PHP e crittografia:

un'introduzione all'utilizzo delle librerie mcrypt e mhash

a cura di **Enrico Zimuel** <**enrico@enricozimuel.net**>

Di che cosa parliamo?

- Il Php e la crittografia
- Esempi d'utilizzo delle funzioni crypt() e md5()
- Uno script per l'autenticazione challenge-response tra client e server
- La libreria mcrypt
- Installazione ed utilizzo della libreria mcrypt
- Modalità di cifratura degli algoritmi simmetrici (block/stream cipher)
- Le modalità ECB, CBC, CFB, OFB
- La libreria mhash
- Installazione ed utilizzo della libreria mhash

Il PHP e la crittografia

- In PHP esistono diverse funzioni crittografiche, esse sono presenti nelle seguenti librerie:
  - String*: con le funzioni crypt(), md5(), md5_file(), sha1(), sha1_file();
  - Mcrypt: con l'implementazione di diversi algoritmi simmetrici di cifratura.
  - Mhash: con l'implementazione di diverse funzioni hash utili per la generazione di checksum, message digest, MAC.
  - Crack: per il test di validità delle password scelte dall'utente.
  - OpenSSL**: per l'utilizzo della crittografia a chiave pubblica.

* libreria standard di PHP

** versione ancora in fase di sviluppo (0.9.7c)

Il PHP e la crittografia

- In queste slide introdurremo solo alcuni esempi d'utilizzo delle librerie mcrypt ed mhash e delle funzioni standard String crypt(), md5() e sha1().

- Prima di parlare di queste funzioni crittografiche è opportuno definire alcuni termini chiave della crittografia che utilizzeremo nel prosieguo della presentazione.

Alcuni termini chiave della crittografia

- **Crittografia**

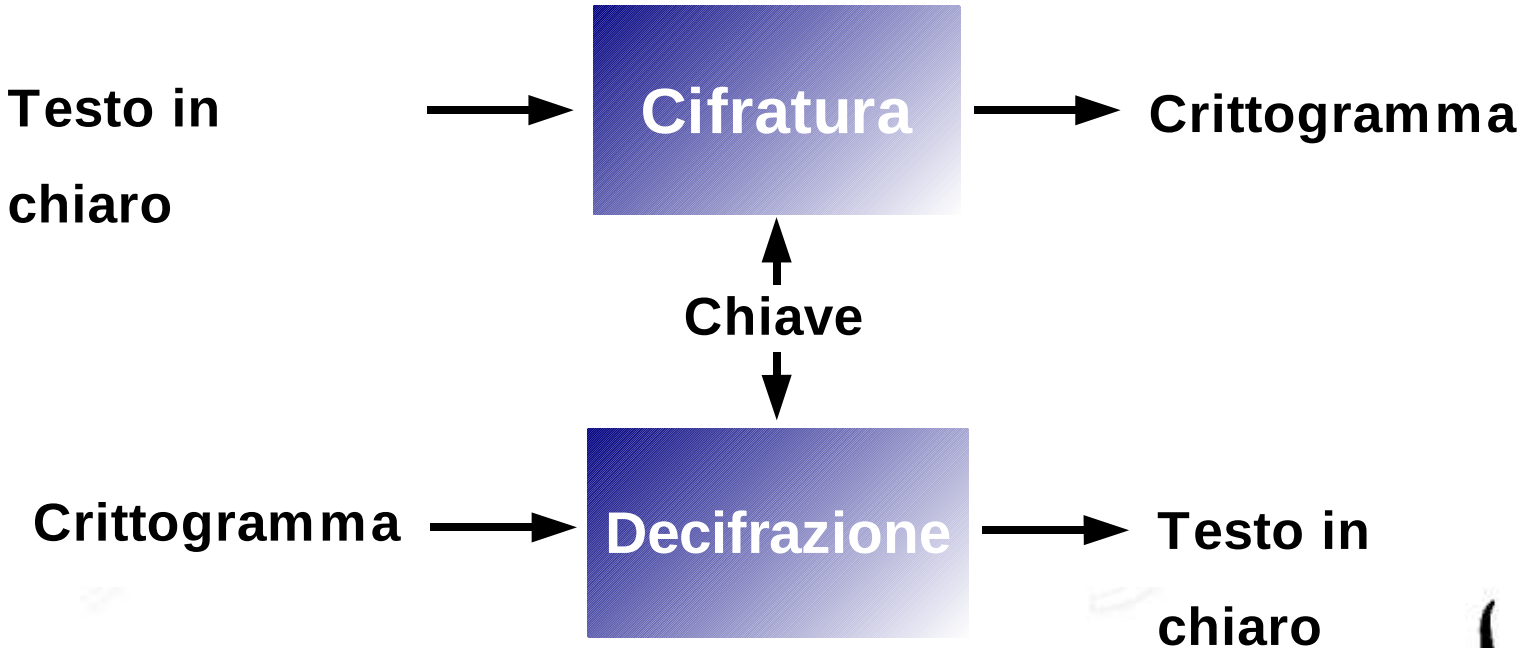
La *crittografia* (dal greco kryptos, nascosto, e graphien, scrivere) è la scienza che studia le tecniche di conversione di un testo leggibile, detto anche *in chiaro*, in un testo incomprensibile, detto anche *crittogramma*, e viceversa.

- **Cifratura e Decifratura**

L'operazione che consente di ottenere un crittogramma da un testo in chiaro viene definita *cifratura*, mentre l'operazione inversa che consente di ottenere il testo in chiaro da un crittogramma *decifratura* o *decifrazione*.

