Program Obfuscation

1 Obfuscation Definitions

Definition 1 (Obfuscation Algorithm.)**.** *An obfuscation algorithm* $\mathcal{O}(\text{program } P, \text{ security parameter } 1^\lambda; \text{ randomness } r)$ *is a polynomial-time randomized algorithm with the following property:*

- ***(Perfect) Functionality:*** *For all programs P,*

$$\Pr_r[\mathcal{O}(P, 1^\lambda; r) \equiv P] = 1$$

Of course, we will also want some notion of security. But what should it be? I encourage you to pause here, and think about what the "right" security notion should be.

1.1 Ideal Obfuscation

Perhaps the first notion that comes to mind is that anything you can compute given the obfuscation of a program, you can compute using only black-box access to the function.

Definition 2 (Ideal Obfuscation)**.** *An obfuscation algorithm* $\mathcal{O}$ *is an* ideal obfuscator *if for every PPT adversary A, there is a PPT simulator Sim such that for all programs P, we have*

$$A(\mathcal{O}(P, 1^\lambda)) \approx_c Sim^P(|P|, 1^\lambda)$$

To illustrate the power of ideal obfuscation, we note that it can be used to generically convert a secret key encryption scheme into a public-key one.

Theorem 3 (Essentially Diffie-Helman '76)**.** *Assume an ideal obfuscator exists. If a secret key encryption scheme exists, then a public-key encryption scheme exists.*

Proof (Sketch). We will not be so formal here for reasons we will see later. Generate the secret key $sk = (sk', k)$ where sk' is a secret key to the private key encryption scheme and k is the key to a PRF. The public key will be $pk = \mathcal{O}(P, 1^\lambda)$ where P is the program $P(m; r) = Enc(sk, m; PRF_k(r))$.

Observe that black box access to P is the same as being able to see chosen plaintexts in the CPA security game. □

In fact, one can even just outright construct a public-key encryption scheme (and much more) using ideal obfuscation. Unfortunately, ideal obfuscators do not exist.

Theorem 4. *An ideal obfuscator does not exist.*

The idea is to consider the adversary that just outputs the program it is given. On the other hand, the simulator can only query a small number of outputs of P, so it is hopeless to come up with a program that exactly computes P. Indeed, one can show that this is impossible if P is a PRF. But we can also show it is impossible unconditionally for P that are "point functions."

Proof. Consider the adversary A that just outputs the program it is given as input, i.e., $A(\tilde{P}) = \tilde{P}$. For $x \in \{0,1\}^\lambda$, let P_x be the program given by $P_x(y) = \mathbb{1}[y = x]$. Let P_{zero} be a program that always outputs zero and has the same length as each P_x program.

Since *Sim* makes poly(λ) queries, we have that

$$\Pr_{x \leftarrow \{0,1\}^\lambda}[Sim^{P_{zero}}(|P_{zero}|, 1^\lambda) \text{ makes an oracle query to } x] \leq \text{poly}(\lambda) 2^{-\lambda} = \text{negl}(\lambda). \tag{1}$$

Then there must exist a fixed x such this bound holds. For this x, we have that

$$\begin{aligned} \mathcal{O}(P_x, 1^\lambda) = A(\mathcal{O}(P_x, 1^\lambda)) &\approx_c Sim^{P_x}(|P|, 1^\lambda) \\ &\approx_s Sim^{P_{zero}}(|P|, 1^\lambda) \\ &\approx_c A(\mathcal{O}(P_{zero}, 1^\lambda)) = \mathcal{O}(P_{zero}, 1^\lambda) \end{aligned}$$

where the first and third line come from the definition of A and the assumption on $\mathcal{O}$ and the middle line comes Equation (1) for this x and the fact that P_x and P_{zero} are identical except at x.

But this is a contradiction because $\mathcal{O}(P_x, 1^\lambda)) \approx_c \mathcal{O}(P_{zero}, 1^\lambda)$ is clearly false (the programs evaluate to different values on x). □

1.2 Virtual Black Box Obfuscation

In light of this impossibility, it is natural to relax our security notion from hiding "many bits" to just hiding "single bits."

