CS6115: Structure versus Hardness in Cryptography Sep 22, 2021

Lecture 14 : Deniable Encryption from iO

Lecturer: *Shweta Agrawal* Scribe: *Senjuti Dutta*

1 Introduction

We have been studying iO and Deniable Encryption from iO for the past three lectures (from [SW14]). We have seen iO, Witness Encryption, pseudorandom functions(PRF), variants of PRFs namely Puncturable PRF, injecting PPRF, extracting PPRF, construction of Deniable Encryption Scheme using iO with "hidden sparse trigger" and the proof structure for indistinguishabily of explanation.

In the following lecture, we go through the concepts and ideas again in order to answer any recurring gaps and questions.

2 Review of Deniable Encryption from iO

1. What assumptions is iO usually based on? How strong of an assumption is iO in comparison to say OWF or DDH?

The iO is considered an extremely strong assumption. It is considered a non-falsifiable assumption [Lec11]. More precisely, the adversary gives the challenger two circuits C_0 and C_1 with the same size and the same truth table. Now the challenger gives the iO of any of the two circuits, chosen at random and the adversary has to guess the bit b. However, as the challenger, in order to check whether the guess was made in a fair manner, it needs to check whether the two circuits that the adversary proposed are functionally equivalent. That is, it needs to check whether both the circuits output the same answer for any given input. If the circuits take n inputs, then the truth table has 2^n values (assuming the circuits are Boolean). So for arbitrary polynomial size circuits, this task is not efficient and it is not possible for it to check the equivalence of the circuits.

So constructing iO from more standard number theoretic assumptions like DDH or LWE would result in an exponential loss in the reduction. Due to the exponential loss, since the challenger has to check for the equivalence, the assumption of iO is very strong. Another reason is, scientifically, constructing iO is very difficult. Only very recently the last heuristic step was crossed and the construction from standard assumptions was completed[JLS21].

2. Explain punctured PRF. Also explain the notion of an adversary outputting a poly-sized set S and the properties holding for all elements in and not in S.

We attempt to answer this question by providing an intuition for the idea. However, for the sake of concreteness, we provide the definition of punctured PRF again.

Definition 2.1 (Punctured PRF)**.** A puncturable family of PRFs F is described by three algorithms, Key_F , $Puncture_F$, and $Eval_F$, and a pair of computable functions $n(.)$ and $m(.)$, satisfying the following conditions:

- **Functionality preserved under puncturing:** For every PPT adversary Adv such that $Adv(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$, then for all $x \in \{0,1\}^n(\lambda)$ and $x \notin S$, we have that:

$$Pr[K \leftarrow Key_F(1^\lambda), K_S = Puncture_F(K, S); Eval_F(K, x) = Eval_F(K_S, x)] = 1$$

- **Pseudorandom at punctured points:** For every pair of PPT adversary (Adv_1, Adv_2) such that $Adv_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$ and state σ. Also let $K \leftarrow Key_F(1^\lambda)$ and $K_S = Puncture_F(K, S)$. Then we have

$$|Pr[Adv_2(S, \sigma, K_S, Eval(K_S, S)) = 1] - Pr[Adv_2(S, \sigma, K_S, U_{m(\lambda).|S|}) = 1]| \leq negl(\lambda)$$

  where $Eval_F(K_S, S)$ is the concatenation of $Eval_F(K_S, x_1), ..., Eval_F(K_S, x_t)$, where $S = \{x_1, ..., x_t\}$. Note that $x_1, ..., x_t$ in S is the enumeration of elements in S according to the lexicographic order and U_l denotes the uniform distribution over l bits.

Now firstly, the main idea of Deniable Encryption is that given a cipher text, there should be more than one way to open the cipher text. So given a cipher text, the attacker should not be able to tell whether the encryption is of message one or message two(say), i.e, there should be an honest randomness by which the encryption of the actual message is done and there should be a fake randomness by which a different message can be associated with the given cipher text. However, it should decrypt the cipher text correctly as well.

For FHE, the cipher text is stored as $CT = As + e + m(\frac{q}{2})$ for a matrix A, secret key s and an error distribution e. Here, the message is m and the scaling factor is q. Now, for an encryption of 0, the ciphertext should be $CT_0 = As + e$ and for an encryption of 1, the ciphertext should be $CT_1 = As + e + (\frac{q}{2})$. Now, in order to deny the message, we must have a representation of an encryption of 0 as an encryption of 1(intuitively). That is, there must exist a secret key s' and error distribution e' such that

$$CT_1 = As' + e'$$

Now since this is the structure of the encryption of 0, we can successfully deny it as an encryption of 1.

Now we look at a different approach. Instead of trying to find a roundabout way of "faking" our encryption, we build the explain function to encode (using the function F) a given ciphertext-message pair (c', m') such that $F(c', m')$ is pseudorandom. So it must follow these two conditions:

1. It can be inverted using a secret key
2. It should be pseudorandom

This brings the notion of "hidden sparse trigger".

