From obfuscation to white-box crypto: relaxation and security notions

Matthieu Rivain

WhibOx 2016, 14 Aug, UCSB

CryptoExperts

What does this program do?

```
([]+/H/)[1&11>>1]+(+[[]+(1-~1<<1)+(~1+1e1)+(1%11)+(1|1>>1|1)+(~1+1e1)+(.1^!1)])[[([]+!![
11])[11^11]+[[{}]+{}][1/1.1&1][1]]+([[]+111/!1][+!1][([{}]+{})[1e1>>1]+[[],[]+{}][1&11>>
1][1|[]]+([]+[][111])[1&1]+[{},1e1,!1+{}][~~(1.1+1.1)][1^1<<1]+(11/!{}+{})[1-~1<<1]+[!!{
}+[]][+(11>11)][[]+1]+(/^/[1.11]+/&/)[.1^!1]+[{},[{}]+{},1][1&11>>1][1+1e1+1]+([]+!!{})[
.1^!1]+([]+{}+[])[[]+1]+[!!{}+{}][!11+!111][[]+1]]+[])[(!/~/+{})[1|1<<1]+[/=/,[]+[][1]][
1&11>>1][1&1>>1]+([]+{})[~~(1.1+1.1)]+[1,!1+{}][1%11][1^1<<1]+(111/[]+/1/)[~1+1e1+~1]+[!
!/-/+[]][+(11>11)][1]]((1<<1^11)+((+(1<1))==([]+/-/[(!![11]+[])[+!1]+(!!/-/+{})[1-~1]+([
]+!/~/)[1-~1]+(!!/-/+{})[!111+!111]])[11%11]),-~11>>1)](~1-~1e1<<1<<1)+([]+{111:1111}+[]
)[11111.1%11.1*111e11|!11]+({}+/W/)[1+~1e1-(~11*1.1<<1)]+(+[[]+(1|1>>1)+(1|1>>1|1)+(11-1
>>1)+(1e1>>1|1)+(1e1>>1)+(1>>11)+(11>>>1)])[[(!!{}+[])[11>>>11]+[[]+{}][.1^!1][111%11]]+
([11/[]+[]][111%111][([{}]+[{}])[1e1>>1]+[[],[{}]+[{}]][1|1>>1|1][1|[]]+([][11]+[])[[]+1
]+[{},1e1,![1]+/~/][1<<!1<<1][1<<1^1]+(1/!1+{})[11+1>>1]+[!!/-/+{}][+(111>111)][111%11]+
([][11]+/&/)[1&1>>1]+[{},[]+{}+[],1][[]+1][11-~1+11>>1]+([]+!!/-/)[11>>11]+([]+{})[1|1>>
1|1]+[[]+!!{}][1>>>1][1&11]]+[])[(!{}+[])[1^1<<1]+[/=/,[]+[][1]][1<<1>>1][!111+!111]+([]
+{}+[])[1<<1^1>>1]+[1,![11]+[]][1|1>>1][1|1<<1|1]+(11/[]+/1/)[-~11>>1]+[!![111]+{}][+[]]
[1|1>>1]]((1e1-1)+((1&1>>1)==([]+/-/[(!!{}+{})[+(1>1)]+(!!/-/+{})[1|1<<1]+(!1+{})[1|1<<1
|1]+(!!/-/+{})[11.11>>11.11]])[1&1>>1]),1-~1<<1)](~1-~1e1<<1<<1)+(/^!/+[])[1+!![11%111]]
```

What does this program do?

```
([]+/H/)[1&11>>1]+(+[[]+(1-~1<<1)+(~1+1e1)+(1%11)+(1|1>>1|1)+(~1+1e1)+(.1^!1)])[[([]+!![
11])[11^11]+[[{}]+{}][1/1.1&1][1]]+([[]+111/!1][+!1][([{}]+{})[1e1>>1]+[[],[]+{}][1&11>>
1][1|[]]+([]+[][111])[1&1]+[{},1e1,!1+{}][~~(1.1+1.1)][1^1<<1]+(11/!{}+{})[1-~1<<1]+[!!{
}+[]][+(11>11)][[]+1]+(/^/[1.11]+/&/)[.1^!1]+[{},[{}]+{},1][1&11>>1][1+1e1+1]+([]+!!{})[
.1^!1]+([]+{}+[])[[]+1]+[!!{}+{}][!11+!111][[]+1]]+[])[(!/~/+{})[1|1<<1]+[/=/,[]+[][1]][
1&11>>1][1&1>>1]+([]+{})[~~(1.1+1.1)]+[1,!1+{}][1%11][1^1<<1]+(111/[]+/1/)[~1+1e1+~1]+[!
!/-/+[]][+(11>11)][1]]((1<<1^11)+((+(1<1))==([]+/-/[(!![11]+[])[+!1]+(!!/-/+{})[1-~1]+([
]+!/~/)[1-~1]+(!!/-/+{})[!111+!111]])[11%11]),-~11>>1)](~1-~1e1<<1<<1)+([]+{111:1111}+[]
)[11111.1%11.1*111e11|!11]+({}+/W/)[1+~1e1-(~11*1.1<<1)]+(+[[]+(1|1>>1)+(1|1>>1|1)+(11-1
>>1)+(1e1>>1|1)+(1e1>>1)+(1>>11)+(11>>>1)])[[(!!{}+[])[11>>>11]+[[]+{}][.1^!1][111%11]]+
([11/[]+[]][111%111][([{}]+[{}])[1e1>>1]+[[],[{}]+[{}]][1|1>>1|1][1|[]]+([][11]+[])[[]+1
]+[{},1e1,![1]+/~/][1<<!1<<1][1<<1^1]+(1/!1+{})[11+1>>1]+[!!/-/+{}][+(111>111)][111%11]+
([][11]+/&/)[1&1>>1]+[{},[]+{}+[],1][[]+1][11-~1+11>>1]+([]+!!/-/)[11>>11]+([]+{})[1|1>>
1|1]+[[]+!!{}][1>>>1][1&11]]+[])[(!{}+[])[1^1<<1]+[/=/,[]+[][1]][1<<1>>1][!111+!111]+([]
+{}+[])[1<<1^1>>1]+[1,![11]+[]][1|1>>1][1|1<<1|1]+(11/[]+/1/)[-~11>>1]+[!![111]+{}][+[]]
[1|1>>1]]((1e1-1)+((1&1>>1)==([]+/-/[(!!{}+{})[+(1>1)]+(!!/-/+{})[1|1<<1]+(!1+{})[1|1<<1
|1]+(!!/-/+{})[11.11>>11.11]])[1&1>>1]),1-~1<<1)](~1-~1e1<<1<<1)+(/^!/+[])[1+!![11%111]]
```

