PENGGUNAAN ENKRIPSI DALAM PHP MENGGUNAKAN LIBRARY *MCRYPT*

Irvan Prama Defindal – NIM : 13503018

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : irvan.prama.defindal@comlabs.itb.ac.id

Abstraksi

Makalah ini kami susun untuk memberikan informasi yang dirasa perlu bagi pengguna bahasa pemrograman PHP yang berbasis web sehingga faktor keamanan benar-benar diperhatikan oleh programmer ketika melakukan pengkodean program. PHP sendiri telah menyediakan library khusus untuk menyandikan paket-paket data yang akan dilewatkan melalui jaringan seperti library *mcryp* dan *mhash.*

PHP merupakan bahasa pemrograman yang paling sering digunakan dalam aplikasi berbasis web. Sebagai sebuah aplikasi *script* yang berjalan di sisi server (*server side script*), semua data akan dikirim melalui jaringan internet yang luas. Keamanan data memegang peranan penting dalam pengiriman data melalui jaringan. Selain adanya prosedur standar yang diterapkan oleh PHP untuk mengamankan data-data yang melewati jaringan , PHP juga menyediakan library khusus yang memungkinkan pengguna bahasa pemrograman tersebut mengenkripsi sendiri data. Dalam PHP dikenal 2 library enkripsi , yaitu *mcrypt* dan *mhash* yang bekerja dengan cara yang berbeda. Pada makalah ini , penulis hanya membahas fungsi-fungsi yang disediakan oleh *mcrypt.*

Seperti normalnya proses enkripsi , library *mcrypt* melalui tahap enkripsi menggunakan kunci rahasia (*rahasia key*) yang ditentukan oleh programmer, dan untuk melakukan proses dekripsi dibutuhkan kunci rahasia (*rahasia key)* tersebut . Library ini mendukung algoritma enkripsi , yaitu ECB (Electronic Code Book) , CBC (Chiper Block Chaining) , CFB (Chiper feedback), OFB (Output Feedback) , NOFB, dan stream.

Kata kunci :

1. Pendahuluan

Pengguna bahasa pemrograman kebanyakan tidak menyadari pentingnya keamanan data yang dilewatkan melalui jaringan. Di samping itu, library enkripsi pada PHP bukan merupakan library dasar, sehingga perlu dilakukan beberapa langkah konfigurasi sebelum menggunakan library tersebut. Dengan informasi yang lebih mengenai fungsi enkripsi pada PHP , penggunaan fungsi-fungsi pada library *mcrypt* akan memudahkan programmer untuk mengamankan data yang bersifat penting dan rahasia.

Dengan adanya pengetahuan mengenai cara kerja dan seluk-beluk library ini, maka programmer bisa mengambil keputusan yang tepat dalam penggunaan fungsi-fungsi enkripsi data.

2. Library mcrypt

Libary mcryt terdiri dari banyak fungsi , di antaranya :

a. mcrypt_module_open

Masukan : Nama algoritma , Direktori Algoritma , Mode yang digunakan , Direktori Mode

Proses : menentukan algoritma dan mode yang digunakan. Direktori algoritma dan mode bias dikosongkan (*null)* . Library yang digunakan ditutup menggunakan *mcrypt_module_close()*

Keluaran : Deskripsi enkripsi atau variable MCRYPT_FAILED bila terjadi kesalahan

b. mcrypt_generic_init

masukan : MCRYPT , kunci , Maksimal thread, IV

Proses : Inisialisasi semua *buffers* untuk thread tertentu yang sedang berjalan. Nilai maksimal lenofkey seharusnya dibangkitkan oleh fungsi *mcrypt_get_key_size()* dan setiap nilai dibawahnya adalah sah. *Lenofkey* dinyatakan dalam byte bukan bit. Iinitial Vector seharusnya mempunyai ukuran blok sesuai algoritma. Untuk mendapatkan ukurannya, bisa menggunakan fungsi *mcrypt_get_iv_size().* IV tidak digunakan dalam mode ECB. IV harus ada pada mode CFB, CBC, STREAM, nOFB dand OFB. Nilai Initial Vector harusnya acak dan unik, namun tidak harus rahasia . IV yang sama harus digunakan pada proses enkripsi dan dekripsi Setelah memanggil fungsi ini, pengguna bisa menggunakan descriptor untuk enkripsi atau dekripsi.

