CS355 Lecture 19: Program Obfuscation

Lecture notes by David Wu from CS355 2018

<u>So far in the course</u>: Many generations of cryptography:

- 1st generation: symmetric primitives [OWFs, PRGs, PRFs, PRPs]
- 2nd generation: public-key encryption [PKE, key exchange]
- 3rd generation: pairings [IBE, short signatures]
- 4th generation: lattices [FHE, post-quantum key exchange]
- 5th generation: multilinear maps [program obfuscation] ⟵ today's lecture

<u>Program obfuscation</u>: Can we hide <u>secrets</u> inside a piece of code [e.g., obfuscated program preserves functionality, but hides everything about implementation other than program's input/output behavior]

Why might we want this?

<u>Application 1</u>: symmetric encryption ⟹ public-key encryption

$sk \leftarrow KeyGen(1^\lambda)$

$pk \leftarrow Obf(Encrypt(k, \cdot))$ obfuscate the encryption function [rely on obfuscation scheme to argue that key is hidden]

<u>Observe</u>: no algebraic assumptions needed (other than existence of this obfuscation scheme!

<u>Application 2</u>: optimally-short signatures from obfusated PRF $(F: \mathcal{K} \times \mathcal{M} \to \{0,1\}^\lambda)$

$k \xleftarrow{R} \mathcal{K}$

$sk: k$ define function $f_k(m, \sigma) = 1$ if $F(k, m) = \sigma$ and 0 otherwise

$vk: Obf(f_k)$

$Sign(sk, m)$: output $\sigma = PRF(k, m)$ [obfuscation scheme hides the PRF key k, so cannot forge without guessing PRF value]

$Verify(vk, m, \sigma)$: Run $Obf(f_k)$ on (m, σ)

<u>Application 3</u>: optimally-short NIZKs (for language $\mathcal{L}_C = \{x \in \{0,1\}^n : \exists w \in \{0,1\}^m \; C(x,w) = 1\}$) from obfuscated PRF:

$k \xleftarrow{R} \mathcal{K}$ define function $f_k(x, w) = F(k, x)$ if $C(x, w) = 1$ and $\perp$ otherwise

define function $g_k(x, \pi) = 1$ if $F(k, x) = \pi$ and 0 otherwise

$Setup(1^\lambda)$: Output $Obf(f_k)$ and $Obf(g_k)$ as common reference string $\sigma = (Obf(f_k), Obf(g_k))$

$Prove(\sigma, x, w)$: Output $\pi = Obf(f_k)(x, w)$

$Verify(\sigma, x, \pi)$: Output $Obf(g_k)(x, \pi)$

} rely on obfuscated program to hide the PRF key k in f_k and g_k

Seems too "easy"... Turns out this notion of obfuscation (hide everything except input/output behavior) is impossible :((called virtual black box (VBB) security)

Weaker notion of obfuscation proposed: indistinguishability obfuscation

"Obfuscation of two programs that compute identical functions are indistinguishable"

Definition. An indistinguishability obfuscation (iO) scheme for general circuits (on n-bit inputs) is an efficient algorithm iO with the following properties:

Functionality-preserving: For all Boolean circuits $C: \{0,1\}^n \to \{0,1\}$, and all inputs $x \in \{0,1\}^n$:

$$[iO(1^\lambda, C)](x) = C(x)$$

Indistinguishability: For all Boolean circuits $C_1, C_2 : \{0,1\}^n \to \{0,1\}$ where $C_1(x) = C_2(x)$ for all $x \in \{0,1\}^n$ and $|C_1| = |C_2|$,

$$iO(1^\lambda, C_1) \overset{c}{\approx} iO(1^\lambda, C_2)$$

Seems very weak... unclear what it hides about the program, if anything at all!

[But in conjunction with OWFs, we can actually get almost all of crypto — one of the most powerful ("crypto-complete") cryptographic primitives!]

How do we use iO? [Sahai-Waters punctured programming paradigm]

Key building block: puncturable pseudorandom functions

Definition. A PRF $F: \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a puncturable PRF if there exists a puncturing algorithm with the following properties:

Puncture$(k, x^*) \to k_{x^*}$: puncturing algorithm takes as input a PRF key k and a point x^* and produces punctured key k_{x^*}

Correctness: $\forall x \neq x^* : F(k,x) = F(k_{x^*}, x)$

"punctured key can evaluate at all $x \neq x^*$"

[Somewhat overloading notation: evaluation using the punctured key could be handled using a different algorithm]

Security: $\{k \xleftarrow{R} \mathcal{K} : (k_{x^*}, F(k,x^*))\} \overset{c}{\approx} \{y \xleftarrow{R} \mathcal{Y} : (k_{x^*}, y)\}$

"value at punctured point looks random even given punctured key"

