DES Algorithm

Generated by Doxygen 1.8.10

Sun Feb 7 2016 12:58:27

Contents

Chapter 1

Encryption using the DES algorithm

Author

Roman Solar

Version

7/2/2016

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 binary.h File Reference

Functions

- void divideBinary (bool binary[ ], int sizeOfBinary, bool LB[ ], bool RB[ ])
- void overwrite (bool what[ ], bool with[ ], int noOfIndexes)

3.1.1 Detailed Description

functions used to manipulate with binary data

3.1.2 Function Documentation

3.1.2.1 void divideBinary (bool *binary[ ],* int *sizeOfBinary,* bool *LB[ ],* bool *RB[ ]*)

divides a binary array into halves

Parameters

LB	first half
RB	second half

3.1.2.2 void overwrite (bool *what[ ],* bool *with[ ],* int *noOfIndexes*)

overwrites one array with another

Parameters

what	what you want to overwrite
with	what you want to overwrite it
noOfIndexes	how many indexes you want to overwrite

3.2 callFunction.h File Reference

```
#include <iostream>
```

Functions

- bool callFunction (std::string input)
  calls corresponding function according to the command line input
- void help ()
  prints the available commands

3.2.1 Detailed Description

functions used to care for the user commands

3.2.2 Function Documentation

3.2.2.1 bool callFunction (std::string *input*)

calls corresponding function according to the command line input

Parameters

input	the command line input

Returns

false if 'end' was typed in, true otherwise

3.2.2.2 void help ( )

prints the available commands

3.3 cin.h File Reference

```
#include <iostream>
```

Functions

- bool is_number (const std::string &s)
  checks if given string is a number
- void clearBuffer ()
  flushes the cin buffer
- std::string trim (const std::string &s)
  trims a given string (deletes spaces from the end and beginning)

3.3.1 Detailed Description

functions that treat the cin input

3.3.2 Function Documentation

3.3.2.1 void clearBuffer ( )

flushes the cin buffer

3.3.2.2 bool is_number (const std::string & *s*)

checks if given string is a number

Parameters

s	string to check

Returns

true if s is a number, false otherwise

3.3.2.3 std::string trim (const std::string & *s*)

trims a given string (deletes spaces from the end and beginning)

Parameters

string	s to trim

Returns

trimmed string

3.4 encodeMessage.h File Reference

Functions

- void encodeMessage (bool binaryMessage[64], bool subkeys[16][48], bool encrypted[64])
- bool XOR (bool a, bool b)
- void XOR (bool where[ ], bool arr1[ ], bool arr2[ ], int noOfNumbers)
- void permutation (int permutation[ ], bool toPermute[ ], bool outcome[ ], int sizeOfPermutation)
- int lookUpInSBox (int which, bool address[6], bool binaryOutcome[4])
- void f (bool R[32], bool K[48], bool outcome[32])

3.4.1 Detailed Description

functions that encode/decode a given input using subkeys

3.4.2 Function Documentation

3.4.2.1 void encodeMessage (bool *binaryMessage[64],* bool *subkeys[16][48],* bool *encrypted[64]*)

encodes the message

Parameters

binaryMessage	message of 64 bytes to be encoded
subkeys	used to encrypt
encrypted	output array to save the encrypted message

3.4.2.2 void f (bool *R[32],* bool *K[48],* bool *outcome[32]*)

used in the algorithm of encoding message

3.4.2.3 int lookUpInSBox (int *which*, bool *address[6]*, bool *binaryOutcome[4]*)

finds with address (array of 6 bits) in S-boxes its corresponding number (array of 4 bits)

Parameters

which	which S-box should we look at
address	address used to find the correct index
binaryOutcome	array to save the answer

Returns

the answer in decimal

3.4.2.4 void permutation (int *permutation[ ],* bool *toPermute[ ],* bool *outcome[ ],* int *sizeOfPermutation*)

permute an array according to an array of permutations

Parameters

toPermute	the array to be permuted
permutation	the table of permutation

3.4.2.5 bool XOR (bool *a,* bool *b*)

adds two bools together using the XOR addition

Returns

result of the addition

3.4.2.6 void XOR (bool *where[ ],* bool *arr1[ ],* bool *arr2[ ],* int *noOfNumbers*)

adds two bool arrays together using the XOR addition

Parameters

where	output array to save the result
noOfNumbers	how many indexes are in the arrays

