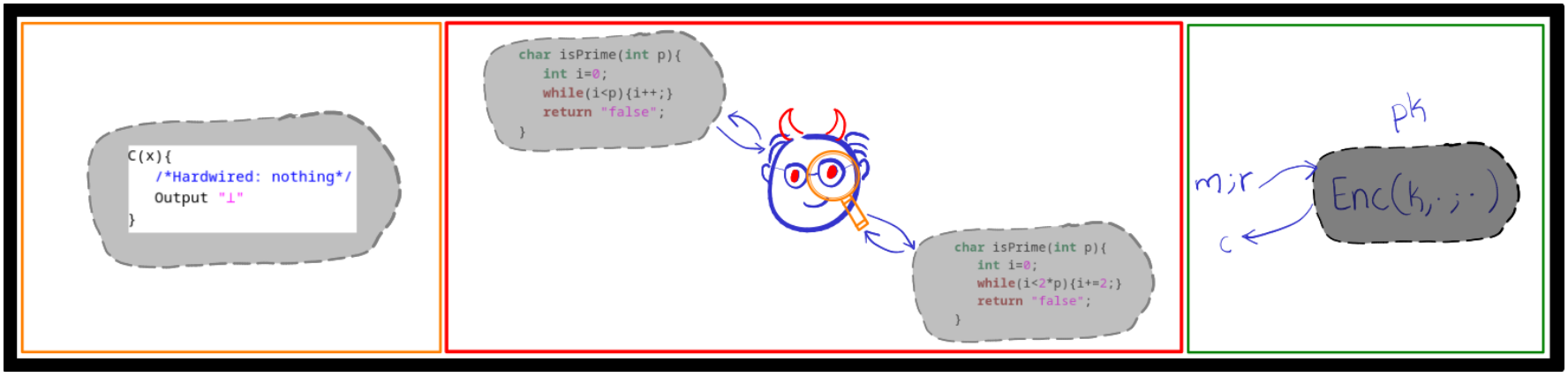


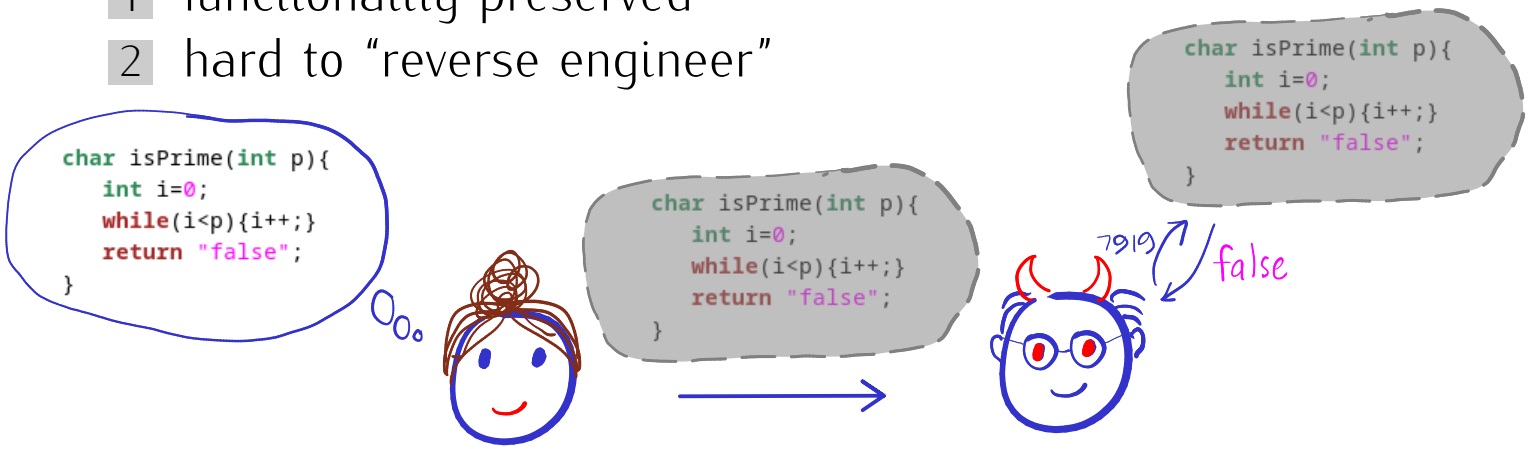
CS783: Theoretical Foundations of Cryptography

Lecture 23 (05/Nov/24)

Instructor: Chethan Kamath

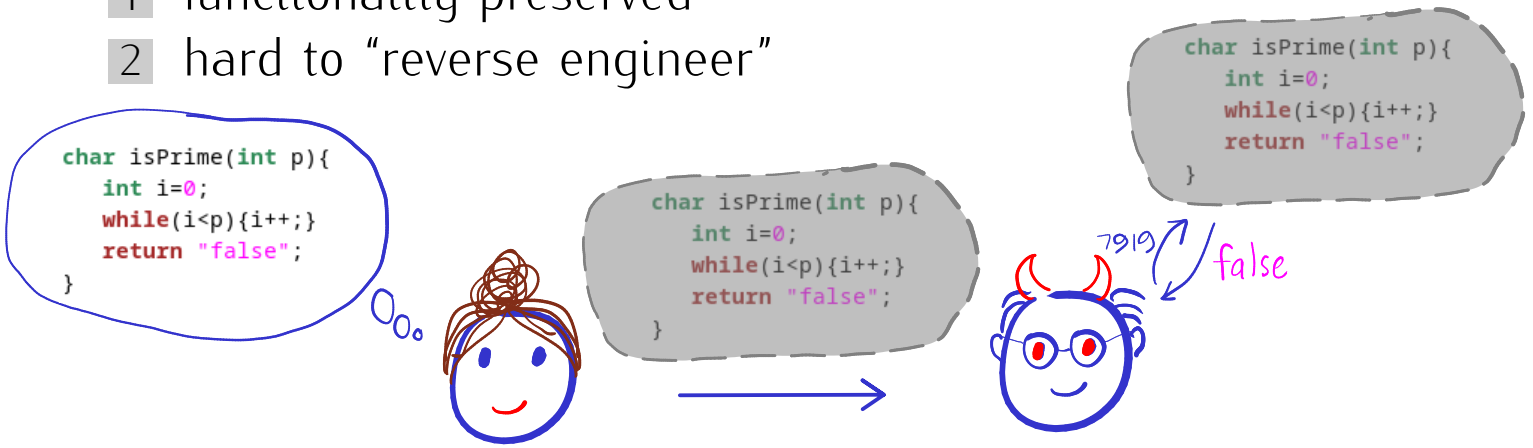
Recall from Last Lecture...

- Program obfuscation: "scramble/encrypt" a program such that
  1. functionality preserved
  2. hard to "reverse engineer"



Recall from Last Lecture...

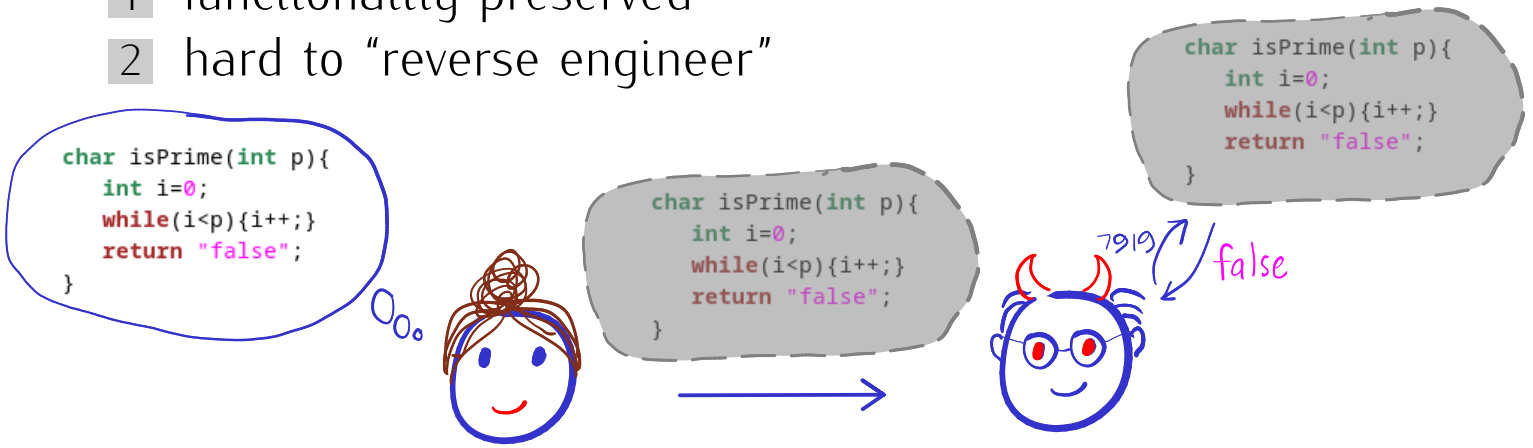
- Program obfuscation: "scramble/encrypt" a program such that
  1. functionality preserved
  2. hard to "reverse engineer"



- How to formalise "hard to reverse engineer"?
  - Virtual black-box obfuscation (VBBO): anything learnable "white-box" given P is learnable 'black-box" given only oracle access to P

Recall from Last Lecture...

- Program obfuscation: "scramble/encrypt" a program such that
  1. functionality preserved
  2. hard to "reverse engineer"

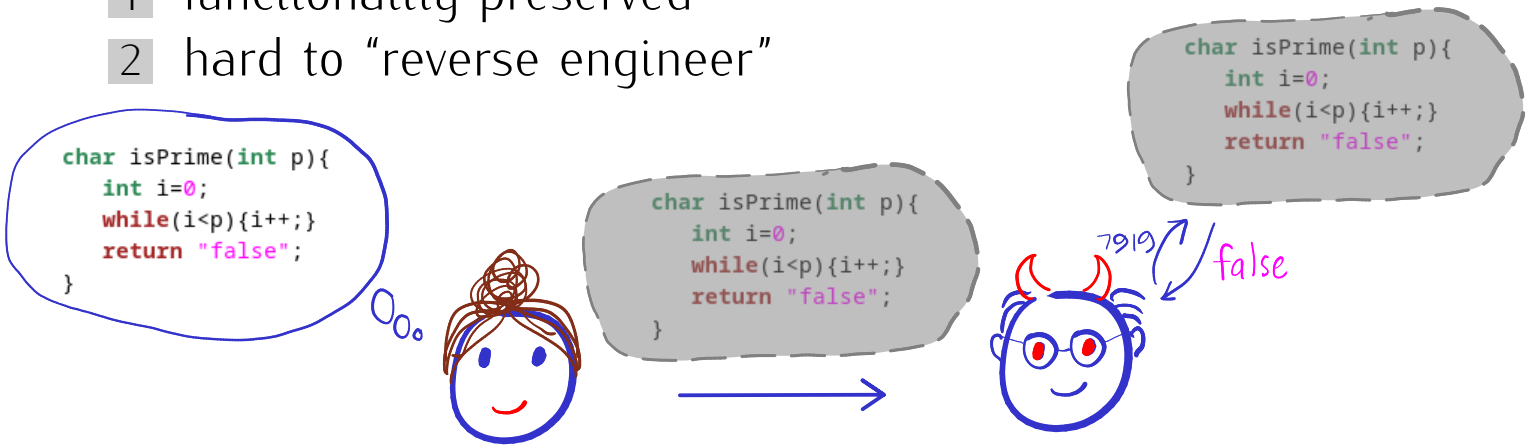


- How to formalise "hard to reverse engineer"?
  - Virtual black-box obfuscation (VBBO): anything learnable "white-box" given P is learnable 'black-box" given only oracle access to P

+ Bypassed black-box separations exploiting primitive's program:
  - OWF $\xrightarrow{\text{VBBO}^*}$ OWP
  - SKE $\xrightarrow{\text{VBBO}^*}$ PKE

Recall from Last Lecture...

