Soumodipta Bose| 2021201086

Build a Provably Secure PRF(Code)

Pseudorandom Function:

F(k, x):Pseudo random function

Args:

k (string): seed (n bits) binary string

x (string): input string (binary)

Returns: n bit truly random binary string

Usage:

1. Create a binary string for key ex. key='10001001' as key
2. Take another binary string as input string called x which can be of any length. This decides the number of iterations in the pseudo random function.
3. The demo code in **start()** allows you to choose a key and message x in binary and returns the random bits of same size as key as output.

Crypto library:

- **gen(x, p=doubler())**:Pseudo Random number generator

Args:

x (binary string): Initial Seed

p (function, optional): A polynomial input can be given. Defaults to doubler. The function can be anonymous function as well as long as it returns an integer and takes length of initial key as input

Returns: (binary string): Pseudo random bits

- **PRG_single(x)** and **PRG_double(x)** are wrappers for gen(x,p) where former retains same length and latter doubles the length

Utility functions:

- **split_string(x)**: splits string into two equal parts
- **dec_to_bin_wo_pad(x)**: Converts decimal to binary without padding
- **dec_to_bin(x, size)**:Converts decimal to binary with padding
- **bin_to_dec(x)**: converts binary to decimal
- **discrete_log(x)**:

DLP One way function GENERATOR = 8173 MOD = 65521

Performs (GENERATOR^x) % (MOD)

Args: x (int): seed value

Returns: one way function value

- **get_hardcore_bit(x):**

Extracts Hardcore bit using Blum Micali Hardcore bit

- **g(x):**Takes input binary string x(L bits) and return hardcore bit and new seed of L+1 bits. Calls discrete_log(x) and get_hardcore_bit(x)