Jornadas de criptografía aplicada:
Bibliotecas de funciones de cifrado

Francisco J. 曹 (Tsao) Santín
e-mail: tsao@linuxbeat.net
Grupo de Programadores y Usuarios de Linux- Coruña Linux Users Group
GPUL-CLUG

12 de Julio de 2004

Cifrado clásico (I)

Algoritmos de sustitución:

- Monoalfabéticos: algortimo de César,ROT13
- Homofónicos:Duque de Mantua (1401)
- "Poligram"?:Playfair (I Guerra Mundial)
- Polialfabéticos: Leon Battista (1568), cifrado de Vigenère, XOR

Cifrado clásico (II)

Algoritmos de trasposición:ADFGVX (I Guerra Mundial)

Maquinas de rotor (polialfabético): Enigma (II Guerra Mundial)

Libretas de uso único:Mayor Joseph Mauborgne (1917)

Cifrado moderno

Tipos de algoritmo:

- Cifrado simétrico
- Cifrado asimétrico o de clave pública
- Funciones de sentido único (one way functions)

Cifrado Simétrico (I)

Tipos de cifrado simétrico:

- "stream ciphers" :bit a bit, o byte a byte
- "block ciphers" :grupos de bits o de bytes

Cifrado Simétrico (II)

Algunos algoritmos de cifrado simétrico:

- Data Encryption Standard (DES): cifrado en bloques de 64 bits, llaves de 56 bits. Desarrollado por IBM a principios de los 70. El referente durante muchos años.

- Otros algoritmos: Lucifer, FEAL, REDOC,LOKI, RC2,RC4,RC6,CAST...

- ... y GOST 28147-89

Cifrado Simétrico (III)

Modos de cifrado simétrico:

Puros en bloques:

- Electronic Codebook Mode (ECB)
- Cipher Block Chaining Mode (CBC)

Cifrado Simétrico (y IV)

Implementados como stream ciphers:

- Cipher-Feedback Mode (CFB)
- Output-Feedback Mode (OFB)

Funciones de sentido único

- No son técnicas de cifrado en sentido estricto. Basados en funciones de compresión
- Directamente relacionadas con cifrado asimétrico y firma digital
- "One way hash functions" : Snefru, N-hash,MD5,CRC, ...y GOST

Cifrado asimétrico

Creo que ya os han dado suficiente caña todo el día con ello: RSA, etc.

Algunas bibliotecas de funciones interesantes

Y libres, por supuesto :-):

- Cifrado simétrico: libmcrypt: multiples algoritmos y modos. Autor: Nikos Mavroyanopoulos
- Cifrado asimétrico: libgcrypt: bibliotecas básicas del proyecto gnupg
- Funciones hash de sentido único: libmhash: multiples funciones, generacion de llaves.Autores: Nikos Mavroyanopoulos y Sascha Schumann

libmcrypt (I)

Funciones de alto nivel:

Iniciación/finalización del contexto de cifrado

```
MCRYPT mcrypt_module_open(char *algorithm,
  char *a_directory, char *mode,
  char *m_directory)\\
int mcrypt_module_close(MCRYPT td)\\
```

Iniciación/liberación de buffers

```
int mcrypt_generic_init(const MCRYPT td, void *key, int lenofkey,
void *IV)
int mcrypt_generic_deinit(const MCRYPT td)

int mcrypt_generic_end(const MCRYPT td) (no usar)
```

libmcrypt (II)

Cifrado/descifrado

```
int mdecrypt_generic(MCRYPT td, void *plaintext, int len)
int mcrypt_generic(MCRYPT td, void *plaintext, int len)
```

Funciones auxiliares:

Funciones de estado

```
int mcrypt_enc_set_state(MCRYPT td, void *st, int size)
int mcrypt_enc_get_state(MCRYPT td, void *st, int *size)
```

libmcrypt (III)

