CS6115: Structure versus Hardness in Cryptography Sept 02, 2021

Lecture 10 : Deniable Encryption from Indistinguishability Obfuscation

Lecturer: *Shweta Agrawal* Scribe: *Aadityan Ganesh*

1 Introduction

Prior to this lecture, we have looked at encryption schemes that base their security on the hardness of the learning with errors (LWE) problem. We have looked at fully homomorphic encryption schemes and deniable encryption schemes based on the LWE problem.

But, the deniable encryption scheme that we had seen previously supports only unidirectional deniability. This barrier has not been breached yet. Hence, schemes that base their security on assumptions other than LWE problem have been studied with the hope of solving challenges posed when security is based on LWE.

2 Indistinguishability Obfuscation

Intuitively, we want to find a way to modify programs such that the functionality of the input program is preserved but the details are lost. Hence, given two programs of the same functionality, the obfuscated output should leave no clue as to which of the two programs was the input.

There exists an obvious way of achieving this- take a program as input and output its truth table. We would want to look at more efficient ways of achieving this. We state the formal definition below.

Definition 2.1. Indistinguishability Obfuscator: An indistinguishability obfuscator iO is a PPT algorithm with

- Input: A program P
- Output: $iO(P)$ satisfying
  1. $iO(P)$ can be computed in time polynomial in the description of P.
  2. $iO(P)$ preserves the functionality of P.
  3. For any PPT adversary A and programs P_1, P_2 of equal complexity and functionality,
  $$|Pr[A(iO(P_1)) = 1] - Pr[A(iO(P_2)) = 1]|$$
  is negligible.

Certainly, any adversary with limited computing power will not be able to learn which of the two programs was obfuscated with the help of the output. But, if that sufficient for cryptography? At the end of the day, we want our programs to hide a piece of secret. Will obfuscating a program help in hiding information? Yes and no!

3 A Relative Viewpoint

Consider programs P and P' of the same functionality and complexity and the indistinguishability obfuscator iO. The definition of indistinguishability obfuscation guarantees that anything that can be learnt about P from $iO(P)$ in polynomial time can also be learnt about P'. We have succeeded in hiding information about the input program which is different from another program, but this need not guarantee hiding some sensitive information about the input which is common to all programs with the same functionality.

Thus, the security provided by an indistinguishability obfuscator need not be absolute. A detail x of a program is hidden by an obfuscator if there exists another equivalent program whose detail x is hidden by the obfuscator but, x need not be hidden otherwise. This relativistic notion of security gives rise to the following theorem (which has some good and bad implications).

Theorem 1. [1] If iO is an efficient obfuscator for circuit family C, then iO is also an efficient (computationally) best possible obfuscator for C.

Proof. Consider an indistinguishability obfuscator iO. Let BiO be a better obfuscator than iO (as a running example, BiO might hide a detail x which iO does not).

Let P be a program. Note that $BiO(P)$ has the same functionality of P. The description of $BiO(P)$ is polynomial in the description of P. Padding P or $BiO(P)$ as required, assume without loss of generality, that P and $BiO(P)$ have the same complexity.

Thus, by the definition of indistinguishability obfuscation, any PPT adversary cannot differentiate between $iO(P)$ and $iO(BiO(P))$. Thus, iO performs as good as BiO (in the running example, the detail x is hidden by iO too). Contradiction.

Thus, iO has to be as good as any other obfuscator. $\square$

Suppose we find an indistinguishability obfuscator for a class of programs. The above theorem suggests that our obfuscator is the best in the market. But, if our obfuscator does not hide some detail that we want to hide, the above theorem suggests it is futile to try to hide the information with a different obfuscator.

Definition 3.1. Virtual Black-box Obfuscator: A virtual black-box obfuscator O is a PPT algorithm with

- Input: A program P
- Output: $iO(P)$ satisfying
    1. $iO(P)$ can be computed in time polynomial in the description of P.
    2. $iO(P)$ preserves the functionality of P.
    3. For any PPT adversary A and program P,

$$|Pr[A(iO(P)) = 1] - Pr[A(Q) = 1]|$$

is negligible, where Q is an oracle that spits $P(x)$ given an input x.

An adversary will not be able to differentiate between the obfuscated function $O(P)$ and an oracle that just gives the output and absolutely nothing else when O is a virtual black-box. Such virtual black-boxes does not exist for all circuit complexity classes, but there are small classes for which such a virtual black-box exists. In such cases, by theorem 1, any obfuscator is as good as a virtual black-box.

4 Witness Encryption

Goal: Let L be a language and x be a string. We want to encrypt message m such that it can be decrypted only by someone possessing a witness w for $x \in L$.

Encryption $WEnc(x, m)$**:** encrypts m relative to statement x
Decryption $WDec(w, c)$**:** output message if w is a witness for $x \in L$, output $\perp$ otherwise.

Securing the Above Description:

Consider $P_{x,m}$ that outputs m on witness that $x \in L$ and $\perp$ otherwise. Consider P that always outputs $\perp$. Let iO be an obfuscator whose domain contains $P_{x,m}$ and P. If $x \notin L$, then $iO(P_{w,m})$ and $iO(P)$ are indistinguishable since both the programs have the same functionality and complexity (after sufficient padding).

Thus, if $x \notin L$, the message m cannot be computed from $iO(P_{x,m})$ and thus the above scheme is secure.

5 A Possible Use of Obfuscation:

Consider a premium dashboard offering various features to its client. Suppose the firm decides to roll out a free version with limited functionalities. The obvious way to do this is to rework all the code and set up a new 'laborious dashboard' allowing access only to the limited set of features.

Another lazy way to achieve the same is to just turn off these functionalities in the dashboard rather than remove them completely to hide them. Even though the new dashboard offers the limited features and achieves the goal, it does not really hide the other features and can easily be reconstructed from this 'lazy dashboard'.

Suppose there is an obfuscator that obfuscates this class of dashboards. Obfuscate the lazy dashboard. Since the lazy and the laborious dashboard have the same functionality and (after padding) complexity, the obfuscations of the two dashboards are indistinguishable. Thus, anything that can be learnt about the removed features from the obfuscation of the lazy dashboard can be learnt from the obfuscation of the laborious dashboard. But, the laborious dashboard has been hard-coded to not have any detail about the removed features.

Thus, nothing about the premium features can be learnt from the obfuscated lazy dashboard.

6 From Relative to Absolute Guarantees

A central theme so far has been the fact that indistinguishability obfuscation gives only relative guarantees. We would want to take a step forward and obtain absolute guarantees after obfuscation.

One possible assumption is regarding the existence of one-way functions. Existence of one-way functions imply the existence of secure obfuscators. Interestingly, if P = NP, even then, there exists secure obfuscators.

We will look at such assumptions and their implications in the succeeding lectures.

References

[1] Goldwasser and Rothblum On Best-Possible Obfuscation. Proceedings of $4th$ Conference on Theory of Cryptography. TCC'07, pages194-213.