CS 6222 Grad Crypto September 18, 2025

Lecture 8: CCA and PRFs from PRGs

Lecturer: Jack Doerner *Scribe: Eric Weng*

1 Topics Covered

- Chosen Ciphertext Attacks
- Pseudorandom Generators imply Pseudorandom Functions (the GGM Theorem)

2 Chosen Ciphertext Attacks

Recall that in a chosen plaintext attack (CPA), adversaries can not only eavesdrop on ciphertexts, but have polynomial access to an encryption oracle before and after selecting a message. This oracle allows the adversary to encrypt any plaintext message and study the resulting ciphertext.

Definition 1 (CPA Indistinguishability Game)**.** *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be an encryption scheme and* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be any two-part NUPPT adversary. Define the INDCPA game,* $\mathsf{IND\text{-}CPA}_b^{\Pi,A}(n)$*, as follows:*

1. $k \leftarrow \mathsf{Gen}(1^n)$.
2. $(m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathsf{Enc}_k(\cdot)}(1^n)$.
3. $c^* \leftarrow \mathsf{Enc}_k(m_b; r^*) : r^* \leftarrow$ *randomness domain of* Enc_k.[1]
4. *Output* $\mathcal{A}_2^{\mathsf{Enc}_k(\cdot)}(s, c^*)$.

Now consider if the adversary also had access to a decryption oracle. Such an attack method is known as a chosen *ciphertext* attack (CCA). There are two variants, depending on when the adversary can use the decryption oracle.

Definition 2 (CCA Indistinguishability Game)**.** *For scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and NUPPT* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the INDCCA1 game,* $\mathsf{IND\text{-}CCA1}_b^{\Pi,\mathcal{A}}(n)$*, is as follows:*

1. $k \leftarrow \mathsf{Gen}(1^n)$
2. $(m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(1^n)$
3. $c^* \leftarrow \mathsf{Enc}_k(m_b; r^*) : r^* \leftarrow$ *randomness domain of* Enc_k.
4. *Output* $\mathcal{A}_2^{\mathsf{Enc}_k(\cdot)}(s, c^*)$

[1] Here we give a name to the random coins used to encrypt m_b, so that we can refer to them later.

In the previous lecture we introduced a IND-CPA-secure encryption scheme from PRFs. That scheme is also IND-CCA1-secure, and the proof is very similar to the one we have already seen. Recall that we defined a set S of the random coins used by the encryption oracle in Step 2, and then reasoned about the probability that $r^* \in S$. In the IND-CCA1 game, the adversary can craft its own cipherexts that include any randomness it desires, and pass them to the decryption oracle. These adversarially-crafted values are also be included in S, and the probability that $r^* \in S$ remains negligible because r^* is sampled uniformly from an exponentially-large domain. The rest of the proof goes through as before.

Definition 3 (Adaptive CCA Indistinguishability Game)**.** *For scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *and NUPPT* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, the INDCCA2 game,* $\mathsf{IND\text{-}CCA2}_b^{\Pi,\mathcal{A}}(n)$*, is as follows:*

1. $k \leftarrow \mathsf{Gen}(1^n)$
2. $(m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(1^n)$
3. $c^* \leftarrow \mathsf{Enc}_k(m_b)$
4. *Output* $\mathcal{A}_2^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}(s, c^*)$*, but refuse to decrypt* c^*

Here, the adversary has additional access to the decryption oracle in Step 4, after it sees c^*. For this reason, CCA2 is also known as *Adaptive* CCA: the adversary can use its knowledge of c^* to craft ciphertexts on which to query the decryption oracle.

Naturally, we set up the game so that the adversary cannot recover m_b by simply decrypting c^*. However, a clever adversary might still be able to *modify* c^* so that it encrypts a message that is different from but identifiably related to m_b. In order to achieve IND-CCA2 security, we have to prevent this kind of behavior.[2] Unfortunately, the PRF-based scheme we introduced last class is *not* IND-CCA2-secure. If (for example) an adversary queries the decryption oracle with the value $c^* \oplus 1$, it will receive a decryption of $m_b \oplus 1$, and this is enough to determine the value of b.