Answer: it prints “hello world”

What does this program do?

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.*-F/OO/OO);}F_OO()
{
            _-_-_-_
       _-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
        _-_-_-_-_-_-_-_
            _-_-_-_
}
```

What does this program do?

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.*-F/OO/OO);}F_OO()
{
            _-_-_-_
       _-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
        _-_-_-_-_-_-_-_
            _-_-_-_
}
```

Answer: it computes π

What is (cryptographic) obfuscation?

What is obfuscation?

Obfuscation is the deliberate act of creating obfuscated code, i.e. [...] that is **difficult for humans to understand**.

Obfuscators make reverse engineering more difficult [...] but **do not alter the behavior** of the obfuscated application.

– *wikipedia*

What is obfuscation?

Obfuscation is the deliberate act of creating obfuscated code, i.e. [...] that is **difficult for humans to understand**.

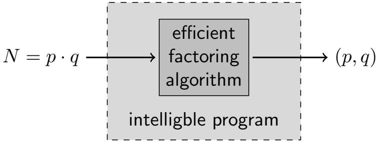
Obfuscators make reverse engineering more difficult [...] but **do not alter the behavior** of the obfuscated application.

– *wikipedia*

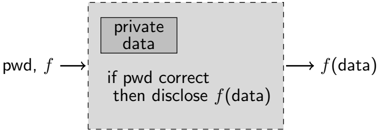
⇒ make a program unintelligible while preserving its functionality

Why obfuscation?

- To protect some secret inside a program
  - the algorithm itself (*e.g.* a factoring program)



  - some private data used by the program (*e.g.* conditional data access)



- Obfuscating a hello-word program is useless

Defining obfuscation

Program

- word in a formal (programming) language $P \in \mathcal{L}$
- function $\mathrm{execute} : \mathcal{L} \times \{0,1\}^* \to \{0,1\}^*$

$$\mathrm{execute} : (P, in) \mapsto out$$

- P implements a function $f : \mathcal{A} \to \mathcal{B}$ if

$$\forall a \in \mathcal{A} \ : \ \mathrm{execute}(P, a) = f(a)$$

  denoted $P \equiv f$
- P_1 and P_2 are functionally equivalent if

$$P_1 \equiv f \equiv P_2 \quad \text{for some } f$$

  denoted $P_1 \equiv P_2$

Defining obfuscation

Obfuscator

- algorithm O mapping a program P to a program $O(P)$ st:
- **functionality:** $O(P) \equiv P$
- **efficiency:** $O(P)$ is efficiently executable
- **security:**
  - (informal) $O(P)$ is hard to understand
  - (informal) $O(P)$ protects its data

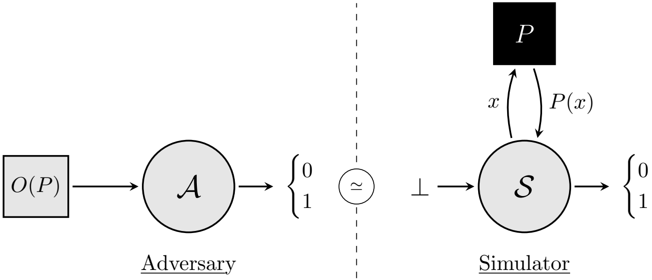
How to formally define the security property?

