criptografía aplicada

Ejemplos de programación criptográfica

Ana Saiz García

ana (at) gpul (dot) org

David Fernández Vaamonde

david_fv (at) gpul (dot) org

II Taller de Criptografía aplicada

Universidade da Coruña

criptografía aplicada

Guión

- Parte I: Introducción a las técnicas de cifrado y autenticación
- Parte II: Programación criptográfica con Cryptlib
- Parte III: Programación criptográfica con Mono
- Conclusiones
- Bibliografía y referencias

criptografía aplicada

Parte I
Introducción a las técnicas de cifrado y autenticación

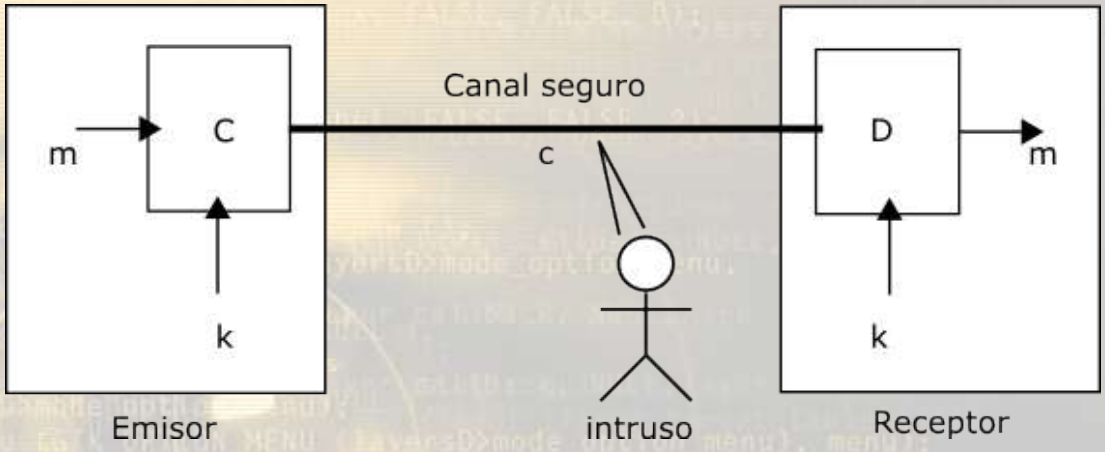
- Introducción
- Cifrado simétrico
- Cifrado de clave pública
- Autenticación

Introducción (I)

- Criptología:
  - Criptografía:
    - mensaje inteligible → no inteligible y viceversa
    - utiliza una clave
  - Criptoanálisis:
    - mensaje no inteligible → inteligible sin conocer la clave
- Un sistema puede ser:
  - Incondicionalmente seguro (ideal)
  - Computacionalmente seguro

Introducción (II)

- Criptosistema:



- Terminología:
  - **m**: **texto en claro**
  - **c**: **texto cifrado** o **criptograma**. Mensaje transformado en ininteligible
  - **algoritmos de cifrado**: convierten m en c
  - **k**: **clave** utilizada por los algoritmos de cifrado
  - **C**: **cifrado**. Proceso de convertir m en c utilizando k y un algoritmo de cifrado: `c=Ck[m]=Ek[m]`
  - **D**: **descifrado**. Proceso de convertir c en m utilizando k y un algoritmo de cifrado: `m=Dk[c]`

Introducción (III)

- Servicios de seguridad
  - Confidencialidad
  - Integridad
  - Autenticación
  - No repudio
- Mecanismos de seguridad
  - Cifrado
  - Generación de tráfico
  - Integridad de los datos
  - Protocolos de autenticación
  - Firma digital

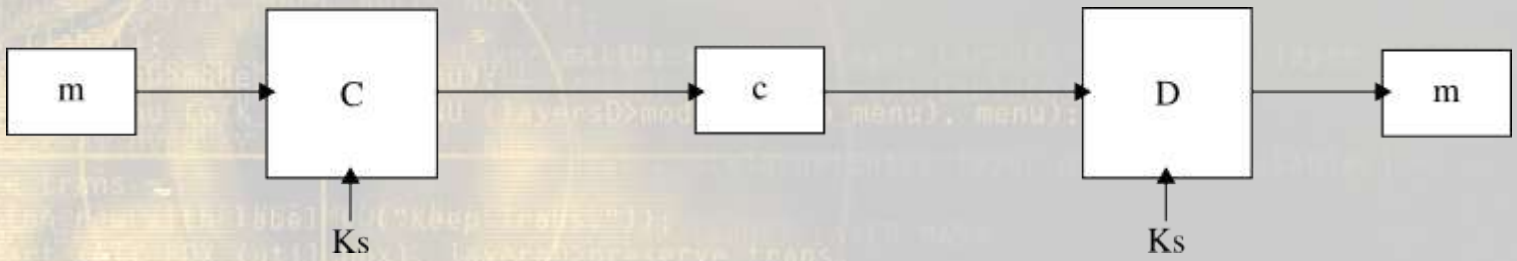
criptografía aplicada

Introducción (IV)

- Ataques a la información:
  - A la confidencialidad:
    - Revelación: averiguar contenido mensaje
    - Análisis de tráfico: descubrir patrón
  - A la autenticidad:
    - Suplantación de la fuente
    - Modificación del contenido del mensaje
    - Modificación de secuencia de mensajes
    - Modificación de tiempo
  - Repudio
    - De origen: niega haber enviado
    - De destino: niega haber recibido

Introducción (V)

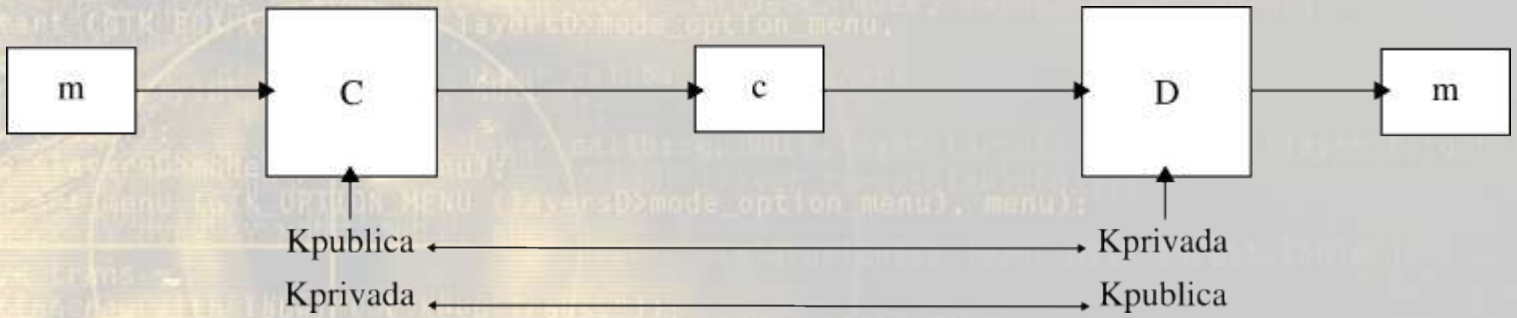
- Sistemas de cifrado simétrico (o de clave secreta)
  - Clave secreta compartida por emisor y receptor
  - Emisor cifra con la clave y receptor descifra con la misma clave



  - Servicios:
    - confidencialidad
    - autenticación
    - integridad
  - Problemas:
    - distribución y gestión de claves

Introducción (VI)

- Sistemas de cifrado asimétrico (o de clave pública)
  - Dos claves: privada y pública asociadas
  - Si se cifra con una clave sólo se puede descifrar con la otra



  - Servicios:
    - cifrado con privada y descifrado con pública → autenticación y no repudio
    - cifrado con pública y descifrado con privada → confidencialidad
  - Problemas:
    - cifrado lento
    - distribución de clave pública

criptografía aplicada

Introducción (y VII)

- Otros mecanismos
  - Intercambio de clave: Diffie-Hellman
  - Protocolos de autenticación
  - Funciones HASH
  - Funciones MAC
  - Firma digital

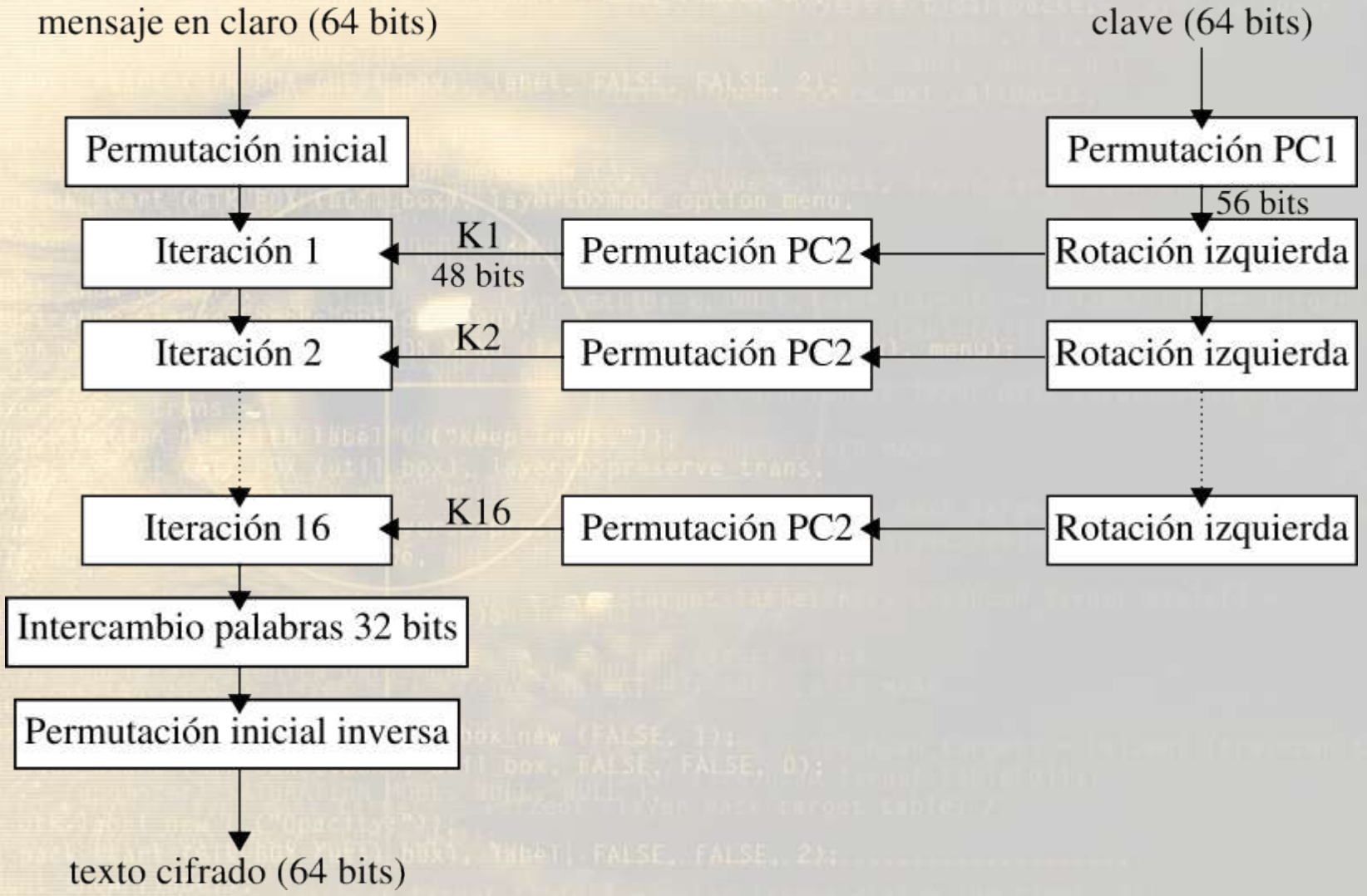
Cifrado simétrico (I)

- Seguridad del cifrado
  - ⇒ privacidad de la clave
  - ⇏ privacidad del algoritmo → no necesita ser secreto
- Bloque cifrador
  - Elemento básico de cifrado
  - Efectos:
    - Avalancha: cambio 1 bit de entrada ⇒ cambio n/2 bits de salida
    - Integridad: cada bit de salida función de todos los de entrada
  - Operaciones básicas:

Substitución:
clave: función de substitución
confusión
cajas-S (S-boxes)