3 Obtaining PRFs from PRGs

Recall that a pseudorandom generator (PRG) is a deterministic polynomial time function $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ such that $\{G(U_n)\}_{n\in\mathbb{N}} \approx_c \{U_{\ell(n)}\}_{n\in\mathbb{N}}$ where U_n is a random variable that is uniformly distributed over $\{0,1\}^n$. In contrast, a pseudorandom *function* (PRF) family is a set of functions $\{F_k : \{0,1\}^{|k|} \to \{0,1\}^{\ell(|k|)}\}_{k\in\{0,1\}^*}$ such that a randomly sampled member of the set is computationally-oracle-indistinguishable from a function sampled randomly from the set of *all* functions with the same domain and range. In other words, if $\mathcal{F}_{n,\ell(n)} = \left\{f : \{0,1\}^n \to \{0,1\}^{\ell(n)}\right\}$ is the set of all functions from n bits to $\ell(n)$ bits, then for all NUPPT $\mathcal{A}$,

$$\left\{\mathcal{A}^{F_k(\cdot)}(1^n) : k \leftarrow \{0,1\}^n\right\}_{n\in\mathbb{N}} \approx_c \left\{\mathcal{A}^{f(\cdot)}(1^n) : f \leftarrow \mathcal{F}_{n,\ell(n)}\right\}_{n\in\mathbb{N}}$$

[2] On the other hand, this kind of behavior is sometimes desirable. For further reading, search up *homomorphic encryption*.

In this lecture, we will prove only one theorem:

Theorem 1 (Goldreich-Goldwasser-Micali [GGM84])**.** $\exists$ *PRG* $\Rightarrow \exists$ *PRF* .

Proof. In Lecture 5, we proved that we can create a PRG with any polynomial stretch from a PRG with one-bit stretch., so without loss of generality, let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG. In this proof we will construct a length-preserving PRF. Given such a PRF and a PRG, it is easy to construct a PRF with any polynomially-bounded output length.

Next, define $G_b : \{0,1\}^n \to \{0,1\}^n$ for $b \in \{0,1\}$ such that $\forall x \in \{0,1\}^n$, $G_0(x) \| G_1(x) = G(x)$. Essentially, we will use $G_b(x)$ to run $G(x)$ and take either the first or last n bits of the $2n$-bit output of G.

Construction 1 (Target PRF)**.** *Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a PRF that accepts n-bit key and n-bit input, and returns a n-bit output. Let F be computed as follows:*

$$F : (k, x) \mapsto G_{x_n}(G_{x_{n-1}}(\ldots G_{x_2}(G_{x_1}(k))\ldots)) : x_1 \| x_2 \| \ldots \| x_{n-1} \| x_n = x$$

or equivalently

$$F : (k, x) \mapsto G_{x_n} \circ G_{x_{n-1}} \circ \ldots \circ G_{x_2} \circ G_{x_1}(k) : x_1 \| x_2 \| \ldots \| x_{n-1} \| x_n = x.$$

In other words, recursively call G on k, and use each bit of x to determine which half of the output to keep at the corresponding level of recursion.

We would like to construct a hybrid argument, using the PRG security of G to show that each pair of hybrids is computationally indistinguishable. It might be temting to represent the evaluation of the PRF as a tree,[3] where each leaf corresponds to a single input (and each level to the intermediate output of a recursive call to G), and then specify one hybrid distribution for each node, changing that node to a uniformly distributed value, in topological order. However, we have 2^n possible inputs for F in n layers, and the hybrid lemma only works with a *polynomial* number of related distributions.

To get around this problem, notice that the adversary can only query the oracle in the PRF game polynomially-many times, which means that it can observe at most polynomially many leaves and interior nodes in this tree. Our strategy will be to replace *only these nodes* with uniformly distributed values, using the hybrid lemma.

We begin with a lemma formalizing our intuition that a PRG can be evaluated polynomially many times in parallel, so long as the inputs are independent of one another.

Lemma 1. *If $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a PRG and ℓ and t are polynomials, then*

$$\left\{\{G(s_j)\}_{j\in[t(n)]} : s_j \leftarrow \{0,1\}^n\right\}_{n\in\mathbb{N}} \approx_c \left\{\{y_j\}_{j\in[t(n)]} : y_j \leftarrow \{0,1\}^{\ell(n)}\right\}_{n\in\mathbb{N}}.$$

[3] See the binary tree in Figure 1.