Virtual Black-Box (VBB) Obfuscation

- $O(P)$ reveals nothing more than the I/O behavior of P
- Any adversary on $O(P)$ can be simulated with a black-box access to P

Virtual Black-Box (VBB) Obfuscation

- $O(P)$ reveals nothing more than the I/O behavior of P
- Any adversary on $O(P)$ can be simulated with a black-box access to P



$$\left|\Pr[\mathcal{A}(O(P)) = 1] - \Pr[\mathcal{S}^P(\perp) = 1]\right| \leq \varepsilon$$

Impossibility result

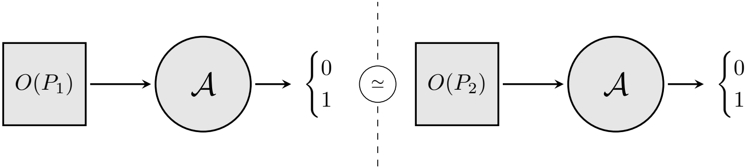
- VBB-O does not exist on general programs (CRYPTO'01)
- Counterexample:

```
uint128_t cannibal (prog P, uint128_t password)
{
    uint128_t secret1 = 0xe075b4f4eabf4377c1aa7202c8cc1ccb;
    uint128_t secret2 = 0x94ff8ec818de3bd8223a62e4cb7c84a4;

    if (password == secret1) return secret2;

    if (execute(P, null, secret1) == secret2) return secret1;

    return 0;
}
```

$$O(\texttt{cannibal})\big(O(\texttt{cannibal}), 0\big) = \texttt{secret1}$$

Indistinguishability obfuscation (iO)

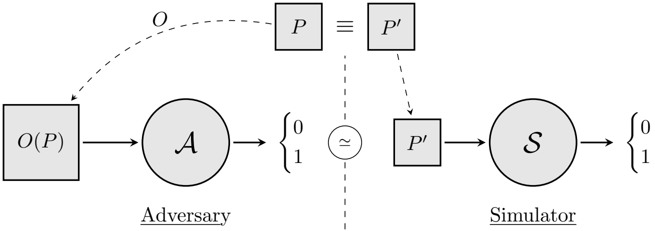
- Restricted to circuits *i.e.* programs without branches/loops
- For any two programs P_1 and P_2 st $P_1 \equiv P_2$ and $|P_1| = |P_2|$, the obfuscated programs $O(P_1)$ and $O(P_2)$ are indistinguishable



$$|\Pr[\mathcal{A}(O(P_1)) = 1] - \Pr[\mathcal{A}(O(P_2)) = 1]| \leq \varepsilon$$

Best possible obfuscation

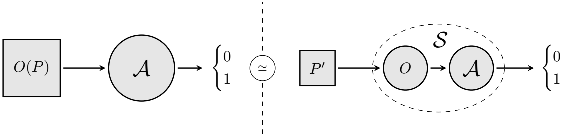
- Anything that can be learned (efficiently) from $O(P)$ can be learned from **any** $P' \equiv P$ with $|P'| \approx |P|$



