CPA Documentation

__init__(self, t):

Input variables: t

- t decides which secure mode the cpa is to be used in, by initializing self.mode

binstring(self, g, p):

Input variables: x

- Takes a number x and outputs it's corresponding binary representation in the form of a
- string
- It's a helper function

genKey(self,x):

Input variables: n

- Generates a random binary string of length n

getint(self,s):

Input variables: s

- Returns the integer value of a string containing the binary representation

setStrLen(self, s, n):

Input variables: s, n

- Sets a binary string to given length n and returns it

getxor(self, s1, s2):

Input variables: s1, s2

- Performs xor between to binary strings and returns the outcome in binary string format

rcm(self, l, m, k=None):

Input variables: prg, l, m, k=None

- L is the number of blocks of length n in the message m
- Here input length is expected to be a multiple of the key length
- If key is not given, then a random key is generated using length n = len(m)/l ang
- Given a key, message and l the rcm is used to make a variable-length prf

ofb(self, l, m, k=None):

Input variables: prg, l, m, k=None

- L is the number of blocks of length n in the message m
- Here input length is expected to be a multiple of the key length
- If key is not given, then a random key is generated using length n = len(m)/l ang
- Given a key, message and l the ofb is used to make a variable-length prf

rcm_dec(self, prg, iv_init, k, c):

Input variables: prg, iv_init, k, c

- Implements decryption algorithm for rcm encoded binary string c

ofb_dec(self, prg, iv_init, k, c):

Input variables: prg, iv_init, k, c

- Implements decryption algorithm for ofb encoded binary string c