University of Illinois, Urbana Champaign
CS/ECE 498AC3/4 Applied Cryptography

Instructor: Dakshita Khurana
Scribe: Ian George & Andrew Liu
Date: August 27, 2020

LECTURE
2

Pseudorandom Generators (PRGs), Stream Ciphers

Last lecture, we were introduced to our first "secure" encryption scheme, **One Time Pad** (OTP). OTP consisted of an encryption and decryption algorithm. As we will see in this lecture, all ciphers are of this form. In the last lecture we also proved that OTP satisfies **Perfect Security**. However, we saw that OTP is defined such that the key space is the same size as the message space, i.e. $|\mathcal{K}| = |\mathcal{M}|$. This assumption is not practical for general real-life applications as it requires using large encryption keys for large messages. We had ended that lecture posing the question as to whether or not there existed encryption schemes with perfect secrecy such that $|\mathcal{K}| < |\mathcal{M}|$.

This lecture we begin by reviewing OTP. Then we will prove two important claims, 1. that OTP satisfies perfect secrecy and 2. answering the previous question in the negative. That is, we show perfect secrecy requires that $|\mathcal{K}| \geq |\mathcal{M}|$. This motivates the need for ideas which would allow us to achieve security in a practical (rather than perfect) sense. To do this, we introduce the **pseudorandom generator** which is a function that expands a small amount of randomness into a larger amount of pseudorandomness. We then introduce the **Stream Cipher** encryption scheme which uses a pseudorandom generator to encrypt messages of much larger length than the length of the available secret key. To justify why the Stream Cipher is practically secure while not being perfectly secure, we introduce the definition of **semantic security** and its abstraction **computational indistinguishability**. Finally, we present formal proof the Stream Cipher is semantically secure justifying our earlier intuitions.

In this lecture we therefore see that while we can't achieve perfect security, by allowing a relaxed notion of security that is still robust in real life applications, semantic security, we can build an encryption scheme which is both (semantically) secure and practical to implement. It is then the notion of computational indistinguishability, which semantic security is a specific instance of for ciphers, which will allow us throughout this course to construct implementable, secure cryptosystems.

2.1 Intro to Shannon Ciphers

How do you obtain secure communication over an untrusted channel? This is done by a **cipher**, two algorithms (Enc, Dec) that are defined over the sets $\{\mathcal{K}, \mathcal{M}, \mathcal{C}\}$. In a cipher, a shared secret key is used by the sender and recipient in order to encrypt message to ciphertext and decrypt from ciphertext to message respectively. The following is the definition of a specific cipher, the **Shannon Cipher**.

DEFINITION 2.1. (Shannon Cipher) Let $\mathcal{K}$ be the set of all possible keys, $\mathcal{M}$ the set of all possible messages, and $\mathcal{C}$ the set of all possible ciphertexts. The Shannon Cipher can be described as $\mathcal{E} = (Enc, Dec)$, where Enc is the encryption function and Dec is the decryption function.
More formally, we can define Enc and Dec as follows, where $m \in \mathcal{M}$,$k \in \mathcal{K}$, $c \in \mathcal{C}$:

$$Enc(k, m) = c$$

$$Dec(k, c) = m$$

The encryption function takes a key and message, and outputs a ciphertext. The decryption function takes a key and ciphertext, and outputs a message. However, a correct cipher should ensure that if a message $m_0 \in \mathcal{M}$ is encrypted with key $k_0 \in \mathcal{K}$, which results in ciphertext $c_0 \in C$, then when c_0 is decrypted with k_0, the result should be m_0.

COROLLARY 2.2. *If $\mathcal{E} = (Enc, Dec)$ is a cipher, then the following must be true for $k \in K$, $m \in M$:*

$$Dec(k, Enc(k, m)) = m$$

This corollary pretty much defines what it means for a scheme to be correct. If the receiver is not able to attain the message that the sender intended, then useful communication between the two parties is not at all possible.

2.2 One Time Pad

So far, we have defined the basics of a cipher, but we are still left with what exactly the $Dec(k, c)$ and $Enc(k, m)$ consists of. Next we will define a specific Shannon Cipher, the **One Time Pad**.