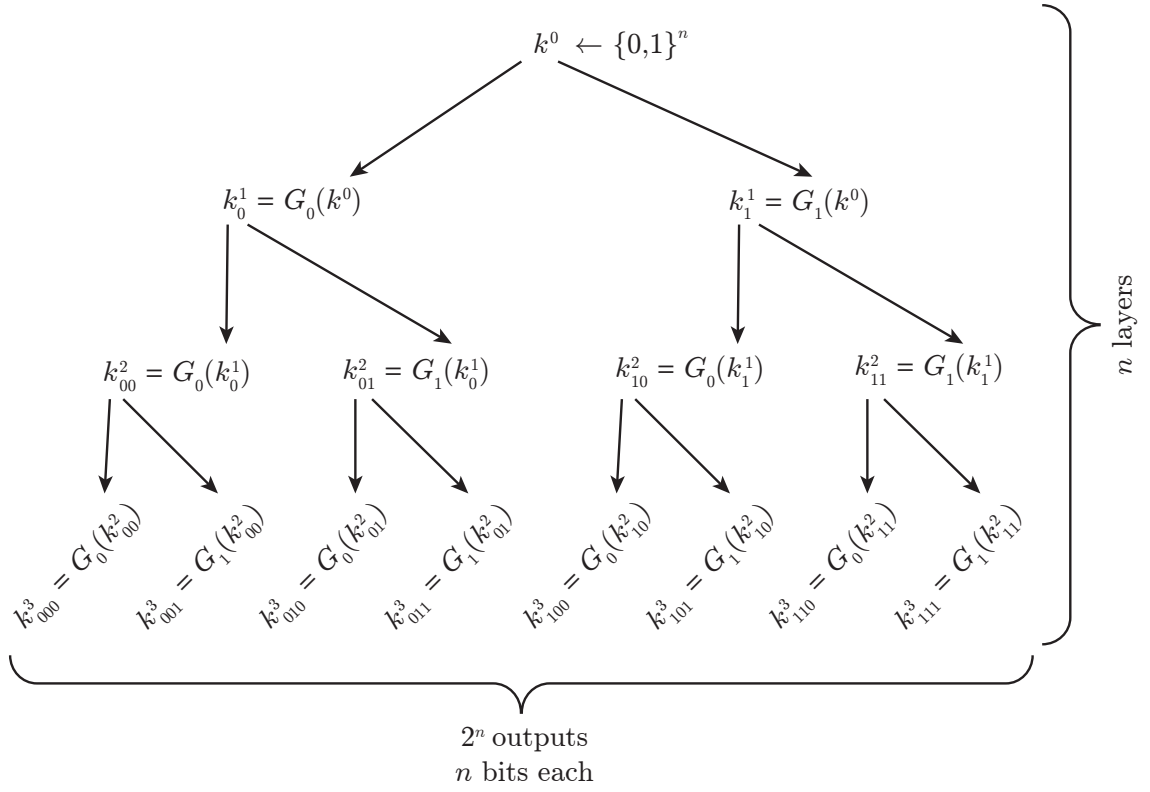

Figure 1: Possible PRF execution paths for $n = 3$. Note that $k^0 = k$ is the PRF key, and at each layer execution takes the left or right fork based upon the value of the corresponding bit of the input x. The output is k_x^n.

Proof. Let $H_n^i = (G(s_1), \ldots, G(s_{t(n)-i}), y_1, \ldots, y_i) : s_j \leftarrow \{0,1\}^n, y_j \leftarrow \{0,1\}^{\ell(n)}$ be a hybrid distribution. The first $t(n) - i$ terms are PRG evaluations, and the remaining i terms are uniform.

