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

Matthieu Rivain

WhibOx 2016, 14 Aug, UCSB
What does this program do?
What does this program do?

Answer: it prints “hello world”
What does this program do?

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

}
```
What does this program do?

```c
#define _ -F<00||--F-00--;
int F=00,00=00;main(){F_00();printf("%1.3f\n",4.*-F/00/00);}F_00()
{


}

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.

⇒ 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

\[
N = p \cdot q \quad \rightarrow \quad \text{efficient factoring algorithm} \quad \rightarrow \quad (p, q)
\]

\[
\text{pwd, } f \quad \rightarrow \quad \text{if pwd correct then disclose } f(\text{data}) \quad \rightarrow \quad f(\text{data})
\]
Defining obfuscation

Program

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

- $P$ implements a function $f : \mathcal{A} \rightarrow \mathcal{B}$ if
  
  $$\forall a \in \mathcal{A} : \text{execute}(P, a) = f(a)$$

  denoted $P \equiv f$

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

  for some $f$

  denoted $P_1 \equiv P_2$
Defining obfuscation

**Obfuscator**

- algorithm $O$ mapping a program $P$ to a program $O(P)$ s.t:
  - **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 $P$

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

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

```c
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;
}
```

\[ O(\text{cannibal})(O(\text{cannibal}), 0) = \text{secret1} \]
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 $|P'| \approx |P|$

\[
\Pr[A(O(P))) = 1] - \Pr[S(P') = 1] \leq \varepsilon
\]
iO and BPO are equivalent

- iO $\Rightarrow$ BPO
iO and BPO are equivalent

- **iO → BPO**

  \begin{align*}
  O(P) & \xrightarrow{} \mathcal{A} \\
  \{0, 1\} \cong & \quad \{0, 1\}
  \end{align*}

- **BPO → iO**

  \begin{align*}
  O(P_1) & \xrightarrow{} \mathcal{A} \\
  \{0, 1\} & \quad \{0, 1\}
  \end{align*}
iO and BPO are equivalent

- iO $\Rightarrow$ BPO

- BPO $\Rightarrow$ iO

We use iO in the rest of the presentation
iO and BPO are equivalent

- iO $\Rightarrow$ BPO

- BPO $\Rightarrow$ iO

We use iO in the rest of the presentation
iO and BPO are equivalent

- **iO \implies BPO**

- **BPO \implies iO**

We use iO in the rest of the presentation.
iO and BPO are equivalent

- iO $\Rightarrow$ BPO

- BPO $\Rightarrow$ iO

We use iO in the rest of the presentation.
iO and BPO are equivalent

- **iO ⇒ BPO**

- **BPO ⇒ 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”
⇒ 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”
⇒ obfuscation restricted to encryption (or another crypto primitive)

“Our main goal is to make key extraction difficult.”
⇒ 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”
⇒ obfuscation restricted to encryption (or another crypto primitive)

“Our main goal is to make key extraction difficult.”
⇒ 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.”
⇒ encryption software ≠ secret key

– Chow et al. (DRM 2002)
What is white-box cryptography?

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

![Diagram of SPN cipher]

- Running example: $\mathcal{F} = \{\text{AES}_k(\cdot) \mid k \in \{0, 1\}^{128}\}$
- White-box obfuscator: $k \mapsto \text{WB-AES}_k \equiv \text{AES}_k(\cdot)$
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 $P_k \equiv P'_k \equiv \text{AES}_k$
- Example of iO AES obfuscator:

  1. $k \leftarrow \text{extract-key}(P_k)$
  2. return reference implem AES

  - probably inefficient obfuscator!

- If a (compact) VBB AES implementation exists

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

![Diagram showing WB-AES and A]

- Easy to satisfy for some variant of AES:
  \[ E_k(\cdot) = AES_h(\cdot) \quad \text{with} \quad h = H(k) \]
  - \( H \) 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**

![Diagram with WB-AES_k > 10 GB and AES_k < 10 KB]

- 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

\[
\text{WB-AES}_{k, \hat{\pi}}
\]

if \((\hat{\pi} == \pi)\)
\[
\text{return AES}_k(m)
\]
else return \(\bot\)

\[c = \text{AES}_k(m)\]

takes time \(O(2^{|\pi|})\)

- WB implem ⏞ software secure element
- Possible use case: payment with token
Include security features