$$|\Pr[\mathcal{A}(O(P)) = 1] - \Pr[\mathcal{S}(P') = 1]| \leq \varepsilon$$

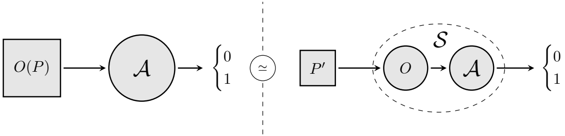
iO and BPO are equivalent

- iO $\Rightarrow$ BPO

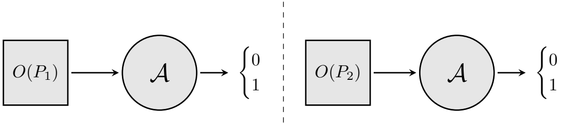


iO and BPO are equivalent

- iO $\Rightarrow$ BPO

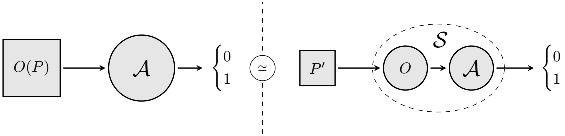


- BPO $\Rightarrow$ iO

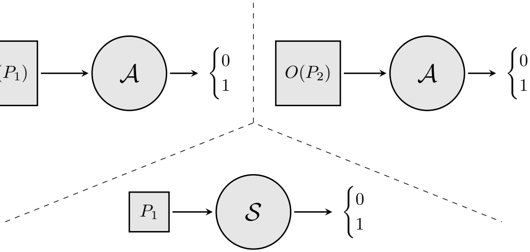


iO and BPO are equivalent

- iO $\Rightarrow$ BPO

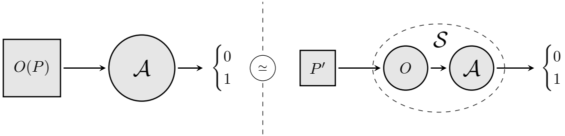


- BPO $\Rightarrow$ iO

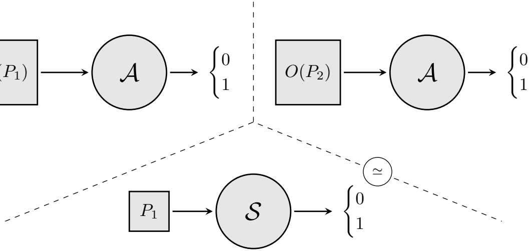


iO and BPO are equivalent

- iO ⇒ BPO

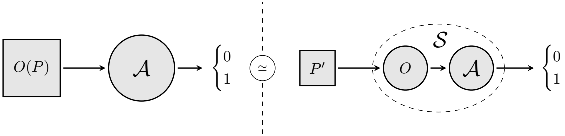


- BPO ⇒ iO

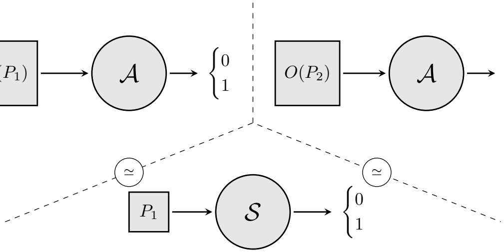


iO and BPO are equivalent

- iO $\Rightarrow$ BPO

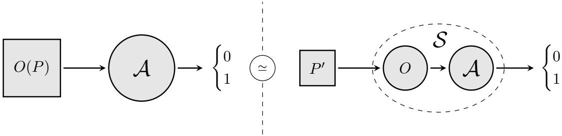


- BPO $\Rightarrow$ iO

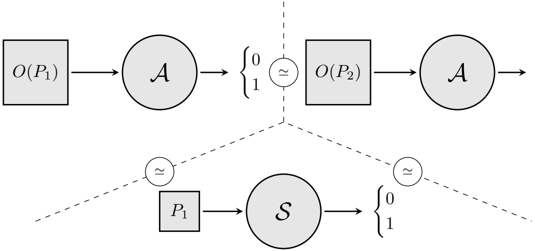


iO and BPO are equivalent

- iO ⇒ BPO

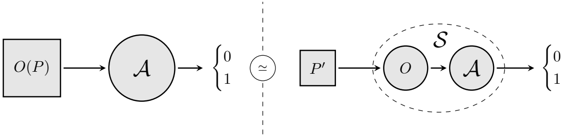


- BPO ⇒ iO

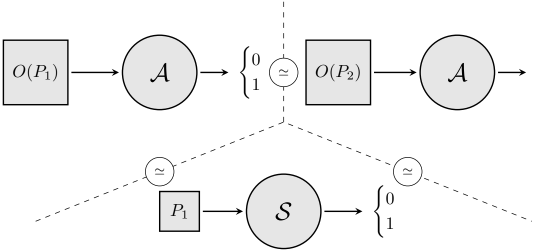


iO and BPO are equivalent

- iO $\Rightarrow$ BPO



- BPO $\Rightarrow$ iO



- We use iO in the rest of the presentation

What is white-box cryptography?

What is white-box cryptography?

"the attacker is assumed to have [...] full access to the encrypting software and control of the execution environment"

"Our main goal is to make key extraction difficult."

"While an attacker can clearly make use of the software itself [...], forcing an attacker to use the installed instance at hand is often of value to DRM systems providers."

– *Chow* et al. *(DRM 2002)*

What is white-box cryptography?

"the attacker is assumed to have [...] full access to the encrypting software and control of the execution environment"

$\Rightarrow$ obfuscation restricted to encryption (or another crypto primitive)

"Our main goal is to make key extraction difficult."

"While an attacker can clearly make use of the software itself [...], forcing an attacker to use the installed instance at hand is often of value to DRM systems providers."

– *Chow* et al. *(DRM 2002)*

What is white-box cryptography?

"the attacker is assumed to have [...] full access to the encrypting software and control of the execution environment"

$\Rightarrow$ obfuscation restricted to encryption (or another crypto primitive)

"Our main goal is to make key extraction difficult."

$\Rightarrow$ relaxed security requirements

"While an attacker can clearly make use of the software itself [...], forcing an attacker to use the installed instance at hand is often of value to DRM systems providers."

– *Chow* et al. *(DRM 2002)*

What is white-box cryptography?

"the attacker is assumed to have [...] full access to the encrypting software and control of the execution environment"

$\Rightarrow$ obfuscation restricted to encryption (or another crypto primitive)

"Our main goal is to make key extraction difficult."

$\Rightarrow$ relaxed security requirements

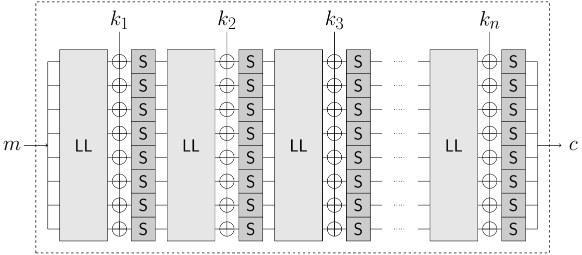
"While an attacker can clearly make use of the software itself [...], forcing an attacker to use the installed instance at hand is often of value to DRM systems providers."

$\Rightarrow$ encryption software $\neq$ secret key

– *Chow* et al. *(DRM 2002)*

What is white-box cryptography?

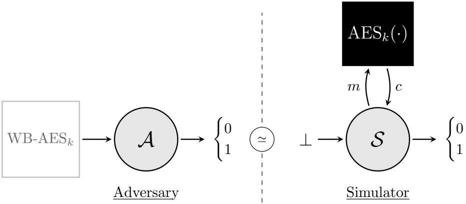
- Obfuscation restricted to a specific class of crypto primitives
- Typically, SPN ciphers:



- Running example: $\mathcal{F} = \left\{ \mathrm{AES}_k(\cdot) \mid k \in \{0,1\}^{128} \right\}$
- White-box obfuscator: $k \mapsto \text{WB-AES}_k \equiv \mathrm{AES}_k(\cdot)$

Strongest possible WBC

- VBB obfuscation restricted to AES



- Impossibility result does not apply
- The AES-LUT program achieves VBB
  - but does not fit into $10^9 \cdot 10^9 \cdot 10^9$ TB
- How to build a compact VBB AES implementation?
  - could be impossible to achieve

What does iO-AES mean?

