

Nama : Dwi Wahyuning Tyas
Nim : 1310652039

Program Enkripsi dan Deskripsi untuk keamanan data pada jaringan

Program Enkripsi Dekripsi

Program enkripsi file teks berfungsi untuk mengenkripsi (mengacak) suatu file teks sehingga informasi di dalamnya tidak bisa dibaca. Pengacakan dilakukan berdasarkan kata kunci (key) tertentu yang diisikan oleh pengguna.

Program juga sekaligus berfungsi untuk meng dekripsi (enkripsi balik) file hasil enkripsi. Agar file dapat didekripsi dengan benar, kata kunci (kode) yang digunakan harus sama dengan kata kunci enkripsi. Jadi disini hanya terdapat 1 key (kunci) untuk melakukan enkripsi dan dekripsi (symmetric encryption).

Cara kerja enkripsi dilakukan dengan menambahkan kode karakter teks sumber dengan teks kunci. Kunci yang lebih pendek dari teks sumber akan berulang-ulang sampai panjangnya sama dengan teks sumber. Misalnya panjang teks sumber 20 karakter, sedangkan kunci = "abcde" (5 karakter), maka faktor penambahan enkripsi adalah abcdeabcdeabcdeabcde.

Contoh:

```
; sumber : "Kucing"  
; kunci : "kode"
```

maka:

```
; sumber : 75 117 99 105 110 103  
; kunci : 107 111 100 101 107 111  
; hasil : 182 228 199 206 217 214
```

Demikian pula dengan proses dekripsi, yaitu dengan melakukan operasi pengurangan teks sumber dengan kunci. Tentunya kunci harus sama dengan kunci enkripsi untuk menghasilkan teks hasil yang benar. Jika kunci tidak cocok, informasi dalam teks tidak akan terbaca..

Seperti dikatakan sebelumnya, program ini terdiri dari 2 bagian. Terdapat 2 listing program:

- enkripsi.asm, berupa kode assembly yang berisi fungsi enkripsi
- enkripsi_main.c, berupa kode C yang berisi program utama yang memanfaatkan enkripsi.asm

listing enkripsi.asm

```
;
```

```

; file ini berisi fungsi enkripsi_dekripsi yang akan digunakan oleh
enkripsi_main.c
;
; tugas besar mata kuliah bahasa assembly
; oleh      :
; Bayu Rizaldhan rayes
;
;
; dites menggunakan Mandrake Linux 10.0
; dengan nasm 0.98.38-1mdk, gcc 3.3.2-6mdk
; teks editor Kate
;
; kompilasi & linking:
; nasm -f elf enkripsi.asm
; gcc -o enkripsiku enkripsi_main.c enkripsi.o
;
    segment .data
    format db "x = %d\n",0

global asm_enkripsi_dekripsi

segment .text

; fungsi asm_enkripsi_dekripsi
; void asm_enkripsi_dekripsi( char * dest, const char * src, char * kunci,
int flag_enkripsi);
; parameters:
;   dest - pointer ke string hasil enkripsi           (pass by
reference)
;   src   - pointer ke string yang akan dienkripsi    (pass by
reference)
;   kunci - string untuk kode enkripsi                (pass by
reference)
;   flag_enkripsi - <>1 -> enkripsi                    (pass by value)
;                 =1   -> dekripsi
;
; aturan:
; dengan menambahkan tiap2 karakter dari teks yang dienkripsi dengan
; karakter dari kunci yang diulang-ulang
; contoh: kunci="abcde" -> 97 98 99 100 101
;
; misal :   src           : "Kucing"
;           kunci         : "kode"
; maka:
;           dest          :  75  117  99 105 110 103
;           kunci         : 107  111 100 101 107 111
;           src           : 182  228 199 206 217 214
;
#define dest [ebp + 8]
#define src  [ebp + 12]
#define kunci [ebp + 16]
#define flag_enkripsi [ebp + 20]
asm_enkripsi_dekripsi:
    enter    0,0
    push     esi
    push     edi