- Program obfuscation: "scramble/encrypt" a program such that
  1. functionality preserved
  2. hard to "reverse engineer"



- How to formalise "hard to reverse engineer"?
  - Virtual black-box obfuscation (VBBO): anything learnable "white-box" given P is learnable 'black-box" given only oracle access to P

\+ Bypassed black-box separations exploiting primitive's program:

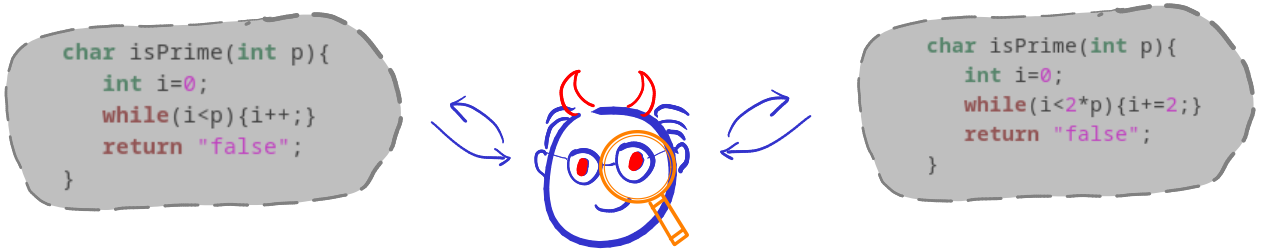
- OWF $\xrightarrow{\text{VBBO}^*}$ OWP
- SKE $\xrightarrow{\text{VBBO}^*}$ PKE

− Impossibility of VBBO for general programs

- Key idea: programs that spit out secret when run "on itself"

Plan for Today's Lecture...

- What do we do in the face of this impossibility?
  - Relax requirement to *indistinguishability* obfuscation (IO)



Plan for Today's Lecture...

- What do we do in the face of this impossibility?
  - Relax requirement to *indistinguishability* obfuscation (IO)

- How to use IO: PRG $\xrightarrow{\text{IO}}$ PKE

Plan for Today's Lecture...

- What do we do in the face of this impossibility?
  - Relax requirement to *indistinguishability* obfuscation (IO)

- How to use IO: PRG $\xrightarrow{\text{IO}}$ PKE

- How to use IO: *punctured* programming
  - Puncturable PRF (PPRF)
  - PPRF $\xrightarrow{\text{IO}}$ PKE
  - PPRF $\xrightarrow{\text{IO}}$ digital signature

Plan for Today's Lecture...

General *template*:

1. Identify the task — program obfuscation
2. Come up with precise threat model M (a.k.a security model)
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure?
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

Plan for Today's Lecture...

General *template*:

1. Identify the task — program obfuscation
2. Come up with precise threat model M (a.k.a security model) — white-box distinguisher
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure? → IO security
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

Plan for Today's Lecture...

General *template*:

1. Identify the task — program obfuscation
2. Come up with precise threat model M (a.k.a security model) — white-box distinguisher
   - Adversary/Attack: What are the adversary's capabilities?
   - Security Goal: What does it mean to be secure? → IO security
3. Construct a scheme Π
4. Formally prove that Π in secure in model M

Next lecture

Plan for Today's Lecture...

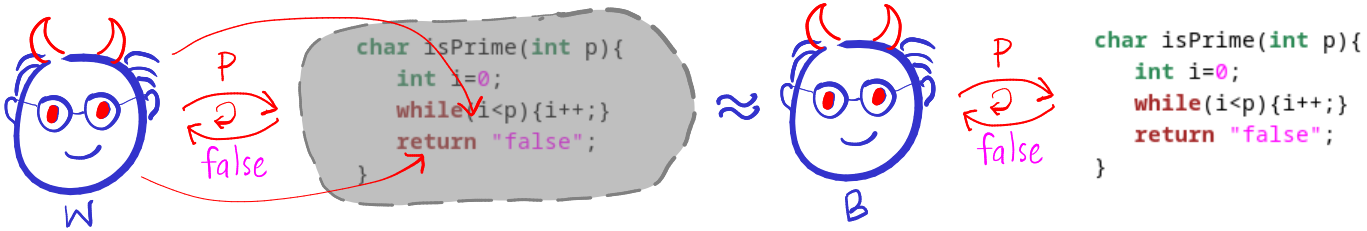
1. Indistinguishability Obfuscation (IO)
2. How to Use IO: PRG $\xrightarrow{\text{IO}}$ PKE
3. How to Use IO: Punctured Programming

Plan for Today's Lecture...

1. Indistinguishability Obfuscation (IO)
2. How to Use IO: PRG $\xrightarrow{\text{IO}}$ PKE
3. How to Use IO: Punctured Programming

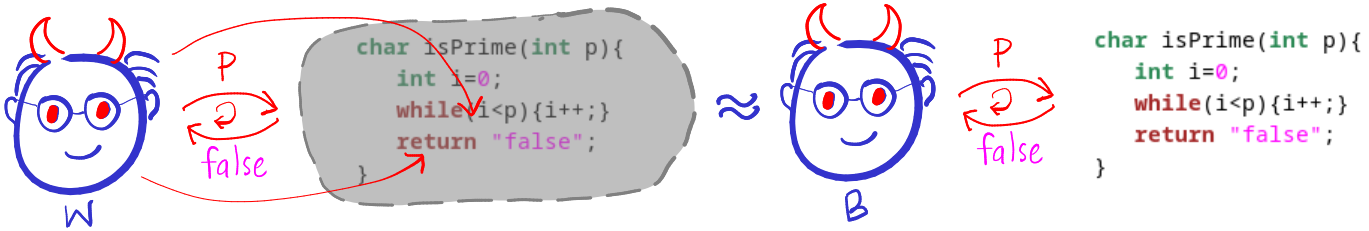
Recall VBBO

- Security via "simulation": anything learnable "white-box" given P is learnable "black-box" given only oracle access to P



Recall VBBO

- Security via "simulation": anything learnable "white-box" given $\tilde{\mathsf{P}}$ is learnable "black-box" given only oracle access to P



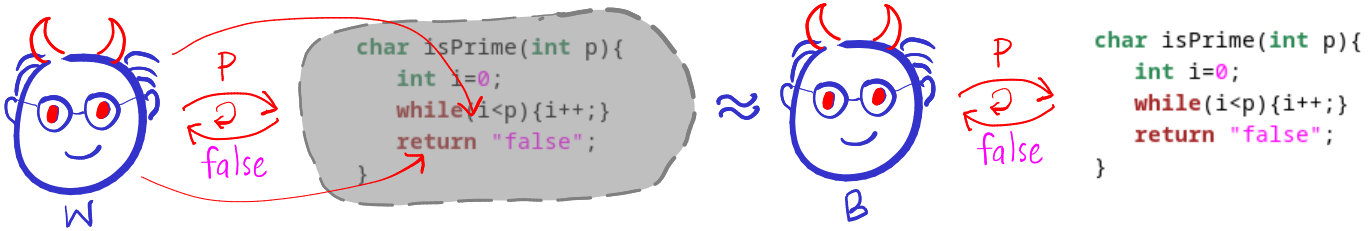
Defintion 1 (VBB obfuscator)

A PPT algorithm Obf *that takes as input any program* P *and a security parameter* n*, and outputs obfuscated program* $\tilde{\mathsf{P}}$ *such that:*

1. *Functionality preserved: for all inputs* x, $\tilde{\mathsf{P}}(x) = \mathsf{P}(x)$
2. *Small slowdown: run-time of* $\tilde{\mathsf{P}}$ *is poly. in* n *and run-time of* P

Recall VBBO

- Security via "simulation": anything learnable "white-box" given $\tilde{P}$ is learnable "black-box" given only oracle access to P



Defintion 1 (VBB obfuscator)

A PPT algorithm Obf *that takes as input any program* P *and a security parameter* n*, and outputs obfuscated program* $\tilde{P}$ *such that:*

1. *Functionality preserved: for all inputs* x, $\tilde{P}(x) = P(x)$
2. *Small slowdown: run-time of* $\tilde{P}$ *is poly. in* n *and run-time of* P
3. *VBBO security: for every PPT* W*, there exists PPT* B *that can simulate* W*'s output on input* $\tilde{P}$ *using only oracle access to* P*. That is, the following is negligible:*

$$\left| \Pr[W(\tilde{P}) = 1] - \Pr[B^{P}(1^n, 1^{|P|}) = 1] \right|$$

$$\tilde{P} \leftarrow \mathrm{Obf}(1^n, P)$$

Let's Define Indistinguishability Obfuscation (IO)...

How to define IO?

Let's Define Indistinguishability Obfuscation (IO)...

$\forall x : P_1(x) = P_2(x)$

How to define IO? Obfuscations of two *functionally-equivalent*, *same-sized* programs are computationally indistinguishable

runtime

```
char isPrime(int p){
   int i=0;
   while(i<p){i++;}
   return "false";
}
```

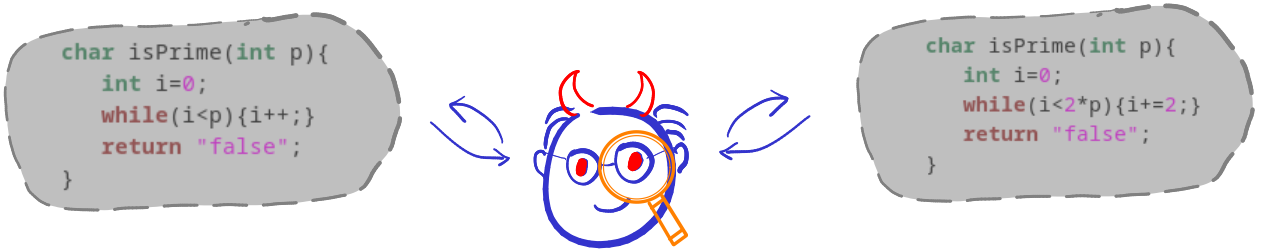
```
char isPrime(int p){
   int i=0;
   while(i<2*p){i+=2;}
   return "false";
}
```

Let's Define Indistinguishability Obfuscation (IO)...