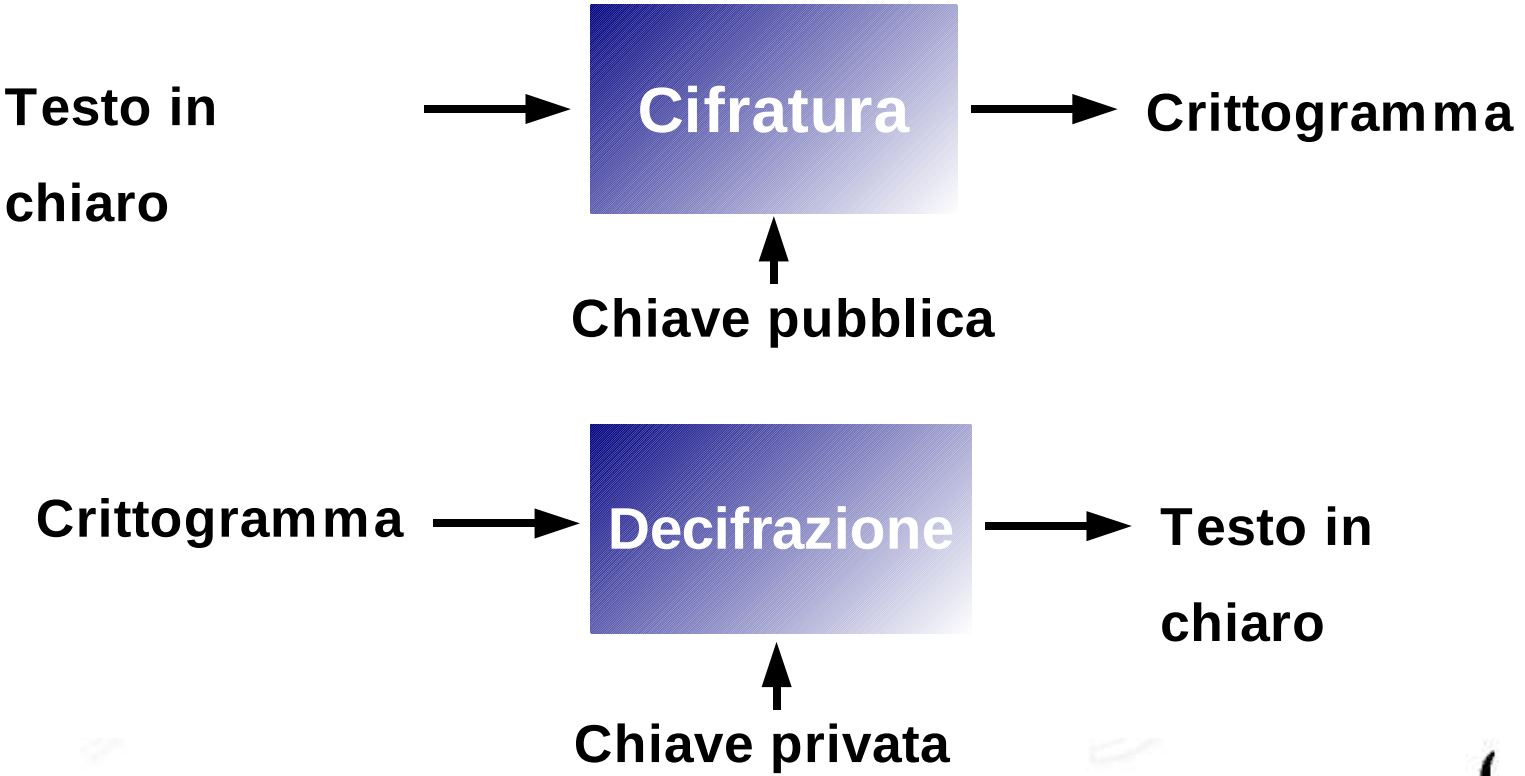
Alcuni termini chiave della crittografia

- La crittografia si divide essenzialmente in due grandi settori: la crittografia *simmetrica* e la crittografia *asimmetrica* o *a chiave pubblica*.
- Nella crittografia simmetrica le operazioni di cifratura e decifrazione avvengono tramite l'utilizzo di una *chiave segreta*.



Alcuni termini chiave della crittografia

- Nella crittografia asimmetrica o a chiave pubblica le operazioni di cifratura e decifrazione avvengono tramite l'utilizzo di due chiavi: una *pubblica* e l'altra *privata* o segreta.



Alcuni termini chiave della crittografia

- Per poter comprendere la differenza tra crittografia simmetrica ed asimmetrica è necessario porsi il *problema della trasmissione della chiave*.
- Ipotizziamo che due utenti, Enrico e Monica, vogliano comunicare su Internet in maniera sicura utilizzando un algoritmo di cifratura simmetrico.
- Sia Enrico che Monica devono essere in possesso della stessa informazione segreta, la chiave, ma dal momento che non abitano nella stessa città non possono incontrarsi di persona per scambiarsi quest'informazione, come fare?
- In pratica il problema è quello di trasmettere, in maniera sicura, un'informazione sensibile attraverso un canale insicuro di comunicazione, ad esempio Internet.

Alcuni termini chiave della crittografia

- Se si ha a disposizione un canale di comunicazione sicuro, utilizzabile ad esempio una tantum e solo per lo scambio di piccole informazioni, il problema può essere risolto inviando la chiave su tale canale informativo.
- Altrimenti l'unica soluzione è quella di adottare un algoritmo di cifratura asimmetrico.
- Con un cifrario asimmetrico Enrico può comunicare con Monica cifrando i messaggi con la chiave pubblica di Monica che è per definizione pubblica e quindi accessibile liberamente.
- Monica è l'unica persona in grado di decifrare il messaggio poichè solo lei è in possesso della sua chiave privata.
- Viceversa Monica può comunicare con Enrico cifrando i messaggi con la chiave pubblica di Enrico e solo Enrico li potrà decifrare utilizzando la sua chiave privata.

Alcuni termini chiave della crittografia

- Oltre a garantire la riservatezza delle informazioni tramite le operazioni di cifratura e decifrazione la crittografia consente anche di garantire l'autenticazione e l'integrità di un messaggio.

- Con l'autenticazione si garantisce la corretta provenienza di un messaggio e con l'integrità si garantisce che il messaggio non sia stato manomesso da un intruso.

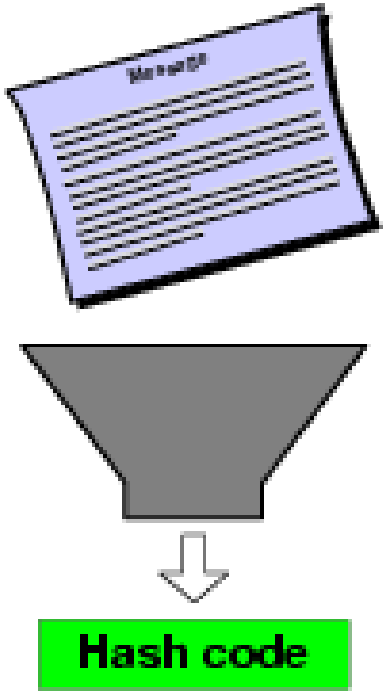
- Una delle funzioni crittografiche fondamentali per i processi di autenticazione e integrità è la funzione hash one-way.

- Una funzione hash one-way è una funzione matematica “difficilmente” invertibile che genera una sorta di “riassunto” del messaggio.

Alcuni termini chiave della crittografia

- Con il termine “difficilmente invertibile” si intende una funzione tale che il calcolo della sua inversa sia un'operazione computazionalmente difficile o meglio che non esista un algoritmo di complessità al più polinomiale che calcoli l'inversa della funzione per un dato valore.