```

```

        mov     edi, dest        ; output dari enkripsi
        mov     esi, src        ; source dari enkripsi

        mov     edx, kunci      ; edx buat menyimpan esi dari kunci

        cld

cari_loop:

        ; blok pemroses kunci
        ; memperoleh karakter kunci ke n (kunci saat ini) dari string kunci,
        ; hasilnya taruh di cx

        push esi                ; simpan dulu, karena esi akan dipakai untuk
pemrosesan string kunci,
        push edi

        mov     esi, edx        ; esi = edx, edx awal = kunci (alamat elemen
ke 1 dari kunci)
        lodsb                  ; load al, inc si      (al = kunci saat ini,
lanjutkan pencarian)

        mov     cx, ax          ; cx menyimpan karakter kunci saat ini (kunci
ke n)

        mov     edx, esi        ; simpan nilai esi ke edx untuk karakter
kunci selanjutnya

        or      al, al          ; set condition flags
        jnz     tidaknol       ; jika 0 (akhir dari string kunci),
        mov     edx, kunci      ; set edx kembali ke alamat awal kunci
(looping)

tidaknol:                            ; kunci tidak nol (bukan akhir kunci)

        ; keluar pemrosesan kunci
        ; kunci ke n sudah didapat (dalam cl), sekarang lakukan enkripsi
        ; jumlahkan karakter ke n dengan kunci ke n

        pop     edi            ; kembalikan esi, edi
        pop     esi

        lodsb                  ; load AL & inc si, al = karakter ke n dari
src

        or      al, al          ; set condition flags
        jz      copy

        cmp     flag_enkripsi, word 1      ; jika flag_enkripsi = 1 -> dekripsi
        je      dekripsi
        add     al, cl          ; (ENKRIPSI) jumlahkan karakter ke n dengan
kunci ke n
        jmp     endif
dekripsi:
        sub     al, cl          ; (DEKRIPSI) kurangkan karakter ke n dengan
kunci ke n
endif:

```

```

copy:
    stosb                ; store AL & inc di, copy al ke dest

    or    al, al         ; set condition flags
    jnz   cari_loop      ; jika bukan nol (end of string), lanjutkan
looping
    pop    edi
    pop    esi
    leave
    ret

```

Penjelasan :

Pada file ini, terdapat 1 fungsi yang bersifat global yaitu fungsi `asm_enkripsi_dekripsi`. Fungsi ini akan melakukan enkripsi sekaligus dekripsi (tergantung parameternya). Penulisan fungsi ini dibuat sesuai standar konvensi fungsi di bahasa C khususnya GCC (GNU C Compiler) di Linux.

```

#define dest [ebp + 8]
#define src [ebp + 12]
#define kunci [ebp + 16]
#define flag_enkripsi [ebp + 20]

```

Jika dalam C, fungsi tersebut akan memiliki bentuk seperti berikut :

```

void asm_enkripsi_dekripsi( char * dest, const char * src, char * kunci, int
flag_enkripsi);

```

Ada 4 parameter dari fungsi, yang akan dipush ke dalam stack, yaitu

- `dest`, sebagai parameter pertama [`ebp+8`]. `dest` merupakan parameter yang di pass by reference, karena ia akan berisi string. (berisi alamat awal dari string) dan nantinya akan menampung string hasil enkripsi/dekripsi.
- `src`, [`ebp+12`]. Berisi string sumber, yaitu teks yang akan dienkripsi/didekripsi. Parameter pass by reference.
- `kunci`, [`ebp+16`]. Berisi string kunci/kode, yaitu string yang akan dimanipulasi dengan `src` untuk proses enkripsi/dekripsi.
- `flag_enkripsi`, [`ebp+20`]. Berisi variabel untuk menandai apakah akan melakukan enkripsi atau melakukan dekripsi (bersifat boolean). Jika nilai $\neq 1$, misalnya 0, maka fungsi akan melakukan enkripsi. Jika `flag_enkripsi = 1`, maka fungsi akan melakukan dekripsi. Parameter ini merupakan pass by value.

Cara kerja.