By the hybrid lemma, if $\exists$ PPT D_n and some value $\delta_n \in \mathbb{R}_{\geq 0}$, s.t.

$$\left|\Pr[D_n(1^n, H_n^0) = 1] - \Pr[D_n(1^n, H_n^{t(n)}) = 1]\right|_{n\in\mathbb{N}} \geq \delta_n,$$

then $\exists\ i_n \in [t(n) - 1]$ s.t.

$$\left|\Pr[D_n(1^n, H_n^{i_n}) = 1] - P[D_n(1^n, H_n^{i_n+1}) - 1]\right| \geq \frac{\delta_n}{t(n)}.$$

Next, consider the reduction

$$R_n : x \mapsto D_n((G(s_1), \ldots, G(s_{t(n)-i_n-1}), x, y_1, y_{i_n})) : s_j \leftarrow \{0,1\}^n, y_j \leftarrow \{0,1\}^{\ell(n)}$$

where i_n is the value known to exist due to the hybrid lemma, above. Notice that if we take U_n to be a random variable uniformly distributed over $\{0,1\}^n$, then $R_n(G(U_n))$ is distributed identically to $H_n^{i_n}$ and $R_n(G(U_{\ell(n)}))$ is distributed identically to $H_n^{i_n+1}$. Therefore,

$$\left|\Pr[R_n(G(U_n)) = 1] - P[R_n(U_{\ell(n)}) = 1]\right| \geq \frac{\delta_n}{t(n)}.$$

So far we have defined both the adversary and its distinguishing advantage for a single security parameter value only. We can construct a single NUPPT adversary $D = \{D_n\}_{n\in\mathbb{N}}$ and an advantage function $\delta(n) = \delta_n$. If such an adversary exists, then there exists a NUPPT reduction R with distinguishing advantage no less than $\frac{\delta(n)}{t(n)}$, which is non-negligible if $\delta(n)$ is. Lemma 1 holds by contraposition. □

Now we will define an oracle that takes the place of F_k or f in the PRG game, and lazily fills in the necessary elements in a truncated version of the tree that was defined in Figure 1, as the adversary queries various leaf values. The tree constructed by our oracle will be truncated: the nodes at some specified level are sampled randomly, and the nodes between that level and the leaves are computing using G. Thus we can use the oracle to define a sequence hybrid experiments, one for each level.

Construction 2 (Lazy Tree Oracle)**.** *Consider the oracle $\varphi_n^i : \{0,1\}^n \to \{0,1\}^n$ that receives input $x_1\| \ldots \|x_n = x$.*

- *If $k^i_{x_1\|\ldots\|x_i}$ is not defined, sample $k^i_{x_1\|\ldots\|x_i} \leftarrow \{0,1\}^n$ and add it to set S^i.*
- *For $j \in [i+1, n]$, if $k^j_{x_1\|\ldots\|x_j}$ is not defined, let $k^j_{x_1\|\ldots\|x_j} := G_{x_j}(k^{j-1}_{x_1\|\ldots\|x_{j-1}})$ and add it to set S^j.*
- *Output k^n_x.*

An illustrated example of the operation of our oracle is given in Figure 2.

Claim 1. $\forall D$, $\Pr\left[D^{F_k}(1^n) = 1 : k \leftarrow \{0,1\}^n\right] = \Pr\left[D^{\varphi_n^0}(1^n) = 1\right]$.

The above claim holds because the oracle produces exactly the same distribution as the pseudorandom function (although it is defined lazily) in this case.

Claim 2. $\forall D$, $\Pr\left[D^{f}(1^n) = 1 : f \leftarrow \mathcal{F}_{n,n}\right] = \Pr\left[D^{\varphi_n^n}(1^n) = 1\right]$.

The above claim holds because the oracle's outputs are all uniformly-random n-bit strings in this case, which is identical to the distribution of outputs produced by a random function.

Claim 3. *$\forall$ NUPPT D, $\exists$ polynomial p s.t. $\forall i \in [n], j \in [n]$,*

$$\Pr\left[|S^j| < p(n) : S^j \text{ is the set constructed by } \varphi_n^i \text{ in } D^{\varphi_n^i}(1^n)\right] = 1.$$

Claim 4. *The difference between the distributions of the random variables $D^{\varphi_n^i}(1^n)$ and $D^{\varphi_n^{i+1}}(1^n)$ is completely characterized by the fact that*

- *In $D^{\varphi_n^i}(1^n)$, $S^{i+1} \subset \{G_b(k_x^i) : k_x^i \leftarrow \{0,1\}^n\}_{b\in\{0,1\}, x\in\{0,1\}^i}$,*