- Mentre con il termine “riassunto” si indica la proprietà della funzione hash di essere legata in maniera univoca al valore dato in input. In pratica applicando la funzione hash su due messaggi differenti il risultato delle due funzioni deve essere differente.



Alcuni termini chiave della crittografia

- Una funzione hash accetta in ingresso un messaggio di lunghezza variabile m e produce in uscita un digest di messaggio H(m) di lunghezza fissa (ad esempio 160 bit).

- Questo digest (impronta, targa, riassunto) è strettamente legato al messaggio m, ogni messaggio m genera un H(m) univoco.

- Anche considerando due messaggi m ed m' differenti solo per 1 bit le loro funzioni hash H(m) ed H(m') saranno differenti, $H(m) \neq H(m')$.

- La probabilità di avere due messaggi x,y differenti $x \neq y$ tali che $H(x)=H(y)$ deve essere molto bassa (dell'ordine di 10^{-30}).

- Ai fini pratici H(m) deve essere facile da calcolare.

Le funzioni crittografiche standard del PHP

- Le funzioni crittografiche presenti nella distribuzione standard del PHP sono le funzioni stringa crypt(), md5() e sha1().

- Queste funzioni consentono di generare solo degli hash code e quindi non possono essere utilizzate per le operazioni di cifratura dei dati.

- Anche la funzione crypt(), a dispetto del nome, è una funzione hash one-way. E' stata chiamata in questo modo per motivi di compatibilità con la funzione crypt di Unix utilizzata per la memorizzazione sicura delle password degli utenti.

- Le altre due funzioni md5() ed sha1() implementano rispettivamente le funzioni hash MD5 ed SHA1.

La funzione hash stringa crypt()

- Il prototipo della funzione crypt() è:

string crypt (string str [, string salt])

dove str è la stringa in input della funzione hash e salt è un parametro stringa opzionale che viene concatenato con la stringa str per il calcolo dell'hash. In pratica questo parametro viene utilizzato per aumentare la sicurezza dell'hash contro attacchi basati su dizionari. Se questo parametro non è specificato il PHP genera automaticamente un salt casuale ad ogni esecuzione della funzione.

- La lunghezza del salt dipende dall'algoritmo utilizzato per il calcolo dell'hash che dipende a sua volta dal sistema operativo utilizzato.
Ad esempio su un sistema Red Hat 9 Linux con Kernel 2.4.20 il salt è di 12 caratteri poiché l'algoritmo utilizzato per l'hash è l'MD5.

La funzione hash stringa crypt()

- Esempio d'utilizzo della funzione crypt():

```
<?php
   $password= “test”;
   $digest= crypt($password);

   echo “Hash('“.$password.”')= “.$digest;
?>
```

- Ogni volta che si esegue questo script il valore di $digest cambia, ad esempio:

Hash('test')= 1dLygajpl$zJ3E92oyju7vAYZ7MuQb8.
Hash('test')= 1Spznq1WT$udgFuVOf8Eyg8Ca815ltX1
Hash('test')= $1$9oJmpneM$.c/f2UwF0zJ5GLMGMlCef0

I caratteri in rosso sono i salt generati automaticamente da Php. La stringa iniziale 1 identifica l'MD5 (costante CRYPT_MD5).

La funzione hash stringa crypt()

- Per eseguire il confronto con un valore hash memorizzato è necessario riapplicare lo stesso salt.
- Esempio di autenticazione della password tramite hash e valore inviato tramite GET http://server/esempio.php?password=test

```
<?php
   $password="test";
   $digest= crypt($password);
   $newdigest= crypt($_GET['password'],$digest);

   if ($newdigest==$digest) {
      echo "Utente autenticato! :-)";
   } else {
      echo "Utente non autenticato! :-(";
   }
?>
```

- Al posto del salt $digest si può anche sostituire substr($digest,0,12) oppure substr($digest,0,11).

Le funzioni hash stringa md5() e sha1()

- L'utilizzo delle funzioni md5() e sha1() è più immediato rispetto a crypt () poiché è sufficiente specificare solo la stringa di input della funzione hash senza il salt.
- Il prototipo delle due funzioni:

```
string md5  ( string str [, bool raw_output] )
string sha1 ( string str [, bool raw_output] )
```

la funzione md5() è disponibile a partire dalla versione 3 del Php mentre l'sha1() a partire dalla versione 4.3.0.

- La funzione md5() restituisce una stringa esadecimale di 32 caratteri. Se il parametro opzionale raw_output è impostato a TRUE la funzione restituisce una valore binario di lunghezza 16.
- La funzione sha1() restituisce una stringa esadecimale di 40 caratteri, con raw_output==TRUE un valore binario di 20 byte.

Le funzioni hash stringa md5() e sha1()

- Esempio di sistema di autenticazione challenge-response tra client e server con l'utilizzo della funzione hash md5() ed un database MySQL.

```
<?php
$host = "localhost"; $user = "test"; $pswd = ""; $db = "utenti";

// Imposto l'autorizzazione a false
$autorizzazione=0;

// Verifico che l'utente abbia introdotto la username e la password
if (isset($_SERVER['PHP_AUTH_USER']) && isset($_SERVER
['PHP_AUTH_PW'])) {

   mysql_pconnect($host, $user, $pswd) or die("Non riesco a
collegarmi al server MySQL!");
   mysql_select_db($db) or die("Non riesco a collegarmi al db!");

?>
```

Le funzioni hash stringa md5() e sha1()

```
<?php
     // Calcolo l'hash del valore inserito dall'utente
     $digest = md5($_SERVER['PHP_AUTH_PW']);

     $query = "SELECT username FROM utenti WHERE username = '" .
$_SERVER['PHP_AUTH_USER'] ." AND password = '$digest'";

    if (mysql_numrows(mysql_query($query)) == 1) {
       $autorizzazione=1;
    }
}

if ($autorizzazione==0) {
  header('WWW-Authenticate: Basic realm="Private"');
  header('HTTP/1.0 401 Unauthorized');
  print "Non sei autorizzato ad accedere al sito.";
  exit;
} else {
  print "Autorizzazione concessa!";
}
?>
```

Le funzioni hash stringa md5() e sha1()

- Ovviamente il database MySQL degli utenti deve essere preventivamente generato con il calcolo dell'MD5 delle password.

- Il vantaggio principale di una soluzione del genere consiste nella memorizzazione sicura del database delle password. In pratica se il database degli utenti viene violato le password rimangono comunque al sicuro proprio perché non è possibile ricavarne il valore in chiaro partendo dall'hash.

