Notes for Lecture 1

1 Basic Cryptography Review

Basic Notation. Essentially every cryptosystem we will see in this course will depend on a security parameter, which we will denote λ. The idea is that increasing λ will provide better security (which we will formalize in a bit). For now, think of λ as the length of the key, though later on we will sometimes allow the key length to be something different than the security parameter.

For the most part, we will not care too much about the precise model of computation. For concreteness, you can take the model of computation to be Turing machines. For randomized/probabilistic algorithms, we will use Turing machines that have access to a random tape.

Cryptographic algorithms will almost always be required to be efficient. Our notion of efficiency will be polynomial time. We will sometimes restrict to deterministic polynomial-time algorithms, and otherwise allow probabilistic algorithms. We will use PPT as shorthand for probabilistic polynomial time.

We say that a function $\epsilon(\lambda)$ is negligible if it goes to zero faster than any polynomial. More precisely, for any constant c, there exists a constant λ_0 such that $\epsilon(\lambda) < \frac{1}{\lambda^c}$ for all $\lambda > \lambda_0$. We will use negligible functions for any quantity that we want to go to zero extremely fast.

Defining Encryption. A (symmetric key or secret key) encryption scheme consists of two algorithms $(\mathsf{Enc}, \mathsf{Dec})$. Enc is a PPT algorithm that takes as input a key and a plaintext, and outputs a ciphertext. Dec is deterministic polynomial time, takes as input a key and a ciphertext, and outputs a plaintext. For correctness, we require that when used with the same key, Dec inverts Enc. More precisely, for all messages m,

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m, k \xleftarrow{\$} \{0,1\}^\lambda] = 1$$

Here, the probability is taken over a random k, and any random coins chosen by Enc. Since the probability is 1, this means that for any key and any coins, Dec will always correctly decrypt a plaintext. It is also possible to consider schemes where the probability 1 is replaced with $1 - \epsilon(\lambda)$ for a negligible ϵ.

For security, we want a definition that captures the following:

- Security holds for arbitrary messaages. We want this so that security holds for English (or any other) language, for numerical data, or for any other use case. As an extreme example, maybe the encryption scheme will only be used to encrypt two messages, "`ATTACK AT DAWN`" and "`ATTACK AT DUSK`". We also want security to hold even in settings where the adversary may have some influence over the messages that are sent. For example, an adversary may attack a particular location, and then wait for the adversary to send a message containing the location's name asking for help. To be most conservative, we will therefore give the adversary complete control over messages that are sent.
- We want to allow multiple messages to be sent, even the same message sent multiple times. Note that we want to hide whether the same message was sent again. For example, if Alice sends a ciphertext c to Bob, and then both attack the next morning, the adversary may very well guess afterward that c encrypted encrypts "`ATTACK AT DAWN`." Imagine a few days later Alice wants to send the same message "`ATTACK AT DAWN`". If the adversary can figure out that the same message was sent (for example, if the encryption scheme always maps "`ATTACK AT DAWN`" to c), then the adversary can now guess that an attack will occur the following dawn, and prepare accordingly.
- The adversary may only care about a single bit of information about the plaintext, or even some arbitrary function of the plaintext. We want to design a scheme that works, no matter what piece of information the adversary is interested in. For example, in the "`ATTACK AT DAWN`" vs "`ATTACK AT DUSK`" setting, it is sufficient for the adversary to learn, say, the last character of the plaintext.

We therefore define security as follows. Let A be an adversary. Let $\mathsf{IND\text{-}CPA\text{-}EXP}_b(A, \lambda)$ be the following experiment on A, parameterized by a bit b:

1. A interacts with a challenger, denoted Ch.
2. At first, Ch chooses a random key $k \xleftarrow{\$} \{0,1\}^\lambda$
3. Next, A sends the challenger two messages m_0, m_1. Ch selects and encrypts m_b: $c \leftarrow \mathsf{Enc}(k, m_b)$. Then Ch sends c back to A.
4. A can repeat step 3 as many times as it wishes. We will charge A one unit of time for every time it repeats step 3.
5. Finally, A outputs a guess b' for b. b' is the output of $\mathsf{IND\text{-}CPA\text{-}EXP}_b(A, \lambda)$

Here, IND refers to indistinguishability, meaning that the adversary is trying to distinguish between two experiments, $b = 0$ and $b = 1$. CPA stands for "chosen plaintext attack". This refers to the fact that the adversary is able to choose the plaintexts that get encrypted.