Source code `enkripsi.asm` dilengkapi dengan komentar program untuk memahami detail cara kerja per instruksi. Secara umum cara kerjanya adalah:

- memproses string kunci, yaitu memperoleh karakter kunci ke n, dimulai dari 0. Pertama-tama nilai edi (milik dest) dan esi (milik src) akan dipush. alamat kunci akan disimpan dalam esi yang merupakan register untuk memanipulasi string. Karakter ke n disimpan dalam cx. Kemudian index n ditambah (increment) lalu disimpan dalam edx. Index ke n perlu disimpan karena selanjutnya nilai esi akan dimanfaatkan untuk hal lain (string_input), sehingga menjaga agar index kunci berikutnya tidak hilang. Jika index ke n merupakan akhir string (kode 0), maka nilai index akan mengulang dari awal, yaitu nilai kunci ([ebp+16]) Setelah karakter ke n didapat, register esi dan edi dipop kembali sehingga sekarang berisi index dari src dan dest.
- berikutnya program akan memproses src. Src berada dalam esi dan dest berada dalam edi. Perintah lodsb akan meload alamat yang ditunjuk esi ke dalam register al kemudian menambah (increment) si. Sekarang esi menunjuk ke alamat berikutnya (n+1).
- Langkah selanjutnya melakukan manipulasi terhadap 2 karakter, yaitu karakter ke n dari kunci dan karakter ke n dari esi, yang berada dalam al. Dilakukan pengecekan flag_enkripsi. Jika 1, maka lakukan dekripsi yaitu src[n]-kunci[n]. Jika tidak 1, maka lakukan enkripsi, yaitu src[n]+kunci[n]. Setelah itu melalui perintah stosb, maka nilai al akan disimpan ke alamat yang ditunjuk edi, kemudian nilai di akan ditambah.
- Program akan berulang sampai ditemukan kode karakter end of string (0).

listing enkripsi_main.c

```
// enkripsi_main.c
//
// file berisi fungsi enkripsi_dekripsi yang akan digunakan oleh
enkripsi_main.c
//
// tugas besar mata kuliah bahasa assembly
// oleh :
// Bayu Rizaldhan rayes

// dites menggunakan Mandrake Linux 10.0
// dengan nasm 0.98.38-1mdk, gcc 3.3.2-6mdk
// teks editor Kate
//
// kompilasi & linking:
// nasm -f elf enkripsi.asm
// gcc -o enkripsiku enkripsi_main.c enkripsi.o
//
#include
#include

#define SIZE_MAX 1024

// prototype untuk fungsi assembly
// standard C calling convention untuk GCC (GNU C Compiler)
void asm_enkripsi_dekripsi( char *, const char * , char *, int) __attribute__((cdecl));

////////////////////
char teks[SIZE_MAX];           // teks sumber, diambil dari file
char teks_output[SIZE_MAX];    // teks hasil pemrosesan
```

```

FILE *filenya;
FILE *fileout;
char *nama_filenya;
char *nama_fileout;

// variabel2 flag untuk option dari getopt
int ada_d = 0;
int ada_e = 0;
int ada_c = 0;
int ada_v = 0;
int ada_r = 0;
int ada_w = 0;
// variabel parameter dari getopt
char param_code[SIZE_MAX];           // argumen option -c
char param_read[100];                 // argumen option -r
char param_write[100];                // argumen option -w

int i;
int proses_sukses = 0;               // flag proses sukses

//
// reset teks, dengan mode r=read, w=write
//
int reset_teks(char *namafilenya, char modenya[4])
{
    if ((filenya = fopen(namafilenya,modenya)) == NULL) {
        printf("filenya nggak ada tuh, bikin baru ya..");
        return 0;
    } else return 1;
}

//
// load teks dari file
//
void load_from_file(char *nama_filenya) {
    char c; // karakter satuan
    int i = 0;

    if (reset_teks(nama_filenya,"r")) {
        while ((c= fgetc(filenya))!=EOF) {
            teks[i] = c;
            i++;
        }
        fclose(filenya);
    }
}

//
// save teks_output ke file
//
void save_to_file(char *nama_filenya) {
    char c; // karakter satuan
    int i = 0;

    if (reset_teks(nama_filenya,"w")) {
        for (i=0;i<strlen(teks);i++)
            fputc(teks_output[i],filenya);
    }
}

