Soumodipta Bose| 2021201086

Build a Provably Secure PRG(Code)

Polynomials:

These are the default polynomials which can be plugged into the generator function:

- doubler(n): Polynomial to double the length
- singler(n): Polynomial to keep length same

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:

If x < prime/2 - 0

x >= prime/2 - 1

- 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)

Pseudo Random Generator:

- 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):

PRG of same bit size

- PRG_double(x):

PRG of double bit size

Usage:

1. Create a binary string ex. X='10001001' as seed
2. Now `new_seed=gen(X,p=lambda(x):x*x+1)` will create a new_seed with length x^2+1. Similary we can pass any polynomial as input.
3. PRG_single(x) and PRG_double(x) are wrappers that returns random bits of length n and 2n if n is number of bits in x.
4. The demo code in **start.py** allows you to choose a seed and returns the bits of 2n size.