NAME

qfDES - a library of DES utilities

SYNOPSIS

```
#include "qfDES.h"
gcc ... -lqfDES

int qfDES(key, data, size, what, mode, initVec)
char *key;
char *data;
unsigned int size;
const QFDES_what what;
const QFDES_mode mode;
char *initVec;

int qfDES_ECB_e(key, data, size), qfDES_ECB_d(key, data, size)

int qfDES_CBC_e(key, data, size, initVec), qfDES_CBC_d(key, data, size,
initVec)

int qfDES_CFB_e(key, data, size, initVec), qfDES_CFB_d(key, data, size,
initVec)

int qfDES_OFB_e(key, data, size, initVec), qfDES_OFB_d(key, data, size,
initVec)

char qfDES_setPad(pad), qfDES_padSpace(), qfDES_padZero()
char pad;

unsigned int qfDES_insertPadding(ptr, size)
char *ptr;
unsigned int size;

unsigned int qfDES_malloc(ptr, size), qfDES_realloc(ptr, size)
char **ptr;
unsigned int *size;

unsigned int qfDES_plainTextSize(ptr, size)
char *ptr;
unsigned int size;

void qfDES_free(ptr)
char *ptr;

char *qfDES_copy(ptr, size)
char *ptr;
unsigned int size;

unsigned int qfDES_bin2hex(binText, hexText, size)
char *binText;
char *hexText;
unsigned int size;

unsigned int qfDES_hex2bin(hexText, binText, size)
char *binText;
char *hexText;
unsigned int size;

void qfDES_setParity(ptr, size, parity)
char *ptr;
unsigned int size;
QFDES_parity parity;
```

```
unsigned int qfDES_checkParity(ptr, size, parity)
char *ptr;
unsigned int size;
QFDES_parity parity;

char *qfDES_str2key(str)
char *str;

char *qfDES_generate(what), qfDES_generateKey(), qfDES_generateIV()
QFDES_generate what;

int qfDES_checkWeakKeys(key)
char *key;
```

DESCRIPTION

PLEASE NOTE: This manual page is an incomplete first draft!

The qfDES library contains functions for performing DES encoding and decoding and also some support functions for memory management. It is intended for use by us poor souls outside of the US who do not have DES hardware or software shipped with our machines.

qfDES() will perform DES encryption or decryption in any of the 4 DES modes: ECB (Electronic Code Book), CBC (Cipher Block Chaining), Cipher Feedback(CFB) or Ouput Feedback (OFB). CFB and OFB is in 64-bits. Encryption or decryption is done in place, i.e. the supplied buffer is overwritten. Also, for all modes other than ECB mode, if a buffer is supplied for *initVect*, the final value of IV (initialisation vector) is written back to *initVec*, allowing large amounts of data to be encrypted in small chunks. *what* can have the value qfDES_encrypt or qfDES_decrypt, and *mode* can have the value qfDES_ecb, qfDES_cbc, qfDES_cfb or qfDES_ofb representing the 4 modes. *size* is the number of bytes of text to be encrypted or decrypted starting at *data*, and must be a positive multiple of 8. *key* (and *initVec*, if supplied) must both be 8 bytes in size.

For convenience, some macros are provided:

qfDES_ECB_e(key data, l) behaves as *qfDES(key data, size, qfDES_encrypt, qfDES_ecb, (char *) 0)*.

qfDES_ECB_e(key data, l) behaves as *qfDES(key data, size, qfDES_decrypt, qfDES_ecb, (char *) 0)*.

qfDES_CBC_e(key data, size, initVec) behaves as *qfDES(key data, size, qfDES_encrypt, qfDES_cbc, initVec)*.

qfDES_CBC_d(key data, size, initVec) behaves as *qfDES(key data, size, qfDES_decrypt, qfDES_cbc, initVec)*.

qfDES_CFB_e(key data, size, initVec) behaves as *qfDES(key data, size, qfDES_encrypt, qfDES_cfb, initVec)*.

qfDES_CFB_d(key data, size, initVec) behaves as *qfDES(key data, size, qfDES_decrypt, qfDES_cfb, initVec)*.

qfDES OFB e(key data, size, initVec) behaves as qfDES(key data, size, qfDES encrypt, qfDES ofb, initVec).

qfDES OFB d(key data, size, initVec) behaves as qfDES(key data, size, qfDES decrypt, qfDES ofb, initVec).