- iO restricted to AES: $O(P_k) \simeq O(P'_k)$ for any $P_k \equiv P'_k \equiv \mathrm{AES}_k$
- Example of iO AES obfuscator:

  1. $k \leftarrow$ extract-key(P_k)
  2. return reference implem AES_k

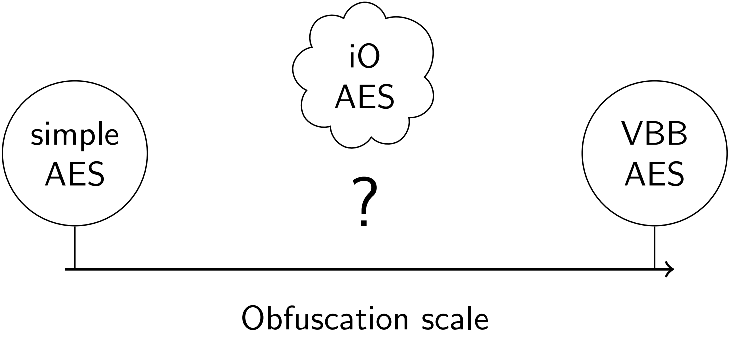
  - probably inefficient obfuscator!

- If a (compact) VBB AES implementation exists

$$O(P_k) \simeq O(\mathrm{VBB\text{-}AES}_k) \quad \Rightarrow \quad \text{efficient iO} \Leftrightarrow \text{VBB}$$

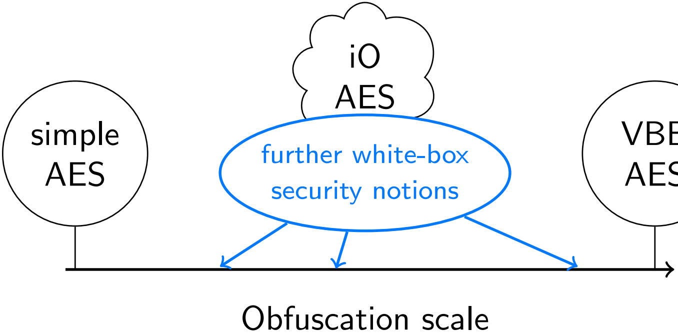
- So what does iO-AES means?

Defining WBC



- We need something
  - relaxed compared to VBB
  - meaningful compared to iO

Defining WBC

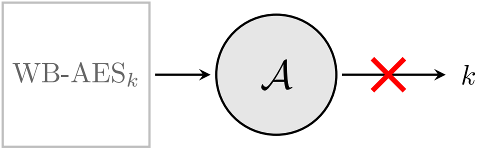


- We need something
  - relaxed compared to VBB
  - meaningful compared to iO ⇒ further notions

What could we expect from WBC?

What could we expect?

- The least requirement: key extraction must be difficult



- Easy to satisfy for some variant of AES:

$$E_k(\cdot) = \mathrm{AES}_h(\cdot) \quad \text{with } h = H(k)$$

  - H one-way $\Rightarrow$ simple AES_h implem unbreakable

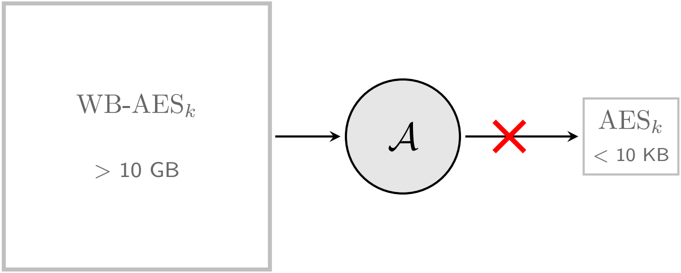
- We should expect more

What could we expect?

- Code-lifting cannot be avoided
  - the adversary can always use the software
- Code-lifting could be made unavoidable
  - force the adversary to use the software
- The software should then constrain the adversary
  - be less convenient to distribute
  - have restricted functionalities
  - include security features

Less convenient to distribute

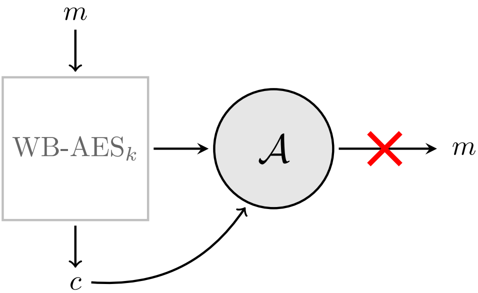
- Example: make the implementation huge and **incompressible**



- Possible use case: DRM

Restrict the software functionalities

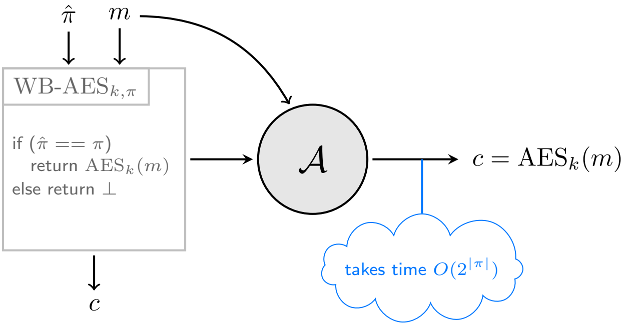
- Example: make the implementation **one-way**



- Namely: turning AES into a public-key cryptosystem
- Possible use case: light-weight signature scheme

Include security features

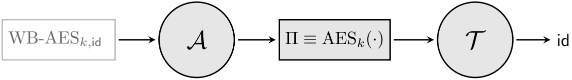
- Example: adding a password



- WB implem $\Rightarrow$ software secure element
- Possible use case: payment with token