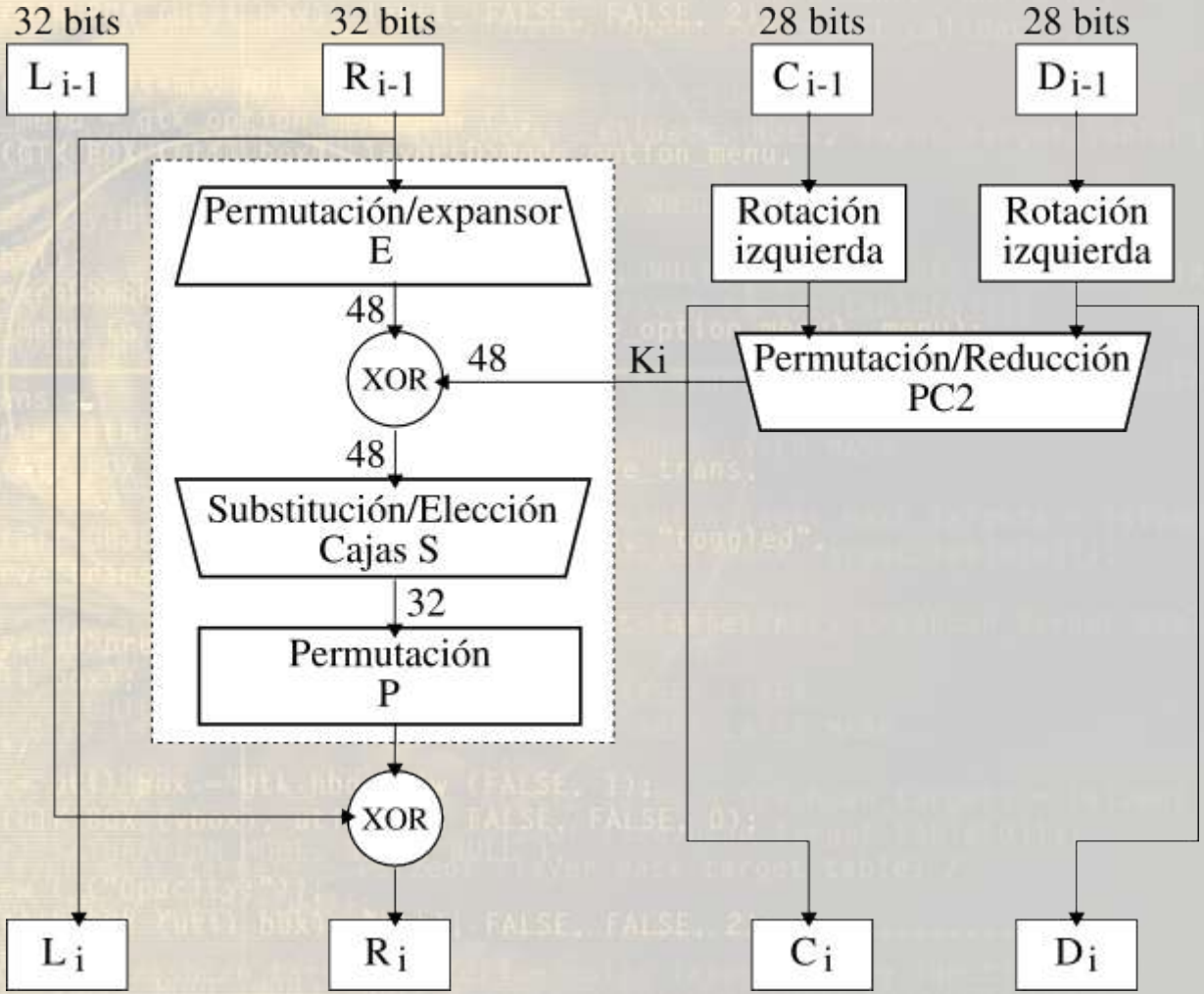
Permutación:
clave: reordenación de bits
difusión
cajas-P (P-boxes)

Cifrado simétrico (II)
DES (I)



Cifrado simétrico (III)
DES (II)

- Detalle de la iteración



criptografía aplicada

Cifrado simétrico (IV)
DES (III): Modos de uso

- ECB: Electronic Codebook
  - cifrado de bloque (64 bits)
  - bloques independientes → repeticiones en bloques en claro se pueden repetir en texto cifrado
  - válido para mensajes cortos (1 bloque)
  - el más rápido
- CBC: Cipher Block Chaining
  - cifrado de bloque (64 bits)
  - realimentación del bloque cifrado anterior
  - necesita vector de inicialización
  - válido para mensajes mayores de 64 bits
- CFB: Cipher Feedback
  - cifrado de bloques de j bits
  - si j=1 o j=8 → cifrado de flujo
  - necesita vector de inicialización
  - válido para tráfico de flujo
- OFB: Output feedback
  - cifrado de flujo
  - necesita vector de inicialización
  - necesita sincronización emisor/receptor
  - no propaga errores

Cifrado simétrico (V)
DES (y IV)

- Ventajas
  - Fuerte efecto avalancha
- Desventajas
  - clave de 56 bits → demasiado corta (ataque de fuerza bruta)
- Triple DES
  - Variante de DES con dos o tres claves:

  - `c=Ck3[Dk2[Ck1[m]]]`
  - No vulnerable a fuerza bruta

Cifrado simétrico (y VI)

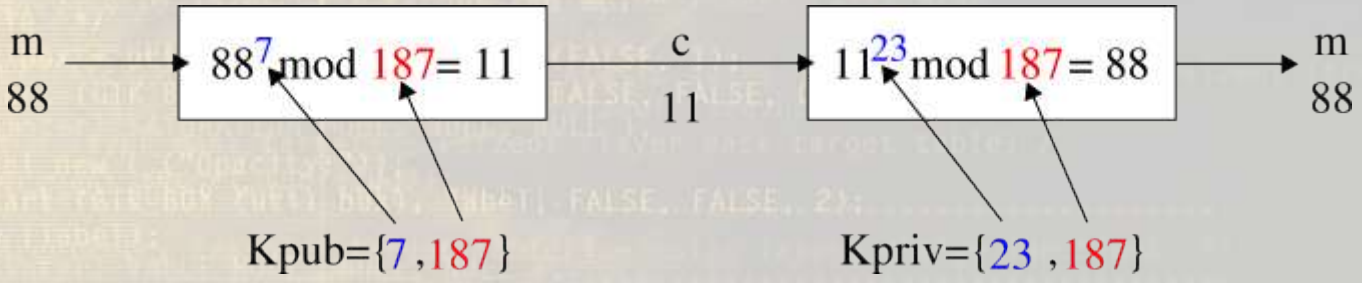
- Otros algoritmos:
  - IDEA
  - Skipjack
  - Blowfish
  - RC2
  - RC5
  - CAST-128
  - AES

Criptografía de clave pública (I)

- Seguridad del cifrado: imposible averiguar Ksecreta conociendo los algoritmos de cifrado y descifrado, Kpública y criptogramas
- Categorías:
  - Cifrado/descifrado ⇒ confidencialidad
  - Firma digital ⇒ autenticación y no repudio
  - Intercambio de clave secreta compartida (cifrado simétrico)
- Confidencialidad + autenticación: cifrar primero con clave pública del receptor y después con clave privada del emisor.
- Ataques:
  - Fuerza bruta sobre clave
  - Fuerza bruta sobre mensaje (mensajes cortos)
  - Cálculo de clave privada

Criptografía de clave pública (II)
RSA (I)

- Texto en claro y cifrado son enteros entre 0 y n-1 (valor típico n=1024 bits)
- Bloques de longitud $\leq \log_2(n)$ bits
- Para un bloque de texto en claro M y un bloque de texto cifrado C:
  - Cifrado: $C = M^e \bmod n$
  - Descifrado: $M = C^d \bmod n = M^{ed} \bmod n$
  - emisor conoce clave pública de receptor: {n, e}
  - receptor conoce clave privada propia: {n, d}



criptografía aplicada

Criptografía de clave pública (y III)
RSA (y II)

- La seguridad se basa en la dificultad de la factorización
- Ataques a RSA
  - Fuerza bruta
    - hacer claves más largas ⇒ cifrado/descifrado más lento
  - Ataque matemático
    - costoso si n grande
  - Ataque de temporización ⇒ observación de tiempos de ejecución para determinados valores aplicados a los algoritmos ⇒ sabiendo tiempo, intuir valor
    - forzar tiempo constante
    - dar retardo aleatorio
    - blindar texto cifrado multiplicándolo por número aleatorio antes de pasarlo al algoritmo

Autenticación (I)

- Procedimiento para verificar la autenticidad de un mensaje:
  - que la fuente es la que dice ser
  - que el contenido no ha sido modificado
- Valor utilizado para autenticar un mensaje: **autenticador**
- Mecanismos para generar el autenticador:
  - **Cifrado del mensaje**: el mensaje cifrado sirve como autenticador del mensaje
  - **Código de Autenticación de Mensaje (MAC)**: función pública del mensaje y una clave secreta ⇒ valor de longitud fija
  - **Función Hash**: función pública que genera un valor de longitud fija a partir de un mensaje de cualquier longitud

Autenticación (II)

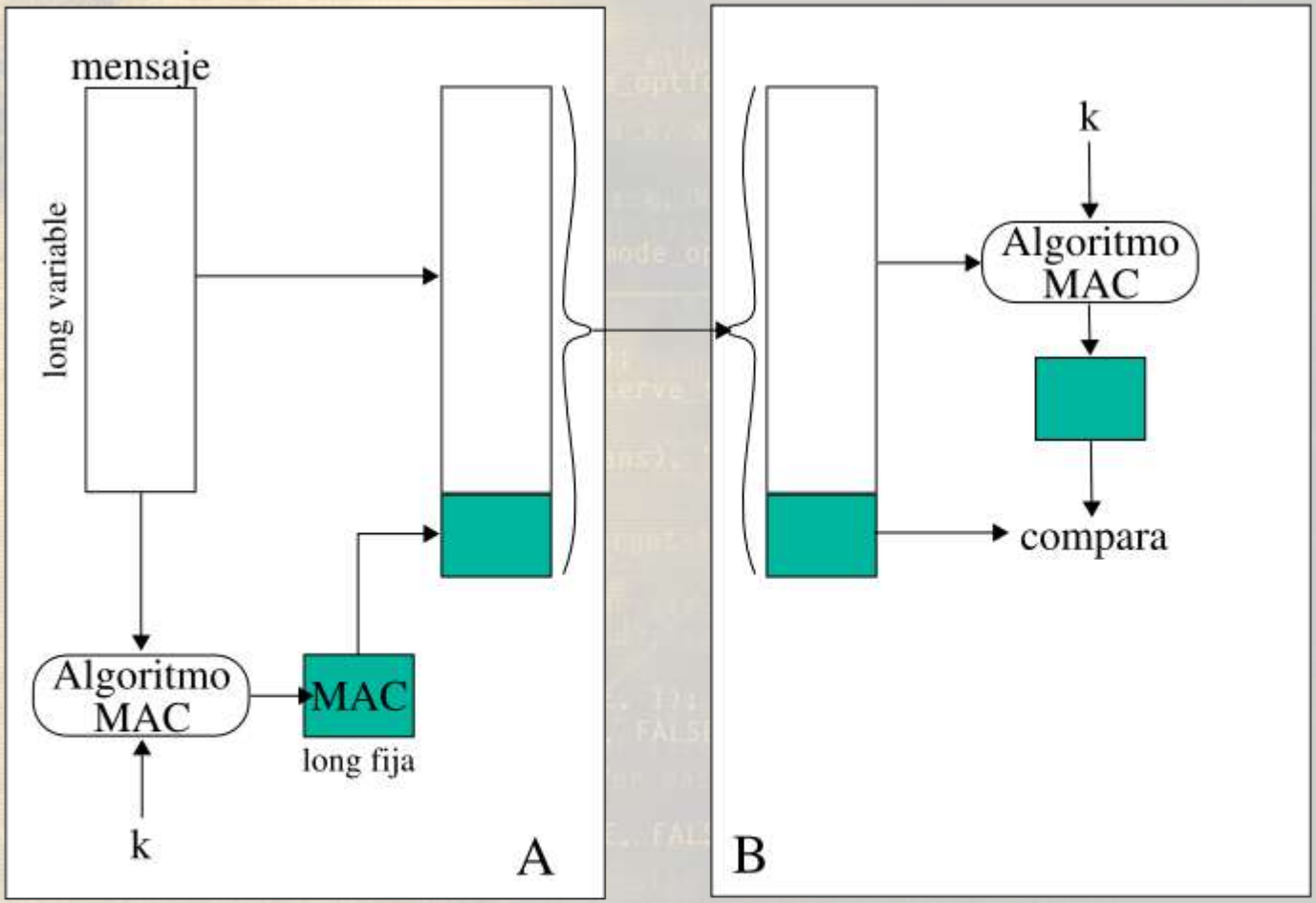
- **Autenticación mediante cifrado**
  - Autenticación del remitente (A):
    - cifrado simétrico:
      - A y B comparten clave secreta, sólo el otro puede haber cifrado con esa clave
      - ofrece confidencialidad
      - no ofrece no repudio
    - cifrado asimétrico:
      - cifrado con clave privada de A, sólo A puede haberlo cifrado
      - ofrece no repudio
      - no ofrece confidencialidad
      - no es eficiente en mensajes largos
  - Autenticación del mensaje:
    - código de detección de errores, número de secuencia, marca de tiempo

Autenticación (III)

- **Autenticación sin cifrado**
  - Se genera una referencia de autenticación y se añade al mensaje, que se transmite en claro
  - No confidencialidad
  - Usos:
    - cuando no importa la confidencialidad
    - mensaje para varios destinos: sólo 1 verifica autenticidad ⇒ más barato y más rápido
    - receptor muy cargado que no puede descifrar todos los mensajes que le llegan ⇒ sólo verifica algunos
    - cuando interesa mantener la protección de autenticación más allá de la recepción
  - Código de autenticación de mensaje
  - Funciones hash

Autenticación (IV)
MAC

- Código de autenticación de mensaje

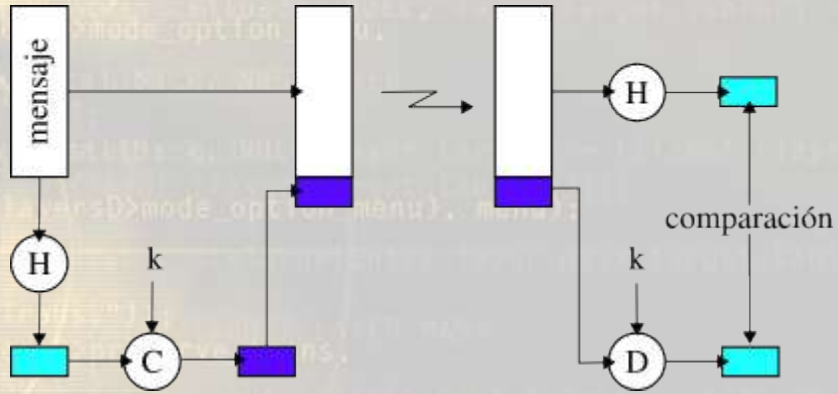


criptografía aplicada

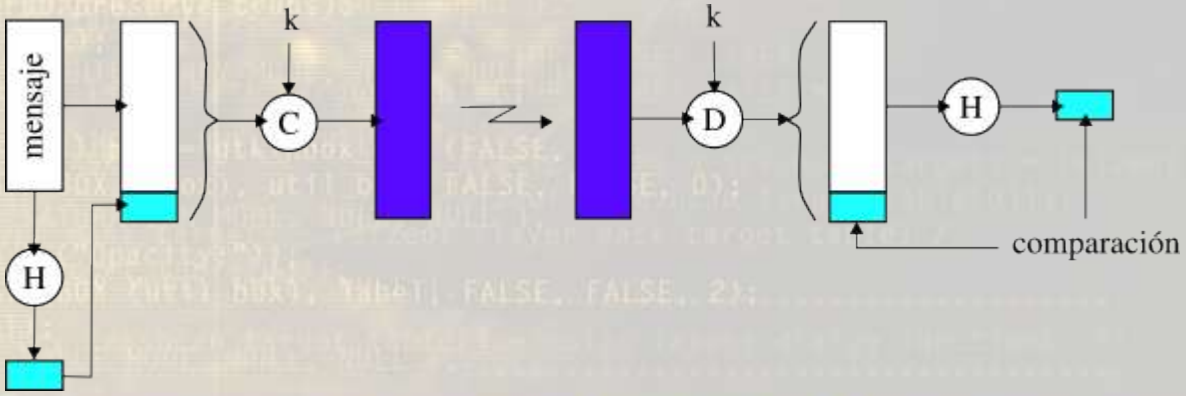
Autenticación (V)
HASH (I)

