CS 65500
Advanced Cryptography

Lecture 25: Obfuscation

Instructor: Aarushi Goel

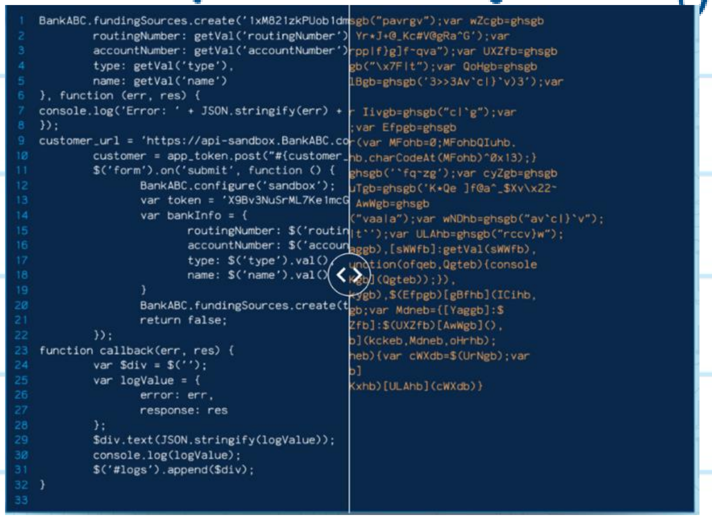
Spring 2025

Agenda

→ Virtual black-box obfuscation (VBB)
→ Applications of VBB obfuscation
→ Impossibility of VBB obfuscation
→ Indistinguishability Obfuscation ($i\mathcal{O}$)
→ How to use $i\mathcal{O}$?

Program Obfuscation

→ Obfuscation is the art of making programs "unintelligible".



→ The program must be fully functional.

→ May contain secrets that shouldn't be revealed to the users.

Use: protecting proprietary algorithms, for hiding potential bugs, for hardwiring cryptographic keys inside apps.

→ Several heuristic approaches to obfuscation exist, but they breakdown under serious program analysis

Virtual Black-Box Obfuscation (Cryptographic Obfuscation)

Having obfuscated source code is no better than black-box access

Definition: A probabilistic algorithm Obf is a VBB obfuscator if

1. Functionality preserving: $\forall$ programs P and security parameter $\lambda \in \mathbb{N}$, Obf outputs $\tilde{P} \leftarrow Obf(1^\lambda, P)$ such that $\forall x$ in the domain of P, it holds that:

$$\Pr[\tilde{P}(x) = P(x)] = 1.$$

2. VBB Security: $\forall$ PPT adversaries A, $\exists$ PPT simulators S such that $\forall$ programs P and security parameter $\lambda \in \mathbb{N}$, it holds that:

$$\left| \Pr[A(Obf(1^\lambda, P)) = 1] - \Pr[S^P(1^\lambda, |P|) = 1] \right| \leq negl(\lambda)$$

Secret-Key Encryption $\xrightarrow{\text{VBBO}}$ Public-Key Encryption

→ We can use VBB obfuscation to design a PKE scheme from SKE.

→ Let (Keygen, Enc, Dec) be a SKE scheme & Obf be a VBBO.

We can design PKE as follows:

- PKE.Keygen(1^λ): $sk \leftarrow \text{Keygen}(1^\lambda)$
  $pk \leftarrow \text{Obf}(\text{Enc}(sk, \cdot))$
- PKE.Enc(pk, m): $ct \leftarrow pk(m)$
- PKE.Dec(sk, ct): $m \leftarrow \text{Dec}(sk, m)$

rely on VBB security to argue sk remains hidden.

Impossibility of Obfuscation

→ VBB obfuscation is impossible in general

→ Example of an unobfuscatable family of functions:

Consider a program $P_{\alpha,\beta,\gamma}$ defined as follows:
(modeled as a Turing machine)

$$P_{\alpha,\beta,\gamma}(x) = \begin{cases} \beta & \text{if } x = \alpha \\ \gamma & \text{if } x(\alpha) = \beta \\ \perp & \text{otherwise.} \end{cases}$$

If α, β, γ are uniformly random strings, observe that:

1. Oracle access to $P_{\alpha,\beta,\gamma}$ is highly unlikely to yield to anything other than $\perp$ with polynomially many queries

2. Given $\tilde{P}_{\alpha,\beta,\gamma} = Obf(1^\lambda, P_{\alpha,\beta,\gamma})$, the functionality preserving property of Obf ensures $\tilde{P}_{\alpha,\beta,\gamma}(\tilde{P}_{\alpha,\beta,\gamma}) = \gamma$

Combining these two observations, we get that $S^{P_{\alpha,\beta,\gamma}}$ is almost never able to retrieve γ, whereas A given $\tilde{P}_{\alpha,\beta,\gamma}$ can retrieve γ.

$\Rightarrow$ Any non-trivial predicate computed on γ will therefore not be simulatable with noticeable probability.

$\Rightarrow$ VBB obfuscation is impossible in general.

Exceptions

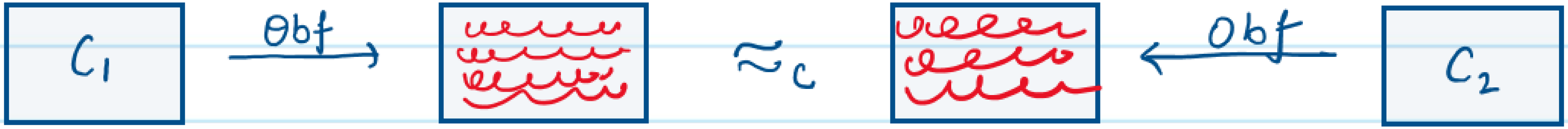
→ Hardware assisted

→ For some simple functions like variants of point functions

* But in general, "low complexity classes" are still unobfuscatable.

