Advanced Topics in Cryptography

Lecture 2: Program Obfuscation - II

April 1, 2009

Lecturer: S. Goldwasser, M. Naor Scribe by: R. Marianer, R. Rothblum Updated: May 3, 2009

1 Introduction

Barak *et-al*[1] formally defined the notion of obfuscation and proved an impossibility result concerning a universal obfuscator by showing a family of functions that cannot be obfuscated. The functions that were shown to be "unobfuscatable" were built specifically for this and thus the question remains whether specific (interesting) families of functions can or cannot be obfuscated.

2 Password functions (Point functions)

Perhaps the most naïve and simple function that we would want to obfuscate is the password function. This obfuscated program would allow anyone to check whether a given password is correct without revealing anything about it.
For example, consider these two possible applications:

1. **Content-Concealing Signatures:** Suppose Alfred is writing a will. He would like to deposit the will with his lawyer while keeping the contents secret. He would also like for his heirs to able to read the will and to verify that it is indeed his. A possible solution would be to write the will as the output of an obfuscated password function and leave the password with his heirs.

2. **Puzzle Application**: Consider a puzzle (Crossword, Sudoku, etc.) published in some newspaper. We would want to allow a potential solver to check his answer but we do not want to reveal the solution itself. To do this we could use an obfuscated password function with the solution of the puzzle as a password. A reader could easily verify his solution but since the program is obfuscated he learns (almost) nothing more and in particular cannot find the solution from the program.

Formally, for any $n \geq 1$ and $p \in \{0,1\}^n$ let $I_p : \{0,1\}^n \to \{0,1\}$ be the following function:

$$I_p = \begin{cases} 1 & \text{if } x = p \\ 0 & \text{otherwise} \end{cases}$$

Let $I_n = \{I_p | p \in \{0,1\}^n\}$ and $I = \bigcup_{n \geq 1} I_n$. Let us consider how we could obfuscate such a function:

Attempt 1

Given a one-way function f, our obfuscator works as follows:

```
$y \leftarrow f(p)$
Output the following code:
    If $f(x) = y$   output 1
    else            output 0
```

At first glance this appears to achieve exactly what we wanted - given code to the obfuscated program one cannot derive p (by the hardness of inverting f). However, our definition of obfuscation is more robust and in particular requires not only that p not be recoverable, but that nothing about p be recoverable. The "obfuscated" program suggested above completely reveals $f(p)$ and thus is not obfuscated in the robust sense we defined. This example illustrates that every obfuscator must be randomized.

Attempt 2

We will show a construction of Canneti[2] which will obfuscate password functions based on a standard cryptographic assumption.

DDH (Decisional Diffie Hellman) assumption

Let G be a cyclic group of prime order and let g be a generator for it. The decisional Diffie Hellman assumption (DDH) is that there are families of groups as above for which:

$$(g^a, g^b, g^{ab}) \cong (g^a, g^b, g^c)$$

(where $a, b, c \in_R \{1, \ldots, |G|\}$).

Claim 1 *DDH implies that $r^p \cong r^x$ with $r \in_R G$ and $x \in_R \{1, \ldots, |G|\}$ (for all p).*

Sketch of Proof Assume to the contrary, we will show that we can decide DDH. We are given (α, β, γ) and know that α and β are random elements in the group (random powers of a generator) and that γ is either a random power of α or just a random element in the group. By using the above we can check in which case we are and decide appropriately. ■

Consider the following obfuscator based on the above:

```
Choose $r \in_R G$
$y \leftarrow r^p$
Output the following code:
    If $r^x = y$   output 1
    else           output 0
```

A potential adversary can see r and r^p. However, since r is random, finding p from these two with nonnegligible probability of success would mean being able to distinguish between r^p and r^{random}.

3 Obfuscating public point functions

Recall that if $L \in \text{NP}$, the witness relation[1] R_L is a polynomially-computable relation (i.e., $R_L \in \text{P}$) and $\exists w$ s.t. $R_L(x,w) \iff x \in L$. We consider the following function, known as a public-point function:

$$f^L_{p,s}(x) = \begin{cases} s & \text{if } (p,x) \in R_L \\ 0 & \text{otherwise} \end{cases}$$

Note that the definition differs from that of password functions in several respects:

- The output value is not 0/1 but 0/s.
- p is not secret, the goal of the obfuscation is to hide s rather than hide p. To illustrate this we could replace the "return 0" in the otherwise clause by "return p".
- The relation is no longer equality.