- No necesita clave
- Usos:
  - Con **cifrado simétrico**:
    - autenticación

$m | C_k[h(m)]$

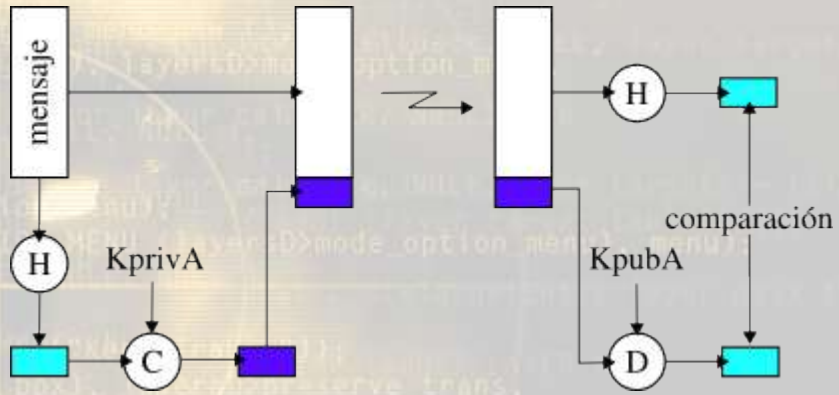


    - autenticación + confidencialidad $C_k[m | h(m)]$

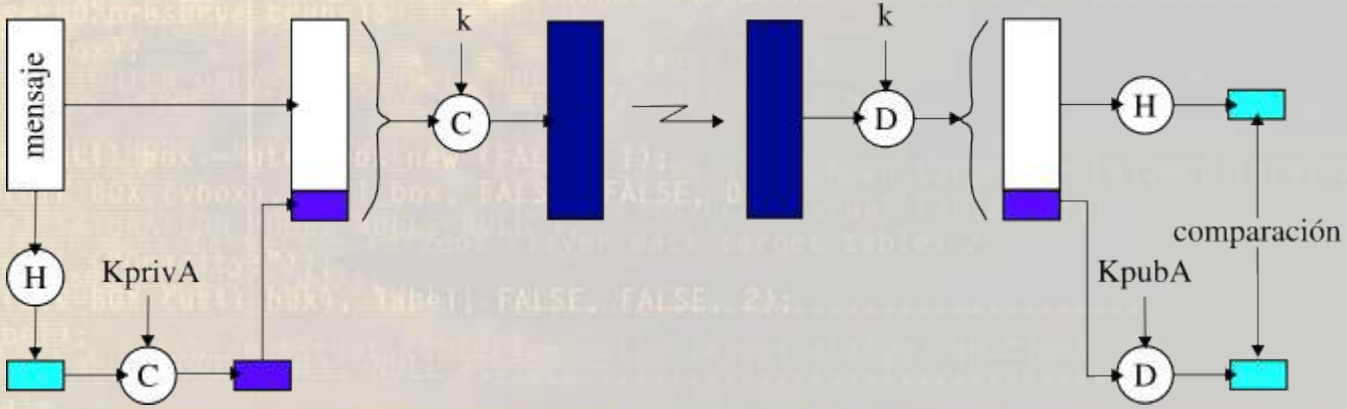


Autenticación (VI)
HASH (II)

- Con **cifrado asimétrico**:
  - autenticación (firma digital) $m|C_{KprivA}[h(m)]$



  - con confidencialidad $C_k[m|C_{KprivA}[h(m)]]$

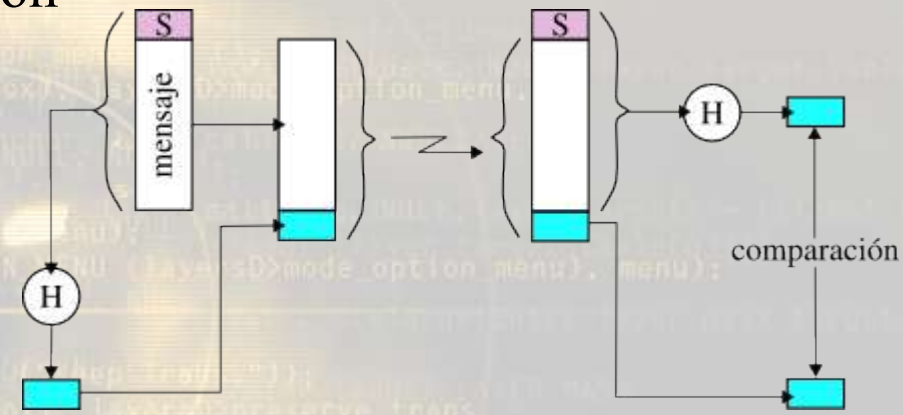


criptografía aplicada

Autenticación (VII)
HASH (y III)

- **Sin cifrado**: secreto compartido S
  - autenticación

`m|h(m|S)`

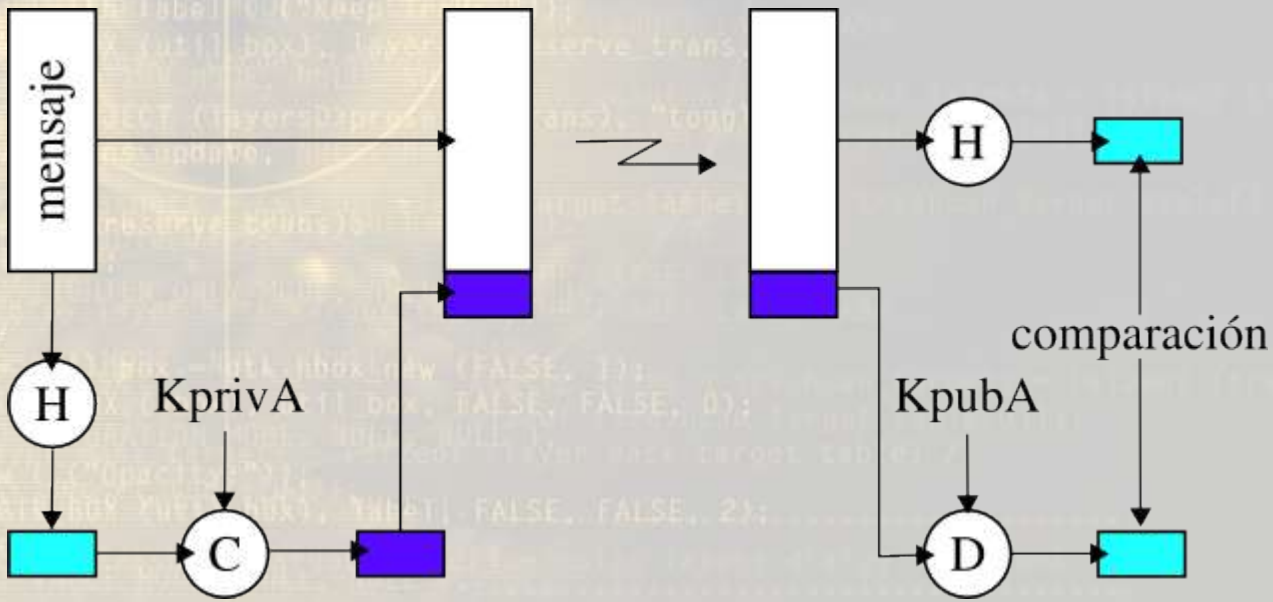


- Algoritmos:
  - MD5
  - SHA-1
  - RIPEMD
- HMAC: MAC con HASH

criptografía aplicada

Autenticación (y VIII)
Firma digital

- Cifrado asimétrico sobre resumen hash de un mensaje ⇒ se reduce tiempo de procesamiento
- No ofrece confidencialidad
- Ofrece no repudio de origen
- Ejemplo: firma digital con RSA



criptografía aplicada

criptografía aplicada

Parte II
Programación criptográfica con Cryptlib

- Introducción
- Ideas generales
- Ejemplo 1: cifrado simétrico
- Ejemplo 2: firma digital

Introducción

- Librerías criptográficas multiplataforma
- Tres niveles de abstracción:
  - alto nivel
  - nivel medio
  - nivel bajo

  mayor conocimiento de las técnicas y algoritmos de cifrado ↓
- Algoritmos
  - Cifrado simétrico: DES, Triple DES, IDEA, Skipjack, CAST-128, Blowfish, RC2, RC4, RC5
  - Cifrado asimétrico: RSA, DSA, El Gamal
  - Hash: MD2, MD3, MD5, RIPEMD-160, SHA
  - MAC: HMAC-MD5, HMAC-SHA, HMAC-RIPEMD-160
  - Intercambio de clave: Diffie-Hellman
  - Certificados
  - Dispositivos hardware

criptografía aplicada

Ideas generales (I)
Inicio/final

- Inicializar cryptlib
- Limpiado y eliminación de objetos que hemos olvidado destruir

```
#include “cryptlib.h”

main{

    cryptInit( );

    ...

    cryptEnd( );

}
```

Ideas generales (II)
Gestión de errores

- Todas las funciones devuelven un entero con un error asociado (definido en cryptlib.h)
- Dos macros:
  - cryptStatusError(status)
  - cryptStatusOK(status)

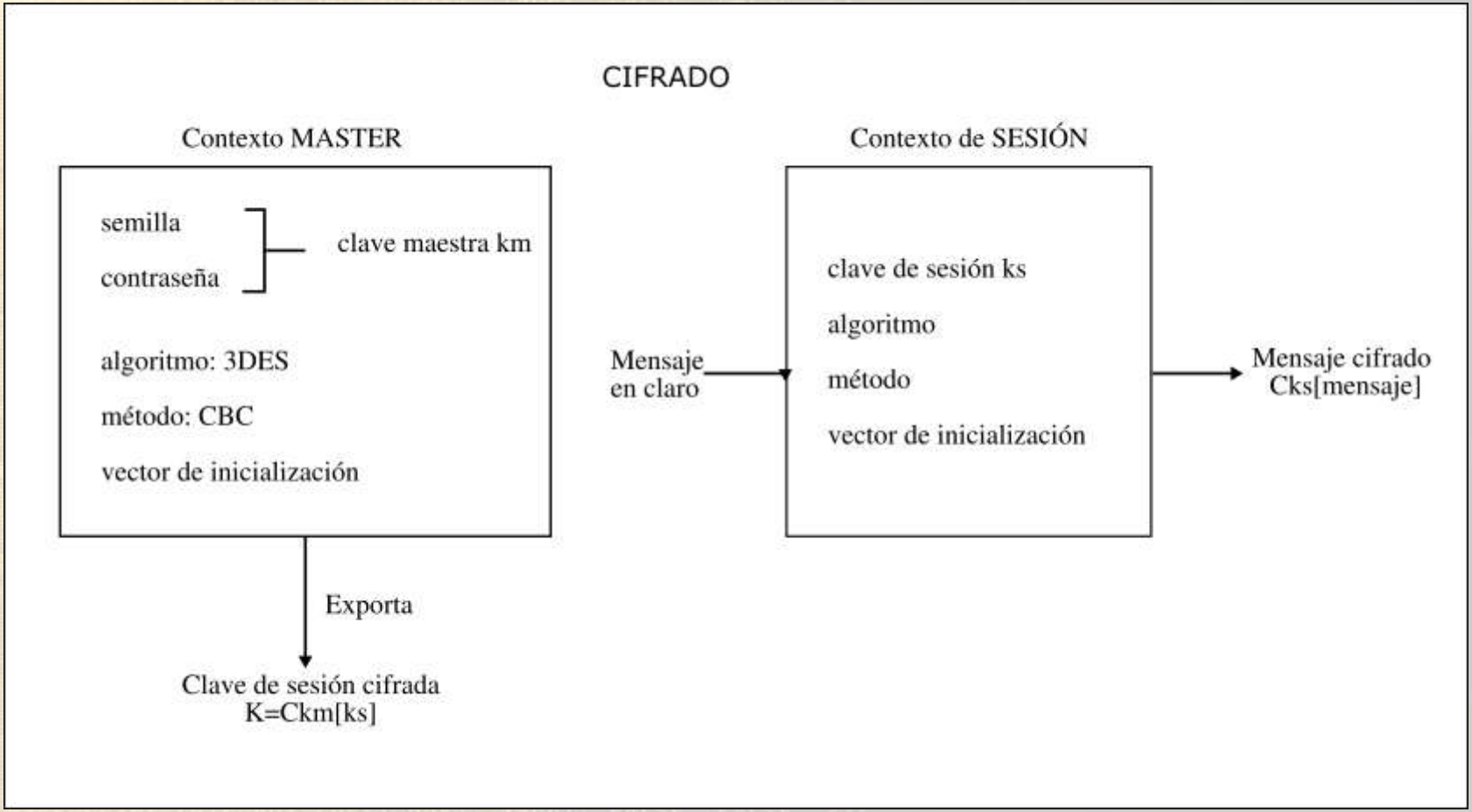
```
status=cryptInit();
  if(cryptStatusError(status))
    printf("Init: %d\n", status);

status=cryptCheckSignature(signature, publicKey,
   shaContext);
  if(cryptStatusOK(status))
    printf("Firma correcta");
```