Definition 1 *An encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *is* IND-CPA secure *(in words,* indistinguishable under a chosen plaintext attack*) if, for all PPT adversaries A, there exists a negligible function ϵ such that*

$$| \Pr[1 \leftarrow \mathsf{IND\text{-}CPA\text{-}EXP}_0(A, \lambda)] - \Pr[1 \leftarrow \mathsf{IND\text{-}CPA\text{-}EXP}_1(A, \lambda)] | < \epsilon(\lambda)$$

We will often simply call such a scheme "CPA secure". Intuitively, this definition means that any guess b' the adversary makes is more or less independent of the actual bit b, since the probabilities for any guess under the two experiments are extremely close.

Question: What happens if Enc is a deterministic scheme? Can it possibly be CPA secure?

2 PRFs

A PRF is a keyed function that looks like a random function if you never get to see the key. That is PRF is a deterministic polynomial time computable function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$, with the following security property.

Let A be an adversary. Let $\mathsf{PRF\text{-}EXP}_b(A, \lambda)$ be the following experiment on A, parameterized by a bit b:

1. A interacts with a challenger, denoted Ch.
2. At first, if $b = 0$, Ch chooses a random key $k \xleftarrow{\$} \{0,1\}^\lambda$. If $b = 1$, Ch initializes an empty list L.
3. Next, A sends the challenger an input $x \in \{0,1\}^{n(\lambda)}$. Ch responds as follows
   - If $b = 0$, Ch responds with $y \leftarrow \mathsf{PRF}(k, x)$.
   - If $b = 1$, Ch looks for a pair (x, y) in L. If it finds an (x, y), it responds with y. Otherwise, it generates a random y and adds the pair (x, y) to L. Then it responds with y.
4. A can repeat step 3 as many times as it wishes. We will charge A one unit of time for every time it repeats step 3.
5. Finally, A outputs a guess b' for b. b' is the output of $\mathsf{PRF\text{-}EXP}_b(A, \lambda)$

Notice that in the $b = 1$ case, Ch is effectively providing A with a truly random function O where all outputs are chosen independently and uniformly at random. In the $b = 0$ case, Ch is providing A with the PRF on a random key k. A's goal is to distinguish the two cases.

Definition 2 *An PRF* PRF secure *if, for all PPT adversaries A, there exists a negligible function ϵ such that*

$$| \Pr[1 \leftarrow \mathsf{PRF\text{-}EXP}_0(A, \lambda)] - \Pr[1 \leftarrow \mathsf{PRF\text{-}EXP}_1(A, \lambda)] | < \epsilon(\lambda)$$

3 CPA-secure secret key encryption from PRFs

Let PRF be a pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$.

Our scheme will be the following:

- $\mathsf{Enc}(k, m)$ for $k \in \{0,1\}^\lambda$ and $m \in \{0,1\}^{m(\lambda)}$ does the following. First it chooses a random $r \in \{0,1\}^\lambda$, and computes $c \leftarrow \mathsf{PRF}(k, r) \oplus m$. It outputs (r, c)
- $\mathsf{Dec}(k, (r, c))$ computes $m \leftarrow \mathsf{PRF}(k, r) \oplus c$

Correctness is straightforward, since Dec computes $\mathsf{PRF}(k, r) \oplus c = \mathsf{PRF}(k, r) \oplus (\mathsf{PRF}(k, r) \oplus m) = m$.

Theorem 3 *If* PRF *is a secure PRF and* $2^{n(\lambda)}$ *is super polynomial, then* $(\mathsf{Enc}, \mathsf{Dec})$ *is CPA secure.*

Proofs in cryptography are usually proofs by contradiction: we assume an adversary violating the security of one primitive (in our case, an encryption scheme), and derive from it an adversary for some starting primitive (in our case, a PRF).

We will also introduce one of the standard proof techniques in cryptography, a hybrid argument. Assume toward contradition that there is an adversary A and a non-negligible function ϵ such that

$$| \Pr[1 \leftarrow \mathsf{IND\text{-}CPA\text{-}EXP}_0(A, \lambda)] - \Pr[1 \leftarrow \mathsf{IND\text{-}CPA\text{-}EXP}_1(A, \lambda)] | \geq \epsilon(\lambda)$$

We will define several "hybrid" games, where the first is $\mathsf{IND\text{-}CPA\text{-}EXP}_0(A, \lambda)$, and the last is $\mathsf{IND\text{-}CPA\text{-}EXP}_0(A, \lambda)$. By our assumption, we know that A distinguishes the first and last hybrid. Therefore, it must also distinguish some pair of adjacent intermediate hybrids. We will use such a distinguishing advantage to break the security of the PRF.