$\forall x : P_1(x) = P_2(x)$

How to define IO? Obfuscations of two *functionally-equivalent, same-sized* programs are computationally indistinguishable

runtime



Defintion 2 (Indistinguishability obfuscator (IO))

A PPT algorithm **Obf** *that takes as input any program* P *and security parameter* **n**, *and outputs obfuscated program* P *such that:*

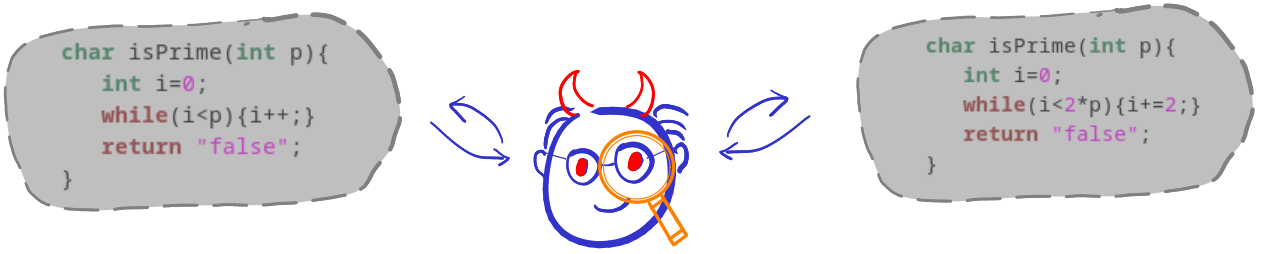
1. *Functionality preserved*
2. *Slowdown is polynomial*

Let's Define Indistinguishability Obfuscation (IO)...

$\forall x : P_1(x) = P_2(x)$

❓ How to define IO? Obfuscations of two *functionally-equivalent, same-sized* programs are computationally indistinguishable

runtime



Defintion 2 (Indistinguishability obfuscator (IO))

A PPT algorithm **Obf** *that takes as input any program* P *and security parameter* **n**, *and outputs obfuscated program* $\tilde{P}$ *such that:*

1. *Functionality preserved*
2. *Slowdown is polynomial*
3. *IO security: for every functionally-equivalent, same-sized* P_1 *and* P_2 *and PPT distinguisher* D, *the following is negligible:*

$$\left| \Pr\left[D\left(\tilde{P}_1\right)=1\right] - \Pr\left[D\left(\tilde{P}_2\right)=1\right] \right|$$

$\tilde{P}_1 \leftarrow Obf(1^n, P_1)$ $\quad$ $\tilde{P}_2 \leftarrow Obf(1^n, P_2)$

Let's Define Indistinguishability Obfuscation (IO)...

Assuming $\mathsf{P} = \mathsf{NP}$, can you construct an IO?

Let's Define Indistinguishability Obfuscation (IO)...

- Assuming $\mathsf{P} = \mathsf{NP}$, can you construct an IO?
  - Output the *lexicographically-first* functionally-equivalent program

Let's Define Indistinguishability Obfuscation (IO)...

Assuming $\mathsf{P} = \mathsf{NP}$, can you construct an IO?

- Output the *lexicographically-first* functionally-equivalent program

Thus hard to show that IO$\rightarrow$OWF (unlike VBBO)

- Will imply $\mathsf{P} \neq \mathsf{NP}$

Let's Define Indistinguishability Obfuscation (IO)...

Assuming $\mathsf{P} = \mathsf{NP}$, can you construct an IO?

- Output the *lexicographically-first* functionally-equivalent program

Thus hard to show that IO$\rightarrow$OWF (unlike VBBO)

- Will imply $\mathsf{P} \neq \mathsf{NP}$

Exercise 1 (VBBO vs. IO)

1. *Show that VBBO $\rightarrow$ IO*
2. *Figure out why Theorem 1 from Lecture 22 fails for IO*

Let's Define Indistinguishability Obfuscation (IO)...

Assuming $\mathsf{P} = \mathsf{NP}$, can you construct an IO?

- Output the *lexicographically-first* functionally-equivalent program

Thus hard to show that IO$\rightarrow$OWF (unlike VBBO)

- Will imply $\mathsf{P} \neq \mathsf{NP}$

Exercise 1 (VBBO vs. IO)

1. *Show that VBBO $\rightarrow$ IO*
2. *Figure out why Theorem 1 from Lecture 22 fails for IO*

Exercise 2 (IO is the "best-possible obfuscation")

If VBBO is possible for a program class $\mathcal{C}$, then an IO Obf for $\mathcal{C}$ is also a VBBO

Plan for Today's Lecture

1. Indistinguishability Obfuscation (IO)
2. How to Use IO: PRG $\xrightarrow{\text{IO}}$ PKE
3. How to Use IO: Punctured Programming

How to Use IO: PRG $\mathsf{G} \xrightarrow{\text{IO}}$ PKE...

How to construct PKE using PRG $\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n}$?

How to Use IO: PRG $\mathsf{G} \xrightarrow{\text{IO}}$ PKE...

How to construct PKE using PRG $\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n}$?

- Secret key is $sk \leftarrow \{0,1\}^n$ and $pk := \mathsf{G}(sk)$
- Ciphertext is obfuscation of function C below

```
C(x){
   /*Hardwired: pk, m */
   if(G(x)=pk) output m
   else output "⊥"
}
```

How to Use IO: PRG $\mathsf{G} \xrightarrow{\mathsf{IO}}$ PKE...

How to construct PKE using PRG $\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n}$?

- Secret key is $sk \leftarrow \{0,1\}^n$ and $pk := \mathsf{G}(sk)$
- Ciphertext is obfuscation of function C below

```
C(x){
   /*Hardwired: pk, m */
   if(G(x)=pk) output m
   else output "⊥"
}
```

Construction 1 ($\mathsf{G} : \{0,1\}^n \to \{0,1\}^{2n} \to \Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$)

- $\mathsf{Gen}(1^n)$:
  - *Sample* $sk \leftarrow \{0,1\}^n$
  - *Output* $pk := \mathsf{G}(sk)$ *as public key and* sk *as secret key*
- $\mathsf{Enc}(pk, m)$: *output* $\mathsf{C} \leftarrow \mathsf{Obf}(\mathsf{C})$
- $\mathsf{Dec}(sk, \mathsf{C})$: *output* $\mathsf{C}(sk)$

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

How to construct PKE using PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$?

- Secret key is $sk \leftarrow \{0,1\}^n$ and $pk := G(sk)$
- Ciphertext is obfuscation of function C below

```
C(x){
   /*Hardwired: pk, m */
   if(G(x)=pk) output m
   else output "⊥"
}
```

"secret"

Construction 1 ($G : \{0,1\}^n \to \{0,1\}^{2n} \to \Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$)

- $\mathsf{Gen}(1^n)$:
  - *Sample* $sk \leftarrow \{0,1\}^n$
  - *Output* $pk := G(sk)$ *as public key and* sk *as secret key*
- $\mathsf{Enc}(pk, m)$: *output* $C \leftarrow \mathsf{Obf}(C)$
- $\mathsf{Dec}(sk, C)$: *output* $C(sk)$

Exercise 3

Prove that Construction 1 is secure if Obf *is VBBO*

How to Use IO: PRG $\mathsf{G} \xrightarrow{\text{IO}}$ PKE...

Theorem 1

If Obf *is an IO and* G *is a PRG then* Π *is a PKE*

How to Use IO: PRG $\mathsf{G} \xrightarrow{\text{IO}}$ PKE...

Theorem 1

If Obf *is an IO and* G *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).

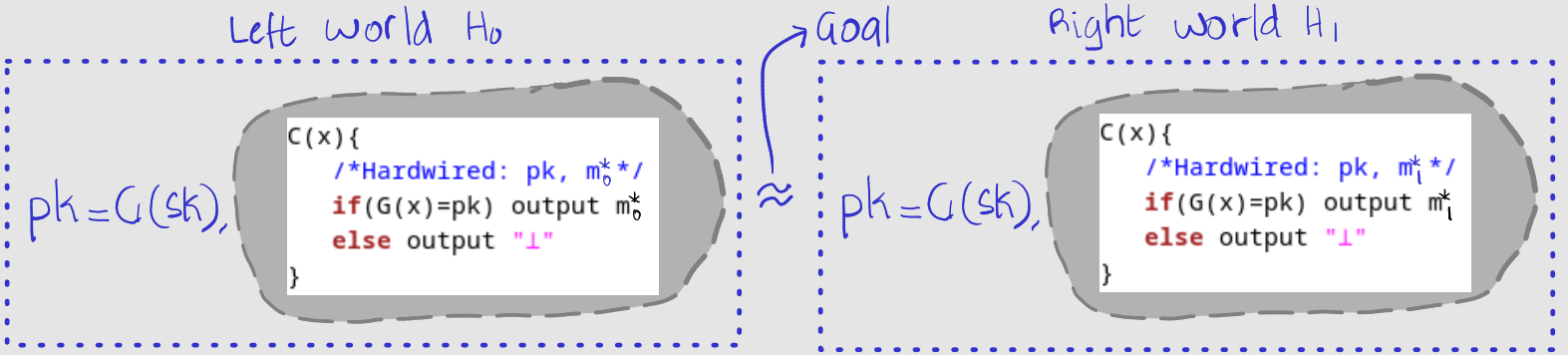
$\square$

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

Theorem 1

If Obf *is an IO and* G *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).