Ideas generales (y III)
Contextos de cifrado y keysets

- Contextos:
  - Objetos a nivel intermedio
  - Se crean con atributos:
    - Propietario
    - Algoritmo y modo de cifrado
    - Vector de inicialización...
  - Se generan, derivan o importan claves
  - Se cifra/descifra en el contexto
- Keysets
  - Contenedores abstractos (alto nivel)
  - Almacenan una o más claves públicas o privadas
  - Evitar conocer formato y parámetros de las claves

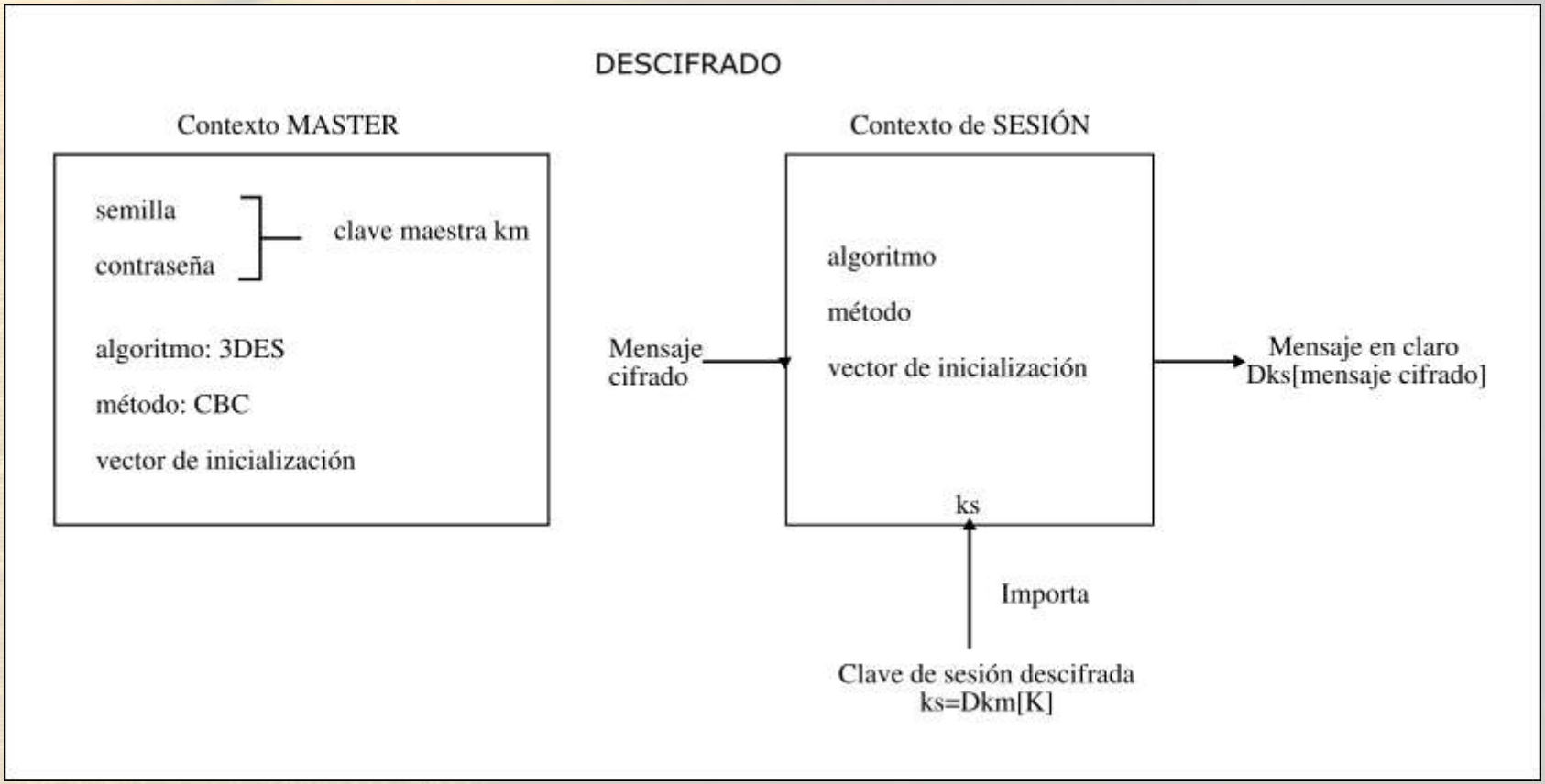
Ejemplo 1: Cifrado simétrico (I)

- Esquema contextual (I)



Ejemplo 1: Cifrado simétrico (II)

- Esquema contextual (y II)



Ejemplo 1: Cifrado simétrico (III)

Definiciones

```
 5  #include <stdio.h>
 6  #include <stdlib.h>
 7  #include <string.h>
 8  //biblioteca para cryptlib
 9  #include "cryptlib.h"
10
11  #define MAXMSJ 1024
12  #define MAXPSWD 16
13  #define SEMILLA "12345678"
14  #define IV "87654321"
15  #define ALGORITMO CRYPT_ALGO_3DES
16  #define MODO CRYPT_MODE_CFB
17
18  //estructura para intercambio de parametros
19  struct datos{
20    char clave_sesion[128];
21    char msj_cifrado[MAXMSJ];
22  };
23  typedef struct datos datos;
24
25  //definicion de funciones de cifrado y descifrado
26  int cifra(datos *mensaje);
27  int descifra(datos mensaje);
```

criptografía aplicada

Ejemplo 1: Cifrado simétrico (IV)

Función principal

```
31  int main(int argc, char *argv[]){
32
33    datos mensaje;
34
35    system("clear");
36
37    printf("Aplicacion que cifra un mensaje con cifrado simetrico\n");
38    printf("-----------------------------------------------\n");
39
40    cifra(&mensaje);
41
42    descifra(mensaje);
43
44    exit(0);
45
46  }
```


criptografía aplicada

Ejemplo 1: Cifrado simétrico (V)

Cifrado (I)

```
50  int cifra(datos *mensaje){
51
52    int n, status, encryptedKeyLength;
53    char password[MAXPSWD];
54    char msj[MAXMSJ];
55    //Estructuras para trabajo con cryptlib
56    CRYPT_CONTEXT masterContext;
57    CRYPT_CONTEXT sessionContext;
58
59    /********** valores de entrada **********/
60    printf("\n\nFUNCION DE CIFRADO\n");
61
62    //Mensaje a cifrar
63    printf("\nIntroduzca el mensaje a cifrar:\n");
64    fgets(msj,MAXMSJ,stdin);
65    n=strlen(msj);
66    msj[n]=0;
67
68    //Contraseña
69    printf("\nIntroduzca su contraseña secreta:\t");
70    scanf("%s", password);
71    n=strlen(password);
72    password[n]=0;
```

Ejemplo 1: Cifrado simétrico (VI)

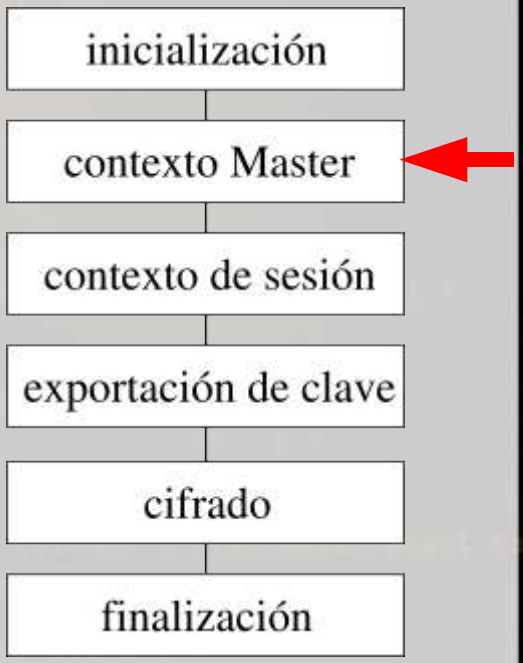
Cifrado (II)

```
75    /************ Inicio del cifrado **********/
76    status=cryptInit();
77    if(cryptStatusError(status))
78      printf("Init: %d\n", status);
```

Ejemplo 1: Cifrado simétrico (VII)
Cifrado (III)

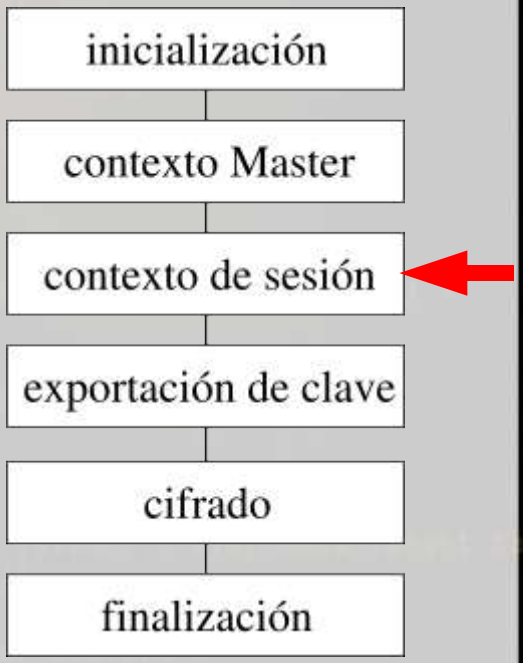
criptografía aplicada

```
81    /******* Contexto Master ********/
82
83    //Creacion del contexto
84    //DES para cifrar la clave de sesion
85    cryptCreateContext(&masterContext,
       CRYPT_UNUSED, CRYPT_ALGO_DES);
86
87    //Derivacion de claves: obtener clave maestra a
       partir de la password del usuario
88    //se introduce la semilla
89    cryptSetAttributeString(masterContext,
       CRYPT_CTXINFO_KEYING_SALT, SEMILLA, 8);
90
91    //se introduce la contraseña
92    cryptSetAttributeString(masterContext,
       CRYPT_CTXINFO_KEYING_VALUE, password,
       strlen(password));
93
94    //se introduce el vector de inicializacion si
       es necesario.
95    //el modo por defecto (CBC), lo necesita
96    cryptSetAttributeString(masterContext,
       CRYPT_CTXINFO_IV, IV, 8);
```



Ejemplo 1: Cifrado simétrico (VIII)
Cifrado (IV)

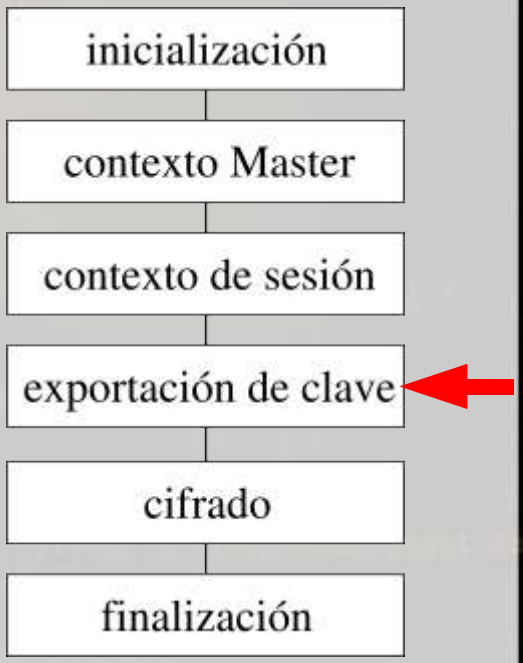
```
 99    /************ Contexto de sesion **********/
100
101    //Generacion del contexto
102    cryptCreateContext(&sessionContext,
        CRYPT_UNUSED, ALGORITMO);
103
104    //Seleccion del modo de operacion
105    cryptSetAttribute(sessionContext,
        CRYPT_CTXINFO_MODE, MODO);
106
107    //Generacion de clave de sesion
        de forma aleatoria
108    cryptGenerateKey(sessionContext);
109
110    //vector de inicializacion si es necesario
        (en todos los metodos menos ECB)
111    cryptSetAttributeString(sessionContext,
        CRYPT_CTXINFO_IV, IV, 8);
```



criptografía aplicada

Ejemplo 1: Cifrado simétrico (IX)
Cifrado (V)

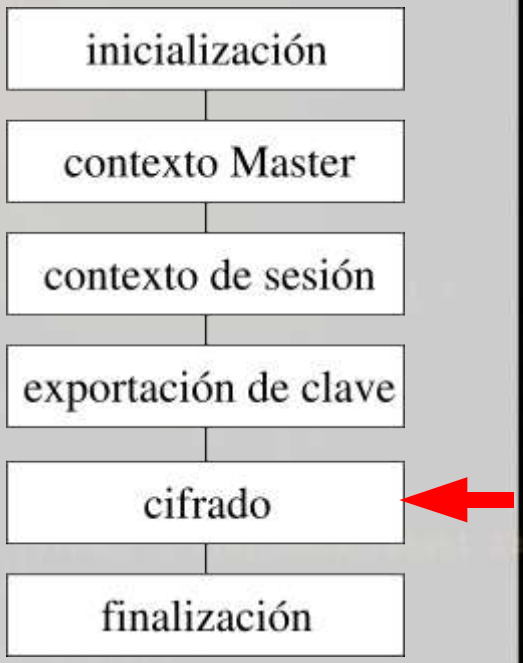
```
114     /***** Exportacion de clave *****/
115
116     //Calculo de la memoria necesaria
117     cryptExportKey(NULL, &encryptedKeyLength,
         masterContext, sessionContext);
118
119     //Exportacion
120     cryptExportKey(mensaje->clave_sesion,
         &encryptedKeyLength, masterContext,
         sessionContext);
```



criptografía aplicada

Ejemplo 1: Cifrado simétrico (X)
Cifrado (VI)

```
123     /******** Cifrado de los datos *******/
124
125     //Cifra el contenido de msj con la clave
         y metodos asociados al contexto de sesion
126     //y lo vuelve a guardar en msj
127     n=strlen(msj);
128
129     cryptEncrypt(sessionContext, msj, n);
```



criptografía aplicada

Ejemplo 1: Cifrado simétrico (XI)
Cifrado (VII)

```
132     /****** Destruccion de contextos *******/
133
134     cryptDestroyContext(sessionContext);
135
136     cryptDestroyContext(masterContext);
137
138     /********* Fin del cifrado ************/
139
140     cryptEnd();
```

