COS 597C: Recent Developments in Program Obfuscation — Princeton University
Lecture 22 (12/8/16)
Lecturer: Mark Zhandry — Scribe: Fermi Ma

Notes for Lecture 22

1 Difficulties in Building iO

Today we'll talk about difficulties in building iO. We'll also discuss difficulties / impossibilities in other obfuscation definitions.

First we'll argue that iO is not really one assumption, but rather a bunch of assumptions. Remember what iO says: $\forall C_0 \equiv C_1$, we have $\mathsf{iO}(C_0) \approx_c \mathsf{iO}(C_1)$.

Normal assumptions in cryptography say things of the form: one distribution is computationally indistinguishable from another distribution. iO has an assumption of this form, but for any pair of circuits. It seems like we could imagine this assumption holding true for some pair of circuits but not others.

To see that this is really a difference, think about how we would gain confidence in a PRG-type assumption. We could try extensive cryptanalysis. We could post challenges online.

We can't gain confidence in the iO assumption in this fashion, since we would have to post challenges for exponentially many pairs of C_0, C_1.

The goal is then to base iO $\forall C_0 \equiv C_1$ on some simple assumption.

Ideally we'll have some reduction that gets some sample s from either $Dist_1$ or $Dist_2$. It's additionally going to get C_0, C_1, and then it outputs $\hat{C}$. If $s \leftarrow Dist_0$, then $\hat{C} = iO(C_0)$, and if $s \leftarrow Dist_1$, then $\hat{C} = iO(C_1)$. So this reduces the problem to assuming that $Dist_0 \approx_c Dist_1$.

So if some adversary A can distinguish between C_0 and C_1, then we have something that can break our simple assumption $Dist_0 \approx_c Dist_1$.

Unfortunately, this is not going to work.

Consider two cases. $C_0 \equiv C_1$ means that there is no adversary A that will work. On the flip side, let's say that we are fed $C_0 \not\equiv C_1$. Then there does exist an A, since there is a point where they are not equivalent, and A will just check that point. The intuition here is that the reduction must decide circuit equivalence, since in one case it must produce a good obfuscation and in another case it does not (but actually testing this is not obvious so this doesn't constitute a full proof).

The way out of this issue is to allow the reduction to run in time 2^n, where n is the input length of the circuit. Then the reduction is allowed to verify circuit equivalence. For this, we need that $Dist_0 \approx_C Dist_1$ even for 2^n-time adversaries.

Examples: There is iO from subexponentially secure FE from assumptions on MMaps. There is also iO from subexponential DDH on 5-linear maps.

2 Point Obfuscation

Point obfuscation is obfuscation restricted to the set of programs that are point functions. The types of programs are $I_{xy}(x')$ which is y if $x = x'$ and 0 elsewhere. Recall that we saw these point functions in the first/second lecture for the Barak impossibility result.

These are useful for password hashing. You obfuscate the point function where the secret input is the password.

Auxiliary Input Point Obfuscation (AIPO). Here we'll consider distributions $(x, y, \mathsf{aux}) \leftarrow Dist$. We'll say that $Dist$ is computationally unpredictable if, given the auxiliary information produced by the distribution, it's still hard to find x. O is an auxiliary input point obfuscator if, for all computationally unpredictable distributions, $(O(I_{xy}), \mathsf{aux}) \approx_c (O(I_{xy'}), \mathsf{aux})$ where y' is random and x, y, aux is drawn from the distribution.

We'll show under a reasonable assumption (namely that iO exists), that this definition is unsustainable.

Consider the following distribution. We draw x at random, y at random, and then aux is $\mathsf{iO}(D_{xy})$ where $D_{xy}(C)$ does the following. It outputs 1 if $C(x) = y$, and 0 else.

If we just had VBB obfuscation, the value of x and y should be hidden, because for any circuit C' we feed in, $C'(x) \neq y$ if y is drawn from a large enough distribution. So this distribution should be computationally unpredictable. For AIPO to hold, we need $(O(I_{xy}), aux) \approx_C (O(I_{xy'}), \mathsf{aux}$ where y' is random. Clearly this won't be the case. In the first case $\mathsf{aux}(O(I_{xy})) = 1$, and in the other case $\mathsf{aux}(O(I_{xy'})) = 0$.

To actually make this proof work with iO, we'll have to say $\mathsf{aux} = \mathsf{iO}(D_{xz})$ where $z = G(y)$ and G is some PRG. $D_{xz}(C)$ works by outputting 1 if $G(C(x)) = z$, and 0 else.

By PRG security, we know that $D_{x,G(y)} \approx_c D_{x,z'}$ for random z', But with high probability $D_{x,z'} \equiv$ all 0's program since there likely won't be a preimage of z'. So by iO we have $\mathsf{iO}(D_{x,G(y)}) \approx_c \mathsf{iO}(\text{all } 0's)$

So in particular this means that given the auxiliary information, you don't learn anything about x and y. So this distribution is computationally unpredictable. The rest of the argument is the same.

3 Differing Inputs Obfuscation

We have a distribution that produces $(C_0, C_1, \mathsf{aux}) \leftarrow Dist$. Dist is differing inputs if for all differing inputs distributions, $diO(C_0), \mathsf{aux} \approx diO(C_1), \mathsf{aux}$. We need at a minimum that given the auxiliary information, it is hard to compute a point where the two circuits differ.

$Dist$ is differing inputs if given C_0, C_1, aux, it's hard to find x such that $C_0(x) \neq C_1(x)$.

We want to concoct a distribution that is differing inputs, but for which we can easily tell apart the obfuscations of the two programs.

Suppose we have some signature scheme. For $b = 0, 1$, let $C_b = C_{vk,b}$. $C_{vk,b}(x, \sigma)$ is b if $Ver(vk, x, \sigma)$ accepts and 0 else. So if the signature σ is valid for a message x, it outputs b and otherwise it outputs 0.

We'll set $D_{sk}(C) = C(x, \sigma)$. x is going to be some hash $H(C)$ of a circuit C. Then σ is $Sign(sk, x)$. So the distribution will be $(C_0, C_1, Obf(D_{sk}))$.

We can tell apart the obfuscations of C_0 and C_1 by simply feeding them into the auxiliary obfuscated program, since C_0 will lead the program to reject (since C_0 will always reject) and C_1 will lead the program to accept (since it will be fed a valid signature of its hash).

It remains to decide that given $Obf(D_{sk})$ if we can we find a differing input for C_0 and C_1. If this obfuscator were actually VBB obfuscation, we would be good. Since we don't have VBB obfuscation, we get impossibility if there exists a "special purpose obfuscator"; if there exists some signature scheme and obfuscation scheme such that given an obfuscation of this circuit containing the signing key, it is hard to produce a signature.

By tweaking this counterexample, we can get an impossibility result for Turing machines, where we no longer have to compute $H(C)$ and can use the code of C directly.

In public coin diO, we only consider Dist where aux is random coins. So in this case we say that they are differing inputs even if the coins flipped are public. Clearly what we did above is not public coins diO since if we have the random coins then we can undo the signature scheme. It turns out that we can build iO for Turing Machines from public coin diO.