- Example: include a tracing mechanism

\[ \exists \mathcal{T} \text{ st } \forall A : \text{WB-AES}_k,\text{id} \leftrightarrow \Pi \equiv \text{AES}_k(\cdot) \Rightarrow \mathcal{T}(\Pi) = \text{id} \]

- Possible use case: pay-TV
Include security features

- Example: include a tracing mechanism

\[ \exists \mathcal{T} \; \text{st} \; \forall A : \text{WB-AES}_{k, id} \leftrightarrow \Pi \equiv \text{AES}_k(\cdot) \implies \mathcal{T}(\Pi) = id \]

- 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:** $C_{\mathcal{E}} : (k, r) \mapsto [E_k^r] \equiv E(k, \cdot)$

- **Attack model:**
  - target: a white-box encryption program $[E_k] = C_{\mathcal{E}}(k, \$)$
  - CPA (chosen plaintext attack) – unavoidable
  - CCA (chosen ciphertext attack) – oracle for $D(k, \cdot)$
  - RCA (recompilation attack) – oracle for $C_{\mathcal{E}}(k, \$)$

- **Attack goals:**
  - break (extract $k$), compress, inverse, be untraced
Unbreakability

\[ k \leftarrow $, r \leftarrow $ \]
\[ [E^r_k] = C_\mathcal{E}(k, r) \]
\[ \hat{k} \overset{?}{=} k \]

Challenger

\[ A \]

\[ [E^r_k] \]
\[ D(k, \cdot) \] UBK-CCA

\[ m' \]
\[ c' \]
\[ C_\mathcal{E}(k, $) \] UBK-RCA

\[ \hat{k} \]

\[ C_\mathcal{E} \text{ is } (\tau, \varepsilon) \text{-secure wrt UBK-\{CPA/CCA/RCA\}} \]
\[ \iff \]
\[ \forall A \text{ running in time } \tau : \Pr[\hat{k} = k] \leq \varepsilon \]
One-Wayness

\[ k \leftarrow \$, r \leftarrow \$ \]
\[ [E^r_k] = C_\varepsilon(k, r) \]
\[ m \leftarrow \$ \]
\[ c = E(k, m) \]
\[ \hat{m} ? = m \]

\[ \text{Challenger} \]

\[ C_\varepsilon \text{ is } (\tau, \varepsilon)\text{-secure wrt OW-}\{\text{CPA/CCA/RCA}\} \]
\[ \iff \forall A \text{ running in time } \tau : \Pr[\hat{m} = m] \leq \varepsilon \]
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

\[ k \leftarrow \$, r \leftarrow \$
\]
\[ [E'_r^k] = C_\varepsilon(k, r) \]

\[ \Delta(P, E(k, \cdot)) \leq \delta \text{ and } |P| < \lambda \]

\[ C_\varepsilon \text{ is } (\tau, \varepsilon)\text{-secure wrt } (\lambda, \delta)\text{-INC-\{CPA/CCA/RCA\}} \]

\[ \iff \forall A \text{ running in time } \tau : \Pr[\Delta(P, E(k, \cdot)) \leq \delta \land |P| \leq \lambda] \leq \varepsilon \]
Incompressibility

\((\lambda, \delta)\)-INC only makes sense for:

\[ \delta \approx 0 \]

and

\[ |\text{ref implem}| < \lambda < \min_{k,r} \left| E^r_k \right| \]
Toy example

- **Encryption scheme** $\mathcal{E}$

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

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

- **White-box compiler** $C_{\mathcal{E}} : (k, r) \mapsto [E_k^r]$

  - $[E_k^r]$ computes $m^f$ in $\mathbb{G}$
  - blinded exponent: $f = e + r \cdot \omega$
Toy example

- $C_\mathcal{E}$ is OW-CPA under RSA[$G$]
  - RSA[$G$]: it’s hard to compute $x^{1/e}$ for $x \leftarrow \$ G$
- $C_\mathcal{E}$ is $(\lambda, 0)$-INC-CPA (with $\lambda \approx \log f$) under ORD[$G$]
  - ORD[$G$]: it’s hard to compute the order of $x \leftarrow \$ G$
  - 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 $C_{\mathcal{E}} = \text{designing a PK encryption scheme}$