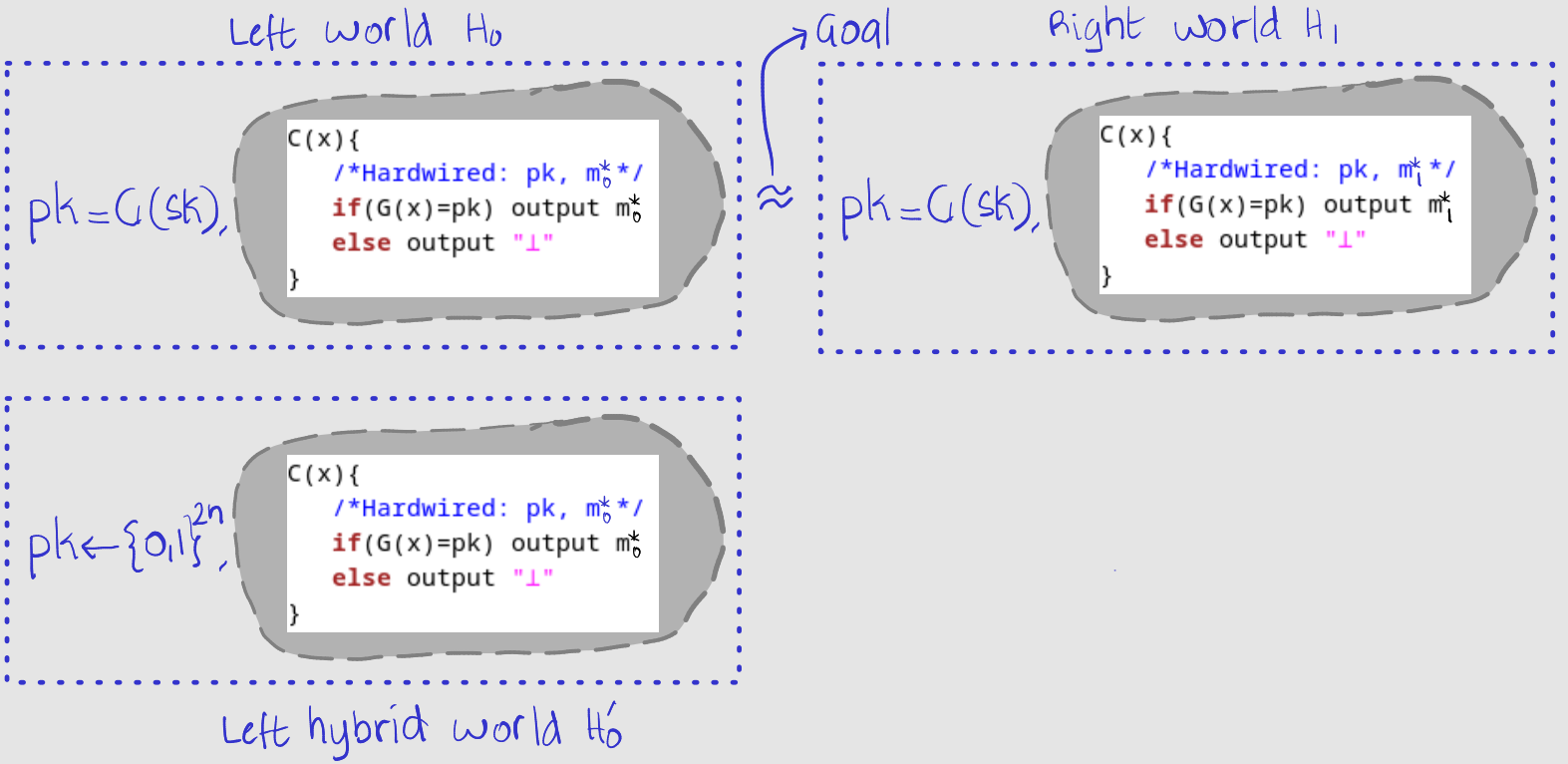
□

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

Theorem 1

If Obf *is an IO and* G *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).



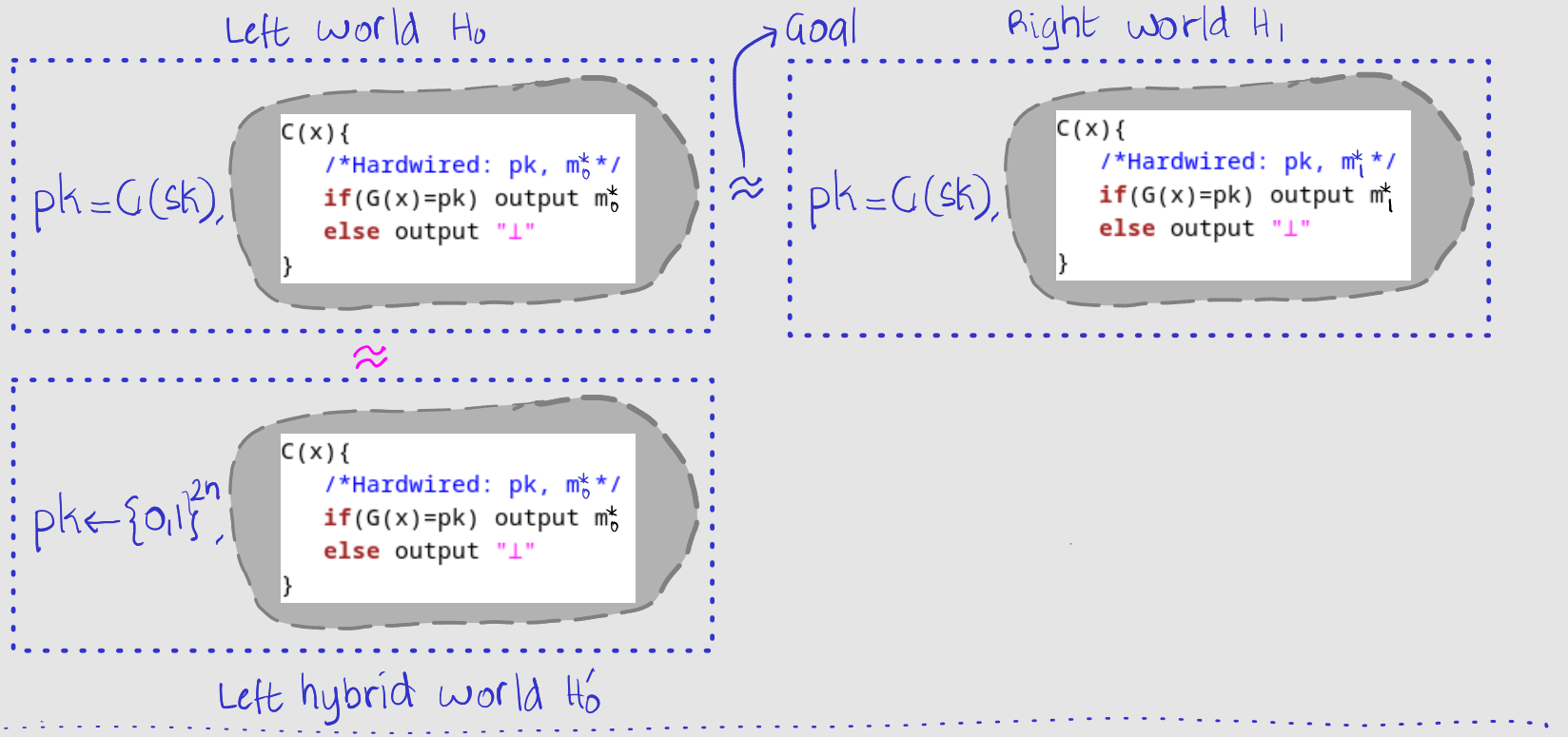
□

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

Theorem 1

If **Obf** *is an IO and* **G** *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).



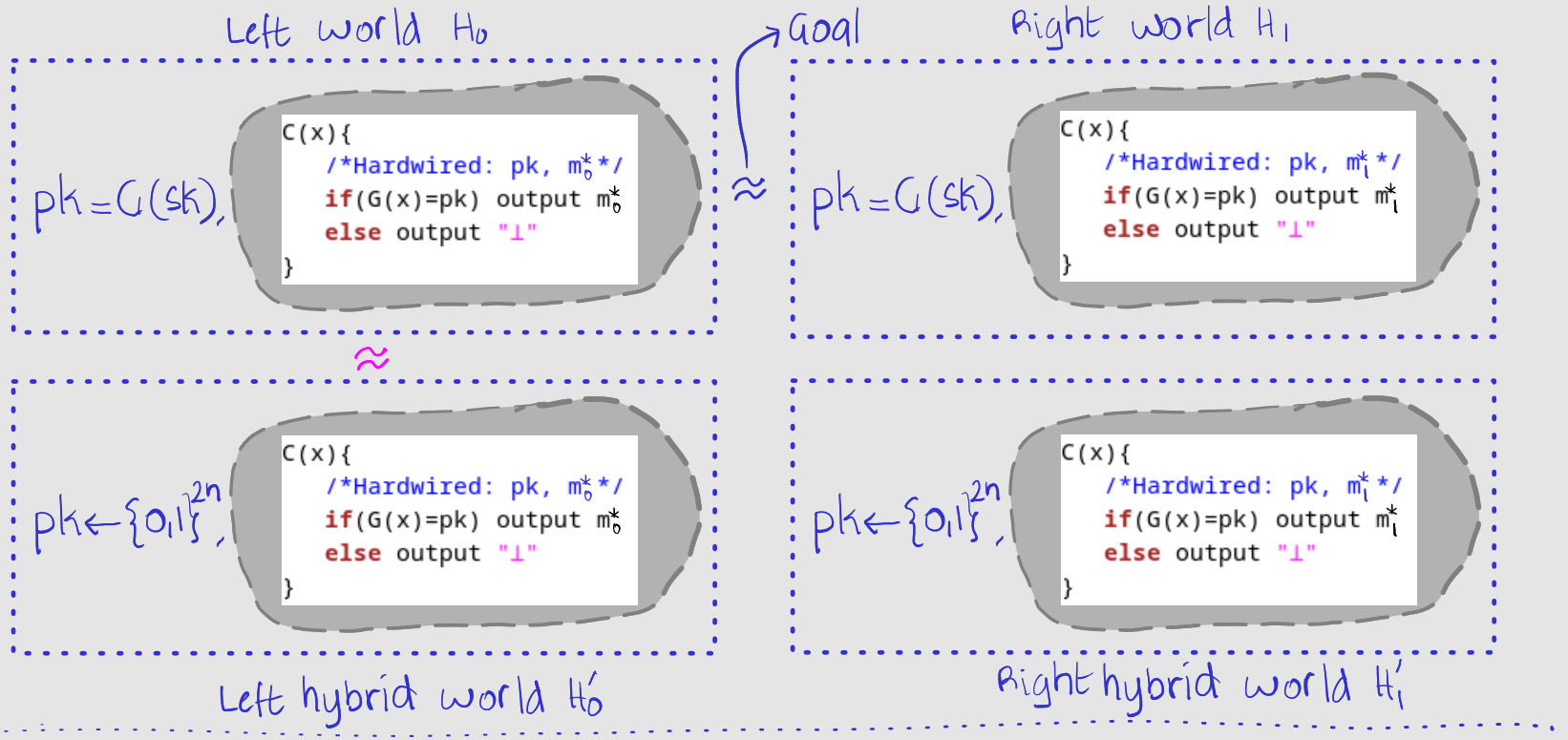
□

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

Theorem 1

If **Obf** *is an IO and* **G** *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).