Include security features

- Example: include a tracing mechanism

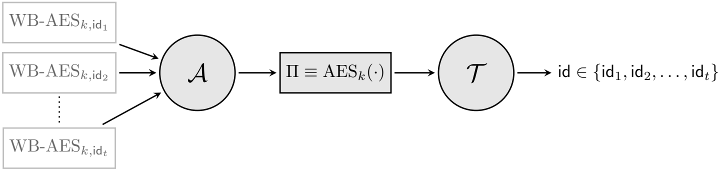


$$\exists\, \mathcal{T} \text{ st } \forall\, \mathcal{A} : \text{WB-AES}_{k,\text{id}} \mapsto \Pi \equiv \text{AES}_k(\cdot) \;\Rightarrow\; \mathcal{T}(\Pi) = \text{id}$$

- Possible use case: pay-TV

Include security features

- Example: include a tracing mechanism



$$\exists\, \mathcal{T} \text{ st } \forall\, \mathcal{A} : \mathrm{WB\text{-}AES}_{k,\mathsf{id}} \mapsto \Pi \equiv \mathrm{AES}_k(\cdot) \Rightarrow \mathcal{T}(\Pi) = \mathsf{id}$$

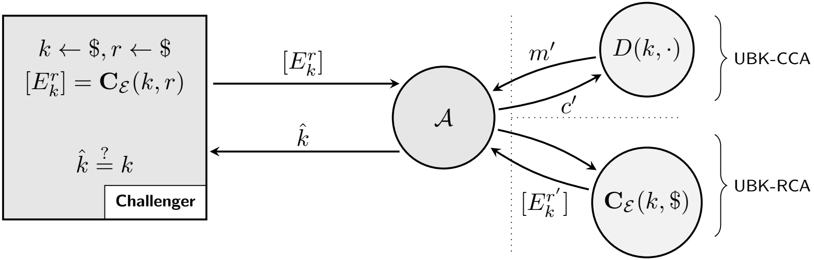
- Possible use case: pay-TV

White-box security notions

Security notions for symmetric ciphers

- Encryption scheme: $\mathcal{E} = (\mathcal{K}, \mathcal{M}, E, D)$
  - $E, D : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$
  - $E(k, \cdot) = D(k, \cdot)^{-1}$
- White-box compiler: $\mathbf{C}_{\mathcal{E}} : (k, r) \mapsto [E_k^r] \equiv E(k, \cdot)$
- Attack model:
  - target: a white-box encryption program $[E_k] = \mathbf{C}_{\mathcal{E}}(k, \$)$
  - CPA (chosen plaintext attack) – unavoidable
  - CCA (chosen ciphertext attack) – oracle for $D(k, \cdot)$
  - RCA (recompilation attack) – oracle for $\mathbf{C}_{\mathcal{E}}(k, \$)$
- Attack goals:
  - break (extract k), compress, inverse, be untraced

Unbreakability

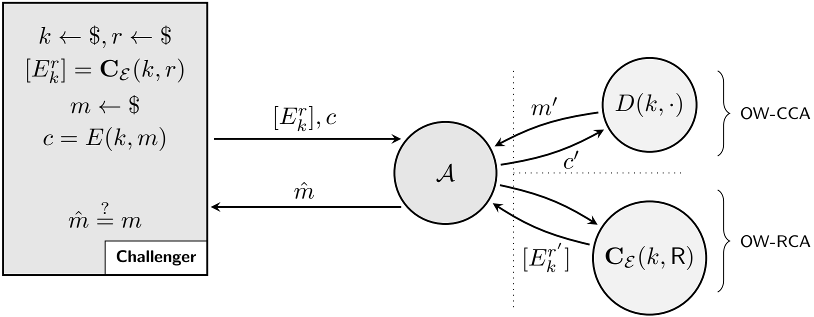


$\mathbf{C}_{\mathcal{E}}$ is (τ, ε)-secure wrt UBK-{CPA/CCA/RCA}

$$\Leftrightarrow$$

$\forall\, \mathcal{A}$ running in time τ : $\Pr[\hat{k} = k] \leq \varepsilon$

One-Wayness



$\mathbf{C}_{\mathcal{E}}$ is (τ, ε)-secure wrt OW-{CPA/CCA/RCA}

$\Leftrightarrow$

$\forall\, \mathcal{A}$ running in time τ : $\Pr[\hat{m} = m] \leq \varepsilon$

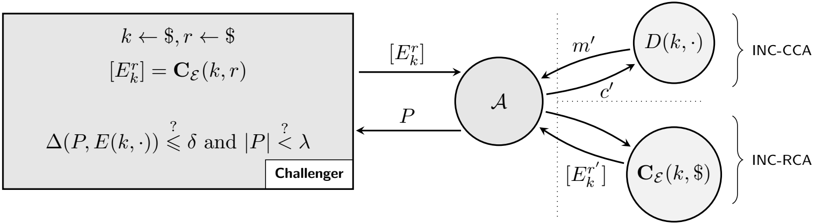
Incompressibility

- Distance between a program P and a function $f : \mathcal{X} \to \mathcal{Y}$

$$\Delta(P, f) = \frac{|\{x \in \mathcal{X} \text{ st } P(x) \neq f(x)\}|}{|\mathcal{X}|}$$