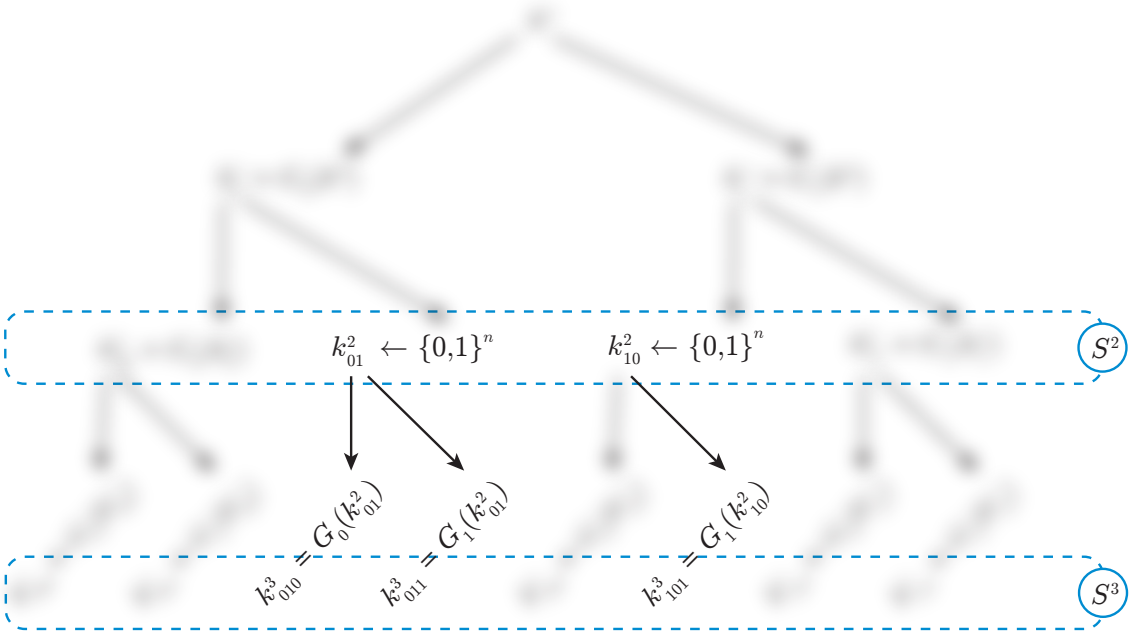

Figure 2: Example of the operation and internal state of oracle φ^2_3 after receiving the queries $\{010, 011, 101\}$. Blurred elements are not defined.

- *In* $D^{\varphi_n^{i+1}}(1^n)$, $S^{i+1} \subset \{k_x^{i+1} : k_x^{i+1} \leftarrow \{0,1\}^n\}_{x \in \{0,1\}^{i+1}}$.

Note that in both cases, we have $S^{i+1} \subset \vec{y}$ where $\vec{y}$ is distributed over $\{0,1\}^{n \times 2i}$. In the former case, $\vec{y}$ comprises i distinct PRG outputs of length $2n$, with the first and last n bits of each output being included in $\vec{y}$ separately. In the latter case, $\vec{y}$ comprises $2i$ random values, each of length n. In both cases, S^{i+1} is formed from $\vec{y}$ by taking a specific subset of the values in $\vec{y}$, and by Claim 3, the size of this subset is at most $p(n)$ for some polynomial p. Thus our oracle defines a sequence of hybrid distributions, as we intended, and each successive hybrid replaces an at-most-polynomial number of PRG outputs with uniformly-sampled values.

We can recast our vector $\vec{y} \in \{0,1\}^{n \times 2i}$ as a vector $\vec{z} \in \{0,1\}^{2n \times i}$ containing either i (undivided) PRG outputs of length $2n$, or i uniform $2n$-bit strings. Notice that S^{i+1} depends upon at most $p(n)$ elements of $\vec{z}$. Thus we can construct S^{i+1} using exactly $p(n)$ length-$2n$ PRG outputs or uniform bitstrings.