Puncturable PRFs can be constructed from OWFs [via Goldreich-Goldwasser-Micali]

The magic of iO: iO + puncturable PRFs $\Longrightarrow$ all of crypto [with a couple exceptions]

Short signatures by obfuscating a PRF (and OWF):

Setup$(1^\lambda) \to (sk, vk)$: $k \xleftarrow{R} \mathcal{K}$

$sk = k$

$vk = iO(C_k)$

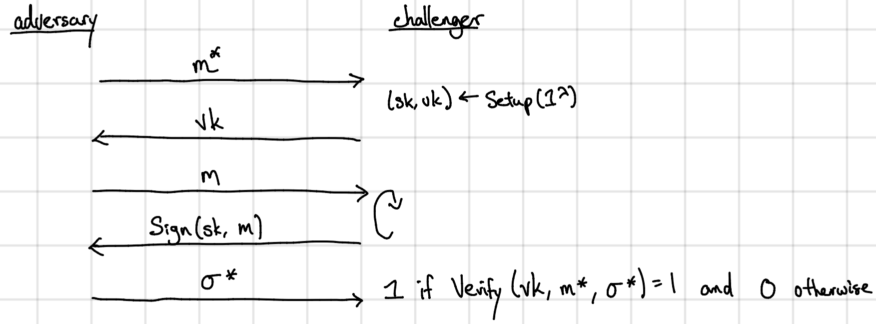
let $C_k(m, \sigma)$ be circuit that outputs 1 if $f(F(k,m)) = f(\sigma)$ [f: OWF; F: puncturable PRF]

Sign(sk, m): Output $F(k,m)$ — Assume PRF output has λ bits

Verify(vk, m, σ): Output $[iO(C_k)](m, \sigma)$

Correctness is immediate by correctness of iO.

We will show "selective unforgeability" where adversary commits to the message it will forge on at the beginning of the security game:



Security of signature scheme:

Hyb_0: real signature game between adversary and challenger

verification program: $C_k^{(0)}(m, \sigma)$: output 1 if $f(F(k,m)) = f(\sigma)$ otherwise, output 0

these two circuits compute identical functionalities [by correctness of puncturable PRF] ↳ hybrids indistinguishable by iO

Hyb_1: verification program replaced by the following:

verification program: $C_{k_{m^*}}^{(1)}(m, \sigma)$: if $m = m^*$: output 1 if $f(F(k, m^*)) = f(\sigma)$ and 0 otherwise (hard-wired value)
if $m \neq m^*$: output 1 if $f(F(k_{m^*}, m)) = f(\sigma)$ and 0 otherwise

(PRF key punctured at m^*)

by puncturing security of F, value of $F(k, m^*)$ looks random given k_{m^*}

Hyb_2: challenger samples random $y \xleftarrow{R} \mathcal{Y}$ and constructs verification program as follows:

verification program: $C_{k_{m^*}}^{(2)}(m, \sigma)$: if $m = m^*$: output 1 if $f(y) = f(\sigma)$ and 0 otherwise
if $m \neq m^*$: output 1 if $f(F(k_{m^*}, m)) = f(\sigma)$ and 0 otherwise

↳ hybrids indistinguishable by puncturing security

Probability that adversary forges in Hyb_2 is negligible since such an adversary can invert the OWF (namely, a forgery on m^* satisfies $f(\sigma) = f(y)$ where y is sampled uniformly at random from $\mathcal{Y}$ $\Rightarrow$ signature forger breaks one-wayness of f)

Advantage of $\mathcal{A}$ in Hyb_2 is negligible $\Rightarrow$ $\text{Hyb}_0, \text{Hyb}_1, \text{Hyb}_2$ are computationally indistinguishable experiments $\Rightarrow$ Advantage of $\mathcal{A}$ in Hyb_0 is negligible

Summary: Signature scheme where signatures is just PRF output (yields λ-bit signatures with λ-bit security provided that underlying primitives provide exponential security

Open Problem: λ-bit signatures without iO?

Secret-key encryption $\Rightarrow$ public-key encryption from iO:

KeyGen(1^λ): sample PRF key k, let $G: \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG

define function $C_k(r,m): (G(r), F(k, G(r)) \oplus m)$

output $sk = k$ and $pk = iO(C_k)$

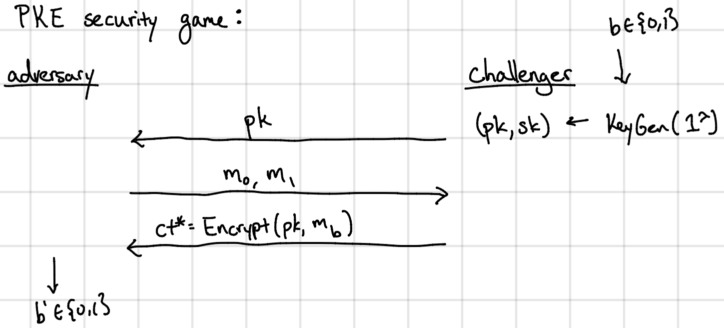
Encrypt(pk, m): $r \xleftarrow{R} \{0,1\}^\lambda$

output $[iO(C_k)](r,m)$

Decrypt(sk, ct): output $F(k, ct_0) \oplus ct_1$

Correctness is immediate.