```

```

        fclose(filenya);
    }
}

//
//  mencetak cara penggunaan program
//
void cetak_usage() {
    printf("Usage: enkripsiku [-dev] [-r nama_file] [-w nama_file] [-c
kode_enkripsi]\n\n");
    printf("daftar_parameter\n-e -> enkripsi\n-d -> dekripsi\n-v ->
tampilkan hasil proses\n\n");
    printf("option e dan d tdk boleh sekaligus, boleh tidak ada
keduanya\n\n");
    printf("contoh penggunaan:\nenkripsiku -r teksku.txt -c kunci -v
-e -w tekshasil.txt");
    printf("\nenkripsiku -r teksku.txt -d kunci -v -e -w
tekshasil.txt\n");
}
////////////////////////////////////

int main (int argc, char *argv[]) {
    int ret, opt_index = 0;

    struct option long_options[] = {
        { "decrypt",          0, NULL, 'd' },
        { "encrypt",          0, NULL, 'e' },
        { "code",             1, NULL, 'c' }, // parameter
        { "view",             0, NULL, 'v' },
        { "read",             1, NULL, 'r' }, // parameter
        { "write",            1, NULL, 'w' }, // parameter
        { 0, 0, 0, 0 }
    };

    // memproses argumen program
    while ((ret = getopt_long(argc, argv, "dec:vr:w:",
long_options,&opt_index)) != -1) {
        switch (ret) {
            case 'd':
                printf("melakukan dekripsi.....\n");
                ada_d = 1;
                break;
            case 'e':
                printf("melakukan enkripsi.....\n");
                ada_e = 1;
                break;
            case 'c':
                ada_c = 1;
                strcpy(param_code,optarg);
                break;
            case 'v':
                ada_v = 1;
                break;
            case 'r':
                ada_r = 1;
                strcpy(param_read,optarg);
                printf("file input = %s\n",param_read);

```

```

        break;
    case 'w':
        ada_w = 1;
        strcpy(param_write,optarg);
        printf("file output = %s\n",param_write);
        break;
    default:
        //tampilkan tata cara (usage)
        cetak_usage();
        exit (1);
}
}

if (ada_r) {
    //load_from_file("email2.txt");
    load_from_file(param_read);
    if (ada_c) {
        // lakukan enkripsi dekripsi
        if (ada_e == 1 && ada_d == 0) {
            asm_enkripsi_dekripsi( teks_output, teks,param_code,0); // enkripsi
            proses_sukses = 1;
        }
        else if (ada_d == 1 && ada_e == 0) {
            asm_enkripsi_dekripsi( teks_output, teks,param_code,1); // dekripsi
            proses_sukses = 1;
        }
        else if (ada_d == 1 && ada_e == 1)
            printf("tidak bisa melakukan enkripsi dan dekripsi sekaligus");
        else {
            asm_enkripsi_dekripsi( teks_output, teks,"",0);          // tidak ada -e
dan -d, cukup copy teks ke teks_output
            proses_sukses = 1;
        }
    }
    else {
        printf("parameter belum lengkap, minimal -r (read file) dan -c
(code)");
    }
}

else
    cetak_usage();

if (proses_sukses && ada_v) {
    // cetak hasil proses
    printf("\nHasil proses : \n\n%s\n", teks_output);
}

if (proses_sukses && ada_w) {
    // simpan ke file
    save_to_file(param_write);
}

printf("\n-----\n");
exit(0);
}

```


Penjelasan:

file enkripsi_main berisi fungsi utama yang memanfaatkan fungsi asm_enkripsi_dekripsi yang ada di file enkripsi.asm. Karena itu kedua file ini harus dilinking agar berjalan.

Pada file ini digunakan library getopt untuk memanfaatkan fungsi getopt_long. getopt_long berfungsi untuk memparse parameter dari argumen program sesuai dengan standar di Linux.

Ada 6 argumen option, yaitu

- -r : untuk membaca file input, option ini harus ada dan memiliki value.
- -w : untuk menulis hasil proses ke file.
- -e : untuk melakukan enkripsi
- -d : untuk melakukan dekripsi
- -v : untuk mencetak hasil proses ke layar
- -c : argumen value berisi kode enkrips

option -d dan -c tidak boleh ada dalam waktu bersamaan. Tetapi boleh tidak ada keduanya.