- Designing $\mathcal{E}$ with (efficient) INC $C_{\mathcal{E}} = \text{designing an incompressible encryption scheme}$

- 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

\[ [D^r_{k,C}] = C\varepsilon(k, r; C) \]

where

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

- Perturbation-Value Hiding (PVH) security:

\[
k \leftarrow \$ , r \leftarrow \$
\]
\[
[D_{k,c}] = C_{\mathcal{E}}(k, r \mid C)
\]
\[
c \leftarrow C
\]
\[
\hat{m} \overset{?}{=} D(k, c)
\]

\[
\hat{m} \rightarrow \mathcal{A}
\]
\[
[D_{k,c}, c]
\]
\[
\mathcal{A} \rightarrow \mathcal{C}_{\mathcal{E}}(k, \$ \mid C')
\]
\[
C_{\mathcal{E}} \text{ is } (\tau, \varepsilon)\text{-secure wrt } C\text{-PVH}
\iff
\]
\[
\forall \mathcal{A} \text{ running in time } \tau: \Pr[\hat{m} = D(k, c)] \leq \varepsilon
\]
Traceability

- User $i$ gets $P_i = C_\varepsilon(k, r_i; C_i)$
  - for random sets $C_1 \subseteq C_2 \subseteq \cdots \subseteq C_n \subseteq M$

- Pirate program from $t$ traitors: $\Pi = A(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible

- PVH security $\Rightarrow$ linear tracing procedure

\[ p(i) = \Pr[c \leftarrow C_i \setminus C_{i-1} : \Pi(c) = D(k, c)] \]
Traceability

- User $i$ gets $P_i = C \varepsilon(k, r_i; C_i)$
  - for random sets $C_1 \subseteq C_2 \subseteq \cdots \subseteq C_n \subseteq \mathcal{M}$
- Pirate program from $t$ traitors: $\Pi = A(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

$$p(i) = \Pr[c \leftarrow C_i \setminus C_{i-1} : \Pi(c) = D(k, c)]$$
Traceability

- User $i$ gets $P_i = C_{\mathcal{E}}(k, r_i; C_i)$
  - for random sets $C_1 \subseteq C_2 \subseteq \cdots \subseteq C_n \subseteq \mathcal{M}$
- Pirate program from $t$ traitors: $\Pi = A(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible
- PVH security $\Rightarrow$ linear tracing procedure

$$p(i) = \Pr[c \leftarrow C_i \setminus C_{i-1} : \Pi(c) = D(k, c)]$$
Traceability

- User \( i \) gets \( P_i = C \mathcal{E}(k, r_i; C_i) \)
  - for random sets \( C_1 \subseteq C_2 \subseteq \cdots \subseteq C_n \subseteq \mathcal{M} \)
- Pirate program from \( t \) traitors: \( \Pi = A(P_{i_1}, P_{i_2}, \ldots, P_{i_t}) \)
  - with \( \Delta(\Pi, D(k, \cdot)) \) negligible
- PVH security \(\implies\) linear tracing procedure
  \[
p(i) = \Pr[c \leftarrow C_i \setminus C_{i-1} : \Pi(c) = D(k, c)]
  \]

![Diagram](image-url)
Traceability

- User $i$ gets $P_i = C_\mathcal{E}(k, r_i; C_i)$
  - for random sets $C_1 \subseteq C_2 \subseteq \cdots \subseteq C_n \subseteq \mathcal{M}$

- Pirate program from $t$ traitors: $\Pi = A(P_{i_1}, P_{i_2}, \ldots, P_{i_t})$
  - with $\Delta(\Pi, D(k, \cdot))$ negligible

- PVH security $\Rightarrow$ linear tracing procedure

$$p(i) = \Pr[c \leftarrow C_i \setminus C_{i-1} : \Pi(c) = D(k, c)]$$
Security hierarchy

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

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

\[
\begin{align*}
\text{VBB} & \Rightarrow \text{OW} \Rightarrow \text{UBK} \Leftarrow \text{PVH} \Leftarrow \text{VBB}
\end{align*}
\]
Security hierarchy

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

VBB \[\Downarrow\] INC \[\Downarrow\] VBB

VBB $\Rightarrow$ OW $\Rightarrow$ UBK $\Leftarrow$ PVH $\Leftarrow$ VBB
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 ?