CUSTOM SECURITY LIBRARY CODE DOCUMENTATION

TABLE OF CONTENTS:

VARIABLES AND CONSTANTS:

Variable Name	Variable Data Type	Variable Utility
p	int	Prime number chosen for the group Z_p^*
g	int	A primitive root / generator of the group Z_p^*
h	int	Another primitive root / generator of the group Z_p^*
ipad	str	Input pad in binary form, used for H-MAC
opad	str	Output pad in binary form, used for H-MAC
iv	str	Initial vector, randomly chosen noise in binary form, introduces probabilistic encryption
seed	str	Seed chosen for PRG, in binary form
BLK_SIZE	int	Size of blocks into which message will be divided into

FUNCTIONS:

UTILITY FUNCTIONS :

1. **decimalToBinary-**

   `This function is used to convert a decimal number to its binary form`

Parameters	Datatype	Description
n	int	Decimal number

Return:
`(str) : binary representation of n`

Called by:

- PRG
- GEN
- ENC
- DEC
- CBC_MAC
- fixedLengthHash
- variableLengthHash
- HMAC

Caller of:
None

2. binaryToDecimal-

This function is used to convert a binary string to its decimal form

Parameters	Datatype	Description
b	str	binary representation of n

Return:
(int): Decimal of b

Called by:

- PRG
- GEN
- ENC
- DEC
- CBC_MAC
- fixedLengthHash
- variableLengthHash
- HMAC

Caller of:
None

3. lengthExpand-

This function is used to return desired polynomial expansion of seed length (say n) of PRG input

Parameters	Datatype	Description
n	int	Bit size of seed

Return:
`(int): expanded poly(n) length that pseudorandom output will be of`

Called by:
`Code block to test PRG`

Caller of:
`None`

4. msgToBinary-

`This function is used to convert a plaintext into its binary form`

Parameters	Datatype	Description
msg	str	String consisting of plaintext

Return:
`(str): Conversion of plaintext to binary representation`

Called by:
- `ENC`
- `HMAC`

Caller of:
`None`

5. binaryToMsg-

`This function converts a binary string to message string`

Parameters	Datatype	Description
binary	str	plaintext

Return:
`(str): String representation of plaintext`

Called by:
- `DEC`

Caller of:
`None`

PRG FUNCTIONS:

1. **hc-**

   `Function to get hardcore predicate bit using Goldreich-Levin Theorem`

Parameters	Datatype	Description
x	str	Input of one-way function

Return:
`(str): 1 bit hardcore predicate of x`

Called by:
- `PRG`

Caller of:
`None`

2. **owf-**

   `One-way function with candidate as Discrete Log Problem`

Parameters	Datatype	Description
x	int	Input of one-way function
g	int	Generator
p	int	Prime

Return:
`(int): Dlp function output`

Called by:
- `PRG`
- `fixedLengthHash`

Caller of:
`None`

3. **PRG-**

   `Function to generate pseudo-random output from a seed`

Parameters	Datatype	Description
seed	str	Binary input seed
l	int	Length to which to expand
g	int	Generator
p	int	Prime

Return:
`(str): pseudo-randomly generated number of l length`
Called by:

- `PRF`

Caller of:

- `hc`

PRF FUNCTIONS:

1. **PRF-**

   `Function to create PRF from PRG`

Parameters	Datatype	Description
input	str	Binary input to PRF function
key	str	randomly chosen key in binary representation (used as initial seed for PRG)
g	int	Generator
p	int	Prime

Return:
`(str): output of prf in binary with same as no. of bits as in input`

Called by:

- `ENC`
- `DEC`
- `CBC_MAC`

Caller of:

- `PRG`

CPA-ENCRYPTION FUNCTIONS:

1. GEN-

`Function to generate key`

Parameters	Datatype	Description
p	int	prime of group Z_p^*. Our valid set of keys will lie between (0, p-2) as the generator can have powers between this range

Return:
`binary format of chosen key`

Called by:
- `ENC`

Caller of:
`None`

2. ENC-

`Function to encrypt plaintext`

Parameters	Datatype	Description
msg	str	Plaintext
g	int	Generator
p	int	Prime
key	str	Randomly chosen key
BLK_SIZE	int	Size of each msg block

Return:
`(str): random noise used for probabilistic encryption`
`(str): cipher text generated`

Called by:
- `CCA_ENC`

Caller of:

- `msgToBinary`
- `GEN`
- `decimalToBinary`
- `binaryToDecimal`
- `PRF`

3. DEC-

`Function to decrypt ciphertext`

Parameters	Datatype	Description
counter	str	Random noise
cipher	str	Cipher text
key	str	Randomly chosen key
g	int	Generator
p	int	Prime
BLK_SIZE	int	Size of each msg block

Return:
`(str): decoded plaintext`

Called by:

- `CCAAuthDec`

Caller of:

- `decimalToBinary`
- `binaryToDecimal`
- `PRF`
- `binaryToMsg`

CBC-MAC FUNCTIONS:

1. CBC_MAC-

`Function to create Message Authentication code of given message`

Parameters	Datatype	Description

msg	str	Plaintext
key	str	Randomly chosen key
g	int	Generator
p	int	Prime
BLK_SIZE	int	Size of each msg block

Return:
`(str) : MAC tag of plaintext`

Called by:
- `CCA_ENC`
- `verifyMAC`

Caller of:
- `decimalToBinary`
- `binaryToDecimal`
- `PRF`

2. verifyMAC-

`Function to authenticate MAC by creating new tag of msg and comparing with old tag received`

Parameters	Datatype	Description
msg	str	input whose tag is to be authenticated
key	str	Key used to create tag
tag	str	Tag against which new tag will be matched
g	int	Generator
p	int	Prime
BLK_SIZE	int	Size of each msg block

Return:
`(bool): True if tag matches, False if tag doesn't match`

Called by:
- `CCAAuthDec`

Caller of:

- `CBC_MAC`

CCA-SECURE ENCRYPTION FUNCTIONS:

1. CCA_ENC-

`Function to encrypt message then create tag of the cipher text`

Parameters	Datatype	Description
msg	str	Plaintext
key1	str	Key to use for encryption
key2	str	Key to use for tag creation
g	int	Generator
p	int	Prime
BLK_SIZE	int	Size of each msg block

Return:
`(str): counter to use for decryption`
`(str): encrypted cipher`
`(str): tag of cipher to be verified`

Called by:
`Main code`

Caller of:
- `ENC`
- `CBC_MAC`

2. CCAAuthDec-

`Function to authenticate and decrypt cipher text. If authentication fails, decryption is not done.`

Parameters	Datatype	Description
key1	str	Key to use for decryption
key2	str	Key to use for tag verification

cipher	str	Cipher to decrypt
tag	str	Tag against which new tag generated on cipher is to be matched
counter	str	Counter to use to encrypt cipher
g	int	Generator
p	int	Prime
BLK_SIZE	int	Size of each msg block

Return:
`(void)`

Called by:
`Main code`

Caller of:

- `verifyMAC`
- `DEC`

HASHING FUNCTION:

1. **fixedLengthHash-**

   `Function used to create fixed length hash of input, using DLP assuming it to be a hard problem`

Parameters	Datatype	Description
x1	str	binary representation of input1 lying between 0 to p-1
x2	str	binary representation of input2 lying between 0 to p-1
g	int	A primitive root of p
h	int	Another primitive root of p
p	int	Prime number p to use in DLP

length	int	Length of fixed hash output

Return:
`(str): Binary fixed length hash output`

Called by:

- `variableLengthHash`
- `HMAC`

Caller of:

- `binaryToDecimal`
- `decimalToBinary`
- `owf`

2. variableLengthHash-

`Function to generate variable length hash from fixed length hash using Merkle-Damgard Transform`

Parameters	Datatype	Description
msg	str	Input whose hash is to be generated
initial_vector	str	Random chosen noise
g	int	A primitive root of p
h	int	Another primitive root of p
p	int	Prime number p to use in DLP
fixed_length	int	Length of fixed hash output

Return:
`(str) : Binary hash of size half of that of input`

Called by:
`Main code`

Caller of:

- `fixedLengthHash`
- `decimalToBinary`

H-MAC FUNCTION:

1. HMAC-

`Function to create MAC from fixed length hash`

Parameters	Datatype	Description
msg	str	Plaintext
key	str	Random chosen key for authentication
iv	str	Random chosen noise
g	int	A primitive root of p
h	int	Another primitive root of p
p	int	Prime number p to use in DLP
fixed_length	int	Length of fixed hash output

Return:
`(str): h-mac code of given plaintext`

Called by:
`Main code`

Caller of:

- `fixedLengthHash`
- `decimalToBinary`
- `binaryToDecimal`