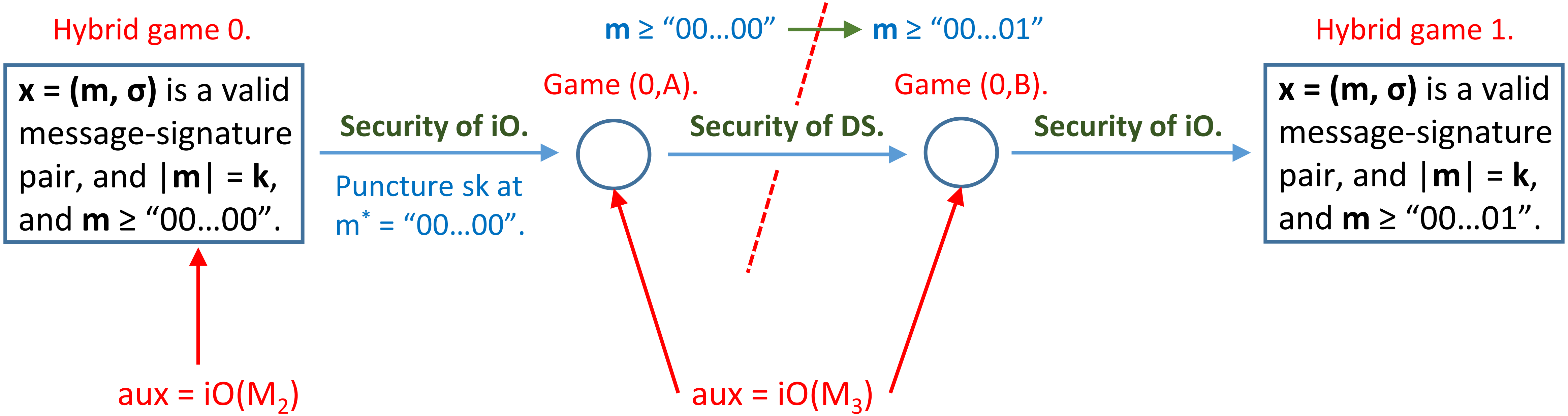
$M_2(\tilde{M})$:

```
m ← $\tilde{M}$
σ ← DS.Sign(sk, m)
b ← $\tilde{M}$(m, σ)
Return b
```

$M_3(\tilde{M})$:

```
m ← $\tilde{M}$
If (m = $m^*$) then return $b^*$
σ ← DS.Sign($sk^*$, m)
b ← $\tilde{M}$(m, σ)
Return b
```

Adversary **I** wins if it outputs **x** such that…



Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…00".

aux = iO(M_2)

Security of iO.

Puncture sk at m^* = "00…00".

Game (0,A).

m ≥ "00…00" → **m** ≥ "00…01"

Security of DS.

Game (0,B).

aux = iO(M_3)

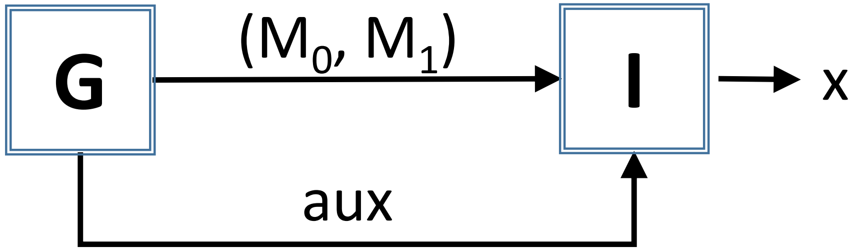
Security of iO.

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ "00…01".