Definition 5 (Virtual Black Box (VBB) Obfuscation [BGIRSVY '01])**.** *An obfuscation algorithm $\mathcal{O}$ is an* VBB obfuscator *if for every PPT adversary A, there is a PPT simulator Sim such that for all programs P*

$$\left| \Pr[A(\mathcal{O}(P, 1^\lambda)) = 1] - \Pr[Sim^P(|P|, 1^\lambda)) = 1] \right| \leq \lambda^{-\omega(1)}.$$

Theorem 6 (BGIRSVY '01)**.** *VBB obfuscation does not exist.*

Proof. For strings $\alpha, \beta \in \{0,1\}^\lambda$ and a bit $b \in \{0,1\}$, let $P_{\alpha,\beta,b}$ be the program given by

$$P_{\alpha,\beta,b}(Q) = \begin{cases} \alpha, & \text{if } Q = \beta \\ b, & \text{if } Q(\alpha) = \beta \\ 0, & \text{otherwise,} \end{cases}$$

Observe that running $P_{\alpha,\beta,b}$ on itself will output b. So the adversary $A(\tilde{P}) = \tilde{P}(P)$ satisfies that for all α, β and b that

$$\Pr[A(\mathcal{O}(P_{\alpha,\beta,b}, 1^\lambda)) = b] = 1.$$

On the other hand, we will show there is no way to recover b using black box access to P in poly(λ) queries for all α and β.

The proof is similar to the ideal obfuscation case. Let *Sim* be an arbitrary PPT algorithm. Let P_{zero} denote the all zero program of the same length as $P_{\alpha,\beta,b}$. Since *Sim* makes poly(λ) many queries

$$\Pr_{\alpha \leftarrow \{0,1\}^\lambda, \beta \leftarrow \{0,1\}^\lambda}[Sim^{P_{zero}}(|P_{zero}|, 1^\lambda) \text{ queries a } Q \text{ with } Q = \beta \text{ or } Q(\alpha) = \beta] \leq \text{poly}(\lambda) 2^{-\lambda+1} \leq \text{negl}(\lambda). \tag{2}$$

Then there are fixed α and β such that the above bound holds. For these α and β and any $b \in \{0,1\}$,

$$Sim^{P_{\alpha,\beta,b}}(|P_{\alpha,\beta,b}|, 1^{\lambda}) \approx_s Sim^{P_{zero}}(|P_{zero}|, 1^{\lambda})$$

because of Equation (2) and the fact that $P_{\alpha,\beta,b}$ and P_{zero} are identical on all points Q with $Q \neq \beta$ and $Q(\alpha) \neq \beta$. Thus, we have that

$$\Pr_{b \leftarrow \{0,1\}}[Sim^{P_{\alpha,\beta,b}}(|P_{\alpha,\beta,b}|, 1^{\lambda})) = b] \leq \frac{1}{2} + \text{negl}(\lambda),$$

as desired. □

The crux of this proof is that Turing machines can eat themselves. One might wonder if it extends to circuits. (Digression: the inability of circuits to eat themselves is related to the difficulty of proving even seemingly "obvious" lower bounds on circuits. While we know that $DTIME[n \log^2 n] \not\subseteq DTIME[n]$, it is still open if $DTIME^{NP}[2^n] \subseteq SIZE[3.2n]$!)

It turns out that one can rule out VBB for circuits. The key idea is to use homomorphic encryption to enable circuits to "eat themselves."

Theorem 7 (BGIRSVY '01)**.** *VBB obfuscation for circuits does not exist.*

Proof (Sketch). We sketch this proof because the details are a bit involved. The first step is to show that a FHE scheme exists if VBB for circuits exists. We won't discuss here how to do this.

For a secret key sk to homomorphic encryption scheme with cipher texts of size λ, strings $\alpha, \beta \in \{0,1\}^{\lambda}$ and a bit $b \in \{0,1\}$, let $C_{sk,\alpha,\beta,b}$ be the circuit that takes as input a string of size at most poly(λ) and outputs

$$C(x) = \begin{cases} \mathsf{Enc}_{sk}(\alpha), & \text{if } x = 0 \\ \alpha, & \text{if } x = \beta \\ b, & \text{if } Dec_{sk}(x) = \beta \\ 0, & \text{otherwise,} \end{cases}$$

Note that

$$b = C(\mathsf{Eval}(C, C(0)).$$

On the other hand, one can show (we won't here) that in the black box setting, it is impossible to recover b. □

1.3 Indistinguishability Obfuscation

In light of these impossibility results, Barak et al. suggested another notion of obfuscation.

Definition 8 (Indistinguishability Obfuscation (iO) [BGIRSVY '01])**.** *An obfuscator $\mathcal{O}$ is an* indistinguishability obfuscator *if for any two circuits C and C' of the same size computing the same function, we have*

$$\mathcal{O}(C, 1^{\lambda}) \approx_c \mathcal{O}(C', 1^{\lambda}).$$

Note: unless otherwise specified, we set $\lambda = |C|$. (We can always pad C to get a larger security parameter.)