Claim 5. *By the hybrid lemma and Claims 1 and 2, if* $\exists$ *PPT* D_n *and some value* $\delta_n \in \mathbb{R}_{\geq 0}$ *such that*

$$\left| \Pr\left[D_n^{F_k}(1^n) = 1 : k \leftarrow \{0,1\}^n\right] - \Pr\left[D_n^f(1^n) = 1 : f \leftarrow F_{n,n}\right] \right| \geq \delta_n,$$

then $\exists\ i_n \in [0, n-1]$ *such that*

$$\left| \Pr\left[D_n^{\varphi_n^{i_n}}(1^n) = 1\right] - \Pr\left[D_n^{\varphi_n^{i_n+1}}(1^n) = 1\right] \right| \geq \frac{\delta_n}{n}.$$

Now consider a reduction R_n which has knowledge of i_n hard-coded.[4] Given some input $\vec{w} \in \{0,1\}^{2n \times p(n)}$,[5] $R_n(1^n, \vec{w})$ emulates $D_n^{\varphi_n^{i_n}}(1^n)$ internally, but uses $\vec{w}$ to build the set S^{i_n+1} inside of the oracle $\varphi_n^{i_n}$. Notice that this reduction is PPT if D_n is PPT.

Claim 6. *By Claim 4 and the structure of R_n,*

$$\Pr\left[R_n(1^n, (G(s_1), \ldots, G(s_{p(n)}))) = 1 : \vec{s} \leftarrow \{0,1\}^{n \times p(n)}\right] = \Pr\left[D_n^{\varphi_n^{i_n}}(1^n) = 1\right]$$

Claim 7. *By Claim 4 and the structure of R_n,*

$$\Pr\left[R_n(1^n, \vec{w}) = 1 : \vec{w} \leftarrow \{0,1\}^{2n \times p(n)}\right] = \Pr\left[D_n^{\varphi_n^{i_n+1}}(1^n) = 1\right]$$

Once again, we have considered only individual values of the security parameter in our previous claims (each with a specific PPT adversary and real-valued advantage). Now we generalize to a NUPPT adversary $D = \{D_n\}_{n \in \mathbb{N}}$, and NUPPT reduction $R = \{R_n\}_{n \in \mathbb{N}}$, and an advantage function $\delta(n) = \delta_n$.

Claim 8. *By Claims 5-7, if $\exists$ NUPPT D and some function $\delta : \mathbb{N} \to \mathbb{R}_{\geq 0}$ s.t.*

$$\left|\Pr\left[D^{F_k}(1^n) = 1\right] - \Pr\left[D^f(1^n) = 1 : f \leftarrow \mathcal{F}_{n,n}\right]\right| \geq \delta(n),$$

Then $\exists$ NUPPT R s.t.

$$\left|\begin{array}{r} \Pr\left[R(1^n, (G(s_j))_{j \in [p(n)]}) = 1 : \vec{s} \leftarrow \{0,1\}^{n \times p(n)}\right] \\ - \Pr\left[R(1^n, \vec{w}) = 1 : \vec{w} \leftarrow \{0,1\}^{2n \times p(n)}\right] \end{array}\right| \geq \frac{\delta(n)}{n}.$$

Notice that in Claim 8, if δ is a non-negligible function, then we can view our reduction R as an adversary that contradicts Lemma 1. In other words, to prove Theorem 1, suppose toward contradiction that $\exists$ NUPPT D that oracle-distinguishes our PRF F_k from a truly random function f with advantage no less than some non-negligible function $\delta(n)$. By combining the reductions from Claim 8 and the proof of Lemma 1, $\exists$ NUPPT R' that distinguishes $G(s) : s \leftarrow \{0,1\}^n$ from a uniformly random value with advantage $\delta(n)/(n \cdot p(n))$, which is non-negligible,[6] contradicting the PRG security of G. Formally,

$$\left|\Pr\left[D^{F_k}(1^n) = 1 : k \leftarrow \{0,1\}^n\right] - \Pr\left[D^f(1^n) = 1 : f \leftarrow F_{n,n}\right]\right| \geq \delta(n)$$

$$\Rightarrow \left|\Pr\left[R'(1^n, G(s)) = 1 : s \leftarrow \{0,1\}^n\right] - \Pr\left[R'(1^n, y) = 1 : y \leftarrow \{0,1\}^{2n}\right]\right| \geq \frac{\delta(n)}{n \cdot p(n)} \qquad \square$$

References

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984.

[4] Note that this R_n and D_n are not the same as the ones we considered when proving Lemma 1.

[5] $\vec{w}$ can be thought of as containing the $p(n)$ elements of $\vec{z}$ on which S^{i_n+1} depends, as per the above discussion.

[6] Recall that p is a polynomial that depends upon D, per Claim 3.