Keluaran : Nilai integer, Nilai negatif bila terjadi kesalahan

c. mcrypt_generic

masukan : MCRYPT td, plaintext, panjang

Proses : Merupkan fungsi utama enkripsi yang merupakan keluaran dari fungsi *mcrypt_generic_init().* Plainteks yang dimaksud merupakan plainteks yang akan dienkripsi dan panjang dari plaintes dinyatakan dalam bit, atau ukuran blok algoritma bila dioperasikan dalam mode blok (cbc, ecb, nofb) plaintext akan digantikan oleh ciphertext.

Keluaran : 0 bilan berhasil .

d. mdecrypt_generic

masukan : MCRYPT , ciphertext, panjang teks

Proses : merupakan fungsi utama dekripsi. Hampir sama dengan fungsi enkripsi.

Keluaran : 0 bila berhasil

e. int mcrypt_generic_end

masukan : MCRYPT

proses : Fungsi ini menhentikan proses enkripsi yang dispesifikkan oleh descriptor enkripsi Membersihkan semua *buffers*, dan menutup semua modul yang digunakan. Seringkali orang menggunakan *mcrypt_generic_deinit()* dan *mcrypt_module_close()*

Keluaran : nilai negatif bila ada kesalahan.

Untuk sourcode dari library , bisa dilihat :

Source code fungsi enkripsi dari library mcrypt

```
/* Encrypt */
      if (ein == TRUE) {
       if (openpgp!=0) x = pgp_encrypt_wrap( einfile, outfile, keyword);
       else x = encrypt_general(algorithm, einfile, outfile, keyword);

       if (x == 0) {
          if (stream_flag == FALSE) {
             sprintf(tmperr, _("File %s was encrypted.\n"), einfile);
             err_warn(tmperr);
          } else {
             sprintf(tmperr, _("Stdin was encrypted.\n"));
             err_warn(tmperr);
          }
#ifdef HAVE_STAT
#ifdef HAVE_UTIME_H
          if (stream_flag == FALSE)
             copyDate(einfile, outfile);
#endif
#endif
          if (unlink_flag == TRUE && stream_flag == FALSE)
             x = unlink(einfile);
          if (x != 0)
             perror("unlink");

       } else {
          if (stream_flag == FALSE) {
             sprintf(tmperr,
                   _
                   ("File %s was NOT encrypted successfully.\n"),
                   einfile);
             err_crit(tmperr);
             if (x != -1) {
              if (nodelete == FALSE)
                 remove(outfile);
             } else {
              x = 1;
             }
          } else {
             err_crit(_("Stdin was NOT encrypted successfully.\n"));
          }

       }


       return_val += x;
      }
   }                    /* the main loop */
/* Clear the last keyword used */
   if (keyword != NULL) {
      Bzero(keyword, strlen(keyword));
      _mcrypt_free(keyword);
   }
```

Source code fungsi dekrips dari mcrypt

```
/* Decrypt */
      if (din == TRUE) {
       if (openpgp!=0) x = pgp_decrypt_wrap( dinfile, outfile,
keyword);
       else x = decrypt_general(algorithm, dinfile, outfile,
keyword);

       if (x == 0) {
          if (stream_flag == FALSE) {
             sprintf(tmperr, _("File %s was decrypted.\n"),
dinfile);
             err_warn(tmperr);
          } else {
             sprintf(tmperr, _("Stdin was decrypted.\n"));
             err_warn(tmperr);
          }
#ifdef HAVE_STAT
# ifdef HAVE_UTIME_H
          if (stream_flag == FALSE)
             copyDate(dinfile, outfile);
# endif
#endif

          if (unlink_flag == TRUE && stream_flag == FALSE)
             x = unlink(dinfile);
          if (x != 0)
             perror("unlink");

} else {
          if (stream_flag == FALSE) {
             sprintf(tmperr,
                   _
                   ("File %s was NOT decrypted successfully.\n"),
                   dinfile);
             err_crit(tmperr);
             if (x != -1) {
              if (nodelete == FALSE)
                 remove(outfile);
             } else {
              x = 1;
             }
          } else {
             err_crit(_("Stdin was NOT decrypted successfully.\n"));
          }
       }

       return_val += x;
      }
```

