Soumodipta Bose| 2021201086

Use the PRF to build a secure MAC(Code)

Message Authentication Code:

- **cbc_mac(k, message):**Calculates MAC tag using Cipher Block Chaining

  Args:

    k (binary string - n bits): input key for PRF in cbc MAC

    message (binary string): string for which tag needs to be calculated

  Returns:

    binary string - n bits: Tag of n bits

- **cbc_mac_verify(k, message, tag):**

Verifies the given tag with the original message

  Args:

    k (binary string - n bits): input key for PRF in cbc MAC

    message (binary string): string for which tag needs to be calculated

    tag (binary string - n bits): tag of n bits generated at sender

  Returns:

    boolean: True if tag matches the message else False

Usage:

1. Create a binary string for key ex. key='10001001' as key
2. Take another binary string as input string called message which can be of any length. This is the message/text whose tag needs to be calculated.
3. Now use **cbc_mac(key, message)** to find the tag at the sender side.
4. The receiver side should receive this key and message and call **cbc_mac_verify(key, message, tag)** to recalculate tag and verify.
5. The demo code in **start()** allows you to choose a key and message string in binary and returns the generated tag in binary, the tag generated is of n bits same as the key.

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

- **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

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)
- **xor(bin_x, bin_y)**: returns xor of the two binary strings
- **str_to_bin(s)**:Converts string into binary string
- **bin_to_str(n)**:Converts binary string into normal string