CS6115: Structure versus Hardness in Cryptography Sep 08, 2021

Lecture 11 : Deniable Encryption from iO

Lecturer: *Shweta Agrawal* Scribe: *Amik Raj Behera*

1 Review of iO

In the last class, we defined iO. Recall that, we get "as good as" type of guarantee from iO i.e. a *relative guarantee*. In particular, we saw that any iO is atleast as good as any other iO. It is not giving us an absolute guarantee.

Suppose we want to obfuscate functionality f. Intuitively, we wanted that an obfuscation converts f into a black box $\mathcal{B}$, such that on input x, $\mathcal{B}$ should return $f(x)$. iO *does not* capture this intuitive notion of obfuscation, and thus it gives us a "weak relative guarantee".

1.1 Non-Falsifiable assumption

Let us recall the game based security of Public-Key Encryption.

- Let $\Pi = (\mathsf{PKE.SecretKeyGen}, \mathsf{PKE.PublicKeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a public-key encryption. λ denotes the security parameter.
- Challenger runs $sk \leftarrow \mathsf{PKE.SecretKeyGen}(1^\lambda)$, $pk \leftarrow \mathsf{PKE.PublicKeyGen}(1^\lambda)$. sk is the secret key and pk is the corresponding public key. Challenger sends pk to the Adversary.
- Adversary chooses two distinct plaintext messages m_0 and m_1 from the message space. Adversary sends m_0 and m_1 to the Challenger.
- Challenger flips a fair coin and gets bit $b \in \{0, 1\}$.
- Challenger runs $c \leftarrow \mathsf{PKE.Enc}(m_b, pk)$ i.e. encrypted m_b. Challenger sends c to the Adversary.
- Adversary guesses the bit b, and sends $\tilde{b}$.

We say that Π is a secure Public-key encryption if the probability of $\tilde{b} = b$ (i.e. adversary correctly guesses b) is only negligibly better than $1/2$ (i.e. random guess).

Now let us look a similar game based security of iO.

- Challenger sends public parameters to the Adversary. (In case of iO, there are no public-keys or secret-keys. In some cases, there may not be any public parameters, so the Challenger can send an empty message.)
- Adversary chooses two circuits $\mathcal{C}_0$ and $\mathcal{C}_1$ of same size (size of a circuit is the number of gates in the circuit). Adversary sends $\mathcal{C}_0$ and $\mathcal{C}_1$ to the Challenger.
- Challenger flips a fair coin and gets bit $b \in \{0, 1\}$.

- Challenger runs $\mathcal{C} \leftarrow \text{iO}(\mathcal{C}_b)$, i.e. obfuscation of $\mathcal{C}_b$. Challenger sends $\mathcal{C}$ to the Adversary.
- Adversary guesses the bit b, and sends $\tilde{b}$.

> The security of iO makes sense only if $\mathcal{C}_0 \equiv \mathcal{C}_1$, i.e. both $\mathcal{C}_0$ and $\mathcal{C}_1$ have the same truth table. If $\mathcal{C}_0$ and $\mathcal{C}_1$ don't have the same truth table, then the Adversary can query $\mathcal{C}$ at any of the points where $\mathcal{C}_0$ and $\mathcal{C}_1$ differ, and always correctly guess b.

More precisely, say the adversary sends $\mathcal{C}_0$ and $\mathcal{C}_1$ such that $\mathcal{C}_0(\mathbf{0}) \neq \mathcal{C}_1(\mathbf{0})$, i.e. $\mathcal{C}_0$ gives a different output than $\mathcal{C}_1$ on input $\mathbf{0}$. The adversary *knows* this fact since he is choosing $\mathcal{C}_0$ and $\mathcal{C}_1$. Now when the Challenger sends $\mathcal{C}$, the Adversary can simply check whether $\mathcal{C}(\mathbf{0})$ is equal to $\mathcal{C}_0(\mathbf{0})$ or $\mathcal{C}_1(\mathbf{0})$. Hence with probability 1, $\tilde{b} = b$.