Ejemplo 1: Cifrado simétrico (XII)
Cifrado (y VIII)

```
143     /*************************************************
144      ** Asignacion de parametros a la estructura de datos **
145      *************************************************/
146
147     //mensaje
148     strcpy(mensaje->msj_cifrado, msj);
149     n=strlen(msj);
150     mensaje->msj_cifrado[n]='\0';
151
152     //presentacion de datos por pantalla
153     printf("\n\nMensaje cifrado: %s\n", msj);
154
155     return(0);
156
157  }
```

criptografía aplicada

Ejemplo 1: Cifrado simétrico (XIII)
Descifrado (I)

criptografía aplicada

```
162  int descifra(datos mensaje){
163
164    int n, status;
165    char password[MAXPSWD];
166    char msj[MAXMSJ];
167    //Estructuras para trabajo con cryptlib
168    CRYPT_CONTEXT masterContext;
169    CRYPT_CONTEXT sessionContext;
170
171
172    /********** valores de entrada **********/
173    printf("\n\n\n----------------------------------------");
174    printf("\n\nFUNCION DE DESCIFRADO\n");
175
176    //Contraseña
177    printf("\nIntroduzca su contraseña secreta:\t");
178    scanf("%s", password);
179    n=strlen(password);
180    password[n]=0;
181
182
183    /**** Asignacion de parametros de la estructura de datos ****/
184    strcpy(msj, mensaje.msj_cifrado);
```

Ejemplo 1: Cifrado simétrico (XIV)
Descifrado (II)

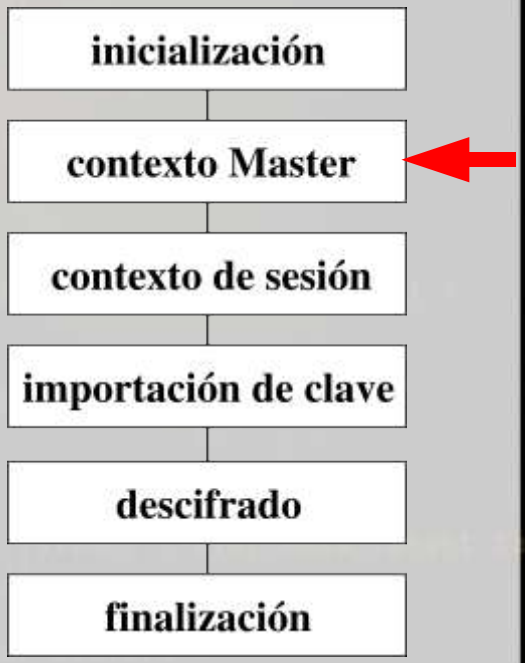
```
191     /******** Inicio del descifrado *********/
192     cryptInit();
```

criptografía aplicada

Ejemplo 1: Cifrado simétrico (XV)

Descifrado (III)

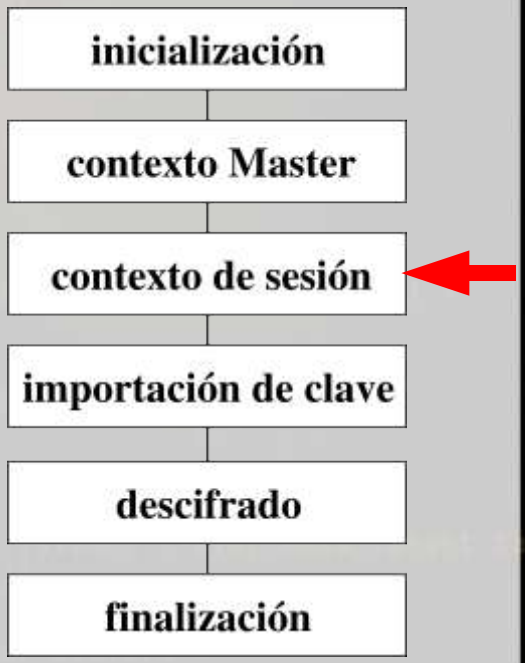
```
195    /*********** Contexto Master ************/
196
197    //Creacion del contexto
198    //en este caso DES para
        descifrar la clave de sesion
199    cryptCreateContext(&masterContext,
        CRYPT_UNUSED, CRYPT_ALGO_DES);
200
201    //Derivacion de claves

202    //se introduce la semilla
203    cryptSetAttributeString(masterContext,
        CRYPT_CTXINFO_KEYING_SALT, SEMILLA, 8);
204
205    //se introduce la contraseña
206    cryptSetAttributeString(masterContext,
        CRYPT_CTXINFO_KEYING_VALUE, password,
        strlen(password));
207
208    //se introduce el vector de inicializacion si es necesario.
209    //el modo por defecto (CBC) lo necesita
210    cryptSetAttributeString(masterContext, CRYPT_CTXINFO_IV, IV, 8);
```



criptografía aplicada

Ejemplo 1: Cifrado simétrico (XVI)
Descifrado (IV)

```
213     /********** Contexto de sesion **********/
214
215     //Generacion del contexto
216     cryptCreateContext(&sessionContext,
         CRYPT_UNUSED, ALGORITMO);
217
218     //Seleccion del modo de operacion
219     cryptSetAttribute(sessionContext,
         CRYPT_CTXINFO_MODE, MODO);
220
221     //vector de inicializacion si es necesario
         (en todos los metodos menos ECB)
222     cryptSetAttributeString(sessionContext,
         CRYPT_CTXINFO_IV, IV, 8);
```

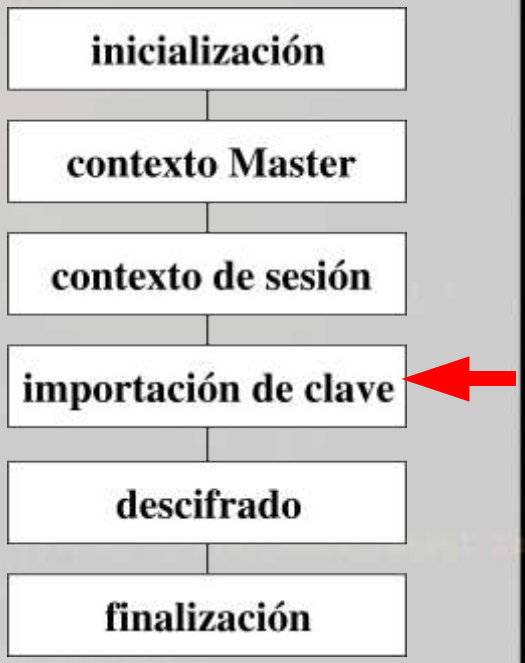


criptografía aplicada

Ejemplo 1: Cifrado simétrico (XVII)

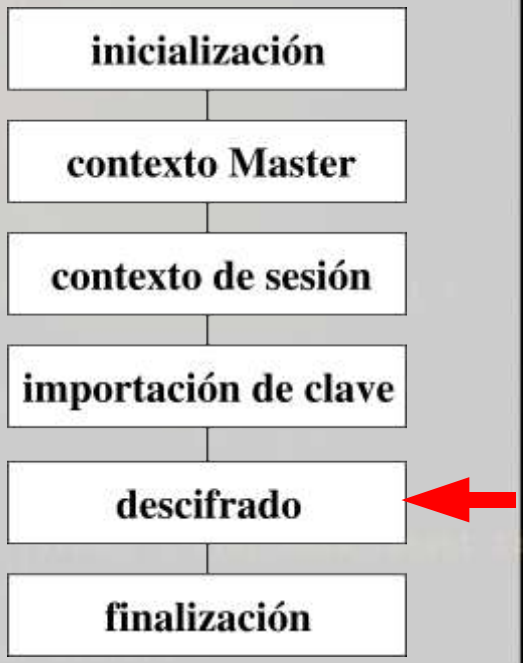
Descifrado (V)

```
225     /***** Importacion de clave *****/
226     cryptImportKey(mensaje.clave_sesion,
         masterContext, sessionContext);
```



Ejemplo 1: Cifrado simétrico (XVIII)
Descifrado (VI)

```
229     /******* Descifrado de los datos *******/
230
231     //Descifra el contenido de msj
          con la clave y metodos asociados
          al contexto de sesion
232     //y lo vuelve a guardar en msj
233     n=strlen(msj);
234
235     cryptDecrypt(sessionContext, msj, n);
```



criptografía aplicada

Ejemplo 1: Cifrado simétrico (y XIX)
Descifrado (y VII)

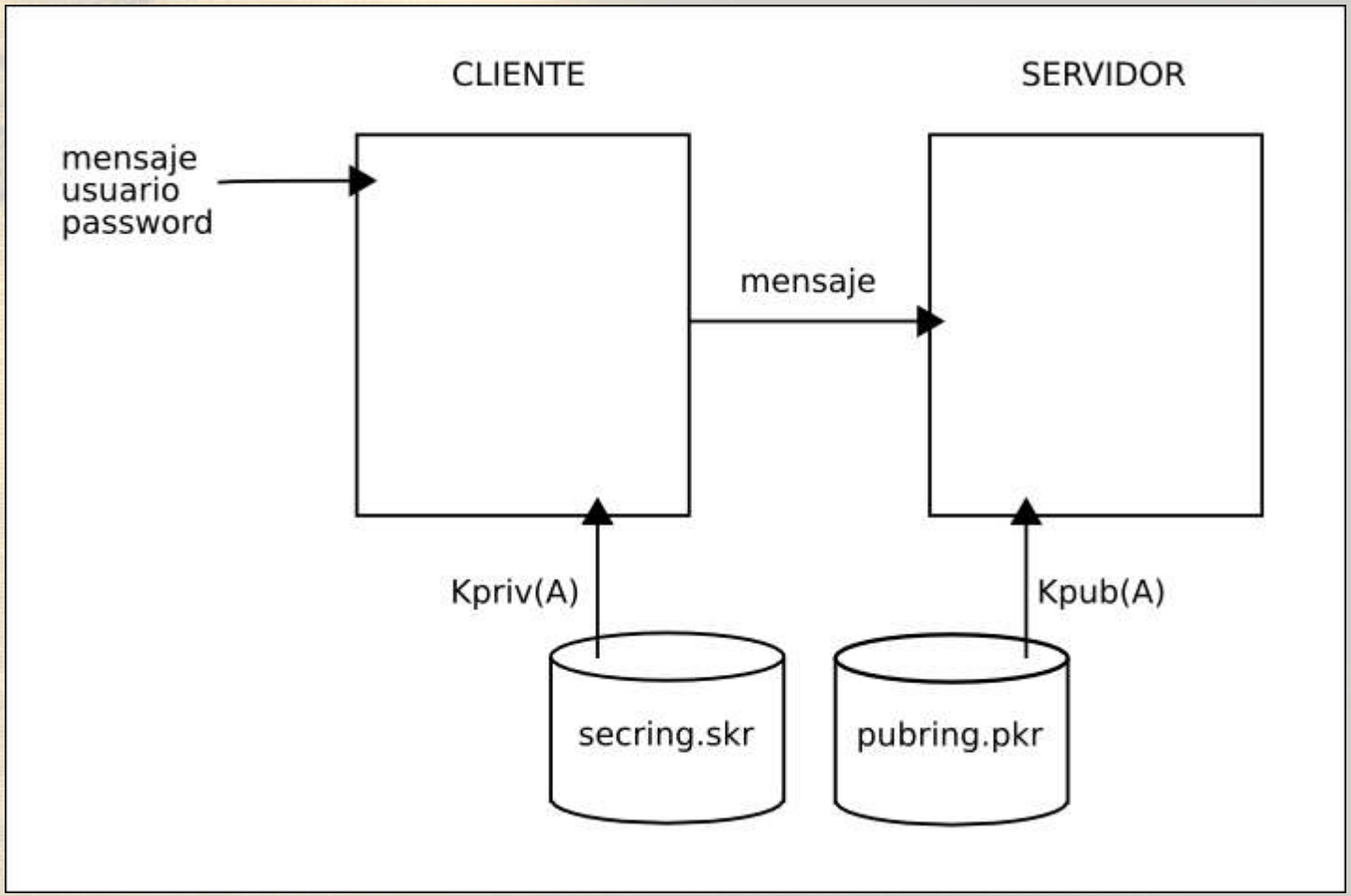
```
238     /******** Destruccion de contextos ********/
239
240     cryptDestroyContext(sessionContext);
241
242     cryptDestroyContext(masterContext);
243
244
245     /******** Fin del descifrado *************/
246
247     cryptEnd();
```


```
249     //presentacion de datos por pantalla
250     printf("\n\nMensaje descifrado: %s\n\n", msj);
251
252     return(0);
253   }
```

criptografía aplicada

Ejemplo 2: Firma digital (I)

- Firma con clave privada del emisor un resumen hash del mensaje
- Algoritmos: SHA-1 (Hash), RSA (Firma)



criptografía aplicada

Ejemplo 2: Firma digital (II)

Definiciones

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3  #include <string.h>
 4  //biblioteca para cryptlib
 5  #include "cryptlib.h"
 6
 7  #define MAXNAME 20
 8  #define MAXMSJ 2048
 9  #define MAXPSWD 10
10  #define MAXSIGN 1024
11  #define KPRIV "secring.skr"
12  #define KPUB "pubring.pkr"
13
14  struct datos{
15    char nombre[MAXNAME];
16    char msj[MAXMSJ];
17    char firma[MAXSIGN];
18  };
19  typedef struct datos datos;
20
21  //definicion de funciones para firma y comprobacion
22  int cifra(datos *mensaje);
23  int descifra(datos mensaje);
```

Ejemplo 2: Firma digital (III)
Función principal