- Volendo migliorare lo script precedente non si dovrebbe inviare in chiaro la password di autenticazione tra il client ed il server. Questo può essere ottenuto tramite un collegamento SSL nel quale si cifra tutto il traffico HTTP tra client e server oppure attraverso l'utilizzo di una funzione hash client, ad esempio in javascript, che calcoli l'hash prima di inviare il valore tramite un submit in POST. Esiste anche un modulo di autenticazione per Apache (mod_auth_digest)...

Le funzioni hash stringa md5_file() e sha1_file()

- Esistono due funzioni per il calcolo dell'hash MD5 e SHA1 applicabili su interi file.
- Il prototipo di queste due funzioni è identico a quello dell'md5() e dell'sha1():

```
string md5_file  ( string filename [, bool raw_output] )
string sha1_file ( string filename [, bool raw_output] )
```

in questo caso il parametro filename indica il nome del file in input alla funzione hash.

- Queste due funzioni, disponibili a partire dalla versione 4.2.0 di PHP, sono molto utili perché consentono di risolvere il problema dell'integrità degli script PHP ed in generale dei file memorizzati su di un server.
- Calcolando l'hash di un sorgente Php è possibile stabilire, a run-time, se lo script è stato modificato o meno da un eventuale intruso...

La libreria mcrypt()

- La libreria mcrypt() non è presente nell'installazione di default di Php.

- Essa è solo un modulo d'interfaccia alla libreria GPL libmcrypt disponibile all'indirizzo mcrypt.hellug.gr.

- La libreria libmcrypt supporta i seguenti algoritmi di cifratura: BLOWFISH, TWOFISH, DES, TripleDES, 3-WAY, SAFER-sk64, SAFER-sk128, SAFER+, LOKI97, GOST, RC2, RC6, MARS, IDEA, RIJNDAEL-128 (AES), RIJNDAEL-192, RIJNDAEL-256, SERPENT, CAST-128 (conosciuto come CAST5), CAST-256, ARCFOUR e WAKE.

- Attualmente la versione stabile è la 2.5.7. E' disponibile anche una versione per sistemi Windows.

Installazione della libreria mcrypt() su Gnu/Linux

- Per installare la libreria libmcrypt-x.x.tar.gz è necessario compilare i sorgenti con la seguente procedura:

```
gunzip libmcrypt-x.x.tar.gz
tar -xvf libmcrypt-x.x.tar.gz
./configure --disable-posix-threads
make
make install
```

- Una volta installata la libreria è possibile ricompilare il PHP con la direttiva -with-mcrypt utilizzando la seguente procedura:

```
cd [dir PHP]
./configure -with-mcrypt=[dir libmcrypt]
make
make install
```

[dir PHP] e [dir libmcrypt] sono le directory di installazione del PHP e della libreria libmcrypt rispettivamente.

Funzioni della libreria mcrypt()

- La libreria mcrypt() del PHP ha a disposizione le seguenti funzioni:

- mcrypt_cbc -- Encrypt/decrypt data in CBC mode
- mcrypt_cfb -- Encrypt/decrypt data in CFB mode
- mcrypt_create_iv -- Create an initialization vector (IV) from a random source
- mcrypt_decrypt -- Decrypts crypttext with given parameters
- mcrypt_ecb -- Encrypt/decrypt data in ECB mode
- mcrypt_enc_get_algorithms_name -- Returns the name of the opened algorithm
- mcrypt_enc_get_block_size -- Returns the blocksize of the opened algorithm
- mcrypt_enc_get_iv_size -- Returns the size of the IV of the opened algorithm
- mcrypt_enc_get_key_size -- Returns the maximum supported keysize of the opened mode
- mcrypt_enc_get_modes_name -- Returns the name of the opened mode
- mcrypt_enc_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm
- mcrypt_enc_is_block_algorithm_mode -- Checks whether the encryption of the opened mode works on blocks
- mcrypt_enc_is_block_algorithm -- Checks whether the algorithm of the opened mode is a block algorithm
- mcrypt_enc_is_block_mode -- Checks whether the opened mode outputs blocks
- mcrypt_enc_self_test -- This function runs a self test on the opened module
- mcrypt_encrypt -- Encrypts plaintext with given parameters
- mcrypt_generic_deinit -- This function deinitializes an encryption module
- mcrypt_generic_end -- This function terminates encryption
- mcrypt_generic_init -- This function initializes all buffers needed for encryption

Funzioni della libreria mcrypt()

- La libreria mcrypt() del PHP ha a disposizione le seguenti funzioni:

- mcrypt_generic -- This function encrypts data
- mcrypt_get_block_size -- Get the block size of the specified cipher
- mcrypt_get_cipher_name -- Get the name of the specified cipher
- mcrypt_get_iv_size -- Returns the size of the IV belonging to a specific cipher/mode combination
- mcrypt_get_key_size -- Get the key size of the specified cipher
- mcrypt_list_algorithms -- Get an array of all supported ciphers
- mcrypt_list_modes -- Get an array of all supported modes
- mcrypt_module_close -- Close the mcrypt module
- mcrypt_module_get_algo_block_size -- Returns the blocksize of the specified algorithm
- mcrypt_module_get_algo_key_size -- Returns the maximum supported keysize of the opened mode
- mcrypt_module_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm
- mcrypt_module_is_block_algorithm_mode -- This function returns if the the specified module is a block algorithm or not
- mcrypt_module_is_block_algorithm -- This function checks whether the specified algorithm is a block algorithm
- mcrypt_module_is_block_mode -- This function returns if the the specified mode outputs blocks or not
- mcrypt_module_open -- Opens the module of the algorithm and the mode to be used
- mcrypt_module_self_test -- This function runs a self test on the specified module
- mcrypt_ofb -- Encrypt/decrypt data in OFB mode
  mdecrypt_generic -- Decrypt data

Funzioni della libreria mcrypt()

- Le funzioni principali della libreria mcrypt() sono le funzioni di cifratura e decifrazione mcrypt_encrypt() e mcrypt_decrypt():

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv])
string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv])

dove:
- *cipher* è il tipo di algoritmo, specificato tramite le costanti MCRYPT_ciphername (ad esempio MCRYPT_BLOWFISH);
- *key* è la chiave di cifratura, è consigliabile non utilizzare una stringa Ascii ma l'ouput della funzione mhash_keygen_s2k (vedremo come);
- *data* è la stringa contenente i dati da cifrare;
- mode è la modalità di cifratura dell'algoritmo utilizzato (ECB,CBC,CFB,OFB,NOFB,STREAM);
- *iv* è un parametro opzionale utilizzato sono nelle modalità di cifratura CBC, CFB, OFB ed in qualche algoritmo di STREAM.