So this explain algorithm would output a fake randomness which is simply a pseudorandom encoding of (c, m'). This is done using trapdoor functions. We use three PRF keys to achieve this and in order to hide these PRF keys, we need obfuscation.

Now consider a reduction algorithm. On one side there is the iO challenger and on the other side there is the DE adversary. It must translate a DE attack to an iO attack. Consider the position

of the reduction. In order for the reduction to use the security of the PRF, the reduction algorithm shouldn't know the keys of the PRF but the DE public key contains obfuscated programs which in turn contain PRF keys. So the reduction must emulate the DE challenger and the iO adversary. The DE challenger needs to give the public keys which contain the obfuscated programs which in turn contain PRF keys. Now for simplicity, assume that explain is giving $PRF(K, (m', c))$ as output. This is pseudorandom only if the PRF keys are not known. If this had been an "oracle style" obfuscation, which says that just the input-output behaviour is maintained, then it could have possibly been simulated. But we don't have an oracle style obfuscation.

So the main question: without "oracle style" obfuscation, given only iO, how can the reduction return the DE PK without knowing the PRF key? And this is where the notion of puncturing comes in. We need to get functional equivalence of PRF$(K, (m', c))$ without knowing the key K. So we use punctured PRF. Let us puncture the PRF on some special point x^* and let the punctured key be K^*. So by our construction, K^* gives me the same functionality as K except on the point x^*. On the point x^*, we want to invoke PRF security. Now PPRF tells us that on x^*, the output of the PRF appears random. However, this needs to be equal for x^* as well. So we insert a "trapdoor branch" at the point x^* in the program. So we add an 'if' condition that if the input is x^* then output y where PRF$(K, x^*) = y$, otherwise replace K with K^*. Now by our construction using punctured PRF, the functional equivalence is maintained. Now the security of the iO can be invoked. Now I can invoke PRF security on the point x^* and in the next hybrid y can be replaced by random. The detailed process is given in [Lec12] and [**?**].

Note that the key K^* can leak x^*. In other words, we do not require x^* to be private. There is a notion of private punctured PRFs that do not give away the x^* point, however we won't be needing it here.

3. In the definition of statistically injective PRF, they define the notion of a failure probability which is the probability with which it is not injective. Do there exist variants which have this failure probability 0? If yes, why not use them to make the ideas more clear?

For our purpose, it can be assumed that the failure probability is 0. The reason for the mention of failure probability is that there is a construction of an injective PRF and it has a failure probability which is inherited from the construction.

4. Explain min-entropy. Also explain the idea of extracting puncturable PRFs.

The formal definition of min-entropy is given in [Lec11] : For a random variable X with support $Supp(X)$ (support of a random variable is the set of all points on which X has non-zero probability). Then $H_\infty(X)$ is the min-entropy of X:

$$H_\infty(X) = \min_{x \in Supp(X)} \log\left(\frac{1}{Pr[X = x]}\right)$$

Intuitively, it gives the maximum amount of uncertainty of a random variable in a distribution. It can be seen as a measure of unpredictability of a set of outcomes.

The idea of an extracting puncturable PRF is that now the key K is known but the point x is unknown. So in the simple case where x is chosen randomly, then the extracting PPRF says that the output is random. Similarly, in the general case, the output should still be random.

5. Is it not possible to find a "very good" PRF which can be used everywhere instead of having 3 different PRF's? In other words, why can't there be a PRF with a large domain and range and somehow use restrictions to modify it for the other cases?

There is no good way of having one PRF having the properties of all the three PRFs. For example, in this particular case, the extracting PRF has to be shrinking and there are other PRFs that has to be expanding in nature. However, there is a notion of "Invertible constrained PRFs" [BKW17] which should suffice by a single key K.

6. Is iO practical? For instance, from what little I know, SMPC(secure multiparty computation) is not highly practical at the moment in most cases but it is slowly changing with more practical applications coming quickly.

No, iO is not practical in the sense that the construction of iO is not efficient. However, iO has practical applications. The notion of converting a private key encryption scheme to a public key encryption scheme is a great practical application and it has become useful in many places(restricted use software).

7. Are other applications like NIZK, TDF and OT be covered?

No.

References

[BKW17] Dan Boneh, Sam Kim, and David J. Wu. Constrained keys for invertible pseudorandom functions. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 237–263, Cham, 2017. Springer International Publishing.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

[Lec11] Lecture. Deniable encryption from io. `http://www.cse.iitm.ac.in/~shwetaag/6115/Lec11.pdf`, 11.

[Lec12] Lecture. Deniable encryption from io. `http://www.cse.iitm.ac.in/~shwetaag/6115/Lec12.pdf`, 12.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484, 2014.