```
27  int main(int argc, char *argv[]){
28
29    datos mensaje;
30
31    system("clear");
32
33    printf("Aplicacion para firmar digitalmente un mensaje\n");
34    printf("-----------------------------------------------\n");
35
36    cifra(&mensaje);
37
38    descifra(mensaje);
39
40    exit(0);
41  }
```

criptografía aplicada

criptografía aplicada

Ejemplo 2: Firma digital (IV)
Firma de los datos (I)

```
45  int cifra(datos *mensaje){
46
47    int  n, status;
48    char password[MAXPSWD];
49    //Variables para trabajo con cryptlib
50    CRYPT_CONTEXT shaContext, privateKeyContext;
51    CRYPT_KEYSET cryptKeyset;
52    int signcliLength;
53
54    printf("\n\nFUNCION PARA FIRMADO\n\n");
55
56    /*********** datos de entrada *************/
57
58    //mensaje
59    printf("\nIntroduzca el mensaje a firmar:\n\n");
60    scanf("%[^\n]", mensaje->msj);
61    n=strlen(mensaje->msj);
62    mensaje->msj[n]='\0';
63
```

Ejemplo 2: Firma digital (V)

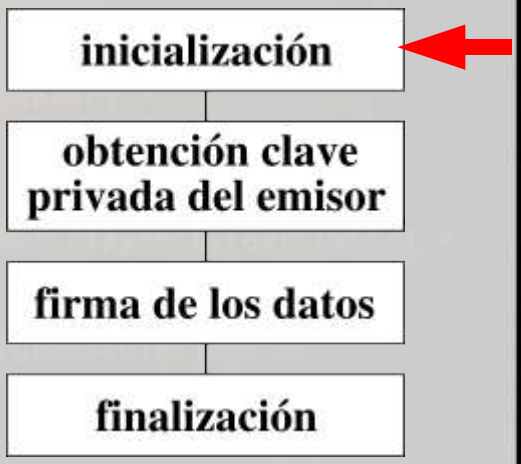
Firma de los datos (II)

```
64    //nombre del cliente
65    printf("\nIntroduzca su nombre:\t");
66    getchar();
67    fflush(stdin);
68    scanf("%[^\n]", mensaje->nombre);
69    n=strlen(mensaje->nombre);
70    mensaje->nombre[n]=0;
71
72    //contraseña
73    printf("Introduzca su contraseña:\t");
74    scanf("%s", password);
75    n=strlen(password);
76    password[n]=0;
```

criptografía aplicada

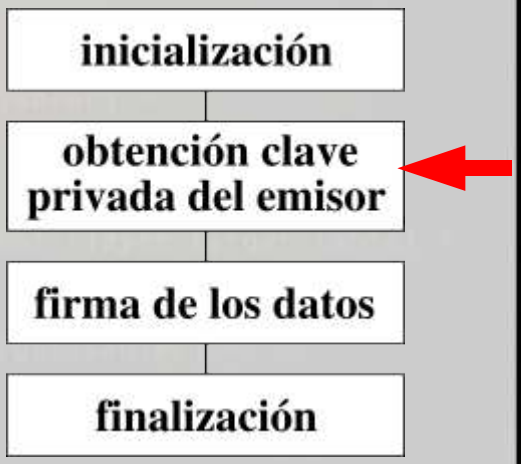
Ejemplo 2: Firma digital (VI)
Firma de los datos (III)

```
83    /***** Inicio de uso de cryptlib ******/
84    cryptInit();
```



Ejemplo 2: Firma digital (VII)
Firma de los datos (IV)

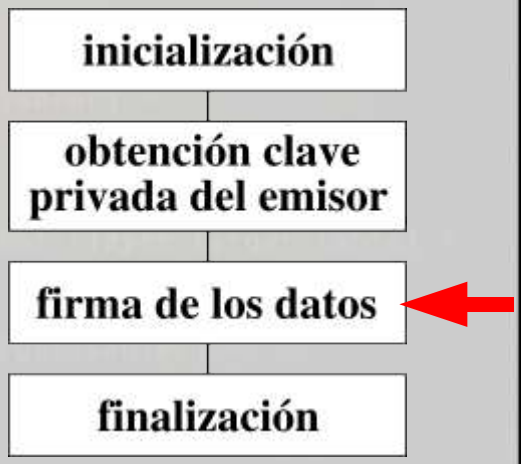
```
87    /***** Obtención de la clave privada *****/
88
89    //Se crea un keyset: contenedor abstracto
       para una o más claves
90    cryptKeysetOpen(&cryptKeyset, CRYPT_UNUSED,
       CRYPT_KEYSET_FILE, KPRIV,
       CRYPT_KEYOPT_READONLY);
91
92    /*** cryptKeySet: keyset que creamos
93         CRIPT_UNUSED: usuario propietario del
               keyset: por defecto
94         CRYPT_KEYSET_FILE: la clave está en
               un archivo plano en el disco duro
95         CRYPT_KEYOPT_READONLY: el acceso es de sólo lectura ***/
96
97    //Se extrae la clave del archivo
98    cryptGetPrivateKey(cryptKeyset, &privateKeyContext,
       CRYPT_KEYID_NAME, mensaje->nombre, password);
99
100   /*** privateKeyContext: contexto de cifrado que devuelve la función
101        CRYPT_KEYID_NAME: el propietario de la clave se
               especifica con el nombre (mensaje->nombre) ***/
```



criptografía aplicada

Ejemplo 2: Firma digital (VIII)
Firma de los datos (V)

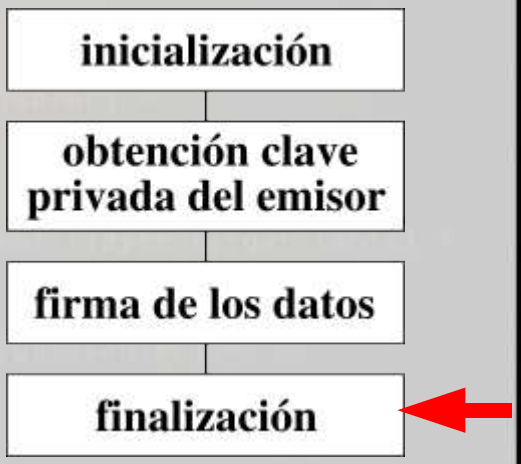
```
104     /******* Firma de los datos *********/
105
106     //Se crea un contexto para utilizar hash SHA:
         shaContext, con algoritmo SHA
107     cryptCreateContext(&shaContext, CRYPT_UNUSED,
         CRYPT_ALGO_SHA);
108
109     //Se crea el resumen hash del mensaje
         y se guarda en el contexto creado
110     cryptEncrypt(shaContext, mensaje->msj,
         strlen(mensaje->msj));
111
112     cryptEncrypt(shaContext, mensaje->msj, 0);
113
114     //Se firma el resumen hash del mensaje
         utilizando el algoritmo SHA en shaCOntext
115     //y la clave privada en privateKeyContext.
116     cryptCreateSignature(mensaje->firma, &signcliLength,
         privateKeyContext, shaContext);
```



criptografía aplicada

Ejemplo 2: Firma digital (IX)
Firma de los datos (y VI)

```
119     /*** Destruccion de contextos ***/
120     cryptDestroyContext(shaContext);
121
122     //Se destruye el keyset utilizado
         para la clave privada
123     cryptKeysetClose(cryptKeyset);
124
125
126     /***** Fin del uso de cryptlib *****/
127     cryptEnd();
```




```
130     //presentacion de resultado por pantalla
131     printf("\nmensaje firmado: %s | %s | %s\n\n",
         mensaje->nombre, mensaje->msj, mensaje->firma);
132
133     return(0);
134   }
```

criptografía aplicada

criptografía aplicada

Ejemplo 2: Firma digital (X)
Comprobación de firma (I)

```
138  int descifra(datos mensaje){
139
140    int status;
141    char modificar='a';
142    enum autent{SI, NO} autent;
143    //Variables para trabajo con cryptlib
144    CRYPT_CONTEXT shaContext;
145    CRYPT_KEYSET cryptKeyset;
146    CRYPT_HANDLE publicKey;
147
148    printf("\n\n\n----------------------------------------------");
149    printf("\n\nFUNCION PARA COMPROBACION DE FIRMA\n\n");
```

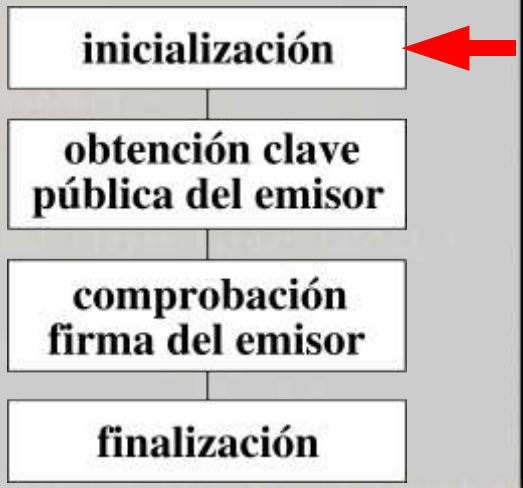
Ejemplo 2: Firma digital (XI)
Comprobación de firma (II)

criptografía aplicada

```
152    /***** Opcion para modificar los datos y comprobar que
        funciona el programa *****/
153    modificar='a';
154    while(modificar!='n' && modificar!='s'){
155      printf("\nComprobacion de funcionamiento: Quiere modificar
           los datos recibidos (s/n)?: ");
156      getchar();
157      fflush(stdin);
158      scanf("%[^\n]", &modificar);
159    }
160    if(modificar=='s')
161      mensaje.msj[0]=(char)((int)mensaje.msj[0]+1);
       //cambia el primer caracter por el siguiente en el alfabeto
```

Ejemplo 2: Firma digital (XII)
Comprobación de firma (III)

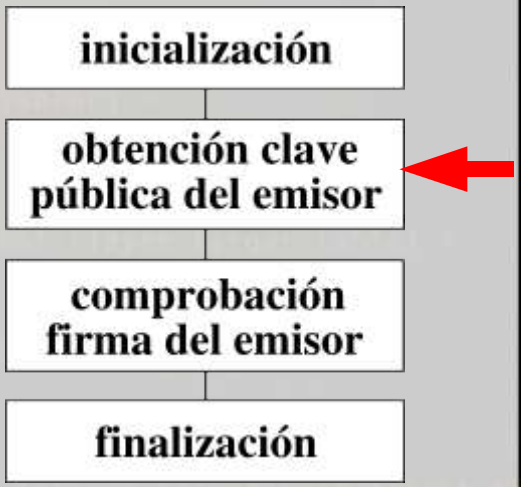
```
168     /*************** Inicio ************/
169     cryptInit();
```



criptografía aplicada

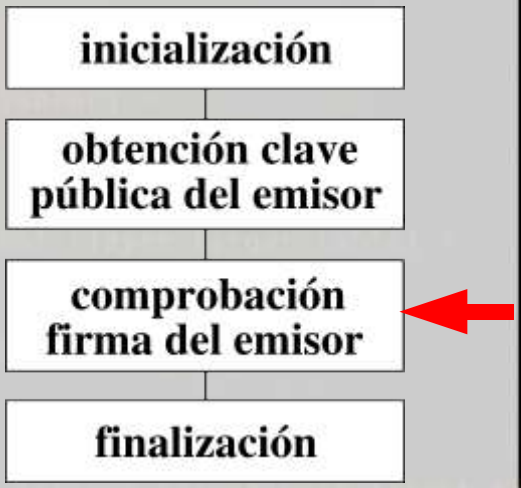
Ejemplo 2: Firma digital (XIII)
Comprobación de firma (IV)

```
172    /*** Obtencion de la clave publica del emisor ***/
173
174    //Se crea un keyset: contenedor abstracto
        para una o más claves
175    cryptKeysetOpen(&cryptKeyset, CRYPT_UNUSED,
        CRYPT_KEYSET_FILE, KPUB,
        CRYPT_KEYOPT_READONLY);
176
177    /*** cryptKeySet: keyset que creamos
178         CRIPT_UNUSED: usuario propietario
                del keyset: por defecto
179         CRYPT_KEYSET_FILE: la clave está
                en un archivo plano en el disco duro
180         CRYPT_KEYOPT_READONLY: el acceso
                es de sólo lectura ***/
181
182    //Se extrae la clave del archivo
183    cryptGetPublicKey(cryptKeyset, &publicKey,
        CRYPT_KEYID_NAME, mensaje.nombre);
184
185    /*** publicKey: contexto en el que se guarda la clave
186         CRYPT_KEYID_NAME: el propietario de la clave se especifica
                con el nombre (mensaje.nombre) ***/
```



Ejemplo 2: Firma digital (XIV)
Comprobación de firma (V)

```
189    /**** Comprobación de la firma del emisor ****/
190
191    //Se crea un contexto para la firma
192    cryptCreateContext(&shaContext, CRYPT_UNUSED,
        CRYPT_ALGO_SHA);
193
194    //Se crea el resumen hash del mensaje
        y se guarda en el contexto creado
195    cryptEncrypt(shaContext, mensaje.msj,
        strlen(mensaje.msj));
196
197    cryptEncrypt(shaContext, mensaje.msj, 0);
198
199    //Se comprueba la firma del emisor
200    status=cryptCheckSignature(mensaje.firma, publicKey, shaContext);
201    if(cryptStatusOK(status)){
202      autent=SI;
203    }
204    else
205      autent=NO;
```



criptografía aplicada

Ejemplo 2: Firma digital (y XV)
Comprobación de firma (y VI)