To be secure against this cheating trick of Adversary, the Challenger needs to verify whether the truth table of $\mathcal{C}_0$ and $\mathcal{C}_1$ are same or not. If the circuits take n inputs, then the truth table has 2^n values (assuming the circuits are Boolean). Since the Challenger runs in polynomial time, it is not possible for it to check the equivalence of the circuits. Thus in iO it is assumed that $\mathcal{C}_0 \equiv \mathcal{C}_1$. This assumption is known as **Non-Falsifiable Assumption**.

2 Publicly Deniable Encryption

Definition 2.1 (Publicly Deniable Encryption). A *publicly deniable encryption scheme* over a message space $\mathcal{M}$ consists of four polynomial time algorithms: $\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Explain}$. λ is the security parameter.

- $\mathsf{KeyGen}(1^\lambda)$: It outputs a secret key sk and a corresponding public key pk.
- $\mathsf{Encrypt}(pk, m; u)$: It takes the public key pk, message $m \in \mathcal{M}$, randomness u, and outputs a ciphertext c.
- $\mathsf{Decrypt}(sk, c)$: It outputs the message m.
- $\mathsf{Explain}(pk, c, m; r)$: It takes the public key pk, ciphertext c, any message m, and randomness r as input. It outputs a string e, that has the same length as the randomness u used in $\mathsf{Encrypt}$.

A publicly deniable encryption scheme is correct if for all messages, the decryption algorithm outputs the correct message with high probability. Formally, for all messages $m \in \mathcal{M}$:

$$\Pr\left[(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda); \quad \mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, m; u)) = m\right] \geq 1 - \text{negl}(\lambda),$$

where negl is a negligible function.

> Recall that in a traditional deniable encryption scheme ([CDNO97]), the sender must remember his *actual randomness* and use it to construct fake randomness for other messages. In a publicly deniable encryption scheme, we *do not* require a sender to know the randomness that was used to generate any particular ciphertext in order to be able to generate fake randomness for it. Thus, even a third party that had nothing to do with the actual honest generation of a ciphertext can nonetheless produce fake randomness for that ciphertext corresponding to any message of his choice.

2.1 Security

Since publicly deniable encryption scheme is a public-key encryption scheme, it must satisfy the standard security definition, i.e. Indistinguishability under Chosen Plaintext Attack. This is the same game-based security which we reviewed in the previous section.

Publicly deniable encryption scheme also satisfies a new security property - **Indistinguishability of explanation**. We will define it using the following game:

- The Challenger runs $(sk, pk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends pk to the Adversary $\mathcal{A}$.
- $\mathcal{A}$ sends a *single* message $m \in \mathcal{M}$.
- The Challenger runs $c \leftarrow \mathsf{Encrypt}(pk, m; u)$ for random string u. It also runs $e \leftarrow \mathsf{Explain}(pk, c, m; r)$. The Challenger flips a fair coin and gets bit $b \in \{0, 1\}$.
- If $b = 0$, the Challenger sends (c, u), otherwise sends (c, e).
- Adversary $\mathcal{A}$ guesses the bit b, and sends $\tilde{b}$.

The *Advantage* of the Adversary $\mathcal{A}$ is the probability of guessing b correctly better than a random guess. Formally,

$$\mathrm{Adv}_{\mathcal{A}} = |\Pr[\tilde{b} = b] - 1/2|.$$

A publicly deniable encryption scheme has Indistinguishability of explanation if for all polynomial time $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{A}}$ is negligible in λ.

> The explanation algorithm $\mathsf{Explain}$ takes as input a ciphertext c, a message m, and some randomness. This explanation algorithm does not care what c is "really" encrypting. It simply outputs a random string which "explains" the c for message m. Note that c could have encrypted $m' \neq m$, but $\mathsf{Explain}$ gives a randomness which explains c with m! Thus, the explanation mode can be used (by anyone) to create fake randomness that can explain any ciphertext as any desired message. Isn't this amazing!!

Remark 1. In their work, [SW14] showed that a publicly deniable encryption scheme implies a deniable encryption scheme as defined by [CDNO97].