3. Mode Operasi

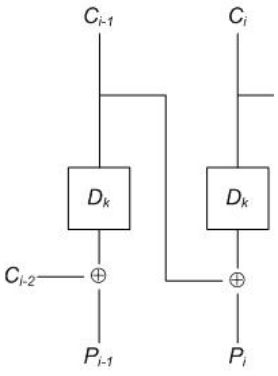
a. CBC (Chiper Blok Chaining)

Sesuai dengan namanya, mode Chiper Blok Chaining menerapkan mekanisme umpan balik ; hasil enkripsi blok sebelumnya diumpanbalikkan ke dalam enkripsi blok yang sedang diproses oleh sistem. Dalam metode ini , operasi XOR diterapkan pada blok plainteks yang sedang diproses dengan blok cipherteks hasil enkripsi sebelumnya. Setalah itu hasil operasi *XOR* yang telah didapat masuk ke dalam fungsi enkripsi. Dalam mode *CBC*, setiap blok cipherteks bergantung tidak hanya pada blok plainteks yang bersesuain , tetapi juga pada seluruh blok plainteks sebelumnya.

Bila dinyatakan dengan notasi persamaan, maka proses enkripsi CBC dinyatakan dalam :

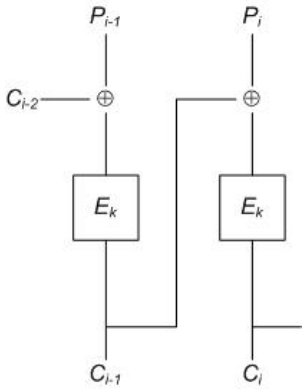
$$C_i = E_k(P_1 \oplus C_{i-1})$$

Pada mode CBC dibutuhkan Initalization Vector (IV) sebagai C_0. *IV* dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program. Jadi, untuk menghasilkan blok cipherteks pertama (C_1), *IV* digunakan untuk menggantikan blok cipherteks sebelumnya, C_0.
Pada dekripsi, blok plainteks diperoleh dengan cara meng-*XOR*-kan *IV* dengan hasil dekripsi terhadap blok cipherteks pertama.



Prose Dekripsi CBC

Karakteristik mode CBC adalah setiap chiperteks bergantung tidak hanya pada blok plainteks , tetapi juga pada seluruh blok plainteks. Keuntungan yang didapat dari mode ini diantaranya ; tidak ada korelasi antara posisi blok



Proses Enkripsi CBC

Pada proses dekripsi , blok cipherteks yang sedang diproses dimasukkan ke fungsi dekripsi. Kemudian melakukan operasi *XOR* antara hasil dekripsi dengan blok cipherteks sebelumnya. Bisa dikatakan bahwa blok cipherteks sebelumnya berfungsi sebagai umpan maju pada akhir proses dekripsi. Skema enkripsi dan dekripsi dengan mode *CBC* dapat dilihat pada skema

Proses Dekripsi bisa dinyatakan dalam persamaan :

$$P_i = D_k(C_i) \oplus C_{i-1}$$

plainteks yang sama dengan posisi blok chiperteks. Kerugian yang bisa saja terjadi dengan mode ini adalah ; kesalahan pada satu bit chiperteks akan membuat chiperteks berikutnya menjadi salah.

b. CFB (Chiper Feed Back)

Dengan menggunakan mode *CFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *CFB*-nya disebut *CFB* 8-bit. Secara umum *CFB* *n*-bit mengenkripsi plainteks sebanyak *n* bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok).