Hybrid Argument: A Single Transition

Bellare, Stepanovs, Waters - EUROCRYPT 2016



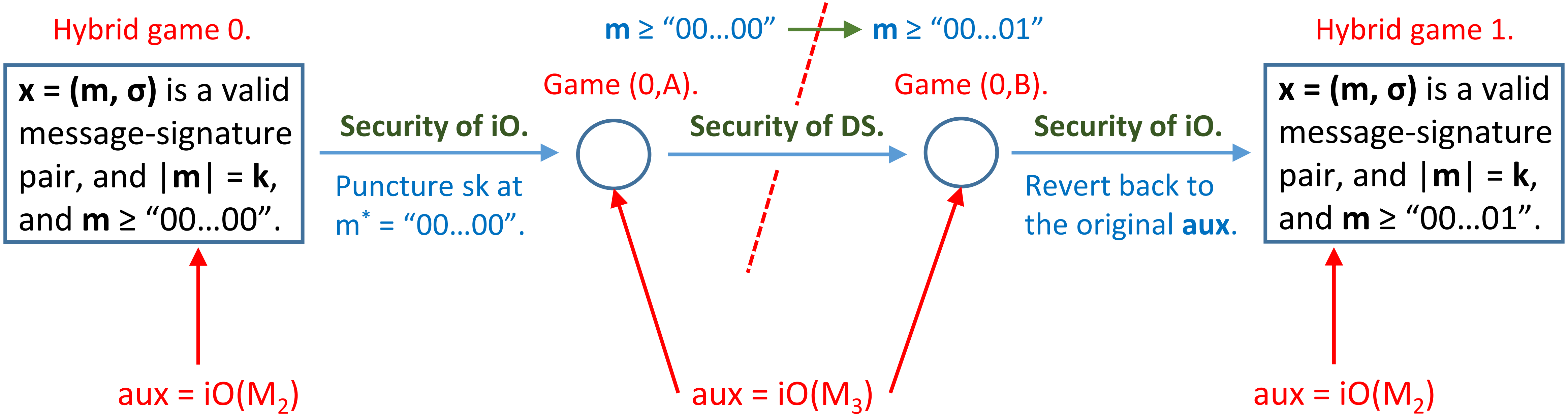
$M_2(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
$\sigma \leftarrow$ DS.Sign(sk, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

$M_3(\tilde{M})$:

```
$m \leftarrow \tilde{M}$
If $(m = m^*)$ then return $b^*$
$\sigma \leftarrow$ DS.Sign($sk^*$, m)
$b \leftarrow \tilde{M}(m, \sigma)$
Return b
```

Adversary **I** wins if it outputs **x** such that…



Hybrid game 0.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…00”.

aux = iO(M_2)

Security of iO.

Puncture sk at m^* = “00…00”.

Game (0,A).

aux = iO(M_3)

m ≥ “00…00” → **m** ≥ “00…01”

Security of DS.

Game (0,B).

Security of iO.

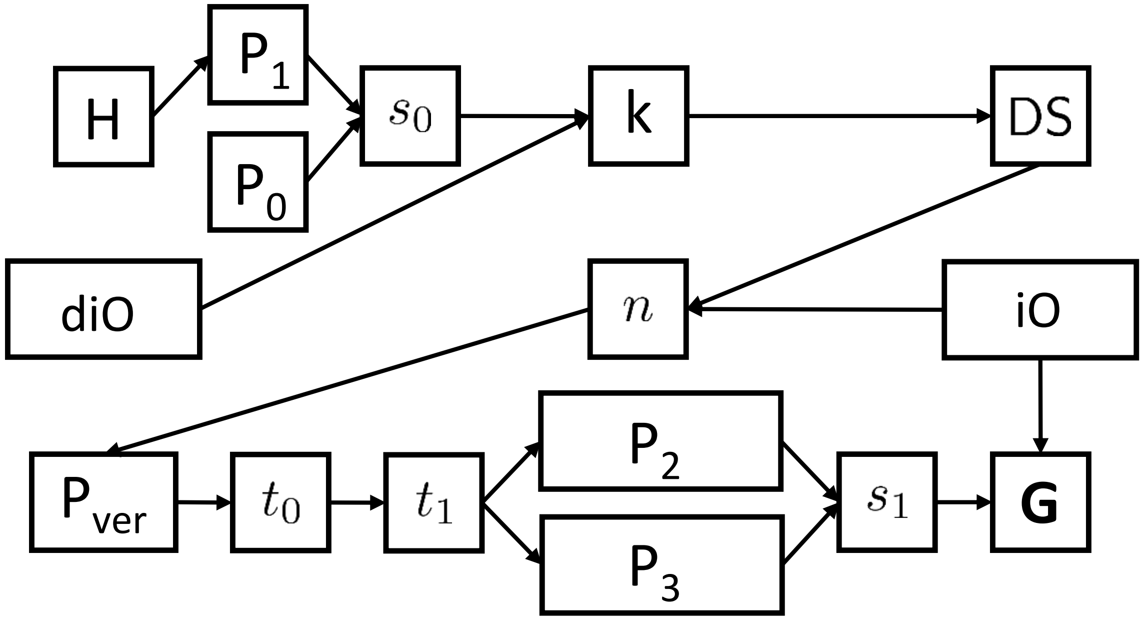
Revert back to the original **aux**.

Hybrid game 1.

x = (m, σ) is a valid message-signature pair, and |**m**| = **k**, and **m** ≥ “00…01”.

aux = iO(M_2)

Parameter Dependencies



A lot of technical details omitted in this talk.

Hard to avoid circular dependencies.