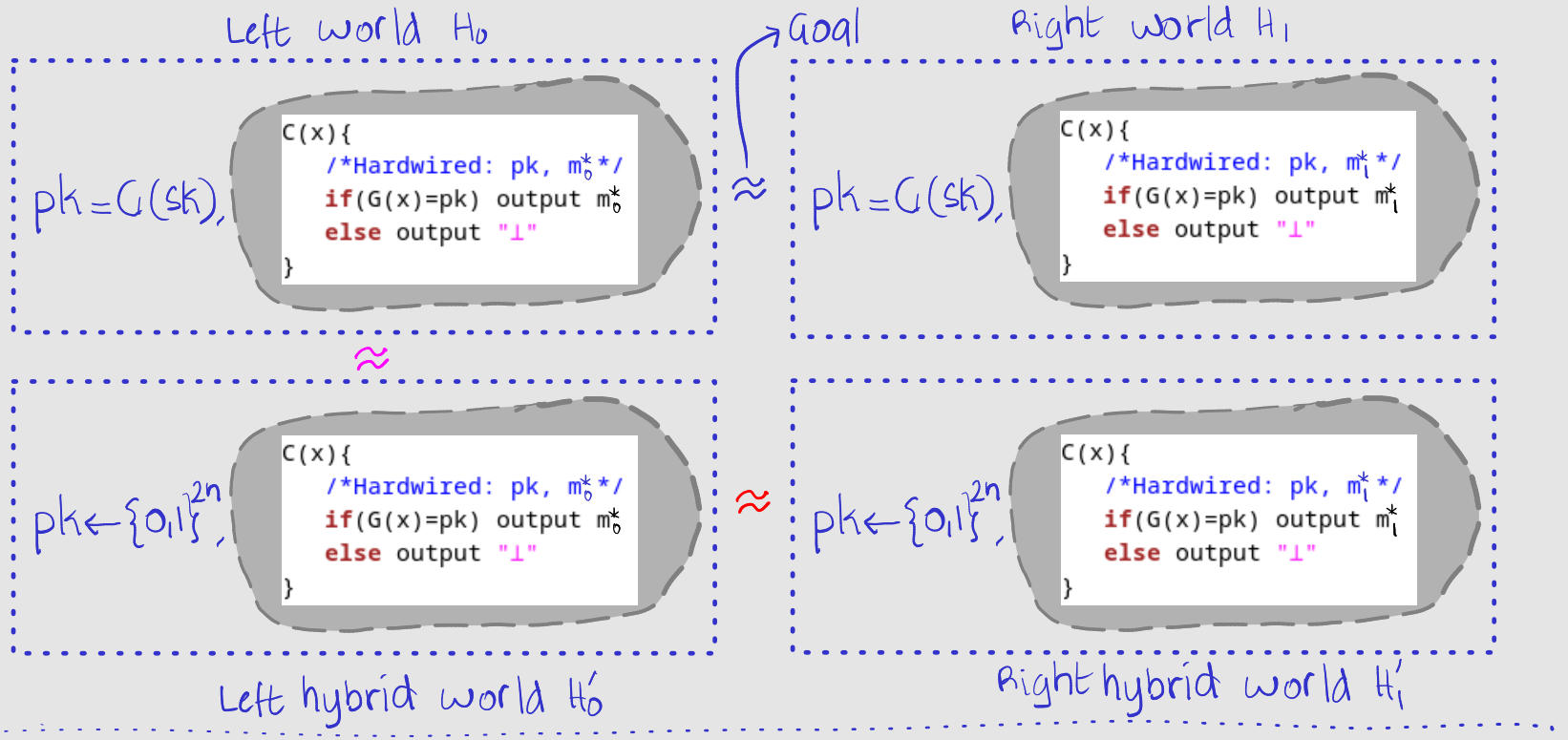
□

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

Theorem 1

If **Obf** *is an IO and* **G** *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).



$\approx$ Claim 1 : $H_0 \approx H_0'$ by pseudorandomness of G

$\approx$ Claim 2 : When $pk \notin \text{Image}(G)$, $H_0' \approx H_1'$ by IO security.
↳ w/ prob. $1 - 1/2^n$

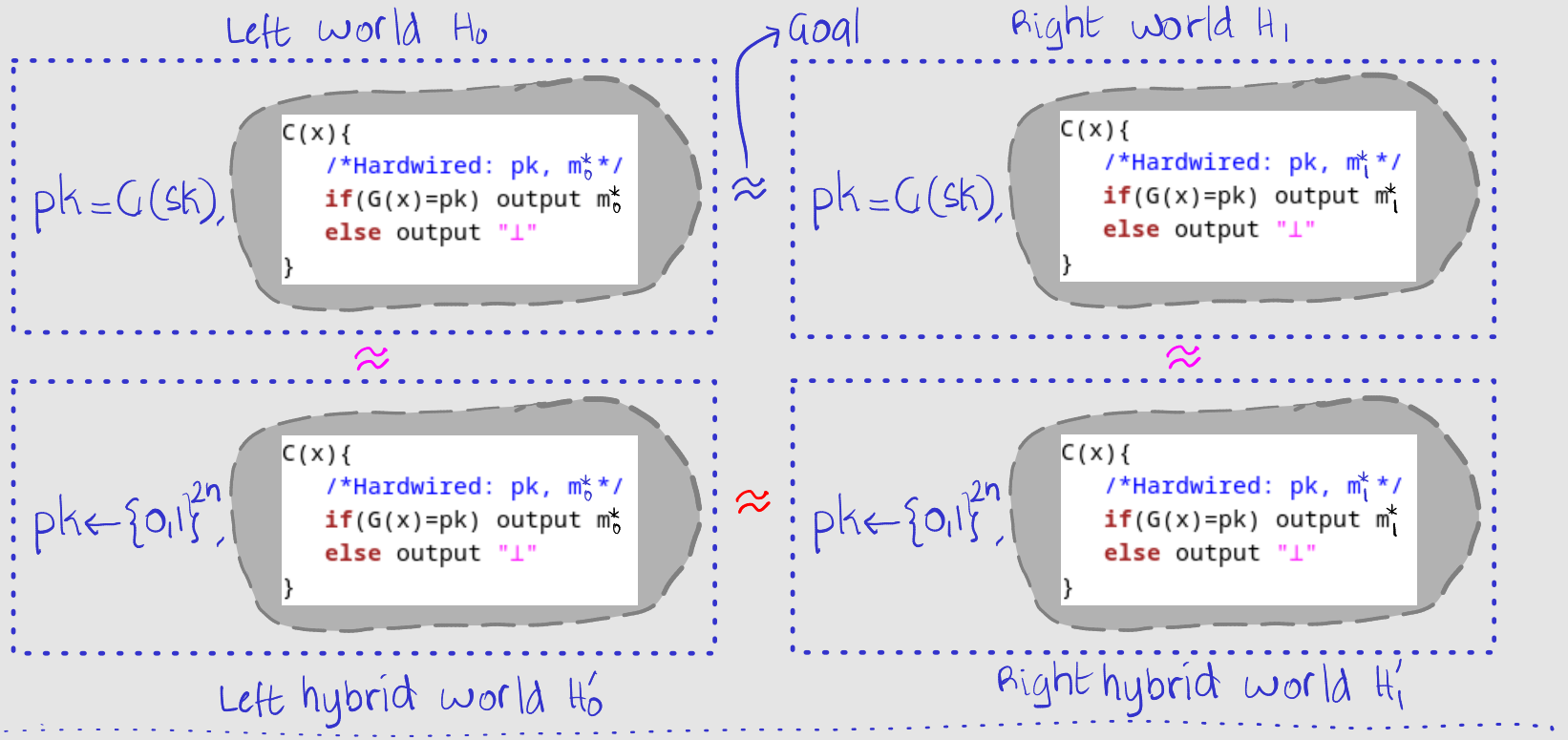
□

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...

Theorem 1

If **Obf** *is an IO and* **G** *is a PRG then* Π *is a PKE*

Proof Sketch (Hybrid argument).

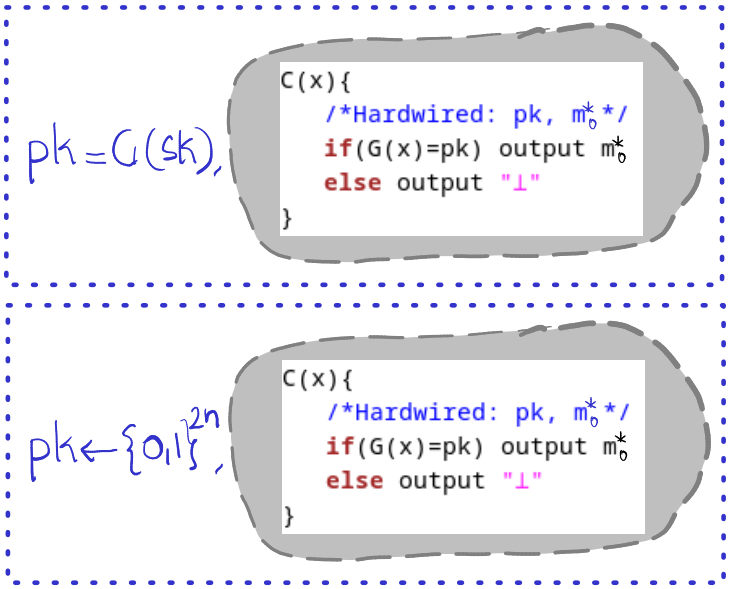


$\approx$ Claim 1 : $H_0 \approx H_0'$ & $H_1 \approx H_1'$ by pseudorandomness of G

$\approx$ Claim 2 : When $pk \notin \text{Image}(G)$, $H_0' \approx H_1'$ by IO security.
↳ w/ prob. $1 - 1/2^n$

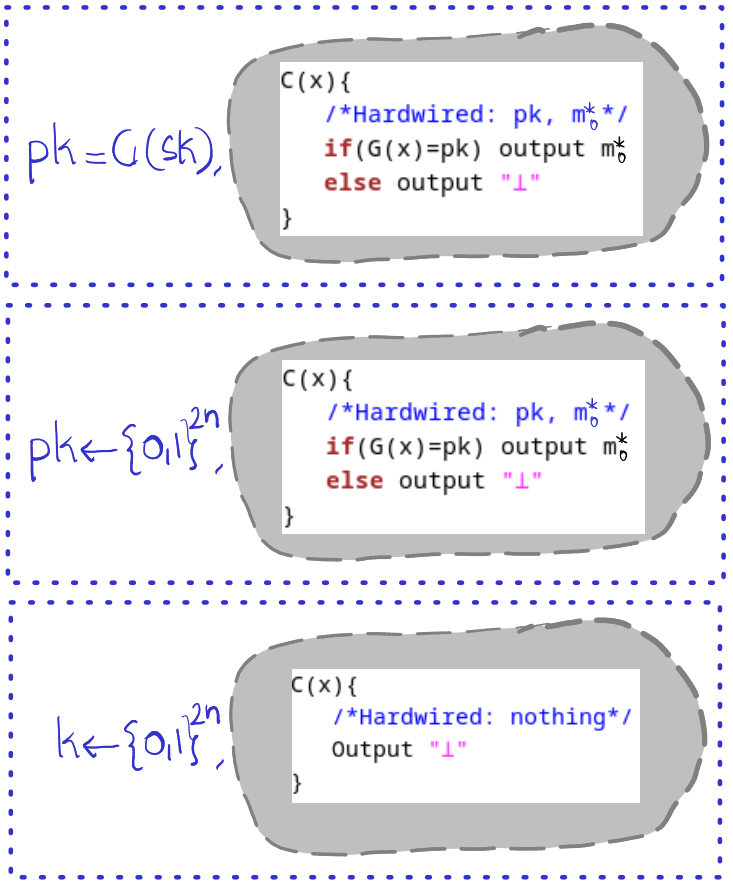
□

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...



- What is going on?
  - IO *can* be used to hide secrets (in this case, messages)

How to Use IO: PRG G $\xrightarrow{\text{IO}}$ PKE...



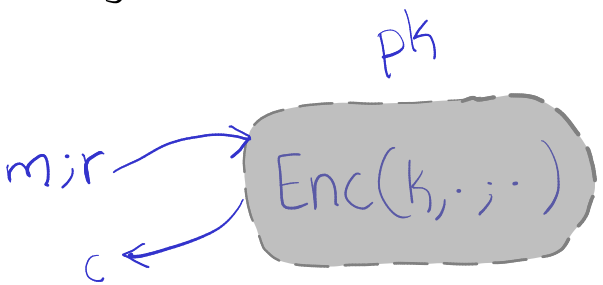
- What is going on?
  - IO *can* be used to hide secrets (in this case, messages)
  - Hiding exploits pseudorandomness of PRG:
    - When y is outside the image of G, message is never used
    - Switch from $pk := G(x)$ to $pk := y$ is indistinguishable

Plan for this Session

1. Indistinguishability Obfuscation (IO)
2. How to Use IO: PRG $\xrightarrow{\text{IO}}$ PKE
3. How to Use IO: Punctured Programming

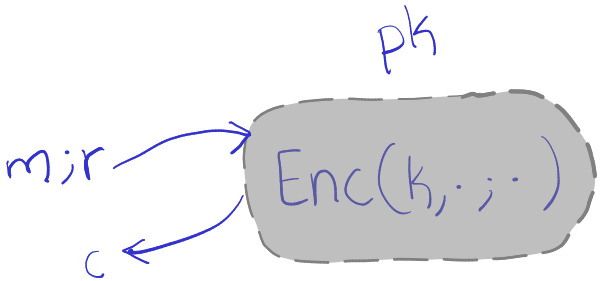
Recall Our Attempt at SKE $\xrightarrow{\text{VBBO}}$ PKE...