Le modalità di cifratura simmetriche

- In crittografia gli algoritmi di cifratura simmetrici si dividono nelle due seguenti categorie: a *blocchi* (Block Cipher) e a *sequenza* (Stream Cipher).

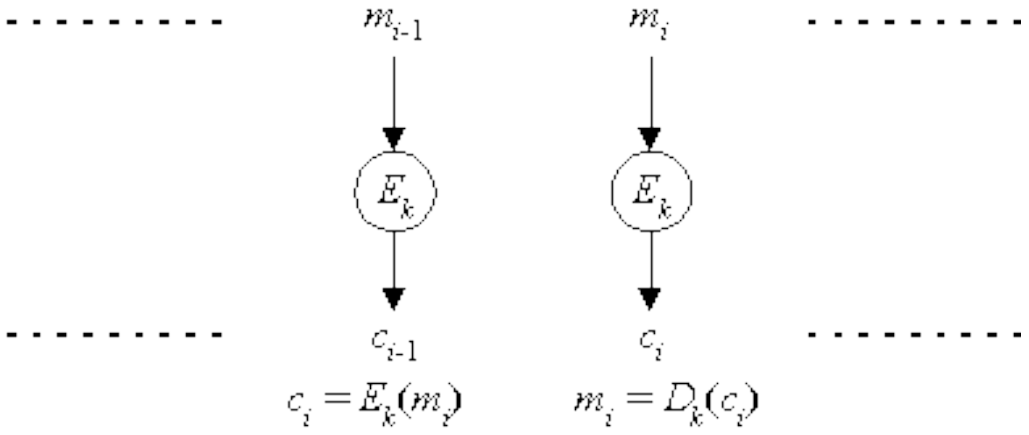
- Nei cifrari a *blocchi* il testo in chiaro viene suddiviso in blocchi di n bit (di solito 64) ed ogni blocco viene cifrato tramite l'algoritmo di cifratura. Nei cifrari a *sequenza* il testo in chiaro viene cifrato un bit o byte per volta come in una catena di montaggio.

- Con un cifrario a *blocchi*, tenendo costante la chiave k di cifratura, ad ogni blocco di testo viene sempre associato lo stesso crittogramma. Invece in un cifrario a *sequenza* il testo in chiaro viene cifrato ogni volta in modo differente.

- A loro volta sia cifrari a *blocchi* che quelli a *sequenza* si differenziano ulteriormente per le loro modalità operative.

Tipologie di cifrari a blocchi: ECB

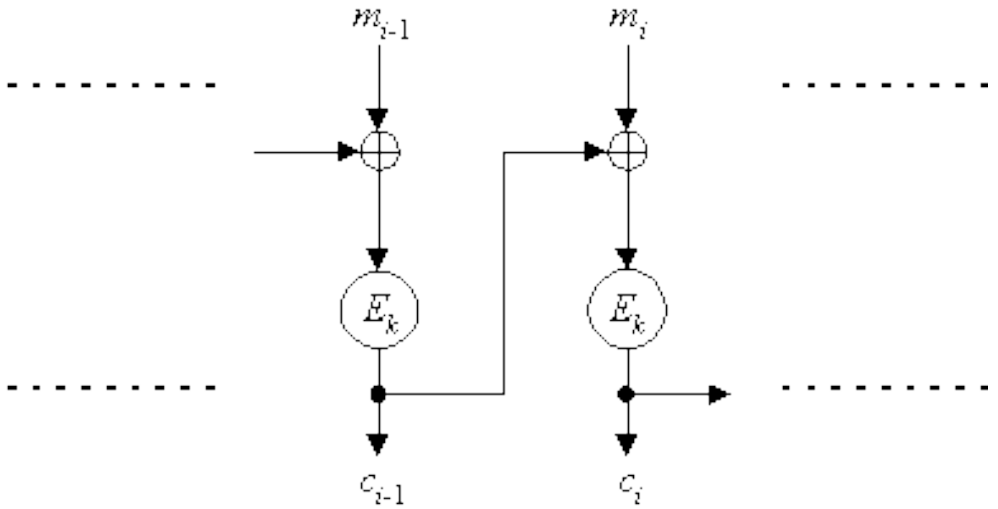
- I cifrari a blocchi possono operare nelle seguenti modalità: ECB (Electronic CodeBook), CBC (Cipher Block Chaining), CFB (Cipher FeedBack), OFB (Output FeedBack).
- La modalità ECB (Electronic CodeBook) è la più semplice ed intuitiva, infatti in essa ogni blocco viene cifrato con la stessa chiave k. Il crittogramma risulta quindi formato da una sequenza di blocchi cifrati ognuno allo stesso modo.



Electronic Code Book

Tipologie di cifrari a blocchi: CBC

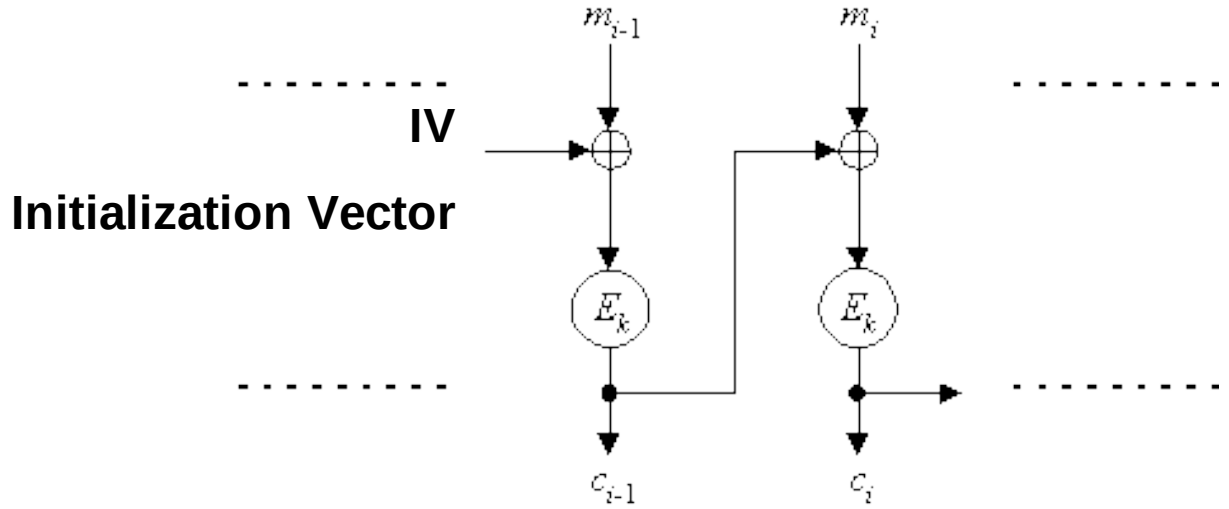
- Nella modalità CBC (Cipher Block Chaining) i blocchi vengono cifrati utilizzando il crittogramma del blocco precedente. Si esegue un'operazione di XOR tra il crittogramma precedente c_{i-1} ed il blocco in chiaro m_i ed il risultato viene cifrato con l'algoritmo E_k ottenendo il crittogramma c_i. In simboli: $c_i = E_k (m_i \oplus c_{i-1})$