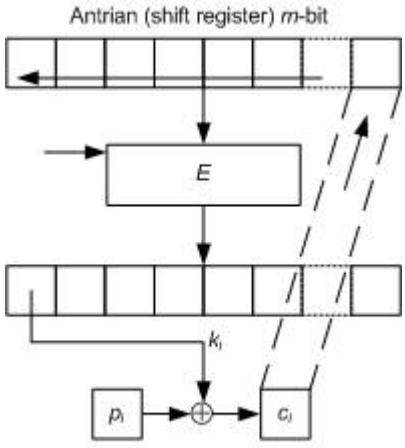
Mode *CFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.

CFB menggunakan skema umpan balik dengan mengaitkan blok plainteks bersama-sama sedemikian sehingga cipherteks bergantung pada semua blok plainteks sebelumnya.

$$C_i = P_i \oplus E_k(C_{i-1})$$
$$P_i = C_i \oplus D_k(C_{i-1})$$

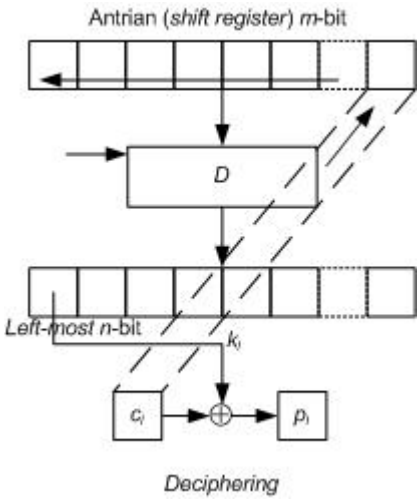
IV pada *CFB* tidak perlu dirahasiakan. *IV* harus unik untuk setiap pesan, sebab *IV* yang sama untuk setiap pesan yang berbeda akan menghasilkan *keystream* k_i yang sama.

Proses enciper



Enchiper

Proses Deciper



Deciphering

Tinjau mode *CFB n*-bit yang bekerja pada blok berukuran *m*-bit. Algoritma enkripsi dengan mode *CFB* adalah sebagai berikut:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil enkripsi berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari plainteks menjadi *n*-bit pertama dari cipherteks. Salinan (*copy*) *n*-bit dari cipherteks ini dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan semua *m*-*n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan.
3. *m*-*n* bit plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

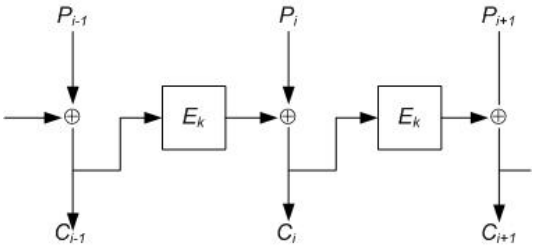
Bila dinyatakan dengan notasi persamaan, maka proses yang terjadi adalah :

$$C_i = P_i \oplus MSB_m(E_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

Dimana :

X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
|| = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte

Kasus khusus : dimana m = n



Enkripsi *CFB*

Sedangkan, algoritma dekripsi dengan mode *CFB* adalah sebagai berikut:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Dekripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil dekripsi berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari cipherteks menjadi *n*-bit pertama dari plainteks. Salinan (*copy*) *n*-bit dari

cipherteks dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan semua *m-n* lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan.

3. *m-n* bit cipherteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

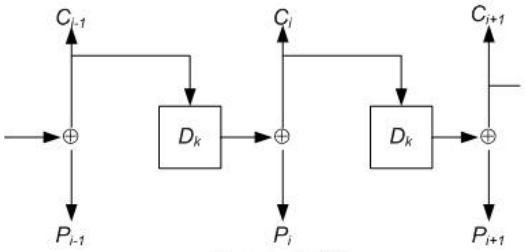
Secara formal, proses dekripsi *CFB n*-bit dapat dinyatakan sebagai:

$$P_i = C_i \oplus MSB_m(D_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

Dimana :

X_i = isi antrian dengan X_i adalah *IV*
D = fungsi dekripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
|| = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte

Kasus khusus : dimana m = n



Dekripsi *CFB*

Catatan: Algoritma *E* = Algoritma *D*

Baik enkripsi maupun dekripsi, algoritma *E* dan *D* yang digunakan sama. Karakteristik mode CFB adalah setiap Pengenkripsian data dilakukan pada unit lebih kecil daripada blok.