3 Variants of Pseudorandom functions (PRFs)

Definition 3.1 (Pseudorandom Function). *Pseudorandom functions* is a family of family of functions - $\{\mathcal{F}_n\} := \{f_s : \{0,1\}^n \to \{0,1\}^k \mid s \in \{0,1\}^k\}$, where s is a k-bit seed, $\mathcal{F}_n$ is the set of functions from n bits to k bits, x is a n-bit input to f_s, satisfying the following properties:

- (**Efficiently computable**): Given $s \leftarrow \{0,1\}^k$ and $x \leftarrow \{0,1\}^n$, $f_s(x)$ should be computable in $\mathrm{poly}(n, k)$ time.
- (**Security**): Let Adv^f denote a PPT algorithm using f as an adaptive oracle (running in $\mathrm{poly}(k)$ time). Fix any n. Let $\mathcal{H}$ denote the set of all functions from mapping inputs of n-bits to k-bits. $\mathcal{F}_n$ is said to be secure if for every PPT algo Adv:

$$|\Pr[s \leftarrow \{0,1\}^k,\ f_s \in \mathcal{F}_n;\ \mathrm{Adv}^{f_s} = 1] - \Pr[f \leftarrow \mathcal{H};\ \mathrm{Adv}^{f} = 1]| \leq \mathrm{negl}(\lambda)$$

Theorem 1 (**Existence of PRFs [GGM84]**). Given any pseudorandom generator (PRG), we can construct a pseudorandom function (Since it is known that PRGs can be constructed from One-way functions (OWFs), this theorem implies that PRFs can be constructed from OWFs. Thus existence of OWFs implies existence of PRFs). This construction is known as GGM construction.

The naive way to compute PRFs takes exponential time. The GGM construction uses binary tree to bring down this exponential time to $\log$ of exponential time i.e. poly time, providing the required efficiency. It uses PRGs to construct the binary tree, and thus security is guaranteed.

Definition 3.2 (**Puncturable PRFs [SW14]**). A *puncturable family of PRFs* F is described by three algorithms, Key_F, $\mathsf{Puncture}_F$, and Eval_F, and a pair of computable functions $n(\cdot)$ and $m(\cdot)$, satisfying the following conditions:

- (**Functionality preserved under puncturing**): For every PPT adversary Adv such that $\mathrm{Adv}(1^\lambda)$ 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:

$$\Pr[K \leftarrow \mathsf{Key}_F(1^\lambda), K_S = \mathsf{Puncture}_F(K, S);\ \mathsf{Eval}_F(K, x) = \mathsf{Eval}_F(K_S, x)] = 1.$$

- (**Pseudorandom at punctured points**): For every pair of PPT adversaries $(\mathrm{Adv}_1, \mathrm{Adv}_2)$ such that $\mathrm{Adv}_1(1^\lambda)$ outputs a set $S \subseteq \{0,1\}^{n(\lambda)}$ and state σ, let $K \leftarrow \mathsf{Key}_F(1^\lambda)$ and $K_S = \mathsf{Puncture}_F(K, S)$. Then we have

$$|\Pr[\mathrm{Adv}_2(S, \sigma, K_S, \mathsf{Eval}(K_S, S)) = 1] - \Pr[\mathrm{Adv}_2(S, \sigma, K_S, \mathcal{U}_{m(\lambda)\cdot|S|}) = 1]| \leq \mathrm{negl}(\lambda)$$

  where $\mathsf{Eval}_F(K_S, S)$ is the concatenation of $\mathsf{Eval}_F(K_S, x_1), \ldots, \mathsf{Eval}_F(K_S, x_t)$, where $S = \{x_1, \ldots, x_t\}$. Note that $x_1, \ldots, x_t$ in S is the enumeration of elements in S according to the lexicographic order. $\mathcal{U}_n$ denotes the uniform distribution over n bits.

> Puncturable PRFs can be understood through the following game. Alice has a PRF F and a key K. Bob gives Alice a subset S of inputs to F. Now Alice can generate a "new' key K_S (depends on original key K and subset S) such that two properties are satisfied: On any input *outside* S, evaluation of F under both the keys returns the same value. Secondly, on any input *inside* S, evaluation of F under both the keys is computationally indistinguishable.

For ease of notation, we will write $\mathcal{F}(K, x)$ to denote $\mathsf{Eval}_F(K, x)$. We also represent the punctured key $\mathsf{Puncture}_F(K, S)$ by $K(S)$.

Theorem 2 (**Existence of Puncturable PRFs [GGM84, BW13, BGI13, KPTZ13]**). If OWFs exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits.