- Public key is an obfuscation of SKE's encrypt algorithm **Enc** with secret key ***k*** *hardcoded*



Recall Our Attempt at SKE $\xrightarrow{\text{VBBO}}$ PKE...

- Public key is an obfuscation of SKE's encrypt algorithm **Enc** with secret key k *hardcoded*



Construction 2 ($\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}) \rightarrow \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \mathsf{Gen}(1^n)$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* $\mathbf{pk}$ *and* k *as secret key*
- $\mathsf{Enc}'(pk, m; r)$: *output* $c := pk(m; r)$
- $\mathsf{Dec}'(k, c)$: *output* $m := \mathsf{Dec}(k, c)$

Recall Our Attempt at SKE $\xrightarrow{\text{VBBO}}$ PKE...

- Using SKE based on PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$:

Construction 3 (PRF $\mathsf{F} \to \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, r) \oplus m; r)$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, r) := pk(m; r)$
- $\mathsf{Dec}'(k, (c, r))$: *output* $m := \mathsf{F}(k, r) \oplus c$

Recall Our Attempt at SKE $\xrightarrow{\text{VBBO}}$ PKE...

- Using SKE based on PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$:

Construction 3 (PRF $\mathsf{F} \to \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, r) \oplus m; r)$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, r) := pk(m; r)$
- $\mathsf{Dec}'(k, (c, r))$: *output* $m := \mathsf{F}(k, r) \oplus c$

Why is Construction 3 insecure?

Recall Our Attempt at SKE $\xrightarrow{\text{VBBO}}$ PKE...

- Using SKE based on PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$:

Construction 3 (PRF $\mathsf{F} \to \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, r) \oplus m; r)$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, r) := pk(m; r)$
- $\mathsf{Dec}'(k, (c, r))$: *output* $m := \mathsf{F}(k, r) \oplus c$

Why is Construction 3 insecure?

- Given challenger ciphertext (c^*, r^*), is it possible recover m^*?

Recall Our Attempt at SKE $\xrightarrow{\text{VBBO}}$ PKE...

- Using SKE based on PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$:

Construction 3 (PRF $\mathsf{F} \to \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, r) \oplus m; r)$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, r) := pk(m; r)$
- $\mathsf{Dec}'(k, (c, r))$: *output* $m := \mathsf{F}(k, r) \oplus c$

Why is Construction 3 insecure?

- Given challenger ciphertext (c^*, r^*), is it possible recover m^*?
  1. Run $pk(0^{2n}; r^*)$ to obtain $\mathsf{F}(k, r^*) \oplus 0^{2n} = \mathsf{F}(k, r^*)$
  2. Recover $m^* := \mathsf{F}(k, r^*) \oplus c^*$

Issue: adversary can control random coins used to encrypt

Let's Fix Construction 2

- How to "hide" random coins used to encrypt

Let's Fix Construction 2

How to "hide" random coins used to encrypt using a PRG G?

Let's Fix Construction 2

How to "hide" random coins used to encrypt using a PRG G?

- Use $\mathsf{G}(r^*)$ for $r^* \leftarrow \{0,1\}^n$ instead of $r^* \leftarrow \{0,1\}^{2n}$ as coin

Construction 4 (PRF $\mathsf{F} \rightarrow \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, \mathsf{G}(r)) \oplus m; \mathsf{G}(r))$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, y) := pk(m; \mathsf{G}(r))$
- $\mathsf{Dec}'(k, (c, y))$: *output* $m := \mathsf{F}(k, y) \oplus c$

Let's Fix Construction 2

How to "hide" random coins used to encrypt using a PRG G?

- Use $\mathsf{G}(r^*)$ for $r^* \leftarrow \{0,1\}^n$ instead of $r^* \leftarrow \{0,1\}^{2n}$ as coin

Construction 4 (PRF $\mathsf{F} \rightarrow \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k, \cdot; \cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, \mathsf{G}(r)) \oplus m; \mathsf{G}(r))$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, y) := pk(m; \mathsf{G}(r))$
- $\mathsf{Dec}'(k, (c, y))$: *output* $m := \mathsf{F}(k, y) \oplus c$

Why does the attack not work now?

Let's Fix Construction 2

How to "hide" random coins used to encrypt using a PRG G?

- Use $\mathsf{G}(r^*)$ for $r^* \leftarrow \{0,1\}^n$ instead of $r^* \leftarrow \{0,1\}^{2n}$ as coin

Construction 4 (PRF F $\rightarrow \Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$)

- $\mathsf{Gen}'(1^n)$:
  - *Sample* $k \leftarrow \{0,1\}^n$
  - *Output* $\mathsf{Enc}(k,\cdot;\cdot)$ *as public key* pk *and* k *as secret key, where*

$$\mathsf{Enc}(k, m; r) := (\mathsf{F}(k, \mathsf{G}(r)) \oplus m; \mathsf{G}(r))$$

- $\mathsf{Enc}'(pk, m; r)$: *output* $(c, y) := pk(m; \mathsf{G}(r))$
- $\mathsf{Dec}'(k, (c, y))$: *output* $m := \mathsf{F}(k, y) \oplus c$

Why does the attack not work now? Need to invert PRG

- To formally prove IND-CPA security, we need additional properties from PRF F...

PRF F Needs to be "Puncturable"

Definition 2 (Puncturable PRF (PPRF))

A PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ that additionally supports

- a puncturing algorithm $k_{x^*} \leftarrow \mathsf{Puncture}(k, x^*)$

PRF F Needs to be "Puncturable"

Definition 2 (Puncturable PRF (PPRF))

A PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ that additionally supports

- a puncturing algorithm $k_{x^*} \leftarrow \mathsf{Puncture}(k, x^*)$

such that

1. Function preserved at non-punctured points:

$$\forall x \neq x^* : \mathsf{F}_{k_{x^*}}(x) = \mathsf{F}_k(x)$$

2. Value of $\mathsf{F}_{k_{x^*}}$ at x^* is uniformly random *even given the punctured key* k_{x^*}

PRF F Needs to be "Puncturable"

Definition 2 (Puncturable PRF (PPRF))

A PRF $\mathsf{F} : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ that additionally supports

- a puncturing algorithm $k_{x^*} \leftarrow \mathsf{Puncture}(k, x^*)$

such that

1. Function preserved at non-punctured points:

$$\forall x \neq x^* : \mathsf{F}_{k_{x^*}}(x) = \mathsf{F}_k(x)$$

2. Value of $\mathsf{F}_{k_{x^*}}$ at x^* is uniformly random *even given the punctured key* k_{x^*}

- PPRF can be obtained by modifying tree-based PRF from Lecture 5
  - PRG→PPRF

Construction 4 is IND-CPA PKE

Theorem 3

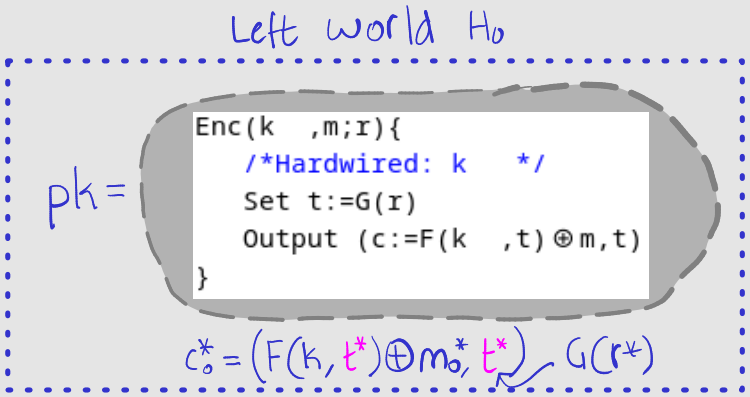
If F *is a puncturable PRF and* Obf *is an IO then* Π' *is a PKE*

Construction 4 is IND-CPA PKE

Theorem 3

If F *is a puncturable PRF and* Obf *is an IO then* Π' *is a PKE*

Proof Sketch (Hybrid argument).



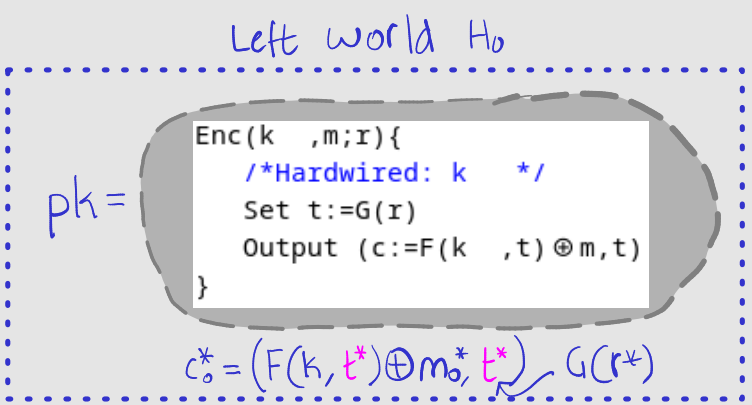
□

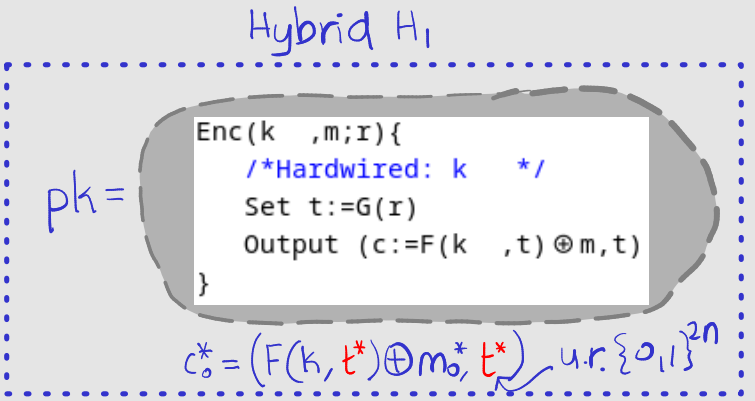
Construction 4 is IND-CPA PKE

Theorem 3

If **F** *is a puncturable PRF and* **Obf** *is an IO then* Π' *is a PKE*

Proof Sketch (Hybrid argument).