DEFINITION 2.3. (One Time Pad) Let $\mathcal{K} := \{0,1\}^n$, $\mathcal{M} := \mathcal{C} := \{0,1\}^{\leq n}$, where n is a fixed natural number. For $k \in \mathcal{K}$, $m \in \mathcal{M}$, $c \in \mathcal{C}$, Enc and Dec of One Time Pad are defined as follows:

$$Enc(k, m) := k \oplus m$$

$$Dec(k, c) := k \oplus c$$

CLAIM 2.4. ***One Time Pad follows Corollary 2.2***

Proof. Given $k \in \mathcal{K}$, $m \in \mathcal{M}$ we will show that $Dec(k, Enc(k, m)) = m$ holds.

$$Dec(k, Enc(k, m)) = m$$

$$Dec(k, k \oplus m) = m$$
$$k \oplus (k \oplus m) = m$$
$$(k \oplus k) \oplus m = m$$
$$(0....0)^{|k|} \oplus m = m$$
$$m = m$$

□

We use the fact that anything XOR with itself is a bit string of 0s, and anything XOR with a bit string of 0s is itself.
The main goal of any cipher is to allow a message to be transfered from a sender to a receiver over an unprotected channel. So how do we determine if a cipher fulfills this goal?
There are answers to this question, as we will later see, but for now we will define **Perfect Secrecy**, or Information-Theoretical Security.

DEFINITION 2.5. (Perfect Secrecy) Let $\mathcal{E} = (Enc, Dec)$ be an encryption scheme defined over (K, M, C). Consider a probabilistic experiment where $k \leftarrow \mathcal{K}$. If $\forall m_0, m_1 \in \mathcal{M}$ and all $c \in \mathcal{C}$, we have

$$Pr[Enc(k, m_0) = c] = Pr[Enc(k, m_1) = c]$$

then we say that $\mathcal{E}$ is perfectly secure.

This definition states that given a choice of two plaintexts, for a given ciphertext and key, it is impossible for an adversary to determine which plaintext was the one that mapped to the ciphertext using the key .
There are alternate definitions of perfect secrecy as well.

DEFINITION 2.6. (Perfect Secrecy alt.) Let $\mathcal{E} = (Enc, Dec)$ be an encryption scheme defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. If $\forall m \in \mathcal{M}$ and all $c \in \mathcal{C}$, we have

$$Pr[M = m|C = c] = Pr[M = m]$$

then we say that $\mathcal{E}$ is perfectly secure.

This states that if an eavesdropper views a ciphertext, no information will be revealed about the plaintext by the ciphertext. Ultimately what this is saying is that any scheme that has perfect secrecy leaks *no information* of an encrypted message, even to adversaries with *unlimited computational power*. Now that we have defined Perfect Secrecy, we will show that One Time Pad satisfies it.

CLAIM 2.7. ***One Time Pad is perfectly secure*** *[1]*

Proof. Let $c \in \mathcal{C}$, $m \in \mathcal{M}$, $k \in \mathcal{K}$, and $l := |\mathcal{C}| = |\mathcal{M}| = |\mathcal{K}|$
First we will compute $Pr[C = c|M = m']$ for arbitrary $c \in \mathcal{C}$ and $m' \in \mathcal{M}$

$$Pr[C = c|M = m']$$
$$= Pr[Enc_k(m') = c]$$

$$= Pr[m' \oplus K = c]$$
$$= Pr[K = m' \oplus c]$$
$$= 2^{-l}$$

Next we will compute $Pr[C = c]$ for any $c \in \mathcal{C}$, using what we just proved

$$Pr[C = c] = \sum_{m' \in M} Pr[C = c | M = m'] * Pr[M = m']$$
$$= 2^{-l} \sum_{m' \in M} Pr[M = m']$$
$$= 2^{-l} * 1$$
$$= 2^{-l}$$

With these results, and using Bayes Thm., we will show that One Time Pad follows Definition 2.6, and is thus perfectly secure

$$Pr[M = m | C = c] = \frac{Pr[C = c | M = m] Pr[M = m]}{Pr[C = c]}$$
$$= \frac{2^{-l} Pr[M = m]}{2^[-l]}$$
$$= Pr[M = m]$$

□

Alternatively, we can prove a more general statement, that if the length of the key is less than the length of a message for a cipher, then that cipher cannot attain perfect secrecy.