Funciones de test e informacion sobre el thread

```
int mcrypt_enc_self_test(MCRYPT td)
int mcrypt_enc_get_block_size(MCRYPT td)
int mcrypt_enc_get_iv_size(MCRYPT td)
int mcrypt_enc_get_key_size(MCRYPT td)
int mcrypt_enc_is_block_algorithm(MCRYPT td)
int mcrypt_enc_is_block_mode(MCRYPT td)
int mcrypt_enc_is_block_algorithm_mode (MCRYPT td)
int mcrypt_enc_mode_has_iv(MCRYPT td)
char *(mcrypt_enc_get_algorithms_name) (MCRYPT td)
char *(mcrypt_enc_get_modes_name) (MCRYPT td)
```

libmcrypt (IV)

```
int *mcrypt_enc_get_supported_key_sizes(MCRYPT td, int *len)
char **mcrypt_list_algorithms(char *libdir, int *size)
char **mcrypt_list_modes(char *libdir, int *size)
```

Funciones de manejo de punteros:

```
void mcrypt_free_p(char **p, int size)
void mcrypt_free(void *ptr)
```

libmcrypt (V)

Funciones de impresión de error:

```
void mcrypt_perror(int err)
const char* mcrypt_strerror(int err)
```

Funciones de informacion general (no necesita inicialización)

```
int mcrypt_module_self_test(char *algorithm, char *a_directory)
int mcrypt_module_is_block_algorithm(char *algorithm,
     char *a_directory)
int mcrypt_module_is_block_algorithm_mode(char *mode,
  char *m_directory)
int mcrypt_module_is_block_mode(char *mode, char *m_directory)
```

libmcrypt (VI)

```
int mcrypt_module_get_algo_key_size(char *algorithm,
                                    char *a_directory)
int mcrypt_module_get_algo_block_size(char *algorithm,
 char *a_directory)
int *mcrypt_module_get_algo_supported_key_sizes(char *algorithm,
char *a_directory,int *len);
int mcrypt_module_algorithm_version(char *algorithm,
    char *a_directory)
int mcrypt_module_mode_version(char *mode, char *a_directory)
```

libmcrypt (y VII)

Funcion para aplicaciones multithread

```
int mcrypt_mutex_register ( void (*mutex_lock)(void) ,
void (*mutex_unlock)(void),
void (*set_error)(const char*),
const char* (*get_error)(void))
```

libmhash (I)

Control del proceso

Inicialización:

```
MHASH mhash_init(hashid type)
```

Copia/actualización:

```
MHASH mhash_cp(MHASH)
int mhash(MHASH thread, const void *plaintext, size_t size)
```

libmhash (II)

Finalización:

```
void *mhash_end(MHASH thread)
void *mhash_end_m(MHASH thread, void *(*hash_malloc) (size_t))
void mhash_deinit(MHASH thread, void *result)
```

Información sobre el thread:

```
size_t mhash_count(void)
size_t mhash_get_block_size(hashid type)
char *mhash_get_hash_name(hashid type)
size_t mhash_get_hash_pblock(hashid type)
hashid mhash_get_mhash_algo(MHASH)
```

libmhash (III)

Generación HMAC

```
MHASH mhash_hmac_init(const hashid type, void *key, int keysize,
      int block)
void *mhash_hmac_end_m(MHASH thread, void *(*hash_malloc) (size_t))
void *mhash_hmac_end(MHASH thread)
int mhash_hmac_deinit(MHASH thread, void *result)
int mhash_save_state_mem(MHASH thread, void *mem, int* mem_size )
MHASH mhash_restore_state_mem(void* mem)
```

libmhash (IV)

Generación de llaves:

```
int mhash_keygen(keygenid algorithm, hashid opt_algorithm,
 unsigned long count, void *keyword, int keysize,
 void *salt, int saltsize, unsigned char *password,
 int passwordlen)
int mhash_keygen_ext(keygenid algorithm, KEYGEN data,
 void *keyword, int keysize,
 unsigned char *password, int passwordlen)
```

libmhash (y V)