Keuntungan yang didapat dari mode ini diantaranya ; algoritma enkripsi yang rumit menyebabkan hasil enkripsi lebih kompleks. Kerugian yang bisa saja terjadi dengan mode ini adalah ; kesalahan pada satu bit chiperteks akan membuat chiperteks berikutnya menjadi salah.

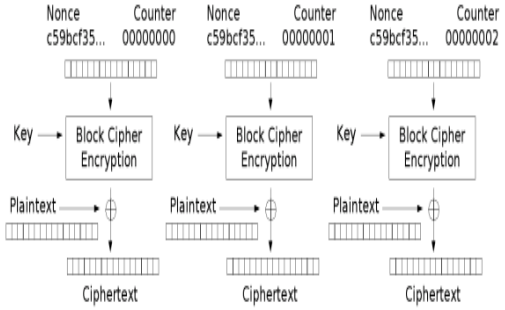
c. CTR (Counter)

Seperti halnya CFB, mode ini menjadikan block menjadi stream. Mode ini membangkitkan blok kunci *stream* berikutnya dengan mengenkripsikan nilai yang berurutan dari *counter*. Fungsi counter sendiri merupakan fungsi yang menghasilkan himpunan yang tidak akan berulang dalam jangka waktu yang lama.

Fungsi Enkripsi :
O[j] = E(K)(T[j]); for j = 1, 2...n;
C[j] = P[j] ^ O[j]; for j = 1, 2...n.

Dimana :

P : plaintext,
C : ciphertext,
E : Fungsi enkripsi
K : Kunci sesi
T : Penghitung.

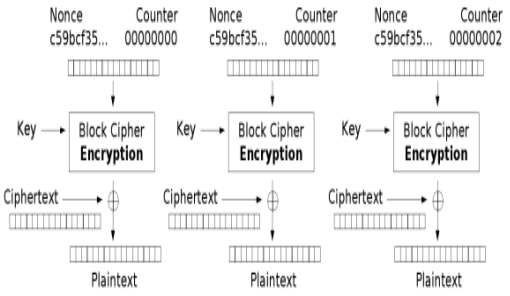


Counter (CTR) mode encryption

Fungsi Dekripsi:
O[j] = E(K)(T[j]); for j = 1, 2...n;
P[j] = C[j] ^ O[j]; for j = 1, 2...n.

Dimana :

P : plaintext,
C : ciphertext,
E : Fungsi enkripsi
K : Kunci sesi
T : Penghitung.

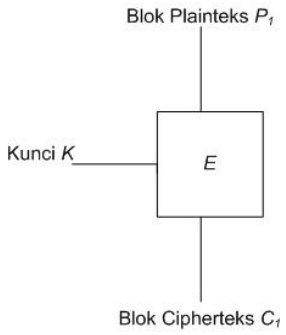


Counter (CTR) mode decryption

d. ECB (Electronic Code Book)

Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen menjadi blok cipherteks C_i. Secara matematis, enkripsi dengan mode *ECB* dinyatakan sebagai

$$C_i = E_k(P_i)$$

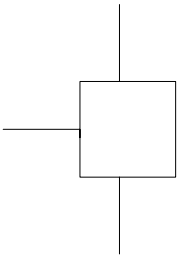


Enkripsi mode EBC

dan dekripsi sebagai

$$P_i = D_k(C_i)$$

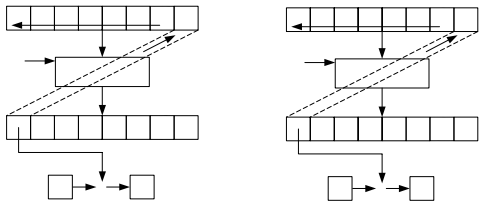
yang dalam hal ini, P_i dan C_i masing-masing blok plainteks dan cipherteks ke-*i*. Skema dekripsi dengan mode *ECB* dapat dilihat pada Gambar di bawah.