We will show that under a somewhat stronger definition of obfuscation, if this function can be obfuscated, then many interesting families of standard cryptographic primitives *cannot* be obfuscated (e.g. pseudo random functions, private key encryption schemes, signature schemes). To show this, we require an obfuscator which is secure against auxiliary input to the challenging algorithm.

Obfuscation w.r.t Auxiliary Input

We define such an obfuscator formally as:

Definition 2 *A probabilistic algorithm $\mathcal{O}$ is called an* Obfuscator with respect to dependent auxiliary input *for a class of circuits $C = \{C_n\}_{n\in\mathbb{N}}$ if the following three conditions hold:*

1. ***Functionality:*** *For every circuit $c \in C$ and for every input x: $c(x) = (\mathcal{O}(c))(x)$.*
2. ***Polynomial Slowdown:*** *There exists a polynomial $p(\cdot)$ such that for every circuit $c \in C$, $|\mathcal{O}(c)| \le p(|c|)$.*
3. ***Virtual Black-Box with auxiliary input:*** *For any PPT algorithm A there exists a negligible function $\mu(n)$ and a PPT simulator S s.t for any $n \in \mathbb{N}$, any $c \in C_n$, any z s.t. $|z| \le \text{poly}(n)$ and any predicate $\pi(c,z)$,*

$$|\Pr[A(\mathcal{O}(c),z) = \pi(c,z)] - \Pr[S^c(1^n,z) = \pi(c,z)]| < \mu(n)$$

Note the strengthening from the original definition of Barak et-al, with the addition of z in the virtual black box property.

Claim 3 *If L is NP-complete and $f^L_n = \{f^L_{p,s}\}$ can be obfuscated (with respect to dependent auxiliary input) then $f^{L'}_n$ can be obfuscated (with respect to dependent auxiliary input) for every L' in NP.*

[1]Technically this isn't well-defined, but we assume a "canonical" witness relation for any given NP-complete language L

Sketch of Proof Assume we can obfuscate $f^L_{p,s}$ for any (p,s), and assume we know how to reduce L to L' and R_L to $R_{L'}$. The following program obfuscates $f^{L'}_{p',s}$:

```
Convert $p'$ to an instance $p$ of $L$ using a known reduction.
Generate an obfuscated program for $f^L_{p,s}$
Output the following code:
    Assuming the input parameter $x'$ is a witness for $p' \in L'$, convert it into a
    witness $x$ for $p \in L$. (If it isn't, this will convert garbage into garbage).
    Run the obfuscated program for $f^L_{p,s}$ on $x'$.
```

■

High Pseudo Entropy Circuits

Definition 4

- *A random variable X has* min-entropy *k (over a set S) if $\mathbb{P}[X = x] \geq \frac{1}{2^k}$ for every $x \in S$.*
- *A sequence (X_n) of random variables has* superpolynomial *min-entropy if its min-entropy is greater than any polynomial $p(n)$ as $n \to \infty$.*
- *A family $\{C_n\}_{n \in \mathbb{N}}$ of circuits has* pseudo-entropy *at least $p(\cdot)$ if there is a sequence of sets (I_n), where $|I_n|$ is limited by a polynomial t, such that the random variable $X_n := C_n(i_1), \ldots, C_n(i_{t(n)})$ where $i_j \in I_n$ is computationally indistinguishable from a random variable with min-entropy $p(n)$ (even for a distinguisher which has oracle access to $C_n(i)$ for $i \notin I_n$).*
- *A sequence of circuits has* superpolynomial *pseudo-entropy if it has pseudo-entropy at least p for every polynomial p.*

The following claim (due to [3]) shows why we are interested in circuits with superpolynomial pseudo-entropy:

Claim 5 *The following are examples of circuits with superpolynomial pseudo-entropy:*

1. *Pseudorandom functions*
2. *The encryption function E_k in a private-key semantically secure encryption scheme*
3. *Randomized digital signatures, as long as each signature has min-entropy at least 1.*

With this in mind, we can prove the theorem stated at the beginning of this section, that if we can obfuscate public point functions then we cannot obfuscate many well-known cryptographic primitives.

