Logistics: HW5 due Friday (6/8)

- No office hours next week – good luck on finals!
- Please fill out course evaluations and give us feedback!
- Come talk to us if you are interested in research in crypto / want to see what’s next...

So far in the course: Many generations of cryptography:

1st generation: symmetric primitives (e.g., Feistel, PRGs, PRFs, PRPs)
2nd generation: public-key encryption (e.g., PKC, key exchange)
3rd generation: pairings (e.g., IBE, short signatures)
4th generation: lattices (e.g., FHE, post-quantum key exchange)
5th generation: multilinear maps (e.g., program obfuscation)

Program obfuscation: Can we hide secrets 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?

**Application 1: symmetric encryption $$\rightarrow$$ public-key encryption**

$$sk \leftarrow \text{KeyGen}(1^n)$$

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

Observe: no algebraic assumptions needed (other than existence of this obfuscation scheme!)

**Application 2: optimally-short signatures from obfuscated PRF**

$$F : \{0, 1\}^n \rightarrow \{0, 1\}^k$$

$$k \leftarrow \text{R}$$

$$sk : k$$

$$vk : \text{Obf}(f_k)$$

$$\text{Sign}(sk, m) : \text{output } \sigma = \text{PRF}(k, m)$$

[obfuscation scheme hides the PRF key k, so cannot forge without guessing PRF value]

$$\text{Verify}(vk, m, \sigma) : \text{Run } \text{Obf}(f_k) \text{ on } (m, \sigma)$$

**Application 3: optimally-short NIZKs from obfuscated PRF**

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

$$\text{Prove}((\sigma, x, w)) : \text{Output } \pi = \text{Obf}(f_k)(x, w)$$ [rely on obfuscated program to hide the PRF key k in f_k and g_k]

$$\text{Verify}(\sigma, x, \pi) : \text{Output } \text{Obf}(g_k)(x, \pi) \text{ called virtual black box (VBB) security}$$

Seems too easy... Turns out this notion of obfuscation (hide everything except input/output behavior) is impossible...
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 \( x \in \{0,1\}^n \):

\[
iO(I^2, C)(x) = C(x)
\]

**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 \( |C_1| = |C_2| \),

\[
iO(I^2, C_1) \approx iO(I^2, C_2)
\]

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 cryptographic primitives!]

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

**Key building block:** puncturable pseudorandom functions

**Definition.** A PRF \( F : k \times X \to Y \) 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: \( \forall k \neq x^* : F(k, x^*) = F(k_x^*, x) \) [Somewhat overloading notation: evaluation using the punctured key could be handled using a different algorithm]

Security: \( \{ k \neq x^* | (k, x^*), F(k, x^*) \} \approx \{ y \neq x^* | (k, y), F(k, y) \} \) [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 \( \implies \) all of crypto [with a couple exceptions]

**Short signatures by obfuscating a PRF (and OWF):**

\[\text{Setup} (1^n) \to (sk, vk) : k \in \mathbb{R} \quad \text{let } C_k (m, \sigma) \text{ be circuit that outputs } 1 \text{ if } f(F(k, m)) = f(\sigma)\]

\[sk = k, \quad vk = iO(C_k)\]

**Sign (sk, m):** Output \( F(k, m) \) Assume PRF output has \( \lambda \) bits

**Verify (vk, m, \sigma):** Output \( I0(C_k)(m, \sigma) \)

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:

adversary

\[ m^* \]

\[ \leftarrow \] uk

\[ \rightarrow (sk, uk) \leftarrow \text{Setup}(1^n) \]

challenger

\[ m \]

\[ \leftarrow \text{Sign}(sk, m) \]

\[ \rightarrow \sigma^* \]

1 if \( \text{Verify}(uk, m^*, \sigma^*) = 1 \) and 0 otherwise

Security of signature scheme:

Hyb₀: real signature game between adversary and challenger

Verification program:

\[ C^0(m, \sigma): \begin{cases} \text{output 1 if } f(F(k, m)) = f(\sigma) \\ \text{otherwise, output 0} \end{cases} \]

Hyb₁: verification program replaced by the following:

Verification program:

\[ C^1(m, \sigma): \begin{cases} \text{output 1 if } f(F(k, m)) = f(\sigma) \\ \text{otherwise, output 0} \end{cases} \]

Hyb₂: challenger samples random \( y \in Y \) and constructs verification program as follows:

Verification program:

\[ C^2(m, \sigma): \begin{cases} \text{output 1 if } f(y) = f(\sigma) \\ \text{otherwise, output 0} \end{cases} \]

Hyb₃: verification program replaces verification program by the following:

Verification program:

\[ C^3(m, \sigma): \begin{cases} \text{output 1 if } f(F(k, m)) = f(\sigma) \\ \text{otherwise, output 0} \end{cases} \]

Probability that adversary forges in Hyb₂ 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 \( Y \) ⇒ signature forger breaks one-in-ness of \( f \))

Advantage of \( A \) in Hyb₁ is negligible ⇒ Hyb₀, Hyb₁, Hyb₂ are computationally indistinguishable experiments ⇒ Advantage of \( A \) in Hyb₀ is negligible.

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

Open Problem: \( \lambda \)-bit signatures without \( iO \)?
Secret-key encryption $\Rightarrow$ public-key encryption from iO:

KeyGen(1^\ast): Sample PRF key k, let $G: \{0,1\}^\ast \rightarrow \{0,1\}^{2k}$ 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): $r \leftarrow \{0,1\}^\ast$
- output $[iO(C_k)](r, m)$

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

Correctness is immediate.

Security: Recall PKE security game:

- adversary
- challenges

\[ \left\langle \begin{array}{c}
\downarrow \\
(pk) \leftarrow \text{KeyGen(1^\ast)} \\
(m_0, m_1) \rightarrow \\
ct^* \leftarrow \text{Encrypt}(pk, m_i) \\
\downarrow \\
b \leftarrow \{0,1\}^\ast
\end{array} \right\rangle \]

Secure if $|\Pr[b=1|b=0] - \Pr[b=1|b=1]| = negl(\lambda)$.

Proceed by hybrid argument:

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

1. Sample $k \leftarrow K$ and $r \leftarrow \{0,1\}^\ast$
2. Construct $pk$ as obfuscation of following program:
   
   $C_k(m, r): \text{Output } (G(r), F(k, G(r) \oplus m))$

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

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

Hyb_2: replace public key with obfuscation of following program:

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

Note: ciphertext is still $ct = (y, F(k, y) \oplus m)$ but $k_y$ punctured at $y$

Hyb_3: replace ciphertext with $ct = (y, z)$ where $z \leftarrow \{0,1\}^{2k}$

Hyb_4: $Hyb_3$: 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 $m$\text{\hspace{1cm}}\text{\textcolor{red}{ct}_{\text{\textcolor{red}{sk}}} \Rightarrow f(m)}$ [and nothing more about $m$]
- Keys are associated with functions $f$

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?

Challenges:
1. Realising 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 maps 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 $\geq 2^n$ encodings or support $\geq 2^{\text{100}}$ 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 not better engineering - need fundamentally better constructions

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

Exciting question: 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)