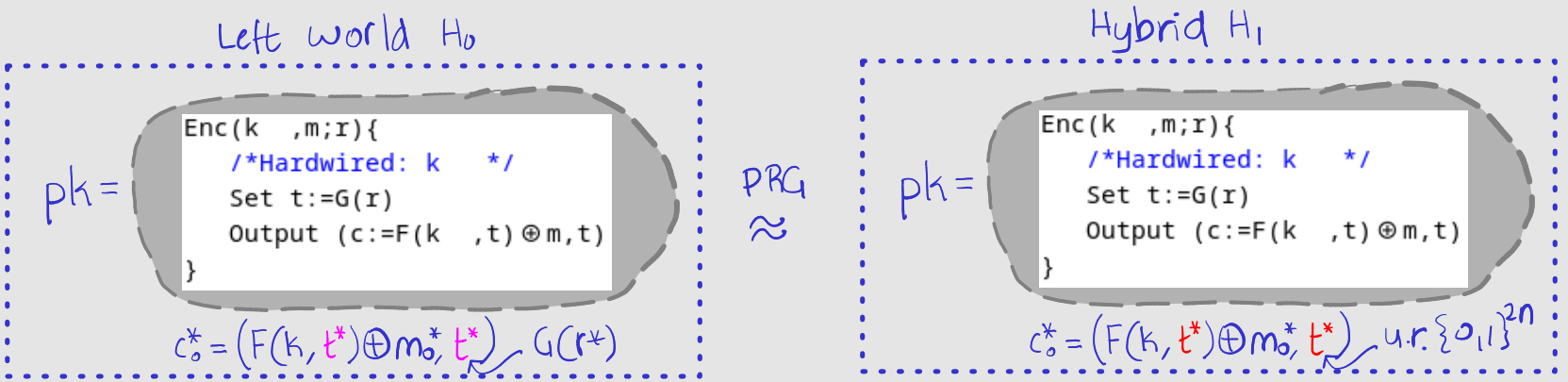
□

Construction 4 is IND-CPA PKE

Theorem 3

If F *is a puncturable PRF and* Obf *is an IO then* Π' *is a PKE*

Proof Sketch (Hybrid argument).



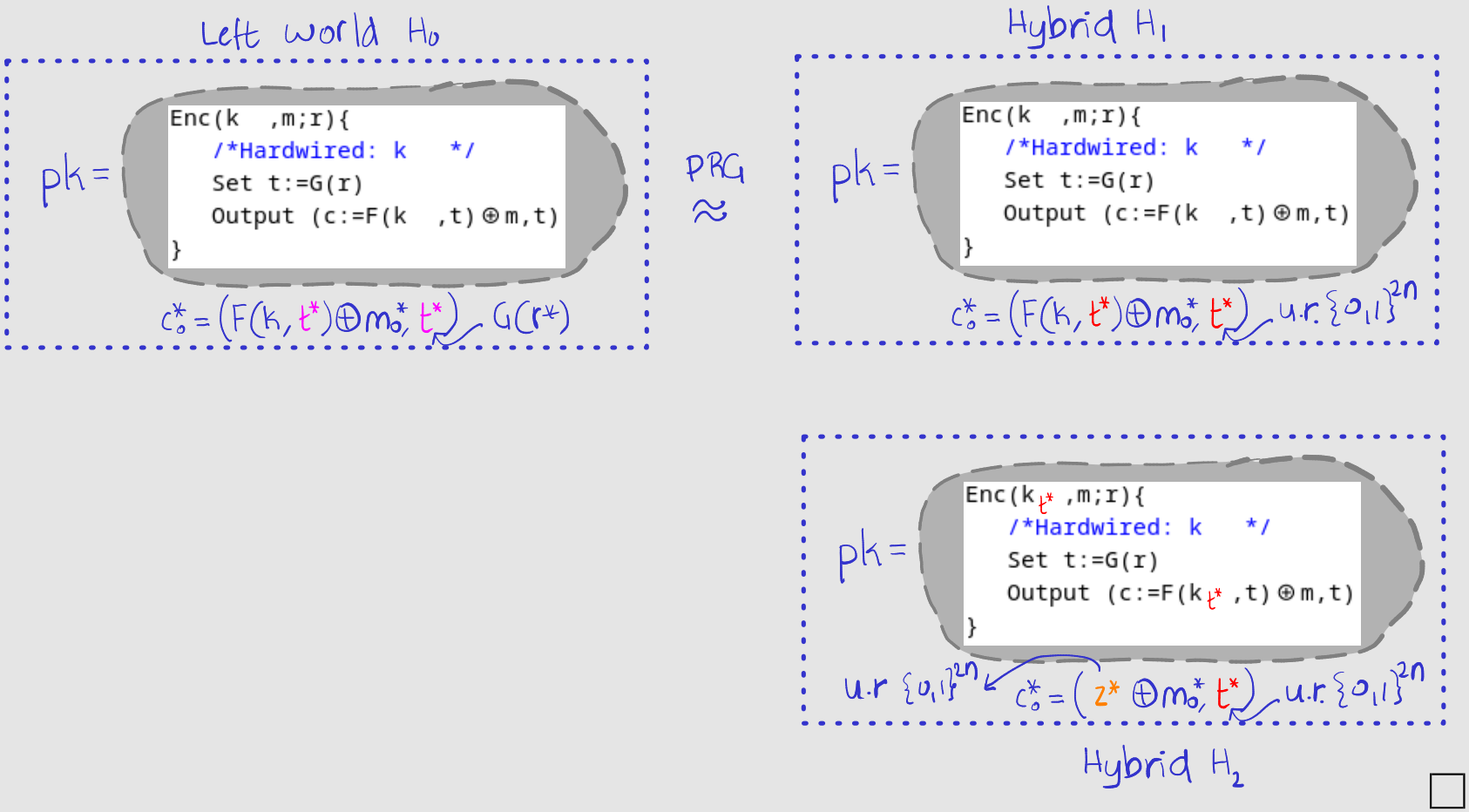
□

Construction 4 is IND-CPA PKE

Theorem 3

If **F** *is a puncturable PRF and* **Obf** *is an IO then* Π' *is a PKE*

Proof Sketch (Hybrid argument).



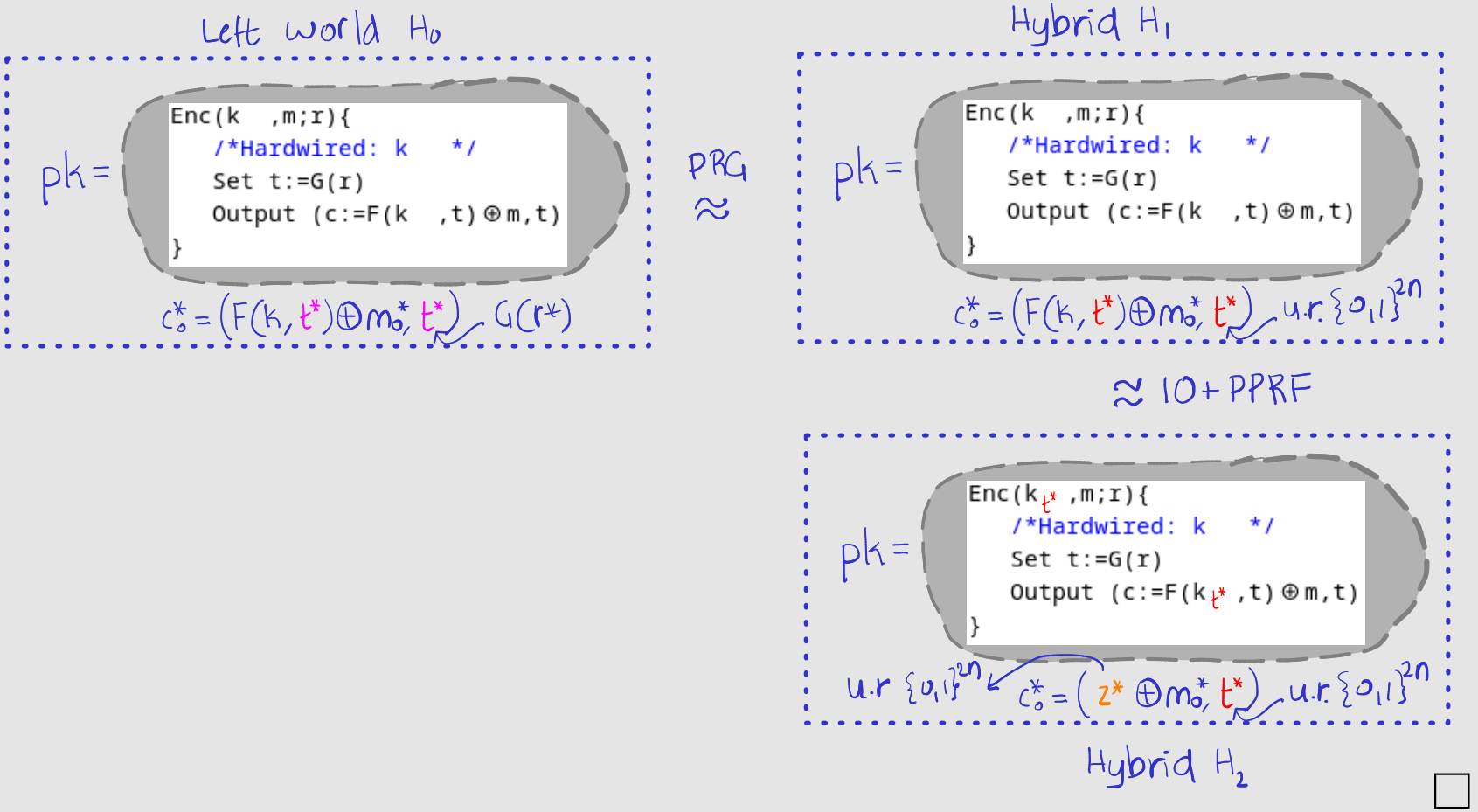
□

Construction 4 is IND-CPA PKE

Theorem 3

If **F** *is a puncturable PRF and* **Obf** *is an IO then* Π' *is a PKE*

Proof Sketch (Hybrid argument).