Theorem 6 *Let $\left\{f^K_{p_n,s}\right\}$ be a family of public point functions for an NP-complete language K, such that $|p_n| = n$. If there exists an obfuscator with auxiliary input that works on this class of functions, then any family $\{C_n | n \in \mathbb{N}\}$ of circuits with superpolynomial pseudo-entropy cannot be obfuscated.*

Sketch of Proof Fix an arbitrary polynomial $p(\cdot)$, and let $t(\cdot)$ be the polynomial whose existence is guaranteed by the definition of superpolynomial pseudo-entropy. Let $l(n) := t(n) - n$, and let L be the following language:

$$L = \{(I_n, C(I_n)) : |C| < l(n)\}$$

(If C is circuit and A is a set, then $C(A) := \{C(a)|a \in A\}$).

We consider the family $\left\{f^L_{p_n,s}\right\}$, where $p_n = \{(I_n, C(I_n)\}$ and $|I_n| = t(n)$. L is obviously in NP, and therefore this family can be obfuscated. Now let $Z_{C,s}(V)$ be the following function, where V is a circuit:

$$Z_{C,s}(V) = \begin{cases} s & \text{if } V(i) = C(i) \text{ for all } i \in I_n \\ 0 & \text{otherwise} \end{cases}$$

Clearly, $Z_{C,s}$ is just $f^L_{p,s}$, where $p = C(I_n)$, so these functions can be obfuscated wrt auxiliary input. To show that C_n cannot be obfuscated wrt auxiliary input, we will take $\mathcal{O}(Z)$ as the auxiliary input. It now suffices to prove the following:

1. Given $\mathcal{O}(C)$ and $\mathcal{O}(Z)$, it is computationally easy to compute s; and
2. Given $\mathcal{O}(Z)$ and oracle access to C, it is computationally difficult to compute s (with probability better than 1/2+negl).

The first part is trivial: $s = Z(C)$, and obfuscation preserves functionality, so $A(C, Z)$ simply returns $Z(C)$ which is always s.

For the second part, we assume towards contradiction that there is a PPT oracle machine S_1 such that

$$Pr[S_1^C(\mathcal{O}(Z)) = s] \geq \frac{1}{2} + \text{nonnegl}$$

Let C' be a machine that, on input i, returns $C(i)$ unless $i \in I_n$. Then we can replace S_1 by an oracle machine that accepts $C(I_n)$ as input, and has oracle access to C':

$$Pr[S_2^{C'}(\mathcal{O}(Z), C(I_n)) = s] \geq \frac{1}{2} + \text{nonnegl}$$

Now, $Z = f^L_{p,s}$, and we can replace this with $Z' = f^L_{p',s}$ where p' is completely random. This is because otherwise we'd have a distinguisher between p and p', and this is impossible given the pseudo-entropy condition. Therefore

$$Pr[S_2^{C'}(\mathcal{O}(Z'), C(I_n)) = s] \geq \frac{1}{2} + \text{nonnegl}$$

We might as well let our algorithm have the whole circuit C now (and no oracle access),

$$Pr[S_3(\mathcal{O}(Z'), C) = s] \geq \frac{1}{2} + \text{nonnegl}$$

and since Z' can be obfuscated wrt auxiliary input, we have the existence of a fourth algorithm such that

$$Pr[S_4^{Z'}(C) = s] \geq \frac{1}{2} + \text{nonnegl}$$

This means that S_4 can find, with noticable probability, a circuit C' of size $< l(n)$ such that $C(I) = p'$. However, since p' is random and of size $t(n)$, a counting argument shows that this is impossible. ■

4 Independent auxiliary input

The proof above relies on the fact that the auxiliary input can depend on the circuit C that is obfuscated. it is possible to define a similar notion, that of **obfuscation wrt independent auxiliary input**, where z must be chosen independently of C (and the result must hold for randomly chosen C).

Definition 7 *A probabilistic algorithm $\mathcal{O}$ is called an* Obfuscator with respect to independent auxiliary input *for a class of circuits $C = \{C_n\}_{n \in \mathbb{N}}$ if the following three conditions hold:*

1. ***Functionality:*** *For every circuit $c \in C$ and for every input x: $c(x) = (\mathcal{O}(c))(x)$.*
2. ***Polynomial Slowdown:*** *There exists a polynomial $p(\cdot)$ such that for every circuit $c \in C$, $|\mathcal{O}(c)| \leq p(|c|)$.*
3. ***Virtual Black-Box with auxiliary input:*** *For any PPT algorithm A there exists a negligible function $\mu(n)$ and a PPT simulator S s.t for any $n \in \mathbb{N}$ and any z s.t. $|z| \leq \mathrm{poly}(n)$ and predicate $\pi(c, z)$,*

$$|\Pr[A(\mathcal{O}(c), z) = \pi(c, z)] - \Pr[S^c(1^n, z) = \pi(c, z)]| < \mu(n)$$

(where the probability is over choosing a random $c \in C_n$ and the random coin tosses of $A, \mathcal{O}, S$).