CLAIM 2.8. ***If*** $\mathcal{E} = (Enc, Dec)$ ***is a perfectly secret encryption scheme defined over*** $(\mathcal{K}, \mathcal{M}, \mathcal{C})$***, then*** $|\mathcal{K}| \geq |\mathcal{M}|$

Proof. Assume $|\mathcal{K}| < |\mathcal{M}|$. Let $c \in \mathcal{C}$ be a ciphertext that occurs with non-zero probability. Let $\mathcal{M}(c) = \{m | m = Dec_k(c) \text{ for some } k \in K\}$. Beacause $Dec()$ is deterministic, and $|\mathcal{K}| < |\mathcal{M}|$, then we can see that there exists $m' \in \mathcal{M}$ such that $m' \notin \mathcal{M}(c)$. Then, $Pr[M = m' | C = c] = 0 \neq Pr[M = m']$, and thus by Definition 2.6, the scheme is not perfectly secure. □

Therefore, we know that One Time Pad is perfectly secure, and a perfectly secure scheme must have key length greater or equal to message length.

2.3 Pseudorandom Generators

Although One Time Pad may seem perfect for sending messages over unsecure channels, there is a major drawback. As we have seen in the previous section, a perfectly secure scheme must have key length greater or equal to message length. This brings up an issue: how to securely guarantee that both sender and receiver receive the secret key they want to use without any adversaries gaining ahold of it.

For example, during the Cuban Missile Crises the US and Soviet Union embassy delivered long rolls of printed bits to act as secret keys. They had agents carry brief cases of paper printed with the keys [2]. This is impractical, and leads us to consider alternate cipher schemes, were the key length does not need to be as long as the message length. However, we just provided in *Claim 2.8* that for a scheme to be perfectly secure, then key length must be greater or equal to message length. So how do we create a scheme that has shorter key length?
The first step for creating such a a scheme is a pseudorandom generator. A **Pseudorandom Generator** is a deterministic function that maps a domain of n bit strings to a domain of m bit strings, such that its output is difficult to determine from a uniform distribution.
More formally, for any Pseudorandom Generator, the following claim must hold true.

CLAIM 2.9. *For PRG:* $\{0,1\}^n \to \{0,1\}^m$*, where* $m > n$*, and* ε *is a negligible value, then for any probabalistic polynomial time adversary* A*,* $k \leftarrow \{0,1\}^n$*,* $j \leftarrow \{0,1\}^m$*, the following is true*

$$|Pr[\mathcal{A}(PRG(k)) = 1] - Pr[\mathcal{A}(j) = 1]| \leq \varepsilon \tag{2.1}$$

This essentially states that it is hard for such an adversary to tell the difference between the m-bit output of a PRG with a random n-bit seed, and a m-bit string chosen uniformly from the set of all m-bit strings.

This claim is similar to the 2 game scenario we discussed in lecture 2. We will discuss what negligibility is in Section 2.5. Some other properties of a pseudorandom generator are as follows:

- $m > n$. The length of the output of the PRG must be longer than the input
- The PRG must satisfy the **next bit test**: Given the first n output of a PRG, a computationally bounded adversary cannot successfully guess the $n+1$th output with probability non-negligibly greater than 50%.
- If a portion of the bits produced by the PRG is revealed, it should be impossible for a computationally bounded adversary to re-construct the previous outputt-ed bits.

2.4 Stream Cipher

Lets use what we just learned, the pseudorandom generator, to create a new cipher. We can input a *short random bit-string* to the PRG, and use the *longer pseudorandom bit-string* as the mask for the message. We will call the encryption and decryption algorithms that use this technique as the **Stream Cipher**.

DEFINITION 2.10. A stream cipher consists of a set of algorithms *Enc* and *Dec*, over message set $\mathcal{M}$, key set $\mathcal{K}$, and cipher text set $\mathcal{C}$. Let $m \in \mathcal{M}$, $k \in \mathcal{K}$, and $c \in \mathcal{C}$. Let l_1 be the size of k, and let l_2 be the size of m,c; $l_1 < l_2$. Define Pseudorandom Generator PRG with input size l_1 and output size l_2.

$$Enc(k, m) = m \oplus PRG(k)$$
$$Dec(k, c) = c \oplus PRG(k)$$

Great! We learned a new cipher that uses a key smaller than the message size. But, is this cipher secure? As shown by Claim 2.8, a Stream Cipher cannot be perfectly secret. However in the next section, we will define a new form of security that is not as strict.