Cipher Block Chaining

Tipologie di cifrari a blocchi: CBC

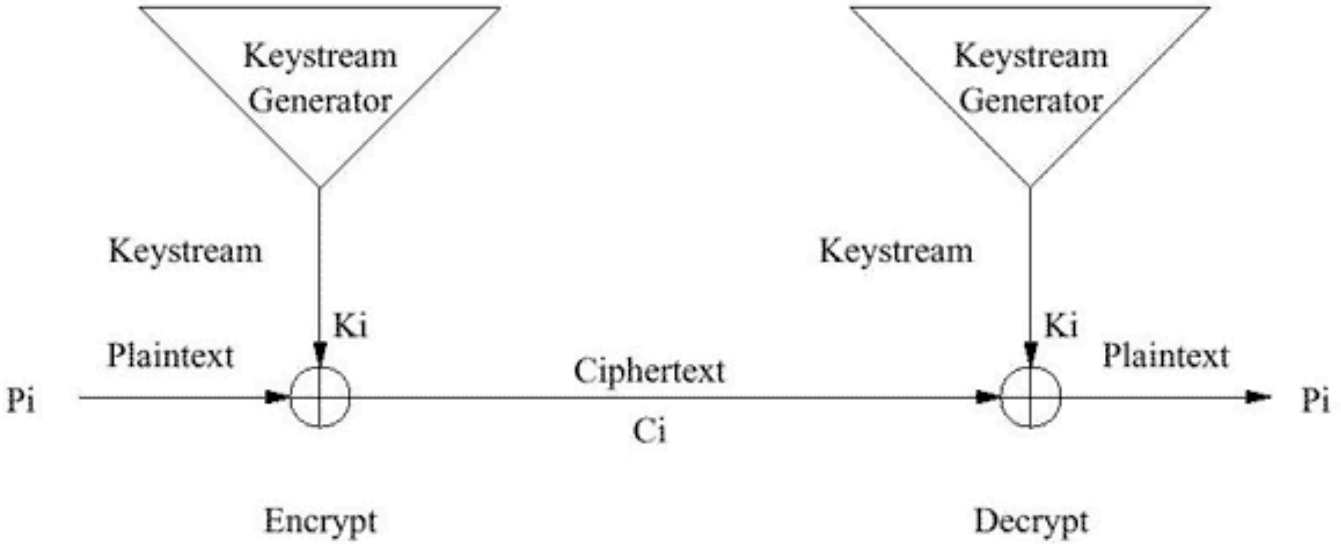
- Per migliorare la sicurezza dei cifrari a blocchi CBC si utilizza un *vettore di inizializzazione* (Initialization Vector, IV), il più delle volte generato casualmente, come input per il primo blocco. Tale vettore consente di variare il crittogramma a parità di testo in chiaro m e chiave k.
- Il vettore di inizializzazione può rimanere tranquillamente in chiaro e venire così trasmesso con il crittogramma.



Cipher Block Chaining

I cifrari a sequenza (stream cipher)

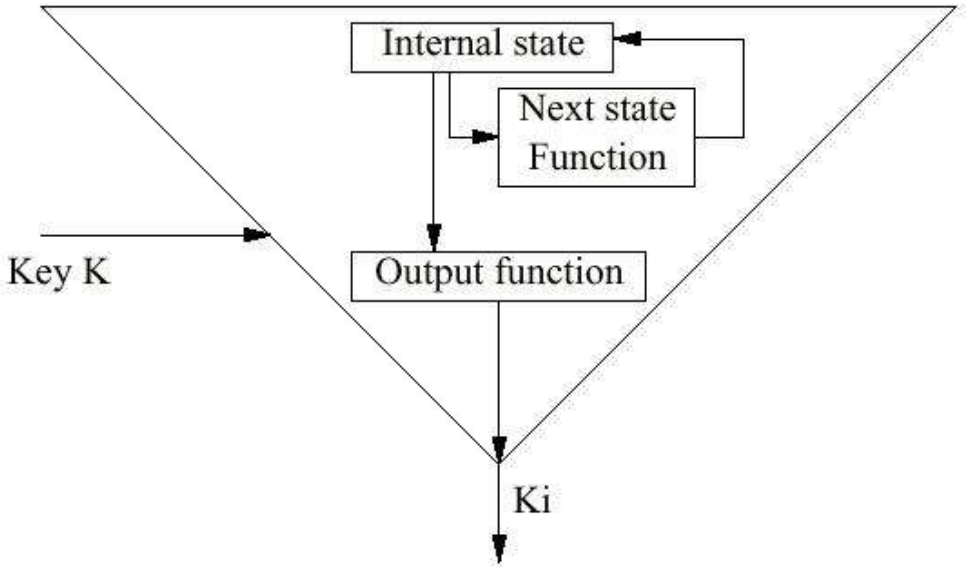
- I cifrari a sequenza (stream cipher) convertono il testo in chiaro nel crittogramma un bit o byte per volta. L'implementazione più semplice di un cifrario a sequenza è quella descritta dal seguente schema:



- La funzione di cifratura e di decifrazione è costituita semplicemente dall'operatore XOR (⊕). Ogni bit o byte del crittogramma C_i viene ottenuto con l'operazione di XOR tra il bit o byte del testo in chiaro P_i e il bit o byte del keystream K_i.

I cifrari a sequenza (stream cipher)

- Il Keystream è un generatore di numeri che accetta come input la chiave k di cifratura.



- La sicurezza dei cifrari a sequenza dipende interamente dal keystream. Una delle caratteristiche fondamentali di un buon keystream è quella di generare numeri i più casuali possibili (pseudo-casuali).