Ada kemungkinan panjang plainteks tidak habis dibagi dengan panjang ukuran blok yang ditetapkan. Hal ini mengakibatkan blok terakhir berukuran lebih pendek daripada blok-blok lainnya. Satu cara untuk mengatasi hal ini adalah dengan *padding*, yaitu menambahkan blok terakhir dengan pola bit yang teratur agar panjangnya sama dengan ukuran blok yang ditetapkan.

e. OFB (Output Feedback)

Pada mode *OFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *OFB*-nya disebut *OFB* 8-bit. Secara umum *OFB n*-bit mengenkripsi plainteks sebanyak *n* bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok). Mode *OFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.



Tinjau mode *OFB n*-bit yang bekerja pada blok berukuran *m*-bit. Algoritma enkripsi dengan mode *OFB* adalah sebagai berikut :

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil enkripsi dimasukkan ke dalam antrian

(menempati *n* posisi bit paling kanan antrian), dan *m*-*n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan. *n* bit paling kiri dari hasil enkripsi juga berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari plainteks menjadi *n*-bit pertama dari cipherteks.

3. *m*-*n* bit plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

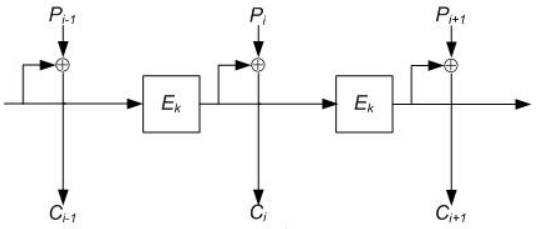
Proses Enkripsi:

$C_i = P_i \oplus MSB_m(E_k(X_i))$

$X_{i+1} = LSB_{m-n}(X_i) \,||\, LSB_n(E_k(X_i))$

Dimana

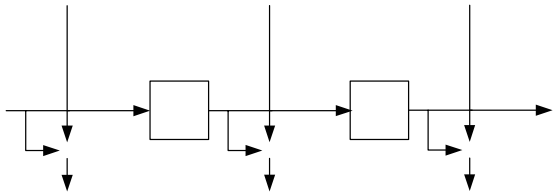
X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
|| = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte



Enkripsi *OFB*

Sedangkan, algoritma dekripsi dengan mode *OFB* adalah sebagai berikut :

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Dekripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil dekripsi dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan *m*-*n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan. *n* bit paling kiri dari hasil dekripsi juga berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari cipherteks menjadi *n*-bit pertama dari plainteks.
3. *m*-*n* bit cipherteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.



Secara formal, Proses Dekripsi mode *OFB* *n*-bit dapat dinyatakan sebagai :

$P_i = C_i \oplus MSB_m(D_k(X_i))$

$X_{i+1} = LSB_{m-n}(X_i) \,||\, LSB_n(E_k(X_i))$

yang dalam hal ini:

X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
D = fungsi dekripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
|| = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte

Baik enkripsi maupun dekripsi, algoritma *E* dan *D* yang digunakan sama. *OFB* menggunakan skema umpan balik dengan mengaitkan blok plainteks bersama-sama sedemikian sehingga cipherteks bergantung pada semua blok plainteks sebelumnya.

f. Chiper Stream

Chiper stream adalah chiper yang simetris dimana digit-digit dari plainteks dienkripsikan secara bersamaan. Transformasi dari digit-digit yang berurutan bervariasi satu sama lain selama proses pengkodean pesan. Chiper *Stream* , atau ada juga yang menyebut dengan nama State Chiper mempunyai waktu eksekusi yang lebih cepat dan kompleksitas perangkat keras yang lebih kecil bila dibandingkan dengan Block Chiper manapun.

Chiper Stream terdiri dari 2 jenis, yaitu :

1) Chiper Stream Sinkron

Pada jenis ini , *stream* yang terdiri dari digit-digit acak dibangkitkan satu-satu tanpa adanya ketergantungan antara plainteks dengan chiperteks. Pada proses enkripsi, hasil pembangkitan acak akan dikombinasikan dengan plainteks. Sedangkan pada proses dekripsi angka-angka itu dikombinasikan dengan chiperteks

2) Chiper Stream Asinkron

Pendekatan ini menggunakan sejumlah N digit chiperteks sebelumnya untuk menentukan stream kunci. Skema ini juga dikenal sebagai CTAK (Chiperteks Autokey) Pada jenis ini, kesalahan kecil pada 1 digit akan mempengaruhi digit lainnya.