2.5 Semantic Security

So far we have presented the intuition of what we want a psuedorandom generator to achieve, specifically the "moral equivalent" of applying a uniform mask, and in Claim 2.9 we stated the formal notion of this "moral equivalent." Specifically, the claim states that Eqn. 2.1 may be interpreted as the ability of a (probabilistic polynomial time) adversary $\mathcal{A}$ to distinguish between being given $j \leftarrow \{0,1\}^m$ and $PRG(k)$ where $k \leftarrow \{0,1\}^n$. We can see that this is the correct interpretation by the setup of a sort of comparison between games. In one game the adversary is given j, in the other, $PRG(k)$. The job of the adversary is then to correctly guess which game it played. The intuition is that if the adversary can't tell the difference between these two games except for some 'negligible amount' then in effect they aren't any different from the adversary's perspective.
Claim 2.9 also required that the adversary $\mathcal{A}$ was a 'probabilistic polynomial time' adversary. What reason did we have for this? Intuitively, one should expect that an adversary that has enough resources should be able to tell the difference between the PRG and the uniform random key because the PRG doesn't even map to all strings in $\{0,1\}^m$. However, from a practical standpoint this shouldn't be an issue because in real life any adversary will have to be an efficient algorithm. The intuition is this limitation is equivalent to limiting the amount of time the adversary has to make a decision on which game was played. We are therefore interested in when the adversary needs to come to its conclusion efficiently, and our choice of modeling this notion of efficiency is that the adversary must make its decision about which game it is playing in polynomial time. This means that the adversary must be a probabilistic polynomial time Turing machine (PPT TM).
These ideas about distinguishing between games and bounding the power of the adversary then may be applied to the security of ciphers. The security of a cipher in which the adversary is a PPT TM and can only negligibly distinguish between the games with negligible probability is called **semantic security**.

Definition 2.11 (Semantic Security as a Game). Consider a cipher $\mathcal{E}$. The cipher $\mathcal{E}$ is semantically secure if for all $m_0, m_1 \in \mathcal{M}$ and probabilistic polynomial time adversary $\mathcal{A}$, it is the case that

$$|\Pr[\mathcal{A}(y) = 1|\text{Game 1}] - |\Pr[\mathcal{A}(y) = 1|\text{Game 2}]| \leq \varepsilon \ , \tag{2.2}$$

where $y = Enc(k, m_1), k \leftarrow \mathcal{K}$ for Game 1 and $y = Enc(k, m_2)$ for Game 2 and ε is negligible.

There are three points to note about this notion of security. First, it is intuitively a natural formal 'loosening' of perfect security in which we might expect $\mathcal{A}$ wouldn't need to be restricted to being a PPT TM and ε could be replaced with zero. Second, one might note that the definition of semantic security is for ciphers but we motivated the definition using the pseudorandom generator which is not a cipher; this gap will be resolved in Section 2.7. Lastly, and most importantly, the definition of semantic security does not explain what it means for ε to be 'negligible.' We must now address this.

Status	Value of ε	Reasoning
Non-negligible	2^{-30}	A good enough computer could probably run at speed 2^{30}
Negligible	2^{-80}	Even for large files a computer can't solve this

FIGURE 2.1: The partitioning of negligible and non-negligible scalar security parameters in implementations of ciphers.

2.6 Negligible Functions

Looking at the equation in the definition of semantic security (Eqn. 2.2) we see it would make most sense for ε to be a scalar as the absolute value between to probabilities clearly is a scalar. In fact, in practical implementations this is the case as presented in Figure 2.6. However, from a theoretic stand point ε is a special type of function called a **negligible function**. The reason for this as we will see in Section 2.7 is that really there is some security parameter $\lambda \in \mathbb{Z}_{\geq 0}$ which of course formalizes the security of the cryptographic tool. This security parameter λ is correlated with certain properties. For example, for a cipher, the length of the bit strings which populate $\mathcal{K}, \mathcal{M}, \mathcal{C}$ grows as λ increases. Given that the adversary is a PPT TM, we want it such that for some sufficiently large security parameter $\lambda > \lambda_0$, our value of 'negligibility' is smaller than the inverse of any polynomial. This is because intuitively this would imply there is no PPT TM which can successfully distinguish the two games fast enough. A negligible function is a function which captures this idea.