Tipologie di cifrari a blocchi: CFB

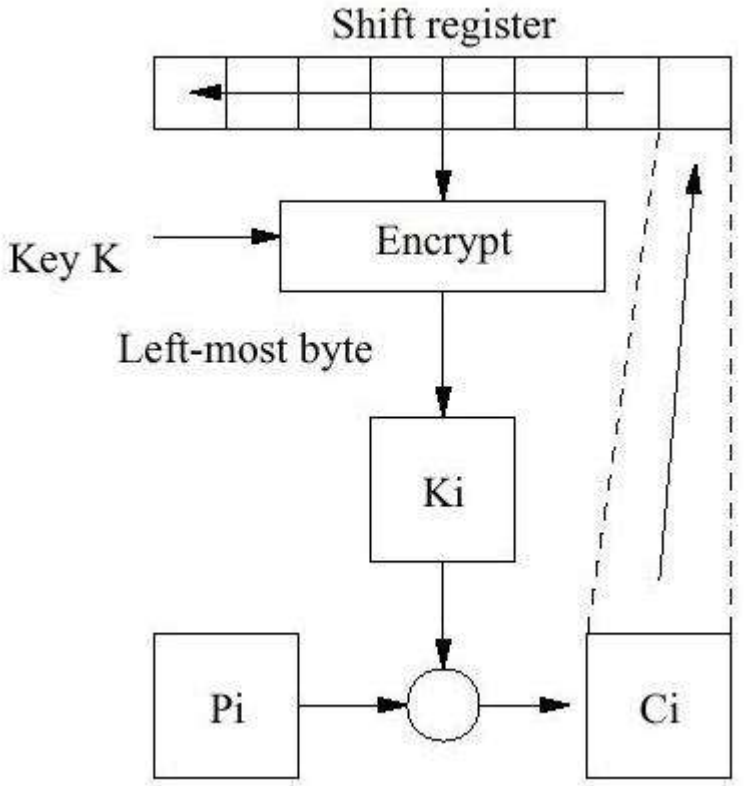
- Nei cifrari a blocchi CFB (Cipher FeedBack) si utilizzano le tecniche dei cifrari a sequenza combinate con la tecnica CBC. Nella modalità CBC i blocchi vengono cifrati utilizzando l'ultimo crittogramma generato con un'operazione di XOR. In questo modo l'operazione di cifratura ha inizio solo quando il blocco in chiaro viene ricevuto interamente. In alcuni ambiti applicativi questa modalità può essere un limite in termini di interattività (si pensi ad esempio agli emulatori di terminale).

- Nella modalità CFB i blocchi vengono cifrati suddividendoli in unità più piccole (di n bit) utilizzando un registro a scorrimento lineare. Queste unità vengono cifrate attraverso l'utilizzo di un CBC con la chiave k di cifratura. Gli n-bit più significativi vengono combinati con gli n-bit del testo in chiaro tramite un'operazione di XOR. Il crittogramma generato viene utilizzato per le operazioni successive di cifratura tramite il reinserimento nel registro a scorrimento lineare.

Tipologie di cifrari a blocchi: CFB

- Sequenza delle operazioni di cifratura per un 8-bit CFB:

1. i=1, lo shift register viene riempito con il vettore di inizializzazione IV (come nel CBC);
2. Il registro viene cifrato con la chiave K;
3. K_i= 8 bit più significativi del crittogramma;
4. P_i= 8 bit del testo in chiaro;
5. $C_i = P_i \oplus K_i$ (stream cipher);
6. Lo shift register viene aggiornato con i valori del crittogramma C_i.
7. Si passa allo step successivo i+1;

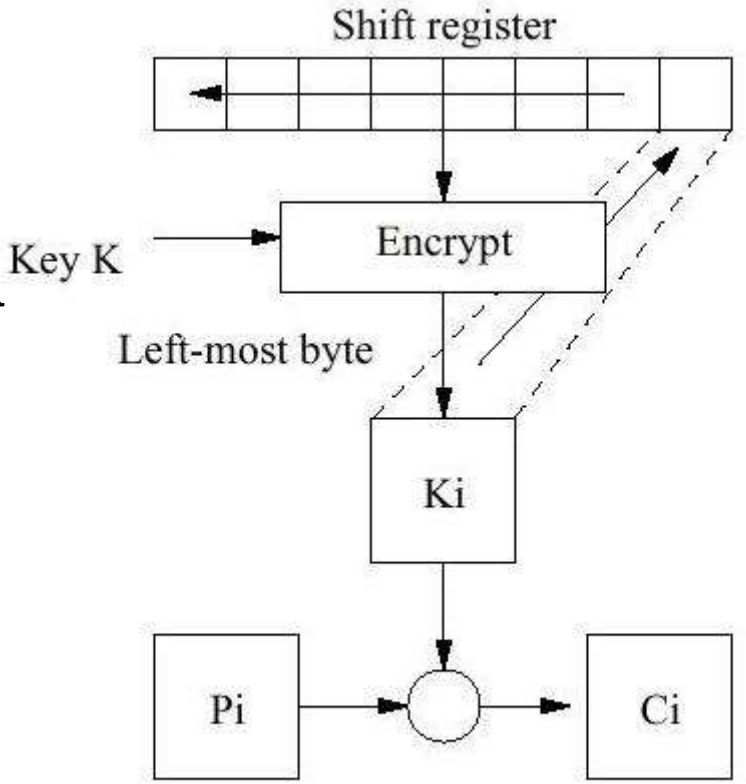


Tipologie di cifrari a blocchi: OFB

- Il metodo OFB (Output FeedBack) è un modo di eseguire un cifrario a blocchi come uno cifrario a sequenza sincrono (synchronus stream cipher).

- E' simile al CFB solo che in questa nuova modalità il registro (shift register) viene rigenerato dagli n-bit più significativi del crittogramma generato dalla chiave K, K_i.

- Sia in cifratura che in decifrazione la tecnica OFB utilizza l'algoritmo di Encrypt nella stessa modalità di cifratura. Questa modalità viene spesso chiamata **internal feedback**.



Consigli sulla scelta delle tipologie di cifrari a blocchi

- In generale valgono le seguenti considerazioni:
  - ECB (Eletronic CodeBook) è consigliabile quando i dati da cifrare sono casuali (random) e di dimensioni limitate;
  - CBC (Cipher Block Chaining) è la modalità adatta per la cifratura dei file;
  - CFB (Cipher Feedback) è la modalità adatta per la cifratura di sequenze di byte dove è necessario cifrare ogni singolo byte (stream).
  - OFB (Output Feedback) è paragonabile al CFB e può essere utilizzato in quelle applicazioni dove la propagazione dell'errore è accettabile. Non è particolarmente sicuro! Ne sconsiglio l'utilizzo, nella libreria mcrypt() è meglio usare il metodo NOFB.
  - STREAM, nella libreria mcrypt() del Php esistono solo due algoritmi “puri” di stream l'RC4 ed il WAKE.