→ <u>Alternate Idea</u>: Consider a weaker definition.

Indistinguistinguishability Obfuscation (iO)



$\forall\, C_1, C_2$, such that

$\forall x: \quad C_1(x) = C_2(x)$

Defining iO

Definition: A PPT algorithm Obf is an indistinguishability obfuscator if $\forall$ pairs of circuits C, C', such that $C(x) = C'(x)$ $\forall$ inputs x, $\forall$ security parameters $\lambda \in \mathbb{N}$, the following distributions are computationally indistinguishable:

$$\{\mathrm{Obf}(1^\lambda, C)\} \approx_c \{\mathrm{Obf}(1^\lambda, C')\}$$

* The functionality preserving property remains identical to that in the definition of VBB obfuscation.

Relationship between iO & OWF

→ Interestingly, unlike other cryptographic primitives, the existence of iO does not imply $P \neq NP$.

* If $P = NP$, iO exists.

if $P = NP$, we can design a simple iO construction where given any circuit C, $Obf(1^\lambda, C)$ outputs the smallest circuit that is functionally equivalent to C. This will trivially produce the same obfuscated circuit for all functionally equivalent circuits.

⇒ Existence of iO does not necessarily imply existence of OWF. Interesting cryptographic primitives follow when we combine iO & OWF.

$i\mathcal{O} \implies$ Witness Encryption.

Definition of WE: Let $\mathcal{L}$ be an NP language with the corresponding relation $R_\mathcal{L}$, i.e., $\forall$ instances x, $R_\mathcal{L}(x,w)=1$ if w is a witness for the statement $x \in \mathcal{L}$.

A witness encryption scheme consists of the following algorithms:

→ $\mathrm{Enc}(1^\lambda, x, m)$: given a message $m \in \{0,1\}$ and an instance x, output a ciphertext ct.

→ $\mathrm{Dec}(w, ct)$: given ciphertext ct & witness w, output a message bit.

These algorithms satisfy the following properties:

Correctness: $\forall \lambda \in \mathbb{N}$, $\forall m \in \{0,1\}$, $\forall$ instances x and $\forall$ witnesses w, if $R_\mathcal{L}(x,w)=1$, then $\Pr[\mathrm{Dec}(w, \mathrm{Enc}(1^\lambda, m, x)) = m] = 1$

Soundness: $\forall x \notin \mathcal{L}$, & $\forall \lambda \in \mathbb{N}$,

$$\{\mathrm{Enc}(1^\lambda, x, 0)\} \approx_c \{\mathrm{Enc}(1^\lambda, x, 1)\}$$

We can construct WE using iO as follows:

→ $\mathrm{Enc}(1^\lambda, x, m)$: · Construct a circuit $C_{x,m}(w) = \begin{cases} m & \text{if } R_L(x,w)=1 \\ \perp & \text{otherwise} \end{cases}$

· output $ct = \mathrm{Obf}(1^\lambda, C_{x,m}(\cdot))$

→ $\mathrm{Dec}(w, ct)$: output $ct(w)$.

* Correctness: trivial

* Soundness: when $x \notin L$, i.e., $\nexists w$, s.t $R_L(x,w)=1$, both $C_{x,0}$ & $C_{x,1}$ will output $\perp$ on all inputs making them functionally equivalent

$\Rightarrow \{\mathrm{Obf}(1^\lambda, C_{x,0})\} \approx_c \{\mathrm{Obf}(1^\lambda, C_{x,1})\}$

$i\mathcal{O}$ + PRG $\Rightarrow$ PKE

We know $i\mathcal{O} \Rightarrow$ WE. We will now show WE + PRG $\Rightarrow$ PKE.

→ Let $f: \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG

→ Let L be an NP language consisting of images of f.

i.e., $R_L(x, w) = 1$ if $f(w) = x$. Let (E, D) be a WE scheme for L.

→ We can design a PKE as follows:

* Keygen(1^λ): sample $sk \xleftarrow{\$} \{0,1\}^\lambda$

compute $pk = f(sk)$.

* Enc(pk, m): $ct = E(1^\lambda, x = pk, m)$

* Dec(sk, ct): $m = D(x = pk, w = sk, ct)$

Correctness of this scheme is easy to see.

Security:

H_0 $\{ (pk, \cdot\ ) \leftarrow \text{KeyGen}(1^\lambda),\ \text{Enc}(pk, m_0) \}$

H_1 $\{ pk \xleftarrow{\$} \{0,1\}^{2\lambda},\ \text{Enc}(pk, m_0) \}$

H_2 $\{ pk \xleftarrow{\$} \{0,1\}^{\lambda},\ \text{Enc}(pk, m_1) \}$

H_3 $\{ (pk, \cdot) \leftarrow \text{Keygen}(1^\lambda),\ \text{Enc}(pk, m_1) \}$

Concluding Remarks:

→ iO can be used to design 2-round MPC [Garg, Gentry, Halevi, Raykova; TCC 2014]

→ There have been multiple attempts to design iO for general circuits all of which were eventually broken.

→ Finally in 2021, Aayush Jain, Amit Sahai & Huijia Lin designed an iO from well-founded assumptions.