DEFINITION 2.12 (Negligible Function [3]). A function $\varepsilon : \mathbb{Z}_{\geq 1} \to \mathbb{R}$ is said to be negligible if for all $d \in \mathbb{R}_{>0}$, there exists $\lambda_0 \in \mathbb{Z}_{\geq 1}$ such that for all $\lambda > \lambda_0$, $\varepsilon(\lambda) \leq \frac{1}{\lambda^d}$.

Naturally a non-negligible function is simply a function which does not satisfy the definition of a negligible function.

EXAMPLE 2.13. It is worthwhile to consider a few examples of negligible and non-negligible functions to better understand what is required of them and what they provide us.

- The function $f(\lambda) := \frac{1}{2^\lambda}$ is a negligible function. This is because for any $d \in \mathbb{R}_{>0}$, one can find λ such that $\lambda > d \log_2(\lambda)$, and
$$\lambda > d \log_2(\lambda) \Rightarrow 2^\lambda > \lambda^d \Rightarrow \frac{1}{2^\lambda} \leq \frac{1}{\lambda^d} \ .$$
- The function $f(\lambda) := \frac{1}{\lambda^{30000}}$ is not a negligible function. This is because for any $d > 30000$, it is the case that $f(\lambda) > \frac{1}{\lambda^d}$.
- The function
$$f(\lambda) := \begin{cases} \frac{1}{2^\lambda} & \lambda \text{ is even} \\ \frac{1}{\lambda^{30000}} & \lambda \text{ is odd} \end{cases}$$
is not negligible. This is because if I pick an odd λ, by the previous example, we know there exists $d \in \mathbb{R}_{>0}$ such that $f(\lambda) > \frac{1}{\lambda^d}$. It follows that for any $d > 30000$, there cannot exist λ_0 such that for all $\lambda > \lambda_0$, $f(\lambda) \leq \frac{1}{\lambda^d}$ as every time λ is odd the condition won't hold.

The takeaway of the above examples is that for any choice of $d \in \mathbb{R}_{>0}$ the negligible function is defined to be well behaved in its negligibility once it achieves it. This is important in our notions of security as not requiring this would imply sometimes one is secure for some choice of security parameter λ_1 but for some choice of security parameter $\lambda_2 > \lambda_1$ the security has gone away.

2.7 Rigorous Definition of PRG

Having presented negligible functions, we are in a good position to present the formal definitions of a PRG which in turn will motivate the concept of **computational indistinguishability**.

Definition 2.14 (Pseudorandom Generator (Formal)). A pseudorandom generator (PRG) is a family of (efficient) deterministic functions, $\{G_\lambda\}_{\lambda \in \mathbb{Z}_{\geq 1}}$ where

$$G_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\text{poly}(\lambda)}$$

such that there exists $\lambda_0 \in \mathbb{Z}_{\geq 1}$ such that for all $\lambda > \lambda_0$ and PPT TM adversary $\mathcal{A}$,

$$|\Pr_{k \leftarrow \{0,1\}^\lambda}[\mathcal{A}(G_\lambda(k)) = 1] - \Pr_{y \leftarrow \{0,1\}^{\text{poly}(\lambda)}}[\mathcal{A}(y) = 1]| \leq \varepsilon(\lambda) \tag{2.3}$$

where ε is a negligible function.

First we note this is exactly what Claim 2.9 was saying in less technical terms. Second, we see that what this says is that a PRG is a set of functions which when the security parameter λ is chosen large enough will be sufficiently indistinguishable from a uniformly random string to any PPT TM adversary. Lastly, it is worth pointing out that the view of security in terms of games is not lost; we just need to say whether or not the condition we place on the game is satisfied depends on the security parameter λ. As the indexing on the security parameter λ seems to not add much to the discussion, it will often be ignored as was done earlier in this lecture. Sometimes in statements we will say 'for sufficiently large λ,' which is shorthand for saying that there does in fact exist $\lambda_0 \in \mathbb{Z}_{\geq 1}$ such that for all $\lambda > \lambda_0$, some negligibility condition we are interested in is met.

2.8 Computational Indistinguishability

We can see from the discussion of semantic security (Definition 2.11) and the above definition of PRG that underlying both of these ideas is that the PPT TM adversary $\mathcal{A}$ cannot tell the difference between the probability distribution over ideal inputs and the probability distribution over inputs generated by our cryptographic tool (whose probability distribution may be induced by a probability distribution over an input itself[1]). It follows we could just abstract this idea of security for arbitrary probability distributions. This is exactly the idea of computational indistinguishability.