Esempio d'utilizzo della libreria mcrypt()

- Un esempio d'utilizzo delle funzioni mcrypt_encrypt() e mcrypt_decrypt().

```
<?php
   $iv_size = mcrypt_get_iv_size(MCRYPT_BLOWFISH,MCRYPT_MODE_CBC);
   $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);

   echo "Dimensione dell'IV: $iv_size<br>";
   echo "IV: ". bin2hex($iv) ."<br><br>";

   $key = "Parola segreta";   $text = "Messaggio di prova.";
   echo "Testo in chiaro: $text<br>"; echo "Chiave (key): $key<br>";

   $crypttext = mcrypt_encrypt(MCRYPT_BLOWFISH, $key, $text,
MCRYPT_MODE_CBC, $iv);
   echo "Crittogramma: ". bin2hex($crypttext). "<br>";

    $decrypted_string = mcrypt_decrypt(MCRYPT_BLOWFISH, $key,
$crypttext, MCRYPT_MODE_CBC, $iv);
    echo "Testo decifrato: $decrypted_string";
?>
```

La libreria mhash()

- La libreria mhash() non è presente nell'installazione di default di PHP.

- Essa è solo un modulo d'interfaccia alla libreria GPL mhash disponibile all'indirizzo mhash.sourceforge.net.

• La libreria mhash supporta i seguenti algoritmi di hash:
SHA1, **GOST**, **HAVAL256**, **HAVAL224**, **HAVAL192**, **HAVAL160**, **HAVAL128**, **MD5**, **MD4**, **RIPEMD160**, **TIGER**, **TIGER160**, **TIGER128.**

- Attualmente la versione stabile è la 0.8.18. E' disponibile anche una versione per sistemi Windows.

Installazione della libreria mhash() su Gnu/Linux

- Per installare la libreria mhash-x.x.tar.gz è necessario compilare i sorgenti con la seguente procedura:

```
gunzip mhash-x.x.tar.gz
tar -xvf mhash-x.x.tar.gz
./configure
make
make install
```

- Una volta installata la libreria è possibile ricompilare il Php con la direttiva --with-mhash utilizzando la seguente procedura:

```
cd [dir PHP]
./configure --with-mhash=[dir mhash]
make
make install
```

[dir PHP] e [dir mhash] sono le directory di installazione del PHP e della libreria mhash rispettivamente.

Le funzioni della libreria mhash()

- La libreria mhash() dispone delle seguenti funzioni:

•mhash_count -- Get the highest available hash id
•mhash_get_block_size -- Get the block size of the specified hash
•mhash_get_hash_name -- Get the name of the specified hash
•mhash_keygen_s2k -- Generates a key
mhash -- Compute hash

- La funzione principale è mhash:

string mhash (int hash, string data [, string key])

dove *hash* è il tipo di algoritmo (ad esempio MHASH_MD5), *data* è la stringa per il calcolo dell'hash e *key* è un parametro opzionale che specifica la chiave per il calcolo dell'HMAC.

La funzione mhash()

- Esempio d'utilizzo della funzione mhash():

```
<?php
•   $input = "esempio di utilizzo di mhash()";
•   $hash = mhash (MHASH_MD5, $input);
•   echo "Hash: ".bin2hex ($hash)."<br>";
•   $hash = mhash (MHASH_MD5, $input, "test");
•   print "HMAC: ".bin2hex ($hash)."<br>";
?>
```

- L'output della funzione mhash() è di tipo binario, quindi è necessario convertirlo in esadecimale per poterlo visualizzare correttamente tramite la funzione bin2hex.
- Nel calcolo dell'HMAC (Hash Message Authentication Code) la stringa “test” rappresenta la chiave di cifratura. Quindi $hash alla fine contiene l'MD5 della stringa $input cifrata con la chiave “test”.

Una funzione per generare password “sicure”

- La libreria mhash() mette a disposizione una funzione molto utile per generare delle password “sicure” a partire dalle password scelte da un utente.

- Il problema delle password scelte dagli utenti è che esse risultano essere troppo prevedibili. Un semplice attacco basato su un dizionario linguistico, meglio se calibrato sulle abitudini dell'utente, è sufficiente per scovare la password di un utente in pochi secondi. La sicurezza di un sistema non può essere affidata alle password degli utenti!

- “*L'anello più debole di un sistema di sicurezza è il fattore umano*” Kevin D. Mitnick

- La funzione mhash_keygen_s2k della libreria mhash() consente di rimediare, almeno in parte, a questo problema.

La funzione mhash_keygen_s2k()

- Il prototipo della funzione mhash_keygen_s2k() è:

string mhash_keygen_s2k (int hash, string password, string salt, int bytes)

dove *hash* è il tipo di algoritmo hash utilizzato per la generazione della chiave (ad esempio MHASH_MD5), *password* è la password utente che dovrà essere trasformata, *salt* è una stringa pseudo-casuale che viene utilizzata per la generazione, *bytes* è la lunghezza in bytes della password che si desidera generare.

- La lunghezza di *salt* è fissa a 8 bytes, ad ogni generazione di una nuova password il valore di *salt* dovrebbe essere differente in modo da aumentare la sicurezza dell'algoritmo.

- Questa funzione di generazione delle password è basata sull'algoritmo Salted S2K specificato nel documento OpenPGP (RFC 2440).

Esempio d'utilizzo di mhash_keygen_s2k()

- Ecco un breve esempio sull'utilizzo della funzione mhash_keygen_s2k() per la generazione delle password:

```
<?php
   $password_utente = "test" ;
   // genero il salt casuale tramite md5
   $salt = substr(pack("h*", md5(mt_rand())) , 0, 8);

   // genero una password di 10 bytes

   $password= mhash_keygen_s2k(MHASH_MD5, $clear_pw, $salt, 10);
   echo "Password utente= $password_utente<br>";
   echo "Password generata= $password<br>";
?>
```

- La funzione mhash_keygen_s2k() può essere utilizzata per generare le chiavi di cifratura per gli algoritmi simmetrici della libreria mcrypt()...

Bibliografia e siti Internet

- Libri:
  - “Applied Cryptography, second edition” Bruce Schneier (John Wiley & Sons, 1996)
  - “Pratical Cryptography” Niels Ferguson, Bruce Schneier (Wiley Publishing, 2003)
  - “L'arte dell'inganno” Kevin D.Mitnick (Feltrinelli, 2003)
- Siti Internet:
  - Sito ufficiale PHP
  - PHP's Encryption Functionality
  - RFC 1321 - The MD5 Message-Digest Algorithm
  - RFC 3174 - US Secure Hash Algorithm 1 (SHA1)
  - MD5 in Javascript
  - Applying crypt() to User Validation
  - Compiling PHP 4 and Apache 2 from source on Linux OS