COS 597C: Recent Developments in Program Obfuscation — Princeton University
Lecture 3 (September 22, 16)
Lecturer: Mark Zhandry — Scribe: Fermi Ma

Notes for Lecture 3

1 Last Time

Last time we saw that VBB obfuscation is impossible [BGI+01]. The issue is that for some programs, every possible implementation will leak information about it. To recover a meaningful formulation of obfuscation, we defined indistinguishability obfuscation (iO), where no adversary can determine which of two programs was obfuscated if both programs have the same functionality. In this lecture, we'll see that iO is still a pretty powerful definition.

2 The "As Good As It Gets" Theorem

Why do we choose iO as a definition, and not try to devise a different definition? After all, iO appears to lose the intuition about preventing reverse engineering, and requires a second equivalent program to even have any meaning. Here, motivate our use of iO, by demonstrating that iO is in some sense the strongest possible obfuscator.

Theorem 1 ([GR07], Informal)**.** *iO is the best possible obfuscator.*

For any program that is VBB obfuscatable, iO obfuscation will achieve VBB obfuscation. Note that this holds true for any reasonable definition of obfuscation, but here we explain why it holds for the VBB definition.

To see why this holds, suppose that $O(P)$ is a VBB obfuscator. It is a simple exercise to show that any further compilation of this program preserves the VBB obfuscation. So $iO(O(P))$, the iO obfuscation of $O(P)$, is still VBB obfuscated. There is a slight technicality here that obfuscated circuits are only indistinguishable if they are the same size, so let $\overline{P}$ be P, but padded to be of length $|O(P)|$. We can finish the argument here, since the iO assumption tells us that $iO(O(P)) \approx_c iO(\overline{P})$, where $A \approx_c B$ means that A is computationally indistinguishable from B given polynomial time.

Thus, if a program can be obfuscated, iO will obfuscate it.

3 Recipe for Obtaining Cryptographic Applicaitons from iO

Now we will see how to use iO for applications.

Our informal recipe will be: 1) Start with a VBB-based method of building the desired application. 2) Tweak the construction by using additional tools so that iO suffices.

4 iO Application: Digital Signatures

Consider the following scenario: Alice and Bob want to communicate over some channel. Previously, we talked about the encryption setting where an adversary can eavesdrop on the channel. In this setting, the adversary is more powerful; he can intercept messages and change them. We want Bob to be able to detect such behavior, and reject any modified messages.

The solution is a digital signature scheme. Alice generates a public verification key vk and gives it to Bob, and generates for herself a private signing key sk. When Alice sends a message m, she uses her signing key sk to generate a signature σ, and forwards this along with m. If Bob receives $m' \neq m$, we want him to reject.

A signature scheme consists of three algorithms:

- $\mathsf{Gen}(\lambda)$ takes the security parameter λ as input and outputs $(\mathsf{sk}, \mathsf{vk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$ takes the secret key sk and the message m as input and outputs σ.
- $\mathsf{Ver}(\mathsf{vk}, m, \sigma)$ takes the verification key vk, the message m, and σ and outputs accept or reject.

The obvious correctness requirement is that if we start with a message m and generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, then $\mathsf{Ver}(\mathsf{vk}, m, \sigma)$ accepts.

We can define security for this in a number of ways. Today we focus on selective CMA-security (chosen message attack), which we will achieve via obfuscation. We say that a scheme is selectively CMA-secure if no polynomial time adversary can attain a non-negligible advantage in the following game.

The adversary first commits to a message m^* and sends it to the challenger. The challenger generates the secret key/verification key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(\lambda)$ and sends vk back to the adversary. At this point, the adversary may make polynomially many queries message queries to the challenger, where he sends a message $m \neq m^*$ and receives $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$. The adversary generates a new signature σ^* and wins if $\mathsf{Ver}(\mathsf{vk}, m^*, \sigma^*) = \textit{accept}$. The adversary's advantage is his probability of winning.

Intuitively, the adversary is trying to generate a signature for a fake message and gets to see signatures for a number of other chosen messages beforehand.

We follow the informal recipe from above to construct a selectively CMA-secure signature scheme. The iO construction and proof are due to [SW13], who introduce a technique known as "punctured programming."

The first step is to give a VBB-based construction. This construction will use a pseudorandom function $PRF(k,x) \to y$. Recall that the security requirement is that if k is unknown and randomly chosen, then $A^{PRF(k,\cdot)} \approx_c A^{H(\cdot)}$, where $H(\cdot)$ is a truly random function. The construction is as follows.

- $\mathsf{Gen}(\lambda)$ samples $\mathsf{sk} = k \leftarrow \{0,1\}^\lambda$. Define $C_k(m,\sigma)$ to be the circuit that outputs the 0/1 value $PRF_k(m) == \sigma$. Then we set vk to be the VBB obfuscation of C_k, $\hat{C}$.
- $\mathsf{Sign}(k,m) = PRF(k,m)$
- $\mathsf{Ver}(\hat{C},m,\sigma) = \hat{C}(m,\sigma)$

With proper VBB obfuscation, giving the adversary vk is no worse than giving the adversary oracle access to C_k. We can think of changing the selective CMA-security experiment for the above scheme. Instead of giving the adversary vk at one step, we can replace that step with polynomially many steps where the adversary can query an oracle on m, σ, and receive an accept/reject. We allow the adversary to interleave these queries with the Sign queries.