4. Fungsi Enkripsi

a. DES

Merupakan algoritma yang dikembangkan oleh IBM dan NSA Amerika. Menggunakan 56 bit kunci dan 64 bit blok. Karena kecilnya kunci, algoritma ini tergolong lemah.

b. TripleDES

Merupakan pengembangan DES dengan 3 kali proses enkripsi. Algoritma ini mengenkripsi plainteks sekali. Lalu mendekripsi dengan kunci kedua, dan menekripsi lagi dengan kunci ke-3. Lebih baik dari DES karena mempunyai 168 bit kunci

c. Blowfish

Algoritma ini dirancang oleh Bruce Schneier. Jauh lebih baik dan cepat daripada DES. Menggunakan 448 bit kunci

d. 3-WAY

Algoritma ini dirancang oleh Joan Daemen. Menggunakan kunci dan blok yang berukuran 96bit.

e. SAFER-SK+

Algoritma ini disusun oleh Prof J.L. Massey, Prof Gurgen H Khachatrian dan Dr Melsik K Kuregian untuk Cylink. SAFER+ berdasarkan pada chiper dari SAFER sendiri dan menyediakan ukuran blok 128 bit untuk 3 macam panjang kunci ; 128,192, 256 bit.

f. TWOFISH, TEA

TwoFish dirancang oleh Bruce Schneier, Doug Whiting , John Kelsey, Chris Hall, David Wagner untuk sistem Couterpane. Dirancang seaman dan sefleksibel mungkin . Menggunakan blok berukuran 128 bit dan kunci sepanjang 128,192,256 bit.

g. RC2

Rivest Chiper dibuat oleh Ron Rivest. Menggunakan blok berukuran 64 bit dan kunci yang panjangnya antara 8 sampai 1024 bit. Sangat optimal untuk microprosesor 16bit.

h. GOST

Merupakan algoritma yang digunakan Uni Sovyet yang merupakan singkatan dari *GOsudarstvenny Standard.* Algoritma ini menggunakan kunci sepanang 256bit dan blok sepanjang 64 bit. Metode S-Box yang dipakai di sini diaplikasikan pada Bank Sentral Federasi Rusia. Standar dari algoritma ini ditulis oleh A Zabotin (Pemimpin proyek) G.P. Gilazkov dan V.B. saeva. Diterima dan diperkenalkan dalam penggunaan oleh Panitia Standarisasi Kenegaraan USSR pada tahun 2 Juni 1989 nomor 1409.

Kesimpulan dari algoritma yang digunakan :

Algoritma	Dibebani	Jumlah Blok	Jenis Kunci
3-Way	Tidak	96	Rahasia
Arcfour	(Mungkin) tidak	Stream	Rahasia
Blowfish	Tidak	64	Rahasia
Cast	Tidak	64	Rahasia
DES	Tidak	64	Rahasia
Enigma	Tidak	Stream	Rahasia
Gost	Tidak	64	Rahasia
Idea	Tidak	64	Rahasia
RC2	Tidak	64	Rahasia
RC6	Ya	128	Rahasia
Loki	Tidak	128	Rahasia
Mars	Ya	128	Rahasia
Panama	Tidak	Stream	Rahasia
Rijndael	Tidak	128, 192, 256	Rahasia
Safer	Tidak	64	Rahasia
Safer+	Tidak		Rahasia

Serpent	No	Block	Rahasia
Skipjack	Ragu-ragu	64	Rahasia
Twofish	Tidak	128	Rahasia
Triple DES	Tidak	64	Rahasia
Wake	Tidak		Rahasia
XTea	Tidak	64	Rahasia