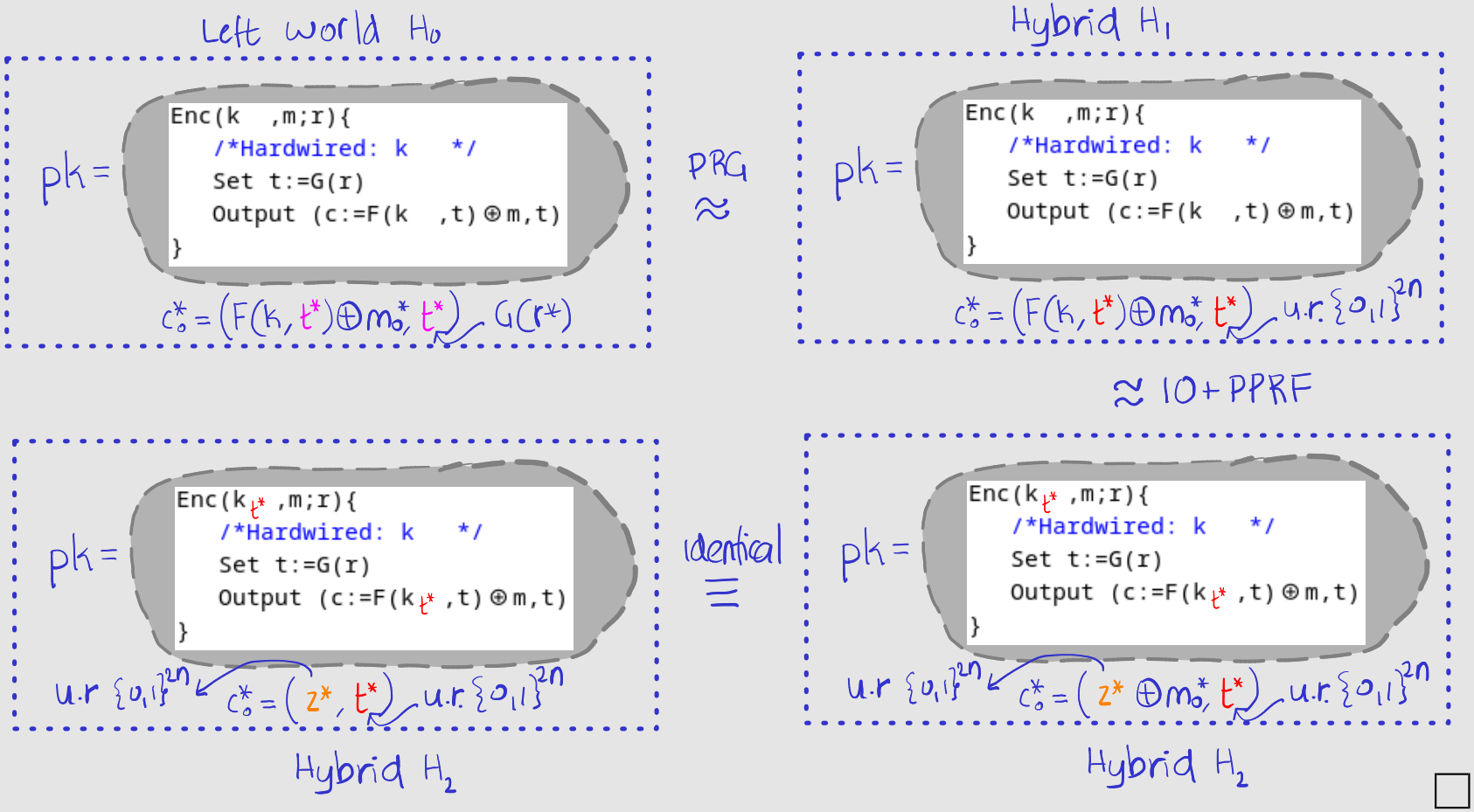
□

Construction 4 is IND-CPA PKE

Theorem 3

If **F** *is a puncturable PRF and* **Obf** *is an IO then* Π' *is a PKE*

Proof Sketch (Hybrid argument).



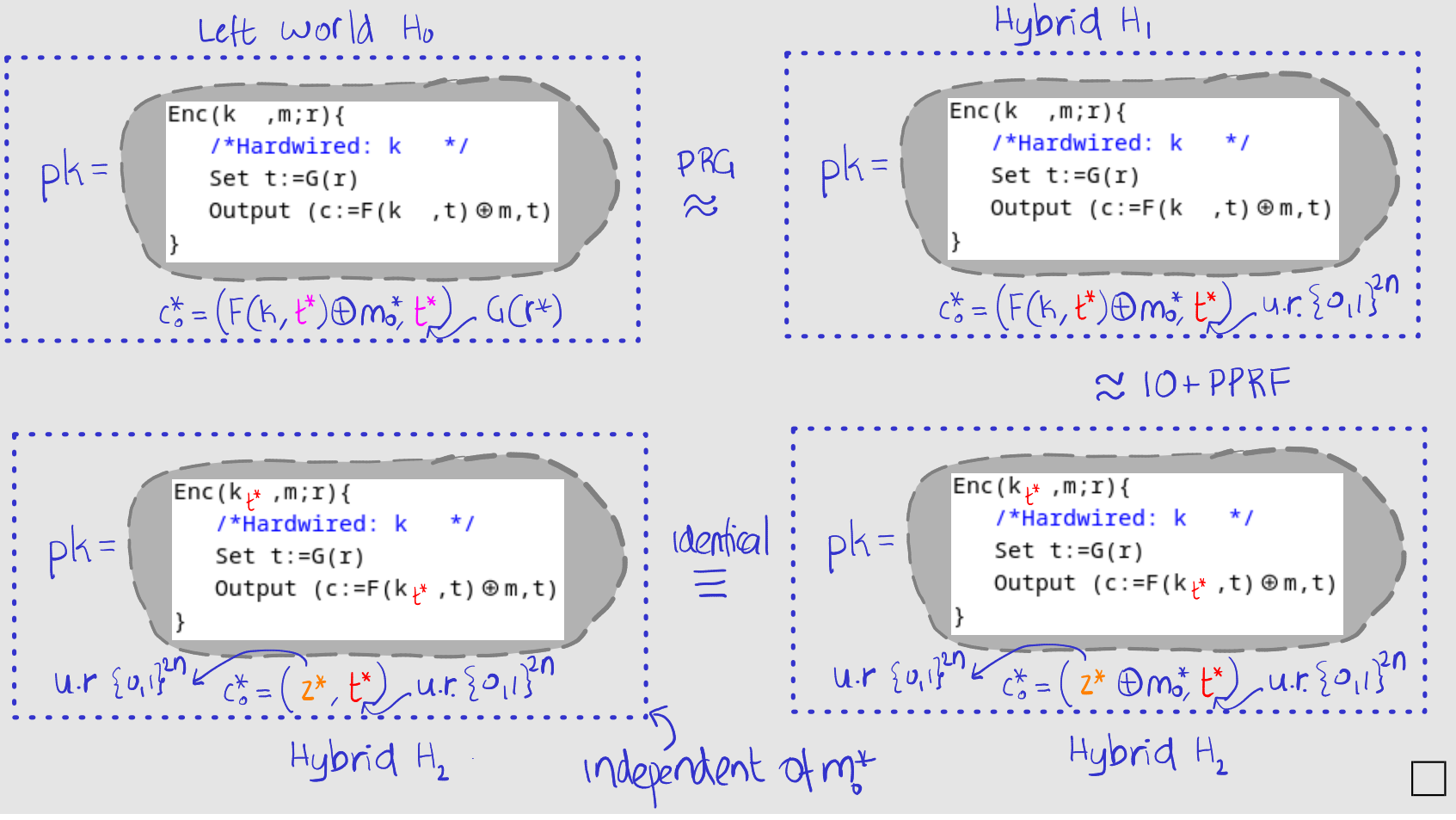
□

Construction 4 is IND-CPA PKE

Theorem 3

If **F** *is a puncturable PRF and* **Obf** *is an IO then* Π' *is a PKE*

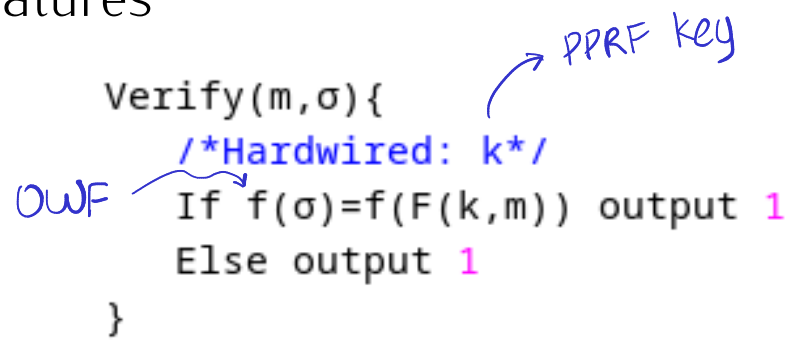
Proof Sketch (Hybrid argument).



□

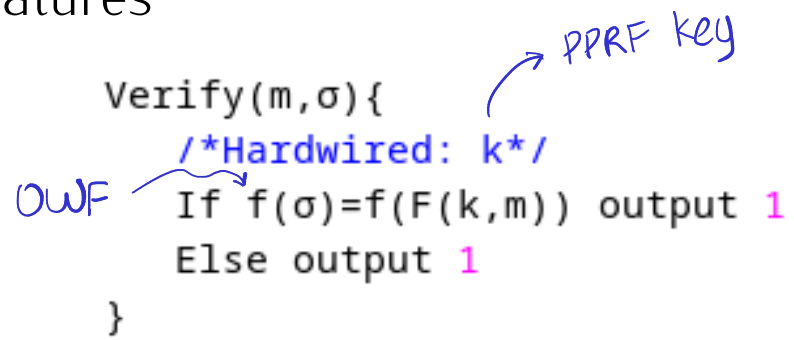
(Short) Digital Signatures via Punctured Programming

- Additionally use OWF f
- Signature on m is evaluation of PPRF F on m
- Public key consists of obfuscation of the following program that verifies signatures



(Short) Digital Signatures via Punctured Programming

- Additionally use OWF f
- Signature on m is evaluation of PPRF F on m
- Public key consists of obfuscation of the following program that verifies signatures



Theorem 4

If F *is a puncturable PRF,* f *a OWP and* Obf *is an IO then* Π' *is a PKE*

Exercise 4

Prove Theorem 4

To Recap Today's Lecture

- Relaxed requirements for obfuscators from VBBO to IO

To Recap Today's Lecture

- Relaxed requirements for obfuscators from VBBO to IO

- How to use IO?
  - PRG $\xrightarrow{\text{IO}}$ PKE

Key idea: how to use IO to hide secrets

To Recap Today's Lecture

- Relaxed requirements for obfuscators from VBBO to IO

- How to use IO?
  - PRG $\xrightarrow{\text{IO}}$ PKE

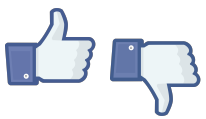
  Key idea: how to use IO to hide secrets

- New tool: *punctured* programming
  - Puncturable PRF (PPRF)
  - PPRF $\xrightarrow{\text{IO}}$ PKE
  - PPRF $\xrightarrow{\text{IO}}$ digital signature

Next Lecture

- How to construct indistinguishability obfuscator (IO)
  - Bootstrapping theorem for IO
  - State of affairs for IO for NC^1

Next Lecture

- How to construct indistinguishability obfuscator (IO)
  - Bootstrapping theorem for IO
  - State of affairs for IO for NC^1
- Course feedback 

References

1. The problem of constructing cryptographic primitives using IO was studied in [SW14]. That paper also introduces the "punctured programming" approach, and uses it to construct PKE, signatures, NIZK and several other primitives from IO.
2. Construction 1 is taken from Lecture 25 of Vinod Vaikuntanathan's MIT6875.
3. Mark Zhandry's COS 597C course (Autumn 2016) is an excellent source to learn further about program obfuscation.
4. Puncturable PRF was introduced in [BW13, BGI14, KPTZ13]. A formal definition can be found in [SW14].


Elette Boyle, Shafi Goldwasser, and Ioana Ivan.

Functional signatures and pseudorandom functions.

In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014.


Dan Boneh and Brent Waters.

Constrained pseudorandom functions and their applications.

In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.


Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias.

Delegatable pseudorandom functions and applications.

In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.


Amit Sahai and Brent Waters.

How to use indistinguishability obfuscation: deniable encryption, and more.

In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.