We claim that this oracle doesn't actually help the adversary. If the adversary queries on m, σ, it should be the case that the adversary got σ from making a sign query on m. But if that is the case, then the adversary already knows the query is accepted, so the oracle does not give any new information. If the adversary did not get σ from a sign query, then with overwhelming probability the query will be rejected. So the adversary can just say the query is rejected without needing an oracle query. So the oracle is unnecessary.

We now introduce a new tool called a puncturable PRF, which will have a different security requirement from a regular PRF. The puncturable PRF is still a function $PRF(k,x) \leftarrow y$, but the difference is that there is an algorithm $k\{x^*\} \leftarrow \mathsf{Puncture}(k,x^*)$ that outputs a punctured key. $PRF(k\{x^*\},x)$ is equal to $PRF(k,x)$ if $x \neq x^*$, and equals $\perp$ if $x = x^*$.

So far this isn't very interesting, since we could have just defined a regular PRF to work like this. But that changes with the following security definition. For all x^*, we require that

$$(k\{x^*\}, PRF(k,x^*)) \approx_c (k\{x^*\}, R),$$

where R is a truly random value. In words, this means the adversary is given the punctured PRF and can compute the value at every point except for x^*. Even with this information, it cannot distinguish the value of the PRF at x^* from a random value.

We go back to the construction we gave above and make the following 3 changes. 1) We replace the general PRF with a puncturable PRF. 2) We change the circuit C_k to have a one way function F applied to each side. 3) We change the obfuscator to be an indistinguishability obfuscator.

- $\mathsf{Gen}(\lambda)$ samples $\mathsf{sk} = k \leftarrow \{0,1\}^\lambda$. Define $C_k(m,\sigma) = F(PRF_k(m)) == F(\sigma)$ where F is a one way function. Then we set $\mathsf{vk} = O(C_k) = \hat{C}$.
- $\mathsf{Sign}(k, m) = PRF(k, m)$
- $\mathsf{Ver}(\hat{C}, m, \sigma) = \hat{C}(m, \sigma)$

We'll show that this is selectively CMA-secure via a sequence of hybrids. We'll give a sequence of indistinguishable hybrids until we get to a situation where the adversary cannot win.

Hybrid 0 is the original selective CMA game.

Hybrid 1 is the same as Hybrid 0, but we change $\hat{C}$ to be $\hat{C} = O(C'_{k\{m^*\},y})$ where

$$C'_{k\{m^*\},y}(m,\sigma) = \begin{cases} F(PRF(k\{m^*\}, m)) == F(\sigma) & \text{if } m \neq m^* \\ y == F(\sigma) & \text{if } m = m^* \end{cases}$$

y is just going to be $F(PRF(k, m^*))$. Whether or not $m = m^*$, this circuit computes exactly the same thing it did before. Thus, $C'_{k\{m^*\},y}(m,\sigma) \equiv C_k$. So by iO, $H_0 \approx_c H_1$.

Hybrid 2 is the same as Hybrid 1, except $y = F(r)$ where r is random. Let's assume that adversary A can distinguish between H_1 and H_2. We can use this to come up with a B that breaks punctured PRF security. B runs A as a subroutine. A produces m^* and gives this to B. B gives m^* to his challenger. His challenger sends back $(k\{m^*\}, r)$. r is either the PRF value at m^* or it is random. B now gives $iO(C'_{m^*,k\{m^*\},F(r)})$ to A. When A makes a query on $m \neq m^*$, B responds with $\sigma = PRF(k\{m^*\}, m)$. Lastly, A outputs σ^*, and B checks that $F(\sigma^*) = F(r)$. Now if r is the value of the PRF, then the view of the adversary is Hybrid 1. If r is a uniformly random value, then the view of the adversary is Hybrid 2. So B can obtain a non-negligible advantage in the punctured PRF experiment, which is a contradiction. So no adversary A can distinguish between H_1 and H_2, and thus we have $H_1 \approx_c H_2$.

Now we show that if an adversary A can win in Hybrid 2, we can use that adversary to break one way function security. We will construct an inverting function $D(y)$, where y will be $F(r)$ for a random r. First, we have adversary A send m^*. D will generate a random PRF key k, will puncture at m^* and then send back $iO(C'_{m^*,k\{m^*\},y})$. Then A makes queries on $m \neq m^*$ and D will answer correctly. Eventually the adversary guesses σ^*. Observe that D is simulating the view of Hybrid 2. If the adversary succeeds and produces a valid σ^*, that means the adversary has come up with something that satisfies $F(\sigma^*) = F(r) = y$.

References

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (Im)possibility of obfuscating programs. In *Advances in Cryptology — CRYPTO 2001*, number Im, 2001. 1

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007. 1

[SW13] Amit Sahai and Brent Waters. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More. 2013. 3