- If $\Delta(P, f) = 0$ then $P \equiv f$

Incompressibility



$\mathbf{C}_{\mathcal{E}}$ is (τ, ε)-secure wrt (λ, δ)-INC-$\{$CPA/CCA/RCA$\}$

$$\Leftrightarrow$$

$\forall\, \mathcal{A}$ running in time τ : $\Pr[\Delta(P, E(k,\cdot)) \leq \delta \wedge |P| \leq \lambda] \leq \varepsilon$

Incompressibility

(λ, δ)-INC only makes sense for:

$$\delta \approx 0$$

and

$$|\,\text{ref implem}\,| < \lambda < \min_{k,r} |\,[E_k^r]\,|$$

Toy example

- Encryption scheme $\mathcal{E}$

$$E : (k, m) \mapsto m^e \in \mathbb{G} \qquad D : (k, m) \mapsto m^{e^{-1} \bmod \omega} \in \mathbb{G}$$

  - $k = (\mathbb{G}, \omega, e)$
  - $\mathbb{G}$: RSA group with secret order ω
  - $e \in [2, \omega)$ coprime to ω

- White-box compiler $\mathbf{C}_{\mathcal{E}} : (k, r) \mapsto [E_k^r]$
  - $[E_k^r]$ computes m^f in $\mathbb{G}$
  - blinded exponent: $f = e + r \cdot \omega$

Toy example

- $\mathbf{C}_{\mathcal{E}}$ is OW-CPA under RSA[$\mathbb{G}$]
  - RSA[$\mathbb{G}$]: it's hard to compute $x^{1/e}$ for $x \xleftarrow{\$} \mathbb{G}$
- $\mathbf{C}_{\mathcal{E}}$ is $(\lambda, 0)$-INC-CPA (with $\lambda \approx \log f$) under ORD[$\mathbb{G}$]
  - ORD[$\mathbb{G}$]: it's hard to compute the order of $x \xleftarrow{\$} \mathbb{G}$
  - wrt an adversary producing algebraic programs

Toy example

- Disclaimer: toy example
  - OW part = RSA
  - INC part inefficient (linear in the size)
- Designing $\mathcal{E}$ with (efficient) OW $\mathbf{C}_{\mathcal{E}}$ = designing a PK encryption scheme
- Designing $\mathcal{E}$ with (efficient) INC $\mathbf{C}_{\mathcal{E}}$ = designing an incompressible encryption scheme
- White-box crypto is about designing a **compiler** for an existing encryption scheme
- Real challenge: design a OW and/or INC compiler for AES

Traceability

- White-box implem of the decryption (pay-TV use case)
- Principle: include secret perturbations of the decryption functionality

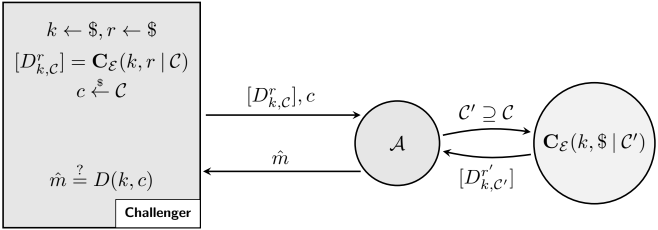
$$[D^r_{k,\mathcal{C}}] = \mathbf{C}_{\mathcal{E}}(k, r; \mathcal{C})$$

  where

$$[D^r_{k,\mathcal{C}}](c) = \begin{cases} \perp & \text{if } c \in \mathcal{C} \subseteq \mathcal{M} \\ D_k(c) & \text{otherwise} \end{cases}$$

Traceability

- Perturbation-Value Hiding (PVH) security:

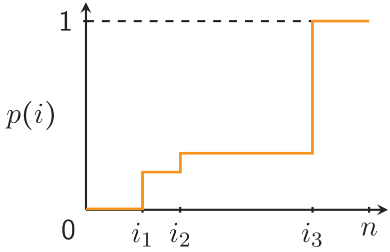


$$\mathbf{C}_\mathcal{E} \text{ is } (\tau, \varepsilon)\text{-secure wrt } \mathcal{C}\text{-PVH}$$
$$\Leftrightarrow$$
$$\forall\, \mathcal{A} \text{ running in time } \tau\text{: } \Pr[\hat{m} = D(k, c)] \leq \varepsilon$$

Traceability

- User i gets $P_i = \mathbf{C}_{\mathcal{E}}(k, r_i; \mathcal{C}_i)$
  - for random sets $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n \subseteq \mathcal{M}$
- Pirate program from t traitors: $\Pi = \mathcal{A}(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

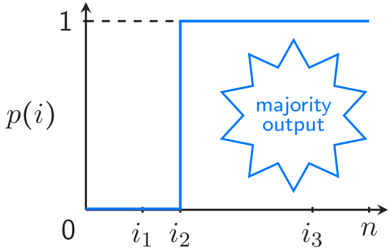
$$p(i) = \Pr[c \xleftarrow{\$} \mathcal{C}_i \backslash \mathcal{C}_{i-1} \ : \ \Pi(c) = D(k, c)]$$



Traceability

- User i gets $P_i = \mathbf{C}_{\mathcal{E}}(k, r_i; \mathcal{C}_i)$
  - for random sets $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n \subseteq \mathcal{M}$
- Pirate program from t traitors: $\Pi = \mathcal{A}(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