Some interpretations:

- The only thing the obfuscated circuit reveals are things about the truth table of the circuit, not things about the implementation of the circuit.
- One can think of it is a pseudocanonicalizer (the meaning might be more clear from the next theorem)
- You might think to yourself. How can iO be useful, since it only shows indistinguishability between functionally identical circuits? This is a very reasonable intuition. Hold on to it for when we see how to use iO!

Unlike almost all other cryptographic objects, iO exists if $\mathbf{P} = \mathbf{NP}$!

Theorem 9 (BGIRSVY '01)**.** *If* $\mathbf{P} = \mathbf{NP}$*, then* iO *exists.*

Proof. Let $iO(C)$ be the lexicographically first circuit equivalent to C. Then clearly for any circuits C and C' computing the same function we have $iO(C) = iO(C')$. Furthermore, this is efficiently computable if $\mathbf{P} = \mathbf{NP}$. □

The culmination of a long line of work starting with [GGHRSW '13] now shows that iO exists under plausible assumptions.

Theorem 10 (JLS '21)**.** *Under "well studied" cryptographic assumptions,* iO *exists.*

2 iO, what is it good for?

Since one-way functions imply that $\mathbf{P} \neq \mathbf{NP}$, this means that, using current techniques, we cannot even prove that iO implies one-way functions. This makes iO seem quite weak. In this section, we will begin showing that iO is actually very strong, as long as you add in (essentially) the assumption that $\mathbf{P} \neq \mathbf{NP}$.

2.1 One-way Functions

First, we will show how to construct one-way functions using iO.

Theorem 11 (KMNPRY '14)**.** *Assume* iO *exists and there is no PPT algorithm solving SAT infinitely often. Then one-way functions exist.*

Proof. Our one-way function will be $f_s(r) = iO(Z_s; r)$ where Z_s denotes a circuit with s gates that computes the zero function. For contradiction, suppose there is a PPT algorithm I that inverts f_s with probability at least $s^{-\Omega(1)}$ for infinitely many s. Now consider the following PPT algorithm $A(\varphi)$ for solving SAT:

1. Let $s = |\varphi|$
2. Sample $\tilde{\varphi} \leftarrow iO(\varphi)$
3. Set $r = I(s, \tilde{\varphi})$
4. Output "unsatisfiable" iff
$$\tilde{\varphi} = iO(Z_s; r). \tag{3}$$

Perfect functionality implies that Equation (3) only occurs when φ is unsatisfiable, so the algorithm is correct on satisfiable φ. On the other hand, when φ is unsatisfiable and I inverts f_s, we have

$$\begin{aligned}\Pr[A(\varphi) = \text{“unsatisfiable”}] &= \Pr_{\tilde{\varphi} \leftarrow iO(\varphi)}[\tilde{\varphi} = iO(Z_s; I(s, \tilde{\varphi}))] \\ &\geq \Pr_{\tilde{\varphi} \leftarrow iO(Z_s)}[\tilde{\varphi} = iO(Z_s; I(s, \tilde{\varphi}))] - \text{negl}(s) \\ &\geq s^{-\Omega(1)}.\end{aligned}$$

where the first line is by definition, the second by iO security since φ is unsatisfiable, and the last by the assumed properties of I.

One can amplify this $s^{-\Omega(1)}$ success probability to $1 - \text{negl}(s)$ by repeating $\text{poly}(s)$ times. Hence, we get a PPT algorithm solving SAT infinitely often, which is a contradiction. □

2.2 Witness Encryption

Now we will use iO to construct an exotic cryptographic primitive we have not mentioned before in class: witness encryption.

Definition 12 (GGSW '13)**.** *A witness encryption scheme (for SAT) consists of two probabilistic polynomial algorithms* Enc(*formula* φ*, bit* b*, security parameter* 1^λ) *and* Dec(*ciphertext* c*, witness* w) *with the following two properties:*

- ***Functionality:*** *If* $\varphi(w) = 1$*, then*

$$\Pr[\mathsf{Dec}(\mathsf{Enc}(\varphi, b, 1^\lambda), w) = b] = 1.$$

- ***Security:*** *If* φ *is unsatisfiable, then*

$$\mathsf{Enc}(\varphi, 0, 1^\lambda) \approx_c \mathsf{Enc}(\varphi, 1, 1^\lambda).$$

Note that this definition does not necessarily say that you need a witness to decrypt b, it only says if you can decrypt b, then φ is satisfiable. In the problem set, we will explore this more.

