Symmetric Encryption

Encryption Scheme

A symmetric encryption scheme is associated with a key space $\{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, a message space $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, and a ciphertext space $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ with two algorithms Enc and Dec with:

$$\text{Enc}_\lambda : \mathcal{K}_\lambda \times \mathcal{M}_\lambda \to \mathcal{C}_\lambda$$

$$\text{Dec}_\lambda : \mathcal{K}_\lambda \times \mathcal{C}_\lambda \to \mathcal{M}_\lambda$$

We define the correctness of the encryption scheme if:

$$\text{Dec}_\lambda(k, \text{Enc}_\lambda(k, m)) = m$$

If our encoders and decoders are probabilistic, then this becomes:

$$\Pr\left[\text{Dec}_\lambda(k, \text{Enc}_\lambda(k, m)) = m\right] = 1$$

Defining Security

Take 1

An encryption scheme is secure if for every λ and pair of messages m_0, m_1, we have:

$$\text{Enc}(k, m_0) \equiv \text{Enc}(k, m_1)$$

for a random k

- This notation means that the probability distributions, random across the different choices of k, are equal for the two messages

A one-time pad, one that just takes $\text{Enc}(k, m) = k \backslash \text{xor} m$, satisfies this

- However, if the adversary sees more than one message, they can just xor them and get what the xor of the two messages is
  - If they see any plaintext, they can get the key directly out!
- Here, ciphertexts are functions of the secrete key and can leak information about it
- We want to be secure even if the adversary sees many ciphertexts

Take 2

An encryption scheme is secure if for every $\lambda, \ell \in \mathbb{N}$ and every message $m_1, \ldots, m_\ell, m'_1, \ldots, m_\ell$ we have:

$$(\text{Enc}(k, m_1), \ldots, \text{Enc}(k, m_\ell)) \equiv (\text{Enc}(k, m'_1), \ldots, \text{Enc}(k, m'_\ell))$$

for a random k

This is sufficiently strong, but is impossible to achieve!

- This cannot be possible if the encryption algorithm is deterministic, since otherwise the adversary can tell if the same message was encrypted twice
- This is still impossible because the ciphertext contains some information about the secret key k, and eventually with enough ciphertexts, all of the information about k will be given away

We relax the security requirement, and instead of requiring the distributions to be the same, we require that they only look the same to polynomial time adversaries

- We refer to this as **computationally indistinguishable** and use $\approx$ to refer to this

Negligible

A function $\mu : \mathbb{N} \to \mathbb{N}$ is neglgible if for every $c \in \mathbb{N}$, there exists $n_c \in \mathbb{N}$ such that for every $n > n_c$, we have $\mu(n) < n^{-c}$

Computationalyl Indistinguishable

Distributions $\mathcal{A}$ and $\mathcal{B}$ are computationally indistinguishable if for every probabilistic polynomial time distinguisher $\mathcal{D}$ there exists a negliglble function μ such that:

$$|\Pr[\mathcal{D}(\alpha) = 1] - \Pr[\mathcal{D}(\beta) = 1]| \leq \mu(\lambda)$$

for random α, β drawn from $\mathcal{A}, \mathcal{B}$

The actual definition is more complicated and even allows the adversary to choose messages adaptively based on previous ciphertexts

CPA Secure

An encryption scheme is secure against adaptively chosen plaintext attacks (CPA secure) if for every PPT adversary $\mathcal{A}$ there exists a negligible function μ such that for every λ, $\mathcal{A}$ wins the following game with probability at most $1/2 + \mu(\lambda)$:

- The challenger chooses a key $k \leftarrow \mathcal{K}_\lambda$
- The adversary, given the length λ, chooses a message m_i and receives c_i
  - This can be repeated polynomially many times
- The adversary $\mathcal{A}$ chooses m_0, m_1
- The challenger chooses a random message out of the two and sends the corresponding ciphertext back
- The adversary attempts to guess which one of its messages was encrypted

- In the end, we say this is secure if the adversary cannot meaningful gain an advantage in discerning which message was encrypted or not

Constructing a CPA-Secure Scheme

If the secret key was infinitely long, we could encrypt all of our messages using a fresh part of the pad, and just tell the user which part of the pad we used

- As long as we do not reuse the same part of the pad, we will have the same security as the one-time pad
- To approximate this infinitely long pad, we could use a perfectly random function F and pass in a random input to it each time
  - $F : \{0,1\}^\lambda \to \{0,1\}$
  - We could just generate a random number to pass into F every time we want to encrypt a message, and as long as we encrypt significantly less than $2^{\lambda/2}$ messages, we do not expect a collision
  - To this end, we use pseudorandomness

Pseudorandom Functions

A pseudorandom function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is a function such that for every PPT algorithm $\mathcal{A}$, there exists a negligible function μ such that:

$$\left|\Pr\left[\mathcal{A}^{F(k,\cdot)} = 1\right] - \Pr\left[\mathcal{A}^{R} = 1\right] \leq \mu\right|$$

where k is chosen randomly and R is a truly random function. The adversary can access F and R a polynomial number of times and attempt to predict whether or not it is interacting with the pseudorandom function

- In other words, the adversary cannot in reasonable time distinguish this function from a truly random one

By just using a pseudorandom function in place of the truly random function we get a CPA secure scheme