Security: recall PKE security game:



Secure if $\left| \Pr[b'=1 \mid b=0] - \Pr[b'=1 \mid b=1] \right| = \text{negl}(\lambda)$.

Proceed by hybrid argument:

Hyb_0: semantic security game between challenger and adversary where adversary encrypts m_0

specifically, challenger does the following:

1. Sample $k \xleftarrow{R} \mathcal{K}$ and $r^* \xleftarrow{R} \{0,1\}^\lambda$
2. Construct pk as obfuscation of following program:

$$\boxed{C_k(m,r): \text{Output } (G(r), F(k, G(r)) \oplus m)}$$

3. When challenger submits messages (m_0, m_1): return $ct \leftarrow (G(r^*), F(k, G(r^*)) \oplus m_0)$

(PRG security)

Hyb_1: replace $G(r^*)$ with uniformly random string $y \xleftarrow{R} \{0,1\}^{2\lambda}$ [in this case, $ct = (y, F(k,y) \oplus m_0)$]

(iO security: since $y \xleftarrow{R} \{0,1\}^{2\lambda}$, $\Pr[\exists x \in \{0,1\}^\lambda : G(x) = y] \le \frac{2^\lambda}{2^{2\lambda}} = \frac{1}{2^\lambda}$ $\Rightarrow$ programs are identical with prob. $1 - \frac{1}{2^\lambda}$)

Hyb_2: replace public key with obfuscation of following program:

$$\boxed{C_{k_y}(m,r): \text{Output } (G(r), F(k_y, G(r)) \oplus m)}$$

$\leftarrow$ key punctured at y

note: ciphertext is still $ct = (y, F(k,y) \oplus m_0)$

(puncturing security of F (the key k_y is punctured at the point y))

Hyb_3: replace ciphertext with $ct = (y, z)$ where $z \xleftarrow{R} \{0,1\}^{|m_0|}$

$\text{Hyb}_4 - \text{Hyb}_6$: unroll the above analysis (with message m_1)

High-level idea in punctured programming: there is some secret information that the adversary needs in order to break security (e.g., the value of the PRF at a particular point) and using puncturing + iO, we can remove that information from the view of the adversary $\Rightarrow$ yields secure cryptographic instantiations

With punctured programming, we can realize applications of VBB obfuscation from iO (which plausibly exists)

In fact, we can do more: can leverage iO + OWFs to obtain functional encryption (FE):

- Ciphertexts are associated with messages m
- Keys are associated with functions f

$\left.\begin{array}{l} ct_m \\ sk_f \end{array}\right\} \Rightarrow f(m)$ [and nothing more about m]

Generalizes notions like public-key encryption (only supports identity function in decryption key)
identity-based encryption (encrypt to (id, m) and functions associated with id' — outputs m if $id = id'$)
attribute-based encryption, predicate encryption, etc. — general umbrella for encryption

If iO is "crypto-complete," what next?

<u>Challenges</u>: 1. Realizing iO from standard assumptions (e.g., DDH, pairings, LWE)

↳ current instantiations rely on multilinear maps, which have been subject to numerous attacks in the last few years (lots of skepticism over their security) — while there exist iO candidates over concrete multilinear map instantiations that are not known to be broken, status is very tenuous

"Cryptographers seldom sleep well." — Joe Kilian (attributed to Silvio Micali)

2. Concrete efficiency of iO: all constructions today are extremely theoretical (and nowhere close to practical)

↳ To obfuscate a PRF like AES, constructions need to publish $\geq 2^{100}$ encodings or support $\geq 2^{100}$ levels of multilinearity [some newer constructions can make do with constant-degree multilinearity (in fact a trilinear map suffices, but these require non-black-box use of the multilinear map, which is also extremely costly]

↳ Solution is <u>not</u> better engineering — need fundamentally better constructions

In spite of the existing limitations, iO informs us about the landscape of cryptography and highlights what is <u>feasible</u>. Techniques from obfuscation and inspired by obfuscation has inspired many new techniques and constructions in the last few years (e.g., round-optimal MPC)

<u>Exciting questions</u>: Now that we have iO, what is the next generation of cryptography?
Can we realize iO from LWE?
(existing constructions of multilinear maps all rely on lattices, but problems not reducible to LWE)