Pada dasarnya program enkripsi_main ini adalah sebagai antarmuka untuk proses enkripsi/dekripsi yang dilakukan oleh fungsi asm_enkripsi_dekripsi pada file enkripsi.asm. Untuk itu perlu disertakan prototype untuk fungsi assembly sesuai dengan standard c calling convention. Pada listing ini, digunakan standar kompiler GCC.

```
void asm_enkripsi_dekripsi( char *, const char * , char *, int) __attribute__((cdecl));
```

Penjelasan program bisa dilihat pada komentar baris pada listing program. Contoh penggunaan fungsi asm_enkripsi_dekripsi sebagai berikut:

```
asm_enkripsi_dekripsi( teks_output, teks,param_code,0);
```

artinya melakukan enkripsi (flag_enkripsi =0), pada string teks. Dan hasilnya dicopy ke teks_output.

Kompilasi dan Linking

Perintah berikut kan mengkompilasi file enkripsi.asm menjadi enkripsi.o.

```
$ nasm -f elf enkripsi.asm
```

Sedangkan mengkompilasi enkripsi_main.c sekaligus melinking dengan enkripsi.o digunakan sintaks berikut:

```
$ gcc -o enkripsiku enkripsi_main.c enkripsi.o
```

Sehingga didapatkan file executable bernama enkripsiku

Contoh pengguaan

Melakukan enkripsi file teksku.txt, dengan kode “abcde”, kemudian hasilnya ditampilkan ke layar dan disimpan pada file tekshasil.txt

```
$ ./enkripsiku -r teksku.txt -c abcde -v -e -w tekshasil.txt
```

Melakukan dekripsi file tekshasil.txt, hasil ditampilkan ke layar dan disimpan pada file tekshasil.txt

```
$ ./enkripsiku -r tekshasil.txt -c abcde -v -d -w tekshasil.txt
```

Screenshot

Keterangan: teks bergaris bawah adalah input dari pengguna.

```
[bayu@bluecat tugasbesarfinal]$ cat teksku.txt
Hallo ini teks bukan sekedar teks ya hehehehe
satu
dua
tiga
empat
lima
enam
tujuh
delapan
sembilan
sepuluh
angka:1234567890[bayu@bluecat tugasbesarfinal]$
```

```
[bayu@bluecat tugasbesarfinal]$ ./enkripsiku -r teksku.txt -c katakunci -v -e
-w tekshasil.txt
file input = teksku.txt
melakukan enkripsi.....
file output = tekshasil.txt
Hasil proses :
³ÀàÍÚ×ÑÒ ßÆßÔ×ãîÊñÔÛÌÐÛÎÖtÐÌçäÖÊîhÐÉÛÊÐáÄÝuuÃéÂué×ÊÊ
ÐÎäÂßÚÎÖauÆâÂØâØÓuÓkØÆ×ÖPÄ×
PÆäÃÖäîÑssÐÑéÍàÝxÄ×gÖÂ®´ç7£¤
-----
```

```
[bayu@bluecat tugasbesarfinal]$ cat tekshasil.txt
³ÀàÍÚ×ÑÒ ßÆßÔ×ãîÊñÔÛÌÐÛÎÖtÐÌçäÖÊîhÐÉÛÊÐáÄÝuuÃéÂué×ÊÊ
ÐÎäÂßÚÎÖauÆâÂØâØÓuÓkØÆ×ÖPÄ×
PÆäÃÖäîÑssÐÑéÍàÝxÄ×gÖÂ®´ç7£¤[bayu@bluecat tugasbesarfinal]$
[bayu@bluecat tugasbesarfinal]$ ./enkripsiku -r tekshasil.txt -c katakunci
-v -d -w tekskembali.txt
file input = tekshasil.txt
melakukan dekripsi.....
file output = tekskembali.txt
Hasil proses :
Hallo ini teks bukan sekedar teks ya hehehehe
satu
dua
tiga
empat
lima
```

```

enam
tujuh
delapan
sembilan
sepuluh
angka:1234567890
-----
[bayu@bluecat tugasbesarfinal]$ cat tekskembali.txt
Hallo ini teks bukan sekedar teks ya hehehehe
satu
dua
tiga
empat
lima
enam
tujuh
delapan
sembilan
sepuluh
angka:1234567890[bayu@bluecat tugasbesarfinal]$