```
208     /***** Destruccion de contextos ******/
209     cryptDestroyContext(shaContext);
210
211     //Se destruye el keyset de la clave pública
212     cryptKeysetClose(cryptKeyset);
213
214     /**** Fin del uso de cryptlib ****/
215     cryptEnd();
218     /**** Visualización de resultados *****/
219
220     //visualiza el mensaje recibido
221     printf("\n\nRemitente: %s\n", mensaje.nombre);
222     printf("\nMensaje recibido: %s\n", mensaje.msj);
223
224     if(autent==SI)
225       printf("\nComprobacion de firma correcta\n\n\n");
226     else
227       printf("\nComprobacion de firma incorrecta\n\n\n");
228
229     return(0);
230   }
```

criptografía aplicada

criptografía aplicada

Parte III
Programación criptográfica con Mono

- ¿Qué es Mono?
- API criptográfica de Mono
- Ejemplo 1: Cifrado simétrico (DES)
- Ejemplo 2: Funciones HASH (SHA512)
- Ejemplo 3: Firma digital (RSA+SHA1)

criptografía aplicada

¿Qué es Mono?

- Mono es una plataforma
- Implementa el estándar del ECMA, .Net
- Implementación OpenSource de .Net
- CLI: Common Intermediate Language
- Objetos reutilizables por todos los lenguajes
- Muchos lenguajes: VB# (monobas), C#,...
- Usaremos en estos ejemplos C#

criptografía aplicada

¿Qué es Mono? (II)

- C#
  - Lenguaje “abanderado” de .Net/Mono
  - Orientado a objetos
  - Sintaxis similar a C/C++
  - Parecido en su concepción a Java
    - paquetes → namespaces
    - excepciones: try/catch
    - ...
    - Herencia, polimorfismo,..., no herencia múltiple, plantillas, boxing de tipos básicos, set y gets en lenguaje.

criptografía aplicada

¿Qué es Mono? (y III)

- Un conjunto grande de clases en su API
  - Programación gráfica: GTK#, WindowForms,...
  - Acceso a base de datos: ADO.NET
  - XML, HTTP, XML-RPC
  - Aplicaciones web: ASP.NET
  - Threading, Net, ...
  - Seguridad y ¡Criptografía!

criptografía aplicada

API Criptográfica de Mono

- *System.Security.Cryptography*
  - Criptografía general
- *System.Security.Cryptography.X509*
  - Gestión de certificados digitales X509
- *System.Security.Cryptography.Xml*
  - Firma con XML (XMLsig) y cifrado XML.

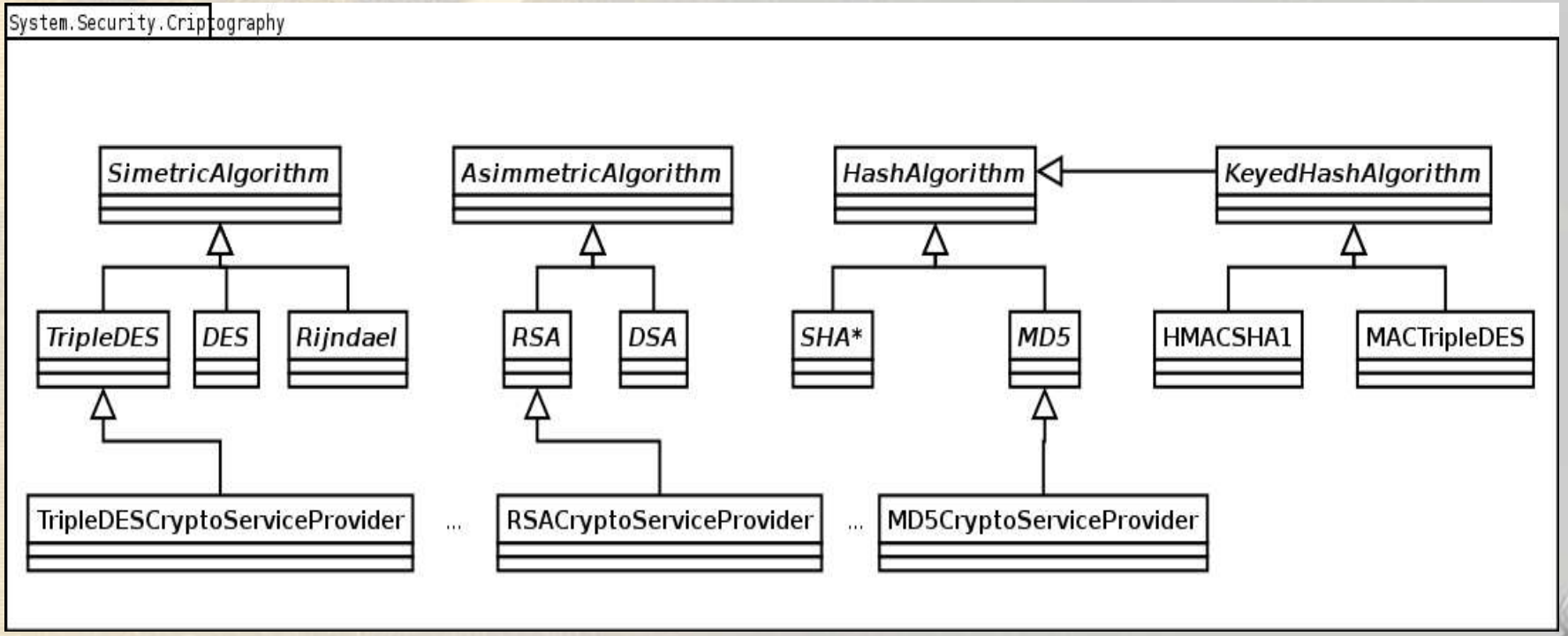
criptografía aplicada

API Criptográfica de Mono (II)

- System.Security.Cryptography
  - Componentes:
    - Clases abstractas de algoritmos de cifrado.
      - Definen las funciones que se han de implementar
      - De ellas se derivan implementaciones de los algoritmos
    - Clases “helper” de elementos de cifrado
      - Clases de ayuda generales para realizar cifrados y descifrados.
    - Clases de implementaciones concretas de algoritmos de cifrado.
      - Implementaciones que se extienden de las clases abstractas

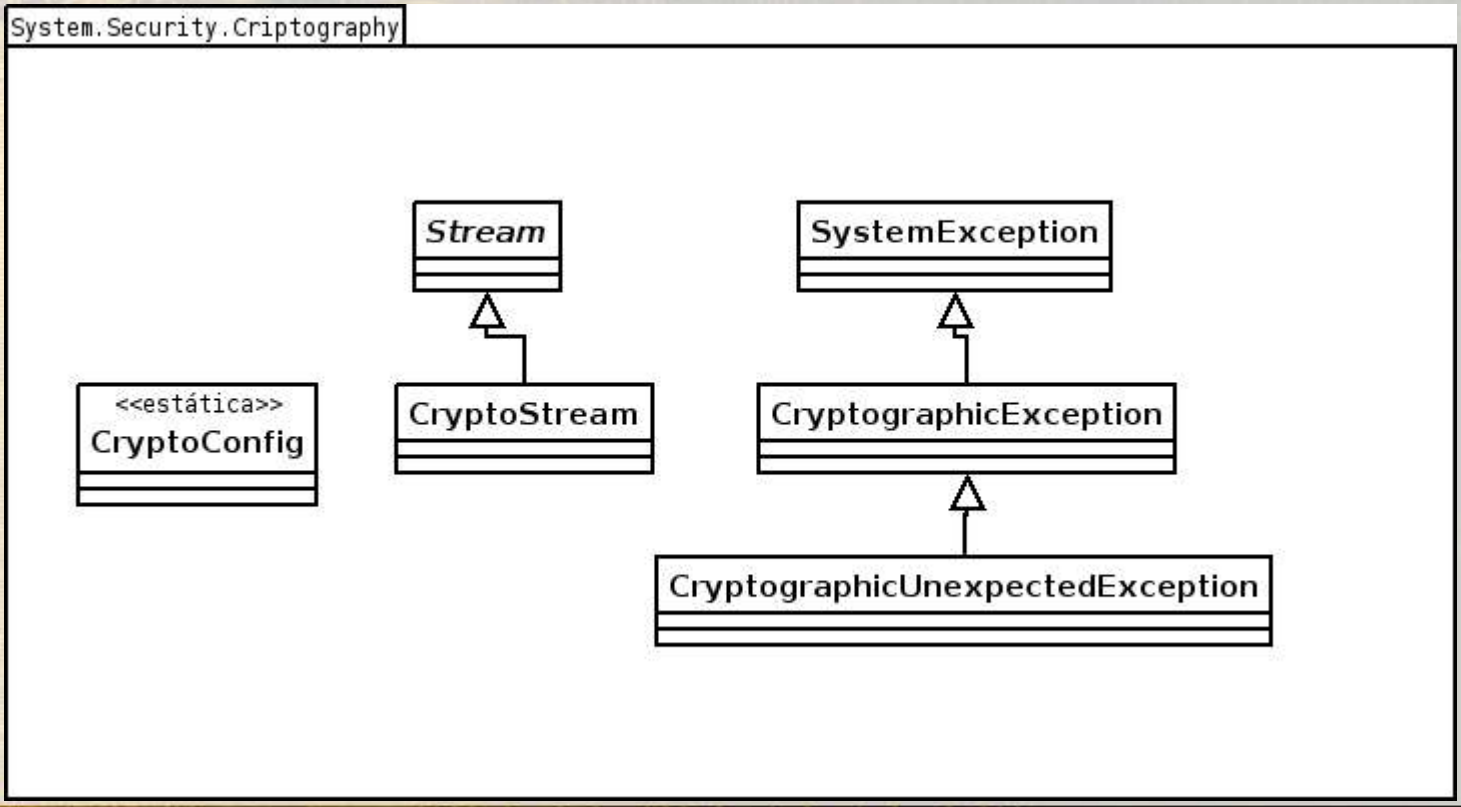
API Criptográfica de Mono (III)

- Esquema de clases del namespace *System.Security.Criptography*



API Criptográfica de Mono (IV)

- Esquema de clases del namespace *System.Security.Criptography*



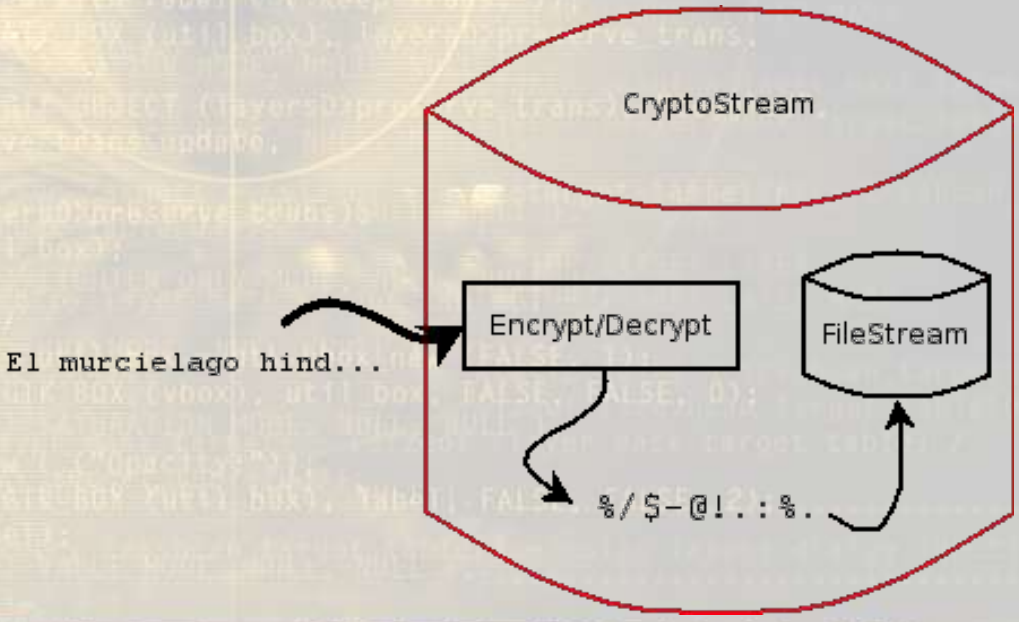
API Criptográfica de Mono (V)

- Otros elementos:
  - Enumerados:
    - CipherMode (CBC,ECB,OFB,CTS)
    - CryptoStreamMode (Read, Write)
    - PaddingMode (None,PKCS7,Zeros)
  - RandomNumberGenerator
    - Usado por muchas clases

criptografía aplicada

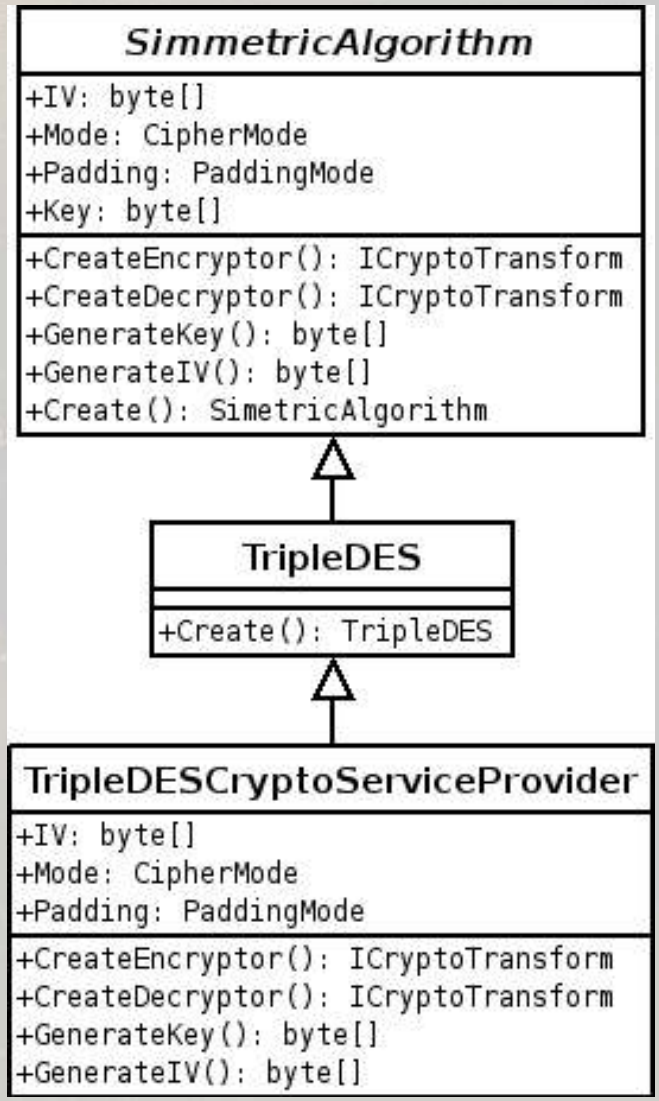
API Criptográfica en Mono (VI)

- Uso de cryptostream
  - Funcionamiento como filtros enganchando streams
  - Se asocia a una transformación que cifra o descifra, dada por el CryptoProvider



API Criptográfica en Mono (VII)

- Algoritmos Simétricos
  - DES,3DES...
- Clase abstracta
  - SimmetricAlgorythm
- Implementación
  - TripleDES
  - TripleDESCryptoProvider



criptografía aplicada

Ejemplo 1. Cifrado Triple DES