The GGM construction with some modifications can yield puncturable PRFs from OWFs, observed by the above-mentioned results.

Definition 3.3 (**Statistically injective (puncturable) PRFs [SW14]**). A *statistically injective* (puncturable) PRF family with failure probability $\varepsilon(\cdot)$ is a family of (puncturable) PRFs F such that with the probability $1 - \varepsilon(\lambda)$ over the random choice of key $K \leftarrow \mathsf{Key}_F(1^\lambda)$, $F(K, \cdot)$ is injective.
If the failure probability $\varepsilon(\cdot)$ is not specified, then $\varepsilon(\cdot)$ is a negligible function.

Theorem 3 (Existence of statistically injective (puncturable) PRFs [SW14]). If OWFs exist, then for all efficiently computable functions $n(\lambda), m(\lambda)$, and $e(\lambda)$ such that $m(\lambda) \geq 2n(\lambda) + e(\lambda)$, there exists a puncturable statistically injective PRF family with failure probability $2^{-e(\lambda)}$ that maps $n(\lambda)$ bits to $m(\lambda)$ bits.

Proof. For sake of writing, we will hide the dependence of n, m, and e on λ. We first 2-universal hash family and pairwise independent hash family.

A family $\mathcal{H}$ of functions from universe $\mathcal{U}$ to T is **2-universal hash functions** if for all $x, y \in \mathcal{U}$ where $x \neq y$, we have

$$\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{|T|}.$$

A family $\mathcal{H}$ of functions from universe $\mathcal{U}$ to T is **pairwise independent hash functions** if for all $x_1, x_2 \in \mathcal{U}$ where $x_1 \neq x_2$, and any $y_1, y_2 \in T$ (y_1 may be equal to y_2), we have

$$\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = \frac{1}{|T|^2}.$$

A union bound shows that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1] = \frac{1}{|T|}.$$

Setup Let $\mathcal{H}$ be a family of 2-universal hash functions mapping n-bits to m-bits. Let $\mathcal{F}$ be family of puncturable PRFs mapping n-bits to m-bits (we know $\mathcal{F}$ exists because of Theorem 2).

Defining the function We define the family $\mathcal{F}'$ as follows: The key space of $\mathcal{F}'$ consists of a key K for $\mathcal{F}$ and a hash function h chosen from $\mathcal{H}$. Then

$$\mathcal{F}'((K, h),\, x) = \mathcal{F}(K, x) \oplus h(x).$$

Observation If $\mathcal{F}$ were a *truly random* function, then for an independent random choice of h from $\mathcal{H}$, $\mathcal{F}(x) \oplus h(x)$ would be truly random function (intuitively one can can think of it this way: XOR operation *preserves the randomness*). Note that we wrote $\mathcal{F}(x)$ instead of $\mathcal{F}(K, x)$ because a truly random function needs no key.

Is the function puncturable PRF? First, is $\mathcal{F}'$ a PRF? $\mathcal{F}$ is a PRF, and thus $\mathcal{F}(K, x)$ is computationally indistinguishable from truly random function. Adding $h(x)$ doesn't affect the distribution of $\mathcal{F}$, and thus $\mathcal{F}'$ is a PRF. From this, one can immediately deduce that since $\mathcal{F}$ is a puncturable PRF, $\mathcal{F}'$ is also a puncturable PRF.

Is the function statistically injective? Consider $x_1 \neq x_2 \in \{0,1\}^n$. Fix any key K. By pairwise independence, we have

$$\Pr_h[h(x_1) = h(x_2) \oplus \mathcal{F}(K, x_1) \oplus \mathcal{F}(K, x_2)] = \frac{1}{2^m}.$$

Taking a union bound over all distinct pairs (x_1, x_2) gives us that:

$$\Pr_h[\exists\, x_1 \neq x_2 \;:\; \mathcal{F}'((K, h), x_1) = \mathcal{F}'((K, h), x_2)] \leq \frac{2^n}{2^m}.$$

Averaging over the choice of K, finishes the proof, by the choice $m \geq 2n + e$. □

Definition 3.4 (Min-Entropy). For a random variable X with support $\mathrm{Supp}(X)$ (support of a random variable is the set of all points on which X has non-zero probability). $H_\infty(X)$ is the *min-entropy* of X:

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

Definition 3.5 (Extracting (puncturable) PRFs [SW14]). An *extracting (puncturable) PRF* family with error $\varepsilon(\cdot)$ for min-entropy $k(\cdot)$ is a family of (puncturable) PRFs $\mathcal{F}$ mapping $n(\lambda)$-bits to $m(\lambda)$-bits such that for all λ, if X is any probability distribution with min-entropy greater than $k(\lambda)$, then the statistical distance between $(K \leftarrow \mathsf{Key}_F(1^\lambda),\ \mathcal{F}(K, X))$ and $(K \leftarrow \mathsf{Key}_F(1^\lambda),\ \mathcal{U}_{m(\lambda)})$ is at most $\varepsilon(\lambda)$.

Theorem 4 (Existence of extracting (puncturable) PRFs [SW14]). If OWFs exist, then for all efficiently computable functions $n(\lambda), m(\lambda), k(\lambda)$, and $e(\lambda)$ such that $n(\lambda) \geq k(\lambda) \geq m(\lambda) + 2e(\lambda) + 2$, there exists an extracting puncturable PRF family that maps $n(\lambda)$ bits to $m(\lambda)$ bits with error probability $2^{-e(\lambda)}$ for min-entropy $k(\lambda)$.

Proof. For sake of writing, we will hide the dependence of n, m, and e on λ.

Setup Let $\mathcal{H}$ be a family of 2-universal hash functions mapping $2n + e + 1$ bits to m bits. Let $\mathcal{F}$ be family of puncturable statistically injective PRFs mapping n bits to $2n + e + 1$ bits with error probability $2^{-(e+1)}$(we know $\mathcal{F}$ exists because of Theorem 3).

Defining the function We define the family $\mathcal{F}'$ as follows: The key space of $\mathcal{F}'$ consists of a key K for $\mathcal{F}$ and a hash function h chosen from $\mathcal{H}$. Then

$$\mathcal{F}'((K, h),\ x) = h(\mathcal{F}(K, x).$$

Observation If $\mathcal{F}$ were a *truly random* function, then for an independent random choice of h from $\mathcal{H}$, $h(\mathcal{F}(x))$ would be truly random function, by the *Leftover Hash Lemma*. Note that we wrote $\mathcal{F}(x)$ instead of $\mathcal{F}(K, x)$ because a truly random function needs no key.

Is the function puncturable PRF? First, is $\mathcal{F}'$ a PRF? $\mathcal{F}$ is a PRF, and thus $\mathcal{F}(K, x)$ is computationally indistinguishable from truly random function. Applying $h(x)$ on doesn't affect the distribution of $\mathcal{F}$, and thus $\mathcal{F}'$ is a PRF. From this, one can immediately deduce that since $\mathcal{F}$ is a puncturable PRF, $\mathcal{F}'$ is also a puncturable PRF.

How close is the function to uniform? Now suppose X is a distribution over $\{0,1\}^n$ such that $H_\infty(X) = k \geq m + 2e - 2$. Fix any K such that $\mathcal{F}(K, \cdot)$ is injective. Then the Leftover Hash Lemma implies that the statistical distance between $(h \leftarrow \mathcal{H}, h(\mathcal{F}(K, X)))$ and $(h \leftarrow \mathcal{H}, \mathcal{U}_m)$ is at most $2^{-(e+1)}$.

The probability that K gives a *non-injective* $\mathcal{F}(K, \cdot)$ is also at most $2^{-(e+1)}$. By union bound, we get that the statistical distance between $((K, h),\ \mathcal{F}'((K, h), X))$ and $((K, h),\ \mathcal{U}_m)$ is at most 2^{-e}, as desired. □

References

BW13 Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. *IACR Cryptology* ePrint Archive, 2013:352, 2013.

BGI13 Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology* ePrint Archive, 2013:401, 2013.

CDNO97 Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *CRYPTO*, pages 90–104, 1997.

GGM84 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the Cryptographic Applications of Random Functions. In *CRYPTO* 1984.

KPTZ13 Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. *IACR Cryptology* ePrint Archive, 2013:379, 2013.

SW14 Amit Sahai and Brent Waters. How to Use Indistinguishability Obfuscation: Deniable Encryption, and More. In *STOC* 2014.