5. Pengujian *mcrypt*

Dalam penggunaanya dalam php, mcypt mempunyai fungsi standar :

a. Fungsi untuk menentukan mode

- MCRYPT_MODE_ECB cocok untuk data acak seperti enkripsi kunci lain .
- MCRYPT_MODE_CBC cocok untuk enkripsi file dimana keamanan meningkat terhadap ECB secara signifikan
- MCRYPT_MODE_CFB mode terbaik enkripsi byte streams dimana tiap bytes harus dienkripsikan.
- MCRYPT_MODE_OFB (in 8bit) hampir sama dengan CFB namun digunakan dalam aplikasi yang tingkat kesalahn tidak bisa ditoleransi. Tidak aman karena beroperasi dalam mode 8bit..
- MCRYPT_MODE_NOFB (output feedback dalam nbit) hampir sama dengan OFB, tapi lebih aman karena beroperasi sesuai dengan ukuran blok algoritma
- MCRYPT_MODE_STREAM merupakan mode extra untuk mendukung algoritma stream seperti WAKE atau RC4.

Fungsi untuk melakukan enkripsi

- `MCRYPT_ENCRYPT` (integer)
- `MCRYPT_DECRYPT` (integer)
- `MCRYPT_DEV_RANDOM` (integer)
- `MCRYPT_DEV_URANDOM` (integer)
- `MCRYPT_RAND` (integer)

Fungsi untuk memilih algoritma

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV
- MCRYPT_ARCFOUR
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT
- MCRYPT_ENIGMA
- MCRYPT_GOST
- MCRYPT_IDEA
- MCRYPT_LOKI97
- MCRYPT_MARS
- CRYPT_PANAMA
- MCRYPT_RIJNDAEL_128
- MCRYPT_RIJNDAEL_192
- MCRYPT_RIJNDAEL_256
- MCRYPT_RC2
- MCRYPT_RC4
- MCRYPT_RC6
- MCRYPT_RC6_128
- MCRYPT_RC6_192
- MCRYPT_RC6_256
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS
- MCRYPT_SERPENT
- MCRYPT_SERPENT_128
- MCRYPT_SERPENT_
- MCRYPT_SERPENT_256
- MCRYPT_SKIPJACK
- MCRYPT_TEAN
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES
- MCRYPT_TWOFISH
- MCRYPT_TWOFISH128
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE
- MCRYPT_XTEA

Pengujian dilakukan dengan code berikut :

```
<?php
   $key = "this is a rahasia
key";
   $input = "Let us meet at 9
o'clock at the rahasia place.";

   $td =
mcrypt_module_open('tripledes',
'', 'ecb', '');
   $iv = mcrypt_create_iv
(mcrypt_enc_get_iv_size($td),
MCRYPT_RAND);
   mcrypt_generic_init($td,
$key, $iv);
   $encrypted_data =
```

```
mcrypt_generic($td, $input);
   mcrypt_generic_deinit($td);
   mcrypt_module_close($td);
?>
```

6. Kesimpulan

Library mcrypt merupakan library yang sangat kompleks. Library ini hampir mendukung semua mode enkripsi yang umum digunakan. Sifatnya yang open source membuat pengembangannya berlangsung dengan cepat. Pengguna bahasa pemrograman dimanjakan dengan fungsi yang benar-benar tinggal digunakan, tanpa harus memikirkan proses yang terjadi di dalamnya. Berbagai macam algoritma yang didukung oleh library ini menjadi faktor penting dalam memudahkan pengguna PHP, karena bila library ini menggunakan algoritma dan mode tertentu, tidak semua orang akan menguasai dan mengatahui seluk beluk keamanan data yang disimpan.

7. Daftar Pustaka

http://www.phpfreaks.com/phpmanual/page/ref.mcrypt.html
http://www.phpfreaks.com/tutorials/128/4.php
http://mcrypt.hellug.gr/lib/mcrypt.3.html#toc
http://en.wikipedia.org/wiki/Mcrypt
http://www.mcs.vuw.ac.nz/technical/software/PHP/ref.mcrypt.html
http://ftp.emini.dk/pub/php/win32/mcrypt/
http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation#Counter_.28CTR.29