```
using System.Security.Cryptography;
using System.IO;
using System;
using System.Text;

public class Cifra3des
  {

  public static void Main(string[] Args)
    {

    // Obtenemos un provider 3DES

    TripleDES td = TripleDES.Create();

    // Tomamos el mensaje a cifrar

    System.Console.Write("Escriba su mensaje a cifrar:
");
    string mensaje=System.Console.ReadLine();
```

Ejemplo1. Cifrado TripleDES (II)

```
// Generamos una nueva llave y un vector de inicialización

td.GenerateKey();
td.GenerateIV();

// Convertimos la cadena a ristra de bytes

byte[] bytemen = Encoding.ASCII.GetBytes(mensaje);

// Abrimos un stream de memoria con escribiremos el cifrado

MemoryStream mem = new MemoryStream();

// Iniciamos un cryptostream que lo enlazamos al stream de memoria

CryptoStream cs = new CryptoStream(mem,td.CreateEncryptor(),
CryptoStreamMode.Write);
```

criptografía aplicada

Ejemplo 1. Cifrado TripleDES (III)

```
// Escribimos en el criptostream el mensaje convertido a bytes

  cs.Write(bytemen,0,bytemen.Length);
  cs.FlushFinalBlock();

  // Obtenemos el resultado del stream de memoria

  byte[] encrypted = mem.ToArray();

  mem.Seek(0,SeekOrigin.Begin);
  StreamReader reader1 = new StreamReader(mem);
  System.Console.WriteLine("Texto Cifrado: "+reader1.ReadToEnd());

  // Creamos un nuevo stream de memoria del que leeremos el descifrado

  MemoryStream mem2 = new MemoryStream();
```

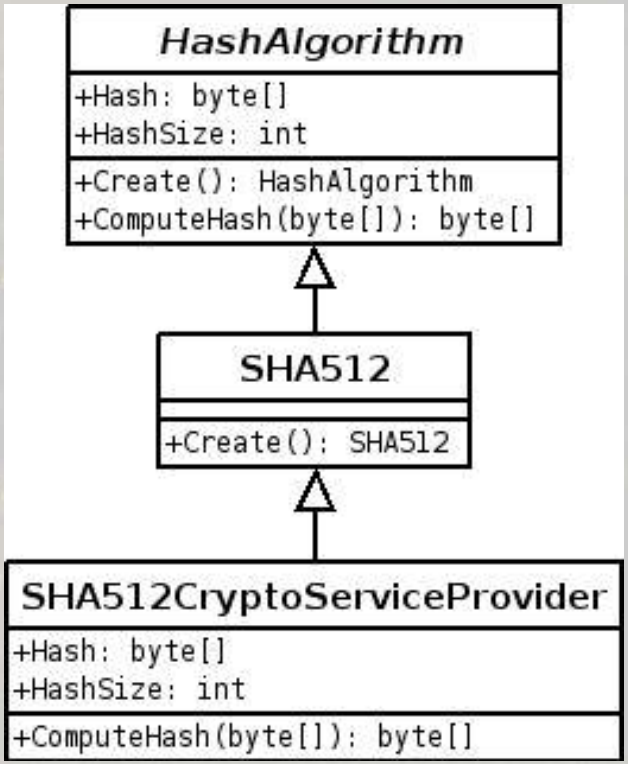
Ejemplo 1. Cifrado TripleDES (y IV)

```
    // Inciamos un cryptostream para descifrar

    CryptoStream cs2 = new CryptoStream(mem2,td.CreateDecryptor(),
CryptoStreamMode.Write);

    // Escribimos el texto cifrado y sale descifrado al stream
    // de memoria

    cs2.Write(encrypted,0,encrypted.Length);
    cs2.FlushFinalBlock();

    // Obtenemos el texto descifrado

    mem2.Seek(0,SeekOrigin.Begin);
    StreamReader reader = new StreamReader(mem2);
    System.Console.WriteLine("Texto Descifrado: "+reader.ReadToEnd());

    }
  }
```

criptografía aplicada

API Criptográfica de Mono (VIII)

- Algoritmos de HASH
  - MD5,SHAx,...
  - Keyed o no Keyed
- Clase abstracta
  - HashAlgorythm
- Implementación
  - SHA512CryptoProvider
  - MD5CryptoProvider



criptografía aplicada

Ejemplo 2. Funciones Hash

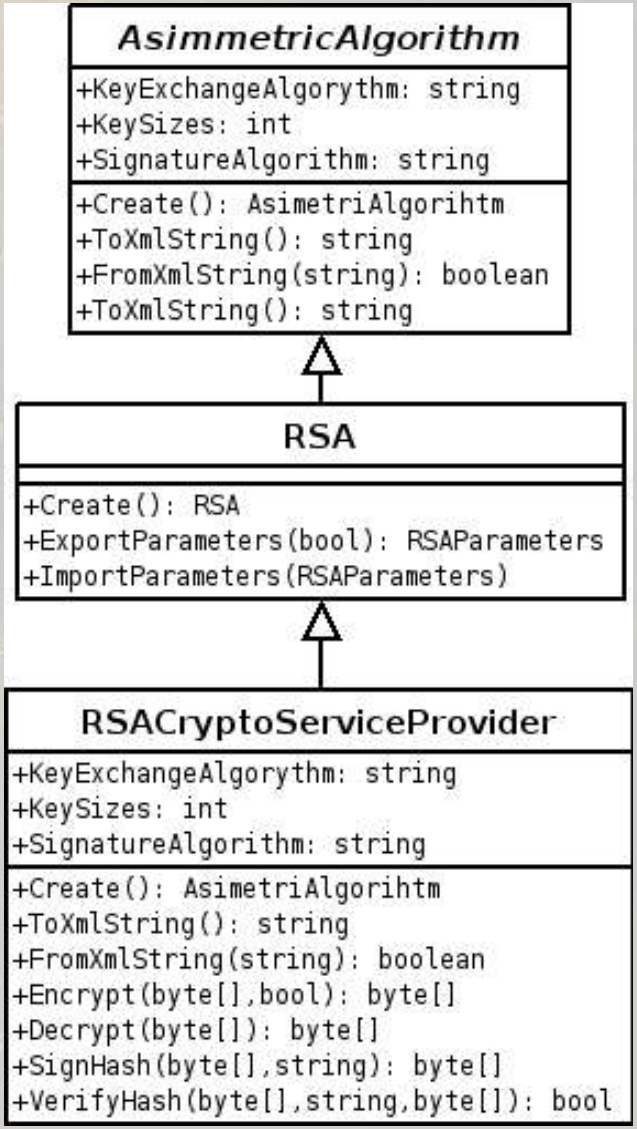
```
 1  using System;
 2  using System.Security.Cryptography;
 3  using System.IO;
 4  using System.Text;
 5
 6  public class HashExample
 7    {
 8
 9    public static void Main(string[] args)
10      {
11
12      if(args.Length==0)
13        {
14        System.Console.WriteLine("\n  Uso: ejemploSHA512.exe
              fichero\n\n");
15        System.Environment.Exit(0);
16
17        }
18
```

Ejemplo 2. Funciones Hash (y II)

```
19
20      StreamReader sr = new StreamReader(args[0]);
21
22      string mensaje = sr.ReadToEnd();
23
24      byte[] mensajeBytes = Encoding.ASCII.GetBytes(mensaje);
25
26      SHA512 sha = SHA512.Create();
27
28      byte[] hash = sha.ComputeHash(mensajeBytes);
29
30      System.Console.WriteLine("Hash:"+BitConverter.ToString(hash));
31
32      }
33
34
35  }
```

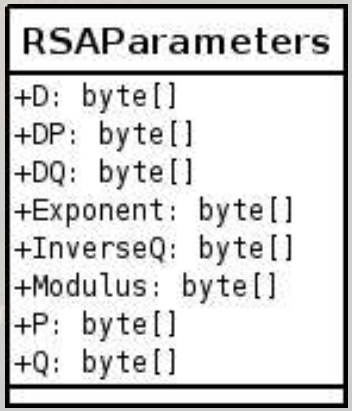
API Criptográfica de Mono (IX)

- Algoritmos asimétricos
  - RSA,DSA
- Clase abstracta
  - AssimetricAlgorithm
- Clases de implentación
  - DSACryptoProvider
  - RSACryptoProvider



API Criptográfica de Mono (y X)

- Parámetros RSA
  - Parámetros públicos:
    - D,Modulus,Exponent
  - Parámetros privados
    - P,Q,DP,DQ,InverseQ
      - **P**: Primo
      - **Q**: Primo
      - **dP**=(1/e)mod(p-1)
      - **dQ**=(1/e)mod(q-1)
      - **qINV**=(1/q)mod p donde p > q



criptografía aplicada

Ejemplo3. Firma Digital RSA+SHA1

```
 1  using System;
 2  using System.Security.Cryptography;
 3  using System.IO;
 4  using System.Text;
 5
 6  public class ejemploSHARSA
 7    {
 8
 9    public static void Main(string[] args)
10      {
11
12      if(args.Length==0)
13        {
14        System.Console.WriteLine("\n  Uso: ejemploSHA512.exe
             fichero\n\n");
15        System.Environment.Exit(0);
16
17        }
```

Ejemplo 3. Firma Digital (II)

```
19      // Leemos el fichero a firmar
20
21      StreamReader sr = new StreamReader(args[0]);
22      string mensaje = sr.ReadToEnd();
23      sr.Close();
24
25      byte[] mensajeBytes = Encoding.ASCII.GetBytes(mensaje);
26
27      // Creamos una clase SHA1
28
29      SHA1 sha = SHA1.Create();
30
31      // Realizamos el HASH del fichero en bytes
32
33      byte[] hash = sha.ComputeHash(mensajeBytes);
34
35      System.Console.WriteLine("Hash: "+BitConverter.ToString(hash));
36
```

criptografía aplicada

Ejemplo 3. Firma Digital (III)

```
37        // Generamos un Provider RSA
38
39        RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
40
41        // Es en el firmado cuando se generan el par de llaves.
          // Firmamos
42
43        byte[] signbytes = rsa.SignHash(hash,"1.3.14.3.2.26");
44
45        // Exportamos los parametros para usarlos en la verificación.
46        // Parte de estos parámetros son la clave pública que enviamos a
         // otro punto.
47
48        RSAParameters rsaparams = rsa.ExportParameters(false);
49
50        System.Console.WriteLine("Hash Firmado: \n\n"+
             BitConverter.ToString(signbytes));
51        System.Console.Write("\n(Cambie el fichero para prueba) Quiere
cambiar la firma? (S/N) ");
52
53        if(String.Compare(System.Console.ReadLine(),"S")==0)
54          signbytes = new byte[signbytes.Length];
```

criptografía aplicada

Ejemplo 3. Firma Digital (IV)

```
56      // Releemos el fichero por si queremos cambiar algo
57
58      StreamReader sr2 = new StreamReader(args[0]);
59      string mensaje2 = sr2.ReadToEnd();
60      sr2.Close();
61
62      // Realizamos de nuevo el hash
63
64      byte[] mensajeBytes2 = Encoding.ASCII.GetBytes(mensaje2);
65      SHA1 sha2 = SHA1.Create();
66      byte[] hash2 = sha2.ComputeHash(mensajeBytes2);
67
68      System.Console.WriteLine("\nHash 2: "+
          BitConverter.ToString(hash2));
69
70      // Creamos un provider RSA
71
72      RSACryptoServiceProvider rsa2 = new RSACryptoServiceProvider();
```

criptografía aplicada

Ejemplo III. Firma Digital (y V)

```
74      // Importamos los parámetros que extrajimos
      //anteriormente
75
76      rsa.ImportParameters(rsaparams);
77
78
79      if(rsa.VerifyHash(hash2,"1.3.14.3.2.26",signbytes))
80        {
81        System.Console.WriteLine("Firma Correcta!");
82        }else{
83        System.Console.WriteLine("Firma Incorrecta!");
84        }
85
86
87      }
88
89    }
```

Conclusiones

- Conocimiento mínimo de algoritmos para hacer programación criptográfica.
- Escoger el API que mejor conozcamos y se adapte a nuestras necesidades.
- Escoger el algoritmo que se adapte a nuestras necesidades (simétrico, asimétrico o hash).
- La programación criptográfica es sencilla y está al alcance de nuestra mano.

Bibliografía y referencias

- Parte teórica:
  - ***Fundamento de seguridad en redes. Aplicaciones y estándares***. 2º edición. William Stallings. PH. 2003.
  - ***Cryptography and network security. Principles and practice.*** 3ª edition. William Stallings. PH. 2003
  - ***Handbook of applied cryptography***: http://www.cacr.math.uwaterloo.ca/hac/
  - ***Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C*** . Bruce Schneir, 96
- Cryptlib:
  - ***Cryptlib website***: http://www.cs.auckland.ac.nz/~pgut001/cryptlib/
- API Criptográfica de Mono
  - ***Mono: A Developer's Notebook***. Edd Dumbill, Niel M. Bornstein. O Reilly. 2004
  - ***Monodoc***: http://www.go-mono.com/docs/
  - ***.NET Security and Cryptography.*** Peter Thorsteinson, G. Gnana Arun Ganesh. Prentice Hall. 2003