Limitations of our results:

1. TMs with poly-bounded inputs.

I want to obfuscate TMs that take inputs of length ≤ a fixed poly.

Our results do not apply if max input length of TMs is apriori bounded by some polynomial.

2. «Short» auxiliary inputs.

[BST14]

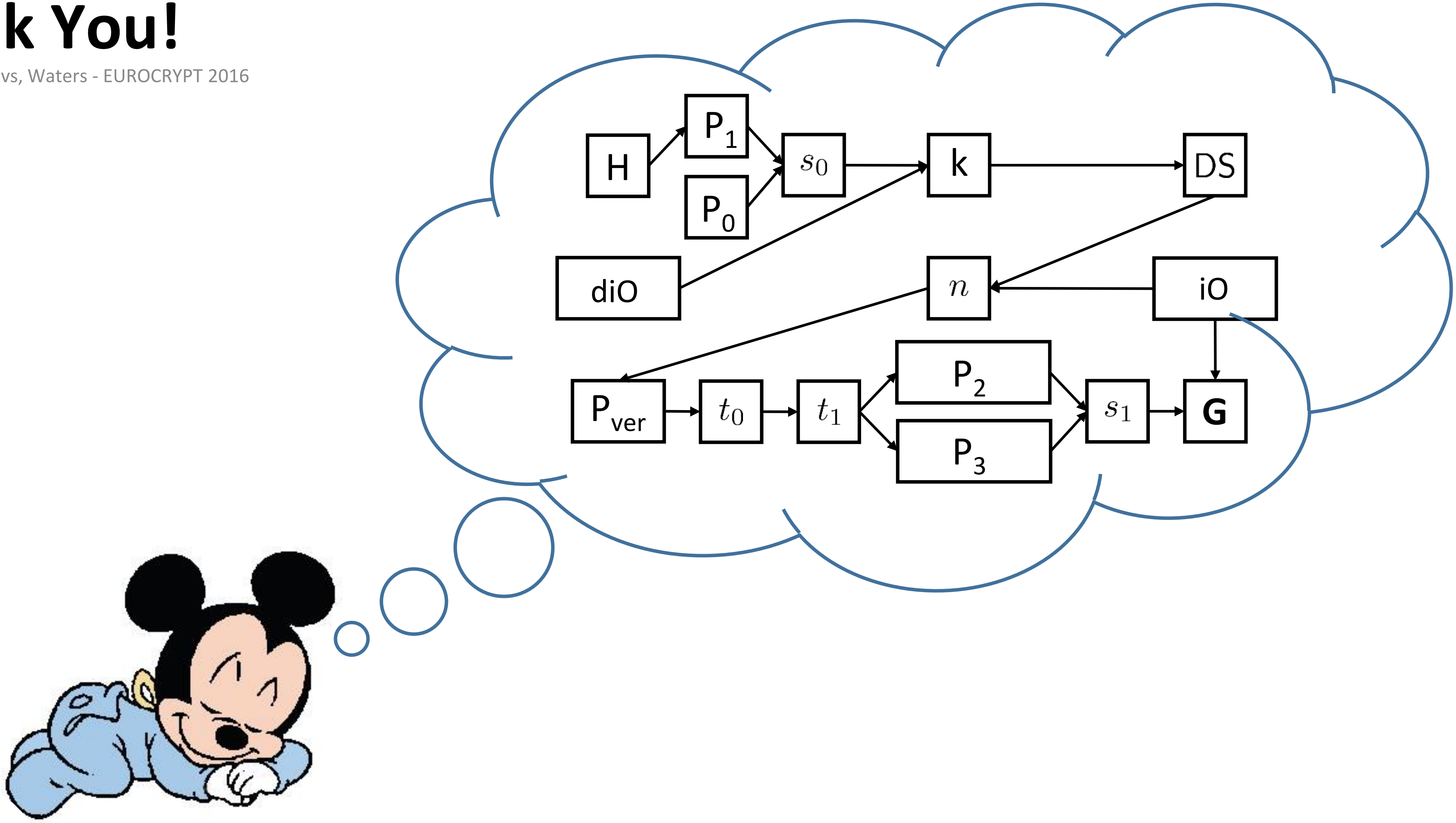
Require |aux| < $|P_0|$ and |aux| < $|P_1|$ to avoid negative results.

[GGHW14] found a workaround by assuming special-purpose obfuscation **for TMs.**

Our attacks do not apply in this case.

Thank You!

Bellare, Stepanovs, Waters - EUROCRYPT 2016



H

P$_1$

P$_0$

s_0

k

DS

diO

n

iO

P$_{\text{ver}}$

t_0

t_1

P$_2$

P$_3$

s_1

G