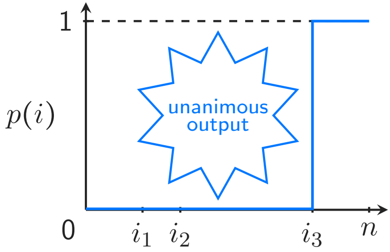
$$p(i) = \Pr[c \xleftarrow{\$} \mathcal{C}_i \backslash \mathcal{C}_{i-1} \ : \ \Pi(c) = D(k, c)]$$



Traceability

- User i gets $P_i = \mathbf{C}_{\mathcal{E}}(k, r_i; \mathcal{C}_i)$
  - for random sets $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n \subseteq \mathcal{M}$
- Pirate program from t traitors: $\Pi = \mathcal{A}(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k,\cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

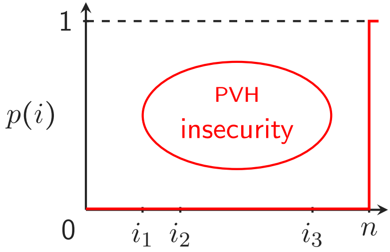
$$p(i) = \Pr[c \xleftarrow{\$} \mathcal{C}_i \backslash \mathcal{C}_{i-1} \ : \ \Pi(c) = D(k,c)]$$



Traceability

- User i gets $P_i = \mathbf{C}_{\mathcal{E}}(k, r_i; \mathcal{C}_i)$
  - for random sets $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n \subseteq \mathcal{M}$
- Pirate program from t traitors: $\Pi = \mathcal{A}(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

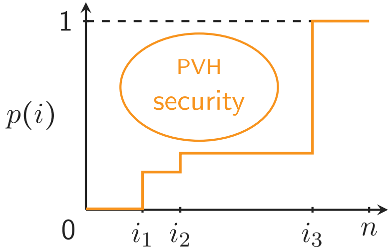
$$p(i) = \Pr[c \xleftarrow{\$} \mathcal{C}_i \backslash \mathcal{C}_{i-1} \ : \ \Pi(c) = D(k, c)]$$



Traceability

- User i gets $P_i = \mathbf{C}_{\mathcal{E}}(k, r_i; \mathcal{C}_i)$
  - for random sets $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n \subseteq \mathcal{M}$
- Pirate program from t traitors: $\Pi = \mathcal{A}(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

$$p(i) = \Pr[c \xleftarrow{\$} \mathcal{C}_i \backslash \mathcal{C}_{i-1} \; : \; \Pi(c) = D(k, c)]$$



Security hierarchy

- If $\mathcal{E}$ is a secure encryption scheme

$$
\begin{array}{ccccc}
 & & \text{INC} & & \\
 & & \Downarrow & & \\
\text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH}
\end{array}
$$

Security hierarchy

- If $\mathcal{E}$ is a secure encryption scheme

$$\begin{array}{ccccccccc} & & & & \text{INC} & & & & \\ & & & & \Downarrow & & & & \\ \text{VBB} & \Rightarrow & \text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH} & \Leftarrow & \text{VBB} \end{array}$$

Security hierarchy

- If $\mathcal{E}$ is a secure encryption scheme

$$
\begin{array}{ccccccccc}
 & & & & \text{VBB} & & & & \\
 & & & & \not\Downarrow & & & & \\
 & & & & \text{INC} & & & & \\
 & & & & \Downarrow & & & & \\
\text{VBB} & \Rightarrow & \text{OW} & \Rightarrow & \text{UBK} & \Leftarrow & \text{PVH} & \Leftarrow & \text{VBB}
\end{array}
$$

Conclusion

Conclusion

- WBC can be define as a restriction of cryptographic obfuscation
  - subset of programs (*e.g.* keyed permutation)
  - relaxed security notions
- More work needed to
  - refine / define alternative security notions
  - build candidate white-box compiler
- Open challenge: INC/OW/PVH-implementation of AES

Final thoughts

- Science is overstepped by industrial usage in the field of WBC
  - Digital content protection (pay-TV, DRM)
  - Mobile payments
  - Software protection
- Yet no secure solution available in the public literature
- Should we rely on the secret-spec model?
  - Academic cryptographer: "over my dead body!"
  - Industrial cryptographer: "only choice I have (for now)"
- Open question: who beats who?
  - secret-spec designer *vs.* state-of-the-art cryptanalyst

Biblio

- Obfuscation notions (VBB, iO, BPO)
  - *"On the (Im)possibility of Obfuscating Programs"* (Barak *et al.* CRYPTO 2001)
  - *"On Best-Possible Obfuscation"* (Goldwasser–Rothblum, TCC 2007)
- White-box crypto (introduction, first constructions)
  - *"A White-Box DES Implementation for DRM Applications"* (Chow *et al.* DRM 2002)
  - *"White-Box Cryptography and an AES Implementation"* (Chow *et al.* SAC 2002)
- Presented white-box security notions
  - *"White-Box Security Notions for Symmetric Encryption Schemes"* (Delerablée *et al.* SAC 2013)
- Related works
  - *"Towards Security Notions for White-Box Cryptography"* (Saxena–Wyseur–Preneel, ISC 2009)
  - *"White-Box Cryptography Revisited: Space-Hard Ciphers"* (Bogdanov–Isobe, CCS 2015)
  - *"Efficient and Provable White-Box Primitives"* (Fouque *et al.* ePrint 2016)

Questions ?