TWO NEW SECRET KEY CRYPTOSYSTEMS

Eurocrypt 1985, Linz, Austria

Henk Meijer & Selim Akl
Department of Computing & Information Science
Queen's University
Kingston, Ontario

1. Introduction

Since the Data Encryption Algorithm DES was accepted as a standard in 1977 [4], few new conventional cryptosystems have been proposed in the open literature [5]. However DES is not necessarily the most suitable encryption procedure for all applications. For example two people desiring to set up a private secure communication channel may not want to use a standardized encryption algorithm; or communicating parties may want to choose from a set of encryption algorithms, trading off speed against security. In this paper we propose two new conventional cryptosystems that are

- adaptable (parameters can be chosen to increase or decrease execution time and level of security),
- efficient (the algorithms are fast, even when implemented in a high level computer language),
- easy to program (both algorithm can be written in less than 100 lines) and
- conceptually simple.

The above properties make the systems attractive to users that do not have the time, expertise and/or money to install special hardware chips or to write long and complicated programs. It is hoped that the last property will increase the trust we can have in the security of the systems. Since no practical cryptosystem can be proven to be secure, we have to use encryption algorithms that we believe to be secure. By using only conceptually simple operations and transformations, we hope that weaknesses are easier to detect. And even if such

F. Pichler (Ed.): Advances in Cryptology - EUROCRYPT '85, LNCS 219, pp. 96-102, 1986.

weaknesses should exist, we claim that in some applications a system with known deficiencies is preferable to an apparently secure, but difficult to analyze cryptosystem.

2. Convential cryptosystem, based on permutations and multiplications.

This cryptosystem consists of three multiplication and two permutation stages. We will first describe the system and then examine its security properties.

2.1 The system

Let m be a message consisting of 2n bits. We write

$$m = \langle m_0, m_1 \rangle$$

where m_0 and m_1 are the n most significant and n least-significant bits of m, respectively. The encryption key k is a block of 3n bits; we can write $k = \langle k_0, k_1, k_2 \rangle$, where each k_i is a block of n bits. P is a permutation of size 2n. The encryption algorithm $E_k(m)$ can be stated as follows:

$E_k(m)$:
$\langle a_0, a_1 \rangle = \langle m_0 * k_0, m_1 * k_2 \rangle$
$\langle b_0, b_1 \rangle = P(\langle a_0, a_1 \rangle)$
$\langle c_0, c_1 \rangle = \langle b_0 * k_1, b_1 * k_1 \rangle$
$\langle d_0, d_1 \rangle = P(\langle c_0, c_1 \rangle)$
$\langle e_0, e_1 \rangle = \langle d_0 * k_2, d_1 * k_0 \rangle$
return $(\langle e_0, e_1 \rangle)$.

In the above algorithm, the operation * is defined by

$a * b = 2^n - 1$ if $a = 2^n - 1$ and $b > 0$

$= ab \bmod 2^n - 1$ otherwise.

If the permutation P is chosen such that $P = P^{-1}$
and if k_0, k_1 and k_2 are such that

$$\gcd(k_i, 2^n - 1) = 1 \text{ for } i = 0,1,2 ,$$

then we have

$$E_{\langle k_0, k_1, k_2 \rangle}(m) = c \text{ iff } E_{\langle k_2^{-1}, k_1^{-1}, k_0^{-1} \rangle}(c) = m,$$

where k_i^{-1} is the multiplicative inverse of k_i modulo $2^n - 1$.

2.2. Efficiency and implementation

The above algorithm can be implemented efficiently by using n-bit integers rather than arrays of length n. Since $2^n \equiv 1 \bmod 2^n-1$, an algorithm for multiplication modulo 2^n-1 can be written as a sequence of regular additions, while adding overflow bits to the least significant bits. For example, in the language C, using 32-bit unsigned integers, we can add modulo 2^n-1 with n=32 by

```
add (a,b) :  if (a+b) <  a return (a+b+1)
                         else return (a+b)
```

since overflow bits are automatically truncated. Or, in Pascal, with 32-bit signed integers and the largest positive integer max = $2^{31}-1$, we can add modulo $2^{31}-1$ by

```
add (a,b) :  if max-a < b then return (a-max+b)
                          else return (a+b).
```

Given an addition function, an algorithm for multiplication modulo 2^n-1 can be written as

```
multiply (a,b) :
     product = 0
     while b > 0 do
          if b is odd then product = add (product,a)
          right-shift (b)
          cyclic-left-shift (a)

     endwhile
     return (product).
```

Notice that the above algorithm returns 2^n-1 if a = 2^n-1 and b>0, as required for the encryption algorithm. The permutation step can be executed by a sequence of modulo reductions, integer divisions and additions, all with powers of 2. In C, this can be done with the standard shift operation. For example the following algorithm swaps bit i of integer a with bit j of integer b:

```
if ((a>>i)&01) != ((b>>j&01)
{       a xor= 01 << i; /* change bit i */
        b xor= 01 << j; /* change bit j */  }
```

In languages in which shift operations do not exist, integers can be mapped into arrays before being permuted, or bits can be swapped directly by code looking like:

$a_i = (a \text{ div } 2^i) \bmod 2$

$b_j = (b \text{ div } 2^j) \bmod 2$

$$\text{if } a_i = b_j \text{ then } a = a + (1 - 2a_i)\ 2^i$$
$$b = b + (1 - 2b_j)\ 2^j .$$

2.3. Analysis

We first note that 2 * x, where * is the operation introduction in section 2.1, is equal to a cyclic left shift of the n-bit integer x. So if cs(x) denotes the cyclic left shift of x, we have

$$2^i * x = cs^i(x).$$

From this we can see that the multiplication step of the encryption algorithm has the property

$$cs^i(a) * k = cs^i(a * k).$$

Therefore, in order to ensure that cyclic shifts will not be preserved under the encryption function, the permutation P has to be chosen such that for all (i,j) $\neq$ (0,0), there exist x,y, such that

$$P(< cs^i(x),\ cs^j(y)>) \neq$$
$$< cs^i(P(<x,y>)_0,\ cs^j(P(<x,y>)_1) >$$

where $P(<x,y>)_0$ and $P(<x,y>)_1$ denote the n most-significant and n least-significant bits of P(<x,y>) respectively.