If qfDES() is called in any of CBC, CFB or OFB mode with an a NULL value for initVec, then an IV of zero is used.

qfDES malloc() and qfDES realloc() make calls to malloc(3) and realloc(3), respectively, and work on *ptr. However, the amount of memory these functions allocate is given by rounding size up to the next 8 byte boundary, i.e. the memory is padded to the next 8 bytes. The final byte of the newly allocated memory will contain the number of padding bytes, and the padding bytes will be ’\0’ characters by default. The padding character can be changed from this default to any other character by using qfDES setPad(), and for convenience, two macros are defined for the commonly used padding values of zero (’\0’) and SPACE (’ ’). qfDES copy() will return a pointer to a copy of the size bytes pointed to by ptr. free(3) can be used on pointers returned from qfDES malloc(), qfDES realloc() and qfDES copy(). qfDES free() is provided for convenience, and is simply a wrap-up of free(3). For padded memory space, qfDES plainTextSize() gives the actual length of the plain text (i.e. without the padding). qfDES insertPadding() will insert padding charcters up to the next 8 byte boundary after size in the buffer given by ptr.

qfDES bin2hex() takes a pointer to memory and gives a "hexdump" of it in the form of a NULL terminated string. binText must be a pointer to memory of size size, and hextText must be of size (2 * size + 1). qfDES hex2bin() does the opposite operation for a printable string "hexdump" of size size, (size does not include the NULL terminating character, i.e the size of the string as returned by strlen(3)).

qfDES str2key() takes the (printable ASCII) NULL terminated string str and makes an 8 byte DES key out of it by shifting the bits in each byte in str left by 1 position and using the last bit as odd parity. If the length of str is less than 8, then zero padding is used. If the length of str is greater than 8, then all characters after the 8th character are ignored. This function returns a pointer to a static area of memory, which can be copied using qfDES copy().

qfDES checkWeakKeys() checks key against a set of known 18 weak keys. The 18 known weak keys are:

center; cB cfC . Weak Keys 0000 0000 0000 0000 1111 1111 1111 1111 0101 0101 0101 0101 fefe fefe fefe fefe 1f1f 1f1f 0e0e 0e0e e0e0 e0e0 f1f1 f1f1 01fe 01fe 01fe 01fe fe01 fe01 fe01 fe01 1fe0 1fe0 0ef1 0ef1 e01f e01f f10e f10e 01e0 01e0 01f1 01f1 e001 e001 f101 f101 1ffe 1ffe 0efe 0efe fe1f fe1f fe0e fe0e 011f 011f 010e 010e 1f01 1f01 0e01 0e01 e0fe e0fe f1fe f1fe fee0 fee0 fef1 fef1

qfDES setParity() sets the parity bits of the bytes in ptr. parity can have the values qfDES_odd or qfDES_even for odd or even parity respecyively. qfDESecheckParity() will return the number of parity errors found in the buffer at ptr.

RETURN VALUES

qfDES() returns 0 on success and -1 on failure.

qfDES malloc() and _qfDES realloc()_ return the number of bytes of memory allocated.

qfDES bin2hex() and _qfDES hex2bin()_ return, respectively, the number of bytes converted to or from binary.

qfDES setPad() returns the value of the previous padding character. _qfDES insertPadding()_ returns the size of the padded buffer.

qfDES checkWeakKeys() returns 0 if the key is not in the set of 18 known weak keys, else it returns (-1).

qfDES checkParity() returns the number of parity errors it found.

NOTES

A description of DES can be found in most good text books on data security or computer network security. A couple that I found useful were:

Security for Computer Networks by D. W. Davies and W. L. Price., published by John Wiley & Sons, 1984.

Cryptography and Data Security, by Dorothy E. Denning, published by Addison-Wesley, 1983.

NOTE: FIPS?

Some of the macros used in the definition of _qfDES()_ are not as efficient as they could be. It is probably possible to make them faster by examining the corresponding DES functions more carefully.

SEE ALSO

qf-des(1), qf-key(1), malloc(3), realloc(3), free(3), strlen(3)

AUTHOR

Saleem N. Bhatti <s.bhatti@cs.ucl.ac.uk>, University College London

DISCLAIMER: If you are not me and you have this then it’s nothing to do with me! :-)