[1] For example, the distribution over PRG's outputs, which is the probability distribution the adversary tests, is induced by the particular PRG along with the fact its input is an input that is uniformly sampled from the PRG's input space.

Definition 2.15 (Computational Indistinguishability). Let $P_{1,\lambda}, P_{2,\lambda}$ be probability distributions over $\{0,1\}^{\text{poly}(\lambda)}$. P_1 and P_2 are computationally indistinguishable, denoted $P_1 \approx_c P_2$, if for all PPT adversaries $\mathcal{A}$, there exists negligible function $\varepsilon(\cdot)$ and sufficiently large λ such that

$$|\Pr[\mathcal{A}(P_{1,\lambda}) = 1] - \Pr[\mathcal{A}(P_{2,\lambda}) = 1]| \leq \varepsilon(\lambda) \ .$$

Thus we can see this should encapsulate the notions of semantic security and the definition of pseudorandom generators by letting $P_{1,\lambda}, P_{2,\lambda}$ be defined appropriately in terms of the respective cryptographic tools (ciphers, PRGs). We show both as examples.

Example 2.16 (PRG Definition as Specific Instance of Computaitonal Indistinguishability). Let $\{G_\lambda\}_\lambda$ be a family of efficient, deterministic functions from $\{0,1\}^\lambda$ to $\{0,1\}^{\text{poly}(\lambda)}$. For every $\lambda \geq 1$, let

$$P_{1,\lambda} = \Pr_{k \leftarrow \{0,1\}^\lambda}[G_\lambda(k)] \qquad P_{2,\lambda} = \text{uniform}(\{0,1\}^{\text{poly}(\lambda)}) \ .$$

It follows that if $P_1 \approx_c P_2$ (i.e. are computationally indistinguishable), then the family of functions $\{G_\lambda\}_\lambda$ satisfies Eqn. 2.3 (i.e. satisfies the PRG game) and consequently satisfies the formal definition of a PRG (Definition 2.14).

Example 2.17 (Semantic Security as Specific Instance of Computational Indistinguishability). Let $\mathcal{E}$ be a cipher. Let $m_0, m_1 \in \mathcal{M}$. For every $\lambda \geq 1$, let

$$P_{1,\lambda}^{m_0} = \Pr_{k \leftarrow \mathcal{K}}[E(k, m_0)] \qquad P_{2,\lambda}^{m_1} = \Pr_{k \leftarrow \mathcal{K}}[E(k, m_1)] \ .$$

It follows that if $P_1^{m_0} \approx_c P_2^{m_1}$ for all choices of $m_0, m_1 \in \mathcal{M}$, then the cipher is semantically secure by Definition 2.11.

Remark 2.18. Note that we can see from the above example that semantic security has put no constraints on the distributions of the outputs of $E(k, m_0)$ and $E(k, m_1)$ beyond that they are computationally indistinguishable. For example, it could be the case that the supports of $E(k, m_0)$ and $E(k, m_1)$ might be disjoint, but as long as they remain computationally indistinguishable, semantic security has been achieved.

2.9 Semantic Security of PRG-based Stream Cipher

We have now shown how the notion of computational indistinguishability allows us to understand semantic security and the definition of PRGs in a compact and natural way. As a final example of the power of computational indistinguishability, we look at how it allows us to prove the semantic security of the PRG-based Stream Cipher. To do this, we first prove a lemma which shows that XORing the outcome of a PRG with a message is computationally indistinguishable from XORing the message with a uniformly sampled key which in effect follows from the definition of PRG. Using this lemma we are then able to prove the semantic security of the stream cipher.