For all permutations P and keys $k = <k_0,k_1,k_2>$ we have

$$E_k(\bar{m}) = E_{\bar{k}}(m) = \overline{E_k(m)} ,$$

where $\bar{x}$ denotes the bitwise complement of x. This can easily be seen from the fact that for all x with $0 <= x <= 2^n-1$,

$$\bar{x} = 2^n-1 - x.$$

This property enables a cryptanalyst to

(i) reduce a search of the keyspace by 50% in case of a known plaintext attack,

(ii) obtain a message-cyphertext pair $(\bar{m},\bar{c})$ for each known pair (m,c).

However, disadvantage (i) is not serious if the key space is sufficiently large and (ii) can be prevented by, for example, requiring that all messages end with a zero.

The encryption algorithm has good statistical properties, even if it is reduced to two multiplications and one permutation. In fact it can be proven that the multiplication step is complete, i.e. the function $f_k(.)$ defined by

$$f_k(x) = k * x$$

is complete [3] for all k with gcd $(k, 2^n-1) = 1$.

3. Secret-key cryptosystem and random-bit generator

The sender and receiver choose and agree on two n-bit (n = 64, say) vectors V and W. The pair (V,W) represents the secret key.

The sender and receiver also choose and agree on two n-integer vectors X and Y such that

$$X = x_1\ x_2\ \ldots\ x_n \qquad \text{where } 1 \le x_i \le i, \text{ and}$$
$$Y = y_1\ y_2\ \ldots\ y_n \qquad \text{where } 1 \le y_i \le i.$$

Each of X and Y is thus a permutation [2]. The pair (X,Y) is a parameter of the system which may - but needs not to - be kept secret.

Each plaintext message consists of n-bit vectors M_i, i = 1,2, ...,r.

3.1. Encryption

With every message to be transmitted a n-bit vector U which is a function of the date and time is created.

This vector is used by the sender to compute two n-bit vectors M_0 and C_0 as follows:

M_0 = middle-n bits of UxV (i.e. $M_0 = \lfloor (UxV \bmod 2^{3n/2})/2^{n/2} \rfloor$),

C_0 = middle-n bits of UxW (i.e. $C_0 = \lfloor (UxW \bmod 2^{3n/2})/2^{n/2} \rfloor$).

Let N_0 equal the reverse of the bit pattern for M_0, i.e. if $M_0 = m_1\ m_2\ \ldots\ m_n$, then $N_0 = m_n\ m_{n-1}\ \ldots\ m_2\ m_1$.

The message is now encrypted using the procedure below.

```
for i=1 to n do
        (M'i-1, Ni) = S(Mi-1 , Ni-1)
            C'i-1 = Ci-1 ⊕ M'i-1
              Ci = P(Mi-1 ⊕ C'i-1, Mi ⊕ C'i-1)
endfor.
```

The functions S and P are defined as follows. Let $K = k_1\ k_2\ \ldots\ k_n$ and $H = h_1\ h_2\ \ldots\ h_n$.

1) function S(K,H) :

$Q = K \times H$ {$Q = q_1 q_2 \cdots q_{2n}$, i.e. Q is a 2n-bit vector}

$K = q_{n/2+1} q_{n/2+2} \cdots q_{3n/2}$ {$K = \lfloor (K \times H \bmod 2^{3n/2}) / 2^{n/2} \rfloor$}

$H = q_{3n/2} q_{3n/2-1} \cdots q_{n/2+2} q_{n/2+1}$ {H is the reverse of K}

return (K,H).

2) function P(K,H) :

```
for i=n to 2 do
    if k_i = 0 then h_i <--> h_{x_i}
               else h_i <--> h_{y_i}
    endif
endfor
return H.
```

The sender now transmits U, C_1, C_2, ..., C_r to the receiver.

3.2. Decryption

The receiver goes through the same steps to compute M_0, C_0, N_0. Then he recovers M_1, M_2, ..., M_r using the following procedure.

```
for i=1 to r do
    (M'_{i-1}, N_i) = S(M_{i-1}, N_{i-1})
    C'_{i-1} = C_{i-1} ⊕ M'_{i-1}
    M_i = C'_{i-1} ⊕ P^{-1}(M_{i-1} ⊕ C'_{i-1}, C_i)        endfor.
```

The function P^{-1} (K,H) is the same as P(K,H) except that the for loop goes from i=2 to n.

4. Conclusions

Both systems introduced in this paper can easily be implemented. They withstand initial attempts to break them and possess no obvious statistical weaknesses [1,3]. More statistical and analytical validation will be done in the future. Notice that the second system is an example of a randomized encryption system, so if a message is encrypted twice under the same key, it will result in two different cyphertexts.

References

[1] H. Beker and F. Piper, Cipher Systems, John Wiley, 1982.

[2] D.E. Knuth, The Art of Computer Programming, Vol.2, Addison Wes-

ley, 1981.

[3] A.G. Konheim, Cryptography: a Primer, John Wiley, 1981.

[4] National Bureau of Standards, Data Encryption Standard, FIPS publication 46, U.S. Department of Commerce, January 1979.

[5] J.A. Thomas and J. Thersites, An infinite encryption system, Dr. Dobb's Journal, August 1984.