Use the PRF in some secure mode of operation to obtain a CPA-secure encryption scheme

CPA –Secure encryption scheme:

Here we assume Adversary (A) can access encryption server, so Adversary can obtain the ciphertexts of any message of his/her choice and to achieve secure encryption in A should not be able to distinguish the encryption of two arbitrary messages, even when A is given access to an encryption server.

Let us define an experiment for any private key encryption scheme $\Pi = (Gen, Enc, Dec)$ and Adversary A, any value n for the security parament

The CPA Indistinguishability Experiment

- Generate a random key k by running $Gen(1^n)$
- The adversary is given input 1^n and oracle access to $Enc_k(.)$, and outputs a pair of messages m_0, m_1 of the same length
- A random bit $b \leftarrow \{0,1\}$ is chosen, and then a cipher text $c \leftarrow Enc_k(m_b)$ is computed and given to A. Lets call this as challenge ciphertext
- The Adversary outputs a bit b'.
- The output of the experiment is defined to be 1 if $b' = b$ and 0 if not. If it is 1 then we say Adversary A succeeded

Defining Computational Security Against CPA:

A private-key encryption scheme $\Pi = (Gen, Enc, Dec)$ has indistinguishable encryptions under a chosen-plaintext attack (or is CPA-secure) if for all probabilistic polynomial-time adversaries A there exists a negligible function negl such that

$$P_r\left[\mathrm{Pr}\,i\,vK_{A,\Pi}^{cpa}(n) = 1\right] \leq \frac{1}{2} + negl(n)$$

Consider this scenario, A outputs (m_0, m_1) and receives $c \leftarrow Enc_k(m_b)$.

An adversary has access to the Encryption server, so he gets ciphertexts of m_0 and m_1 and he will compare with challenge ciphertext, and he will succeed here. Our encryption scheme fails here, it means we need different ciphertext for every time we encrypt it, which means the encryption scheme should be probabilistic, that is, it must use some randomness as part of the encryption process to ensure that two encryptions of the same message are different.

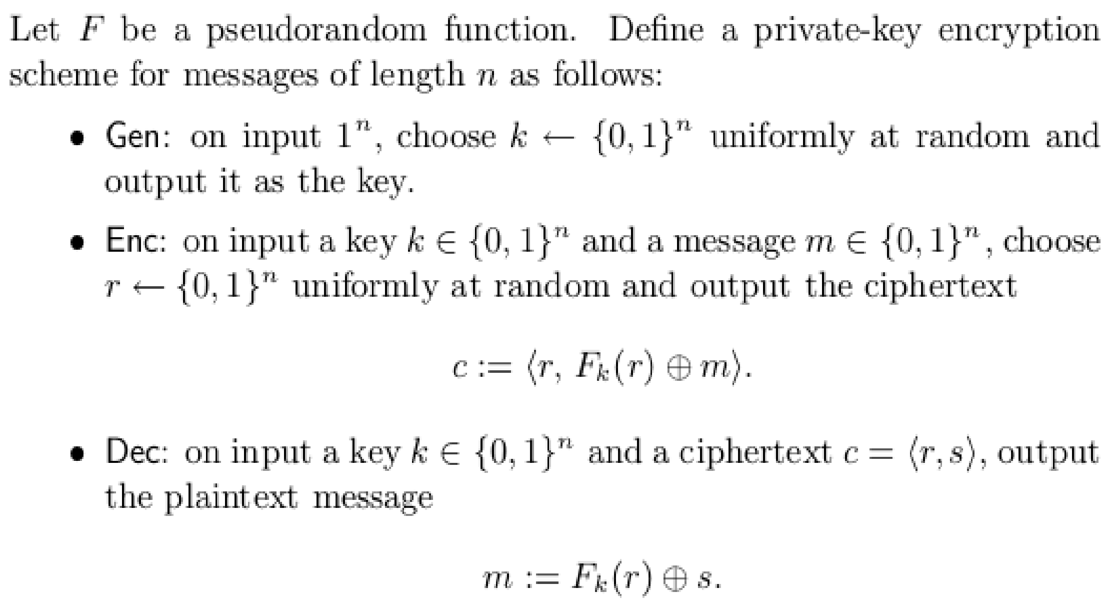
No deterministic Encryption algorithm can be CPA-secure.

Probabilistic Encryption:

The same key, same message should give different ciphertexts when they encrypted once again, so we can avoid CPA attacks, but we should get deterministic encryption. We should get the same message when we decrypt it.

We encrypt the message by applying a pseudorandom function to a random value r and XORing the results with the plaintext.

CONSTRUCTION:



Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- Gen: on input 1^n, choose $k \leftarrow \{0,1\}^n$ uniformly at random and output it as the key.
- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ uniformly at random and output the ciphertext

$$c := \langle r, F_k(r) \oplus m \rangle.$$

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message

$$m := F_k(r) \oplus s.$$

If F is a pseudorandom function, then above Construction is a fixed-length private-key encryption scheme with length parameter $l(n) = n$ that has indistinguishable encryptions under a chosen-plaintext attack.

Here we are adding one term in encryption that is F which is a pseudorandom function and guessing its output for polynomial-time Adversary is hard. So, it is secure.

Modes of operation:

The way of encrypting arbitrary-length messages using a block cipher (means Pseudorandom function). Here we divide message into fixed length blocks of size n and if the last block is partial then we pad with zeros.

1) Cipher Block Chaining (CBC) mode:

In this mode, a random initial vector (IV) of length n is chosen. Then first block of ciphertext is generated using applying F to (IV XOR m_1)

Next ciphertext block is computed by applying F to (c1 XOR m_2)

For $i > 0$, $c_i = F_k(c_{i-1}\ XOR\ m_i)$ and final ciphertext is $< IV, c_1, c_2, .. c_b >$ where b is total number of blocks.

In this mode, the encryption should be processed sequentially to get one cipher block we need previous cipher block.

2) Output Feedback mode (OFB):

In this mode, a random initial vector (IV) of length n is chosen. Then we apply F on that and will be generated r_1 , to get the first cipher block, apply XOR between r_1 and m_1, to get the next cipher block we apply F on r_1 (No need of message block) which gives r_2 and will apply XOR between r_2 and m_2

Generally, $r_i = F_k(r_{i-1})$and $c_i = m_i \, XOR \, r_i$

Here cipher blocks are independent of other cipher blocks, so it may be possible to prepare a stream ahead of time using pre-processing, after which point the encryption of the plaintext (once it is known) is incredibly fast.

This mode is a nice way to convert block ciphers to stream ciphers (nothing but PRFs to PRGs)

Here encryption and decryption should be done in a sequential process.

3) Counter (CTR) mode:

In this mode, a random initial vector (IV) of length n is chosen and denoted as ctr . Then, a stream is generated by computing

$$r_i = F_k(ctr + i)$$

And to get i^{th} cipher block is computer by $c_i = m_i \, XOR \, r_i$

Here encryption and decryption can be done in parallel mode, means to decrypt i^{th} block we do not need to decrypt other blocks.