Lemma 2.19. *For sufficiently large λ, a message $m \in \{0,1\}^{poly(\lambda)}$ and PRG $G_\lambda : \{0,1\}^\lambda \to \{0,1\}^{poly(\lambda)}$ is such that*

$$m \oplus G_\lambda(k) \approx_c m \oplus \textit{uniform} \ .$$

Proof. We proceed by contradiction. Let it be the case that for all $\lambda \geq 1$,

$$m \oplus G_\lambda(k) \not\approx_c m \oplus \text{uniform} \ .$$

Then, for all $\lambda \geq 1$, there exists a PPT TM adversary $\mathcal{A}$ such that

$$\left| \Pr_{k \leftarrow \{0,1\}^\lambda}[\mathcal{A}(m \oplus G_\lambda(k)) = 1] - \Pr_{y \leftarrow \{0,1\}^{\text{poly}(\lambda)}}[\mathcal{A}(m \oplus y) = 1] \right| > \frac{1}{\text{poly}(\lambda)} \ .$$

Define the PPT TM $\mathcal{B}$ as the PPT TM adversary that first XORs its input with m and then does the same thing as $\mathcal{A}$. It follows that $\mathcal{B}$ is a PPT TM adversary such that

$$\left| \Pr_{k \leftarrow \{0,1\}^\lambda}[\mathcal{B}(G_\lambda(k)) = 1] - \Pr_{y \leftarrow \{0,1\}^{\text{poly}(\lambda)}}[\mathcal{B}(y) = 1] \right| > \frac{1}{\text{poly}(\lambda)} \ .$$

However, this means the PRG does not satisfy the definition of a PRG (Definition 2.14) which is a contradiction. □

THEOREM 2.20. *The PRG-Based Stream Cipher is semantically secure.*

Proof. Let $\mathcal{E}$ be the stream cipher (Definition 2.10). Denote its PRG as G (where we assume sufficiently large λ and cease denoting it). We therefore wish to prove $\forall m_0, m_1 \in \mathcal{M} = \{0,1\}^{\text{poly}(n)}$, $P_1^{m_0} \approx_c P_2^{m_1}$ where

$$P_1^{m_0} = \Pr_{k \leftarrow \mathcal{K}}[E(k, m_0)] \qquad P_2^{m_1} = \Pr_{k \leftarrow \mathcal{K}}[E(k, m_1)] \ .$$

Let $m_0, m_1 \in \mathcal{M}$. Then

$$\begin{aligned} Pr_{k \leftarrow \mathcal{K}}[E(k, m_0)] =& Pr_{k \leftarrow \mathcal{K}}[m_0 \oplus G(k)] \\ \approx_c& Pr_{k \leftarrow \mathcal{M}}[m_0 \oplus k] \\ =& Pr_{k \leftarrow \mathcal{M}}[m_1 \oplus k] \\ \approx_c& Pr_{k \leftarrow \mathcal{K}}[m_1 \oplus G(k)] \\ =& Pr_{k \leftarrow \mathcal{K}}[E(m_1, k)] \ , \end{aligned}$$

where the first and final equality are from the definition of the cipher stream, the computational indistinguishabilities follow from Lemma 2.19, and the middle equality is the property of bitwise addition with uniform randomness (see Lecture 1). As $m_0, m_1 \in \mathcal{M}$ were arbitrary, for all $m_0, m_1 \in \mathcal{M}$, $P_1^{m_0} \approx_{2c} P_2^{m_0}$. The factor of 2 in the subscript of the relation denotes needing $\approx_c$ twice in the chain of relations. We note this factor of 2 does not effect the Stream Cipher being computationally indistinguishable as the definition of computational indistinguishability is transitive by the triangle inequality and that the sum of negligible functions is also negligible. Thus, the PRG-based Stream Cipher is semantically secure. □

2.10 Conclusion

In this lecture we began with the one-time pad which promises perfect security. Unfortunately, we showed that perfect security always requires the key space to be at least as large as the message space which is not practical for large messages. Motivated by this we

introduce the psuedorandom generator which takes a perfect secret key to a larger pseudorandom key. We then introduced the Stream Cipher encryption scheme which makes use of the pseudorandom generator to encrypt a larger message. As the pseudorandom generator cannot guarantee computational security, we introduced the weaker notion of semantic security. In formalizing semantic security and the definition of a pseudorandom generator, we abstracted to the concept of computational indistinguishability which can then be used for definitions of cryptographic tools and different notions of security. Finally, we used computational indistinguishability to prove the semantic security of the PRG-based stream cipher, combining all of the ideas presented in this lecture.

Acknowledgement

These scribe notes were prepared by editing a light modification of the template designed by Alexander Sherstov.

References

[1] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC Press, 2007.

[2] D. Khan. *The Code Breakers.* Macmillan, first edition, 1967.

[3] E. Kushilevitz and N. Nisan. *Communication complexity.* Cambridge University Press, 2nd edition, 2006.