Theorem 13. *If* iO *exists, then witness encryption exists.*

Proof. The construction is:

- $\mathsf{Enc}(\varphi, b, 1^\lambda; r) = iO(x \mapsto b \wedge \mathbb{1}[\varphi(x) = 1], 1^\lambda; r)$
- $\mathsf{Dec}(C, w) = C(w)$.

It is easy to see that functionality holds. It remains to show security. If φ is unsatisfiable, we have that

$$\begin{aligned}\mathsf{Enc}(\varphi, b, 1^\lambda) &= iO(x \mapsto b \wedge \mathbb{1}[\varphi(x) = 1], 1^\lambda) \\ &\approx_c iO(x \mapsto b \wedge 0, 1^\lambda) \\ &\approx_c iO(x \mapsto 0, 1^\lambda),\end{aligned}$$

so we have that $\mathsf{Enc}(\varphi, 0, 1^\lambda) \approx_c \mathsf{Enc}(\varphi, 1, 1^\lambda)$, as desired. □

A few remarks about this proof:

- We did not need to assume that **P** $\neq$ **NP** for this proof. Indeed, witness encryption is possible if **P** = **NP**.
- We only really needed the security guarentee of iO to hold for unsatisfiable circuits.
- You could ask if we can construct a stronger variant of witness encryption where one can decrypt if and only if you know a witness. In a certain sense, this scheme actually has this property: if a scheme with the property exists, then a (slight modification) of this scheme also has this property. You will explore this in the problem set. It is related to a phenomena where iO is "best possible obfuscation."

2.3 Public Key Encryption

Building on the witness encryption construction, we can construct public-key encryption.

Theorem 14 (GGSW '13)**.** *If iO exists and no PPT algorithm solves SAT infinitely often, then public-key encryption exists.*

Proof. Let $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a pseudorandom generator (which follows from the existence of one-way functions which follows from iO and the hardness of SAT). Let WE and WD be witness encryption and witness decryption algorithms respectively.

The construction is:

- $Gen(1^\lambda)$ samples a secret key $sk \leftarrow \{0,1\}^\lambda$, computes $r = G(sk)$ and sets the public key $pk = G(sk)$
- $\mathsf{Enc}(pk, b)$ outputs $WE(\varphi_{pk}, b, 1^\lambda)$, where

$$\varphi(x) = \mathbb{1}[G(x) = pk].$$

- $\mathsf{Dec}(c, sk) = WD(c, sk)$.

Functionality is clear. Now we need to show security. Specifically, we need to show that

$$(pk, \mathsf{Enc}(pk, 0)) \approx_c (pk, \mathsf{Enc}(pk, 1)).$$

This follows from the following hybrid argument.

$$\begin{aligned}(pk, \mathsf{Enc}(pk, b)) &= (pk = G(s), WE(\varphi_{pk}, b))_{s \leftarrow \{0,1\}^\lambda} \\ &\approx_c (pk = r, WE(\varphi_{pk}, b))_{r \leftarrow \{0,1\}^{2\lambda}} \\ &\approx_c (pk = r, WE(\varphi_{pk}, 0))_{r \leftarrow \{0,1\}^{2\lambda}}\end{aligned}$$

the first line is by definition, the second is by PRG security, the third line is by WE security since with $1 - \text{negl}(\lambda)$ probability φ_{pk} is unsatisfiable (because a random string $r \leftarrow \{0,1\}^{2\lambda}$ is in the range of $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ with probability $2^{-\lambda}$). The last hybrid is independent of b, completing the proof.

□

Remarks

- Try revisiting the intuition that "iO looks useless because it only shows indistinguishability for functionally identical circuits." How did this proof get around this?
- This proof does not look like the Diffie-Hellman way of getting PKE from obfuscation. In the next class, we will see a different PKE scheme that looks more like the Diffie-Hellman approach.

2.4 Witness PRF (or "Designated Verifier SNARGs")

Suppose Alice wants to prove to Bob that a public n-input formula φ is satisfiable with as little communication between them as possible. Alice could send a witness w over to Bob, but that requires n-bits. We would like to be more succinct? Ideally polylog(n) bits.

One idea to solve this is to use witness encryption. Bob could send Alice several witness encryptions under φ and then if Alice decrypts all of them, Bob is convinced that φ is satisfiable.

The problem is that the witness encryption scheme we have constructed has long ciphertexts (way longer than n bits), so this is worse than just sending over the witness.