Obfuscation wrt independent auxiliary input is weaker than obfuscation wrt dependent auxiliary input (as in the previous definition) not only in z not being dependent on c, but also in requiring the virtual black box property for a random c and not every c. Note that this definition is incomparable to the original definition of Barak et-al[1] since it is stronger in allowing auxiliary input and weaker in requiring security only for random choices of c (any not every c).

This definition, though weak, suffices for proving the following negative result[3]:

Theorem 8 *Define $C^L(x, w) := C(x)$ if w is a witness for x (i.e., $(x, w) \in R_L$), where L is some NP language. The family of such functions cannot be obfuscated (wrt independent auxiliary input) if*

1. *L is NP-complete;*
2. *C is of super-polynomial min-entropy; and*
3. *For random C and x, $C(x)$ is unpredictable, even given access to $C(x')$ for every other x'.*

The proof is based on similar ideas.

5 Hard-core predicates for languages

Definition 9 *Let L be some NP-complete language, and let $B(x,r)$ be some predicate whose first parameter is a word in that language and whose second parameter is an random string of the same length, i.e., $x \in L$ and $r \in \{0,1\}^{|x|}$. B is a* hard-core predicate *for L if it is*

1. ***Easy to compute given a witness:*** *There exists a PPT algorithm that can compute $B(x,r)$ given $(x,w) \in R_L$ and $r \in \{0,1\}^{|x|}$; and*
2. ***Hard to compute otherwise:*** *There exists a PPT oracle machine such that given oracle access to $B(x,r)$ (or even to something that outputs $B(x,r)$ with probability $\frac{1}{2}+\mathrm{nonnegl}(|x|)$ over random choice of r), it can compute a relevant w with probability $\frac{1}{2}+\mathrm{nonnegl}(|x|)$ (again, over random choice of r). (Assuming $P \neq NP$, this means that there is no PPT algorithm that can compute $B(x,r)$ with probability $\frac{1}{2}+\mathrm{nonnegl}(|x|)$ from (x,r) and no witness).*

Remark There is a similar notion of hard-core predcates in the world of one-way functions: A hard-core predicate can be easily computed given (x,r) but not given $f(x,r)$. However, the two notions don't necessarily have similar properties. As an example, any one-way function can be converted into a one-way function with a hard-core predicate ([4]), but there is no known similar result for the notion of hard-core predicates for NP languages.

Theorem 10 *If the language L has a known hard-core predicate B, we can obfuscate (with respect to auxiliary input) the function $f^L_{p,s}$.*

Sketch of Proof The following obfuscator works, albeit in exponential time:

```
Choose $r \in_R \{0,1\}^{|p|}$.
Find a witness $w$ such that $(p,w) \in R_L$. (This is the part that takes exponential time)
Calculate $s' \leftarrow B(p,r) \oplus s$ (We can calculate $B(p,r)$ because we have $w$).
Output the following code:
    If $(p,x) \in R_L$   output $B(p,r) \oplus s'$ (We can calculate $B(p,r)$ because we have $x$)
    else                 output 0
```

■

References

[1] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai and K. Yang, "On the (Im)possibility of Obfuscating Programs", *In Proceedings of the 21st Annual international Cryptology Conference on Advances in Cryptology*, 2001.

[2] R. Canneti, "Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information ", *In Proceedings of the 17th Annual international Cryptology Conference on Advances in Cryptology*, 1997.

[3] S. Goldwasser, Y. Tauman Kalai, “On the Impossibility of Obfuscation with Auxiliary Input”, *FOCS 2005. 46th Annual IEEE Symposium on Foundations of Computer Science, 2005*, 2005.

[4] O. Goldreich, L. A. Levin, “A hard-core predicate for all one-way functions”, *In Proceedings of the 21st Annual ACM Symposium on Theory of Computing, 1989*