- **Hybrid 0** is identical to $\mathsf{IND\text{-}CPA\text{-}EXP}_0(A, \lambda)$. Substituting in to the experiment our construction, the experiment works as follows:
  1. At first, Ch chooses a random key $k \xleftarrow{\$} \{0,1\}^\lambda$

2. Next, A sends the challenger two messages m_0, m_1. Ch chooses a random r in $\{0,1\}^{n(\lambda)}$, and computes $y \leftarrow \mathsf{PRF}(k, r)$. Then Ch sends $(r, y \oplus m_0)$ back to A. A can repeat this step as many times as it wishes. s

- **Hybrid 1** is the following modifications to the $\mathsf{IND\text{-}CPA\text{-}EXP}_0(A, \lambda)$ experiment.
  - Instead of generating a random key k, Ch initializes an empty list L.
  - Instead of computing $y \leftarrow \mathsf{PRF}(k, r)$, Ch looks for (r, y) in L, using y if found. Otherwise, it generates a fresh random y, and adds (r, y) to L.
- **Hybrid 2** is the same as **Hybrid 1**, except that Ch sends $(r, y \oplus m_1)$ back to A.
- **Hybrid 3** is the same as **Hybrid 2**, except that Ch goes back to choosing a random key k and setting $y \leftarrow \mathsf{PRF}(k, r)$. Notice that **Hybrid 3** is identical to $\mathsf{IND\text{-}CPA\text{-}EXP}_1(A, \lambda)$

Now, by our assumption that $(\mathsf{Enc}, \mathsf{Dec})$ is insecure and the triangle inequality, we have that there must exist an $i \in \{0, 1, 2\}$ such that

$$\mid \Pr[1 \leftarrow \text{Hybrid}_i(A, \lambda)] - \Pr[1 \leftarrow \text{Hybrid}_{i+1}(A, \lambda)] \mid \geq \frac{1}{3}\epsilon(\lambda)$$

We now consider the three cases:

- $i = 0$. Notice that the only difference between **Hybrid0** and **Hybrid1** is that in **Hybrid 0**, y is set to $\mathsf{PRF}(k, r)$, whereas in **Hybrid 1**, y is set to random. It is straightforward to construct a PRF adversary B which distinguishes PRF from random with advantage $\epsilon(\lambda)/3$. B works as follows: it simulates A, playing the role of CPA-security challenger to A. Whenever A makes a query (m_0, m_1), B chooses a random r, and queries its own PRF challenger on r, obtaining y. Then it responds to A with $(r, y \oplus m_0)$. Finally, when A outputs a bit b', B outputs b'.

  In $\mathsf{PRF\text{-}EXP}_0(A, \lambda)]$, B successfully simulates the view of A in **Hybrid 0**. Similarly, in $\mathsf{PRF\text{-}EXP}_1(A, \lambda)]$, B successfully simulates the view of A in **Hybrid 1**. Therefore, B's advantage is the same as A's in distinguishing these two hybrids, namely $\epsilon(\lambda)/3$. This is non-negligible, a contradiction to the assumed security of PRF.

- $i = 1$. In **Hybrid 1** and **Hybrid 2**, y is set to random; the only difference is that it is XORed with m_0 or m_1 before responding to A. However, a random string XORed with anything is still random, so A essentially receives random strings in both hybrids.

The only potential problem if the same r is used to encrypt in two different queries. In this case, the response to each query is random, but the two responses are correlated.

Such collisions in r are, however, not a common occurrence: the probability any two queries collide is $2^{-}n(\lambda)$. The probability that some pair of queries collide is therefore at most $q^2 \times 2^{-n(\lambda)}$ where q is the number of queries. Recall that q is a polynomial, and that $2^{-n(\lambda)}$ is negligible. Since a negligible function times a polynomial is still negligible, we have that the probability of a collision is negligible. Therefore, A's distinguishing advantage $\epsilon(\lambda)/3$ must be negligible, a contradiction.

- $i = 2$. This is handled identically to $i = 0$, except that B encrypts m_1 instead of m_0.

Therefore, in any of the three cases, we reach a contradiction. Therefore, our assumed adversary A could not possibly exist. This completes the proof of security.