Información sobre generación de llaves:

```
char *mhash_get_keygen_name(keygenid type)
const char *mhash_get_keygen_name_static(hashid type)
size_t mhash_get_keygen_salt_size(keygenid type)
size_t mhash_get_keygen_max_key_size(keygenid type)
size_t mhash_keygen_count(void)
int mhash_keygen_uses_salt(keygenid type)
int mhash_keygen_uses_count(keygenid type)
int mhash_keygen_uses_hash_algorithm(keygenid type)
```

Ejemplo 1: Cifrado simétrico con GOST

```
#include <stdio.h>
#include "mcrypt.h"
#include <stdlib.h>
#include <string.h>
#include "mhash.h"
main (int argc,char* argv[]){
  MCRYPT td;
  char password[32];
  FILE *ifile;
  FILE *ofile;
  char *block_buffer;
  char *key;
```

```
char *IV;
int keysize;
int blocksize;
int flag,i;

ifile=fopen(argv[1],"r");
ofile=fopen(argv[2],"w");
strcpy(&password[0],argv[3]);
flag=atoi(argv[4]);
td=mcrypt_module_open("gost",NULL,"cfb",NULL);

keysize=mcrypt_enc_get_block_size(td);
key=calloc(1,keysize);
  mhash_keygen(KEYGEN_MCRYPT, MHASH_GOST,0, key, keysize,NULL, 0, passw
```

```
if(td==MCRYPT_FAILED){
  return 1;
}

blocksize=mcrypt_enc_get_block_size(td);
block_buffer=malloc(blocksize);
IV=malloc(mcrypt_enc_get_iv_size(td));

  for (i=0;i<mcrypt_enc_get_iv_size(td);i++){
    IV[i]=rand();
  }
mcrypt_generic_init(td,key,keysize,IV);
  while(fread(block_buffer,1,blocksize,ifile)==blocksize){
```

```
      if (flag==0){
mcrypt_generic(td,block_buffer,blocksize);
      }
      if(flag==1){
mdecrypt_generic(td,block_buffer,blocksize);
      }

    fwrite(block_buffer,1,blocksize,ofile);
    }
/* Y EL PADDING!!!! */
  mcrypt_generic_deinit(td);
  mcrypt_module_close(td);
  close(ifile);
```

```
  close(ofile);

}
```

Ejemplo 2: One-way hashing con GOST

```
#include <stdio.h>
#include <stdlib.h>
#include "mhash.h"
main(int argc,char* argv[]){
  MHASH td;
  FILE *ifile;
  unsigned char *buffer;
  unsigned char *hash;
  int block_size,i;

  ifile=fopen(argv[1],"r");
  td=mhash_init(MHASH_GOST);
```

```
  while (fread(buffer,1,1,ifile) == 1){
    mhash(td,buffer,1);
  }
  hash=mhash_end(td);
  block_size=mhash_get_block_size(MHASH_GOST);
  for (i=0;i<block_size;i++){
    printf("%.2x",hash[i]);
  }
  printf("\n");
}
```

Makefile para los dos ejemplos

```
gostc : mainc.o
gcc -o gostc -lmcrypt -lmhash -lltdl mainc.o
mainc.o : mainc.c
gcc -c mainc.c
gosth : mainh.o
gcc -o gosth -lmcrypt -lmhash -lltdl mainh.o
mainh.o : mainh.c
gcc -c mainh.c

clean :
rm mainc.o mainh.o gostc gosth
```

Bibliografía

- "The GOST Encryption Algorithm" , de Bruce Scheiner. Dr.Dobb's Journal, enero de 1995

- "Applied cryptography", de Bruce Schneier.Wiley & Sons, 2ª ed.1996

- "The Codebreakers",de David Kahn. Scribner,1996

- "Criptonomicón", de Neal Stephenson.Ediciones B,2004 ;-)

Direcciones de interés

- Página del proyecto mcrypt: http://mcrypt.sourceforge.net
- Página de Bruce Schneier: http://www.counterpane.net