3.5 encrypt.h File Reference

```
#include <iostream>
```

Functions

- void encryptFlags (std::string call, bool encrypt)

  separates and reads the flags from the encrypt/decrypt command, then calls `encrypt` *function and passes it the flags as arguments*
- void printDefaultSettings ()

  prints the default settings
- void changeRoot ()

  asks the user to change the root for input/output files
- void printRoot ()

  prints the root for input/output files

3.5.1 Detailed Description

functions used to care for the input/output and calling the encodeMessage function

3.5.2 Function Documentation

3.5.2.1 void changeRoot ( )

asks the user to change the root for input/output files

3.5.2.2 void encryptFlags (std::string *call,* bool *encrypt*)

separates and reads the flags from the encrypt/decrypt command, then calls `encrypt` function and passes it the flags as arguments

Parameters

call	the user command
encrypt	true if user called 'encrypt' or 'e', false if he called 'decrypt' or 'd'

3.5.2.3 void printDefaultSettings ( )

prints the default settings

3.5.2.4 void printRoot ( )

prints the root for input/output files

3.6 key.h File Reference

Functions

- void getKey (int key[8])
- void keyTo8Bytes (char keyWord[ ], int key[8])

3.6.1 Detailed Description

functions used to get and prepare the key

3.6.2 Function Documentation

3.6.2.1 void getKey (int *key[8]*)

gets string from the user and calls keyTo8Bytes to convert it into key

Parameters

key	array to save the key

3.6.2.2 void keyTo8Bytes (char *keyWord[ ],* int *key[8]*)

converts a string to 8-byte key

Parameters

keyWord	the string to be converted
key	array of 8 bytes to save the key

3.7 message.h File Reference

```
#include <iostream>
#include <vector>
```

Functions

- int padding (int elementsInMessage)
- void getMessage (std::vector< unsigned char > &message, int inputS)
- void messageExpand (std::vector< unsigned char > &message)
- void messageToBinary (bool binaryMessage[64], std::vector< unsigned char > &message, size_t fromPos)

  converts the message in form of unsigned chars into 64-bit binary message

3.7.1 Detailed Description

functions used to get and prepare the message

3.7.2 Function Documentation

3.7.2.1 void getMessage (std::vector< unsigned char > & *message,* int *inputS*)

get the message from the user, character after character <unsigned char>=""">

Parameters

message	an array to store the message
numbersIn↩ Message	the size of the message (in bytes)

3.7.2.2 void messageExpand (std::vector< unsigned char > & *message*)

expands the message in order to be a multiple of 8 ints (bytes) long
adds 0 in the empty indexes

3.7.2.3 void messageToBinary (bool *binaryMessage[64],* std::vector< unsigned char > & *message,* size_t *fromPos*)

converts the message in form of unsigned chars into 64-bit binary message

Parameters

binaryMessage	to store the output
&message	the message to be converted

fromPos	to specify from which element of message to start

3.7.2.4 int padding (int *elementsInMessage*)

Returns

how many characters we need to add to the message in order to be a multiple of 8 characters long (64 bits)

Parameters

elementsIn↩ Message	how many elements the message contains

3.8 subkeys.h File Reference

Functions

- void createSubkeys (bool binaryKey[8 ∗8], bool subkeys[16][48])
- void leftShift (bool toShift[28], int noOfShifts, bool destination[28])

3.8.1 Detailed Description

functions used to create subkeys from the key

3.8.2 Function Documentation

3.8.2.1 void createSubkeys (bool *binaryKey[8 ∗8],* bool *subkeys[16][48]*)

creates the subkeys from the key (in binary)

3.8.2.2 void leftShift (bool *toShift[28],* int *noOfShifts,* bool *destination[28]*)

shifts an array to the left

3.9 typeConverter.h File Reference

Functions

- int toDecimal (bool binary[ ], int noOfNumbers)
- void toBinary (int origin[ ], bool binary[ ], int noOfNumbers)

3.9.1 Detailed Description

functions used for converting between different data types

3.9.2 Function Documentation

3.9.2.1 void toBinary (int *origin[ ],* bool *binary[ ],* int *noOfNumbers*)

converts decimal to binary and stores it in a given array

3.9.2.2 int toDecimal (bool *binary[ ],* int *noOfNumbers*)

converts binary to decimal

Returns

converted decimal number

Index