```

Salah satu hal yang penting dalam komunikasi menggunakan computer untuk menjamin kerahasiaan data adalah enkripsi. Enkripsi adalah sebuah proses yang melakukan perubahan sebuah kode dari yang bisa dimengerti menjadi sebuah kode yang tidak bisa dimengerti (tidak terbaca). Enkripsi dapat diartikan sebagai kode atau chipper. Sebuah sistem pengkodean menggunakan suatu table atau kamus yang telah didefinisikan untuk mengganti kata dari informasi atau yang merupakan bagian dari informasi yang dikirim. Sebuah chipper menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (stream) bit dari sebuah pesan menjadi cryptogram yang tidak dimengerti (unitelligible). Karena teknik cipher merupakan suatu sistem yang telah siap untuk di automasi, maka teknik ini digunakan dalam sistem keamanan komputer dan network.

Pada bagian selanjutnya kita akan membahas berbagai macam teknik enkripsi yang biasa digunakan dalam sistem sekuriti dari sistem komputer dan network.

A. Enkripsi Konvensional.

Proses enkripsi ini dapat digambarkan sebagai berikut :

Plain teks -> Algoritma Enkripsi -> Cipher teks -> Algoritma Dekripsi -> Plain teks

User A || User B

|-----Kunci (Key)-----|

Gambar 1

Informasi asal yang dapat di mengerti di simbolkan oleh Plain teks, yang kemudian oleh algoritma Enkripsi diterjemahkan menjadi informasi yang tidak dapat untuk dimengerti yang disimbolkan dengan cipher teks. Proses enkripsi terdiri dari dua yaitu algoritma dan kunci. Kunci biasanya merupakan suatu string bit yang pendek yang mengontrol algoritma. Algoritma enkripsi akan menghasilkan hasil yang berbeda tergantung pada kunci yang digunakan. Mengubah kunci dari enkripsi akan mengubah output dari algortima enkripsi.

Sekali cipher teks telah dihasilkan, kemudian ditransmisikan. Pada bagian penerima selanjutnya cipher teks yang diterima diubah kembali ke plain teks dengan algoritma dan kunci yang sama.

Keamanan dari enkripsi konvensional bergantung pada beberapa faktor. Pertama algoritma enkripsi harus cukup kuat sehingga menjadikan sangat sulit untuk mendekripsi cipher teks dengan dasar cipher teks tersebut. Lebih jauh dari itu keamanan dari algoritma enkripsi konvensional bergantung pada kerahasiaan dari kuncinya bukan algoritmanya. Yaitu dengan asumsi bahwa adalah sangat tidak praktis untuk mendekripsikan informasi dengan dasar cipher teks dan pengetahuan tentang algoritma dekripsi / enkripsi. Atau dengan kata lain, kita tidak perlu menjaga kerahasiaan dari algoritma tetapi cukup dengan kerahasiaan kuncinya.

Manfaat dari konvensional enkripsi algoritma adalah kemudahan dalam penggunaan secara luas. Dengan kenyataan bahwa algoritma ini tidak perlu dijaga kerahasiaannya dengan maksud bahwa pembuat dapat dan mampu membuat suatu implementasi dalam bentuk chip dengan harga yang murah. Chips ini dapat tersedia secara luas dan disediakan pula untuk beberapa jenis produk. Dengan penggunaan dari enkripsi konvensional, prinsip keamanan adalah menjadi menjaga keamanan dari kunci.

Model enkripsi yang digunakan secara luas adalah model yang didasarkan pada data encryption standard (DES), yang diambil oleh Biro standart nasional US pada tahun 1977. Untuk DES data di enkripsi dalam 64 bit block dengan menggunakan 56 bit kunci. Dengan menggunakan kunci ini, 64 data input diubah dengan suatu urutan dari metode menjadi 64 bit output. Proses yang sama dengan kunci yang sama digunakan untuk mengubah kembali enkripsi.