So, we need a different apprach. One attempt at doing this involves a witness PRF. Witness PRFs (which we define in a few sentences) will allow Alice to convince Bob that φ is satisfiable with little communication, with the caveat that one allows a first setup phase, where Bob sends Alice a single long message "once and for all" before anyone knows φ.

A witness PRF is a PRF with two types of keys, a secret key that allows evaluation on all points, and a public key that allows evaluation only on points that correspond to satisfiable formulas (that one produces a witness to).

Definition 15 (Zhandry '14)**.** *A witness PRF consists of a pseudorandom function family PRF_k and two PPT algorithms $PublicGen(\text{key } k) = pk$, and $\mathsf{Eval}(\text{public key } pk, \text{formula } \varphi, \text{ witness } w)$ with the following two guarantees:*

- ***Functionality:*** *For all formulas φ with $\varphi(w) = 1$, we have*

$$\Pr_{\substack{k \leftarrow \{0,1\}^\lambda \\ pk \leftarrow PublicGen(k)}} [Eval(pk, \varphi, w) = PRF_k(\varphi)] = 1$$

- ***Security:*** *For every unsatisfiable formula $\varphi^\star$, we have that when $k \leftarrow \{0,1\}^\lambda$ and $pk \leftarrow PublicGen(k)$ and $z \leftarrow \{0,1\}^\lambda$*

$$(pk, \varphi^\star, PRF_k(\varphi^\star)) \approx_c (pk, \varphi^\star, z)$$

Assuming witness PRFs exist, Bob can send Alice pk, and then Alice can convince Bob that φ is satisfiable by sending $(\varphi, PRF_k(\varphi))$.

Theorem 16 (Essentially Sahai-Waters '13)**.** *If iO and PRFs exists, then a witness PRF exists.*

Fake Proof. A natural first attempt: Let PRF_k be an arbitrary PRF. Let

- $PublicGen(k)$ outputs $iO\left((\varphi, w) \mapsto \begin{cases} PRF_k(\varphi), & \text{if } \varphi(w) = 1 \\ \bot, & \text{otherwise.} \end{cases} \right)$
- $\mathsf{Eval}(pk, \varphi, w) = pk(\varphi, w)$.

Functionality is clear. It only remains to argue for security. If $\varphi^\star$ is unsatisfiable, then we have that

$$iO\left((\varphi, w) \mapsto \begin{cases} PRF_k(\varphi), & \text{if } \varphi(w) = 1 \\ \perp, & \text{otherwise.} \end{cases}\right) \approx_c iO\left((\varphi, w) \mapsto \begin{cases} PRF_k(\varphi), & \text{if } \varphi(w) = 1 \text{ and } \varphi \neq \varphi^\star \\ \perp, & \text{otherwise.} \end{cases}\right)$$

So we eliminated the circuit's dependence on $PRF_k(\varphi^\star)$, so we are done, right? Not so fast, the key k is still in the code, which determines the value $PRF_k(\varphi^\star)$, so we are not done. □

What we need to show is that there is some code which enables us to attain the same functionality as above without the code revealing anything about the value of $PRF_k(\varphi^\star)$.

To do this, we strengthen the notion of a PRF to a "puncturable PRF." This is a great example of how one usually works with iO. Oftentimes, one needs to make objects "iO friendly" in order to prove intuitive looking statements.

Definition 17. *A puncturable PRF consists of a pseudrandom function family PRF_k and two randomized polynomial-time algorithms $PuncGen$(key k, puncture point $x^\star$) $= pk$, and $PuncEval$(punctured key pk, point x) with the following two properties:*

- ***Evaluates on Non-punctured Points:*** *For all $x \neq x^\star$ when $k \leftarrow \{0,1\}^\lambda$ and $pk \leftarrow PuncGen(k)$*

$$\Pr[PuncEval(pk, x) = PRF_k(x)] = 1$$

- ***Hides Punctured Output:*** *For all $x \neq x^\star$ when $k \leftarrow \{0,1\}^\lambda$ and $pk \leftarrow PuncGen(k)$ and z is uniformly random*

$$(pk, x^\star, PRF_k(x^\star)) \approx_c (pk, x^\star, z)$$

Using a punctured PRF, one can complete the proof of security of the Witness PRF. Furthermore, punctured PRFs can be obtained generically from PRFs.

Theorem 18. *If PRFs exist, then puncturable PRFs exist.*

Proof Idea. For those students who remember the GGM construction of PRFs from PRGs: Recall the GGM tree. Instead of giving the key at the top. Give the key at the top of every maximal subtree that does not include $x^\star$. □