B. Enkripsi Public-Key

Salah satu yang menjadi kesulitan utama dari enkripsi konvensional adalah perlunya untuk mendistribusikan kunci yang digunakan dalam keadaan aman. Sebuah cara yang tepat telah ditemukan untuk mengatasi kelemahan ini dengan suatu model enkripsi yang secara mengejutkan tidak memerlukan sebuah kunci untuk didistribusikan. Metode ini dikenal dengan nama enkripsi public-key dan pertama kali diperkenalkan pada tahun 1976.

Plain teks -> Algoritma Enkripsi -> Cipher teks -> Algoritma Dekripsi -> Plain teks

User A || User B

Private Key B —|

|—————Kunci (Key)—————|

Gambar 2

Algoritma tersebut seperti yang digambarkan pada gambar diatas. Untuk enkripsi konvensional, kunci yang digunakan pada prosen enkripsi dan dekripsi adalah sama. Tetapi ini bukanlah kondisi sesungguhnya yang diperlukan. Namun adalah dimungkinkan untuk membangun suatu algoritma yang menggunakan satu kunci untuk enkripsi dan pasangannya, kunci yang berbeda, untuk dekripsi. Lebih jauh lagi adalah mungkin untuk menciptakan suatu algoritma yang mana pengetahuan tentang algoritma enkripsi ditambah kunci enkripsi tidak cukup untuk menentukan kunci dekripsi. Sehingga teknik berikut ini akan dapat dilakukan :

1. Masing – masing dari sistem dalam network akan menciptakan sepasang kunci yang digunakan untuk enkripsi dan dekripsi dari informasi yang diterima.
2. Masing – masing dari sistem akan menerbitkan kunci enkripsinya (public key) dengan memasang dalam register umum atau file, sedang pasangannya tetap dijaga sebagai kunci pribadi (private key).
3. Jika A ingin mengisim pesan kepada B, maka A akan mengenkripsi pesannya dengan kunci publik dari B.
4. Ketika B menerima pesan dari A maka B akan menggunakan kunci privatenya untuk mendeskripsi pesan dari A.

Seperti yang kita lihat, public-key memecahkan masalah pendistribusian karena tidak diperlukan suatu kunci untuk didistribusikan. Semua partisipan mempunyai akses ke kunci publik (public key) dan kunci pribadi dihasilkan secara lokal oleh setiap partisipan sehingga tidak perlu untuk didistribusikan. Selama sistem mengontrol masing – masing private key dengan baik maka komunikasi menjadi komunikasi yang aman. Setiap sistem mengubah private key pasangannya public key akan menggantikan public key yang lama. Yang menjadi kelemahan dari metode enkripsi publik key adalah jika dibandingkan dengan metode enkripsi konvensional algoritma enkripsi ini mempunyai algoritma yang lebih kompleks. Sehingga untuk perbandingan ukuran dan harga dari hardware, metode publik key akan menghasilkan performance yang lebih rendah. Tabel berikut ini akan memperlihatkan berbagai aspek penting dari enkripsi konvensional dan public key.

Enkripsi Konvensional

Yang dibutuhkan untuk bekerja :

1. Algoritma yang sama dengan kunci yang sama dapat digunakan untuk proses dekripsi – enkripsi.
2. Pengirim dan penerima harus membagi algoritma dan kunci yang sama.

Yang dibutuhkan untuk keamanan :

1. Kunci harus dirahasiakan.
2. Adalah tidak mungkin atau sangat tidak praktis untuk menerjemahkan informasi yang telah dienkripsi.
3. Pengetahuan tentang algoritma dan sample dari kata yang terenkripsi tidak mencukupi untuk menentukan kunci.

Enkripsi Public Key

Yang dibutuhkan untuk bekerja :

1. Algoritma yang digunakan untuk enkripsi dan dekripsi dengan sepasang kunci, satu untuk enkripsi satu untuk dekripsi.
2. Pengirim dan penerima harus mempunyai sepasang kunci yang cocok.

Yang dibutuhkan untuk keamanan :

1. Salah satu dari kunci harus dirahasiakan.
2. Adalah tidak mungkin atau sangat tidak praktis untuk menerjemahkan informasi yang telah dienkripsi.
3. Pengetahuan tentang algoritma dan sample dari kata yang terenkripsi tidak mencukupi untuk menentukan kunci.

Referensi :

PC Assembly Language, Paul A. Carter, 2003