Jurnal Teknologi, bil. 26, Jun 1997 hlm. 57–63

PRIVATE KEY GENERATING METHOD OF A NEW PUBLIC KEY CIPHER SYSTEM

MD. RAFIQUL ISLAM
HARIHODIN SELAMAT & MOHD. NOOR MD. SAP
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
Jalan Semarak, 54100 Kuala Lumpur
Malaysia

Abstract. Public key cipher system was invented in order to solve the key management problem. Here we describe the generating process of private keys of a new public key cipher system based upon the diophantine equations proposed by Lin, Chang and Lee. Some algorithms are encoded to compute the keys. We also describe time complexity for computing the keys.

1 INTRODUCTION

Traditional cryptography is based on the sender and receiver of a message knowing and using the same secret key: the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message. This method is known as secret-key cryptography. The main problem is getting the sender and receiver to agree on the secret key without anyone else finding out. If they are in separate physical locations, they must trust a courier, or a phone system, or some other transmission system to not disclose the secret key. Anyone who overhears or intercepts the key in transit can later read all messages encrypted using that key. The generation, transmission and storage of keys are called key management; all cryptosystems must deal with key management issues. Secret-key cryptography often has difficulty providing secure key management. In 1976 Diffie and Hellman [1] proposed their pioneering idea of public key cryptosystem in order to solve key management problem. In the public key system, each person gets a pair of keys, called the public key and the private key. Each person's public key is published while the private key is kept secret. In this paper we describe the generating procedure of private keys of a new public key cipher system based upon the diophantine equations proposed by Lin, Chang and Lee [3].

The organization of this paper is as follows. Public key cryptosystem and diophantine equation are described in Section 2. The underlying conditions, DK-conditions to generate the private keys will appear in Section 3. Algorithms to compute keys are described in Section 4. We also discuss about experimental results in Section 5. Finally, the conclusion is given in Section 6.

2 PUBLIC KEY CRYPTOSYSTEM AND DIOPHANTINE EQUATION

In a public key cryptosystem, each user U uses the encryption algorithm $E(K_p, P)$ and decryption algorithm $D(K_r, C)$, where K_p is the public key, K_r is the private key of U, P

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX and C are plaintext and ciphertext respectively. Each user publishes his encryption key by putting it on a public directory, while the decryption key is kept secret by himself. The need for the sender and receiver to share secret information is eliminated: all communications involve only public keys, and no private key is ever transmitted or shared. No longer is it necessary to trust some communications channel to be secure against eavesdropping or betrayal. Anyone can send a confidential message just using public information, but it can only be decrypted with a private key that is in the sole possession of the intended recipient. Furthermore, public-key cryptography can be used for authentication (digital signatures) as well as for privacy (encryption). Here is how it works for encryption: Suppose that user X wants to send a message M to user Y. First, X finds the public encryption key, namely K_{py} for Y from the public directory. Then X encrypts the message M to C by $C = E(K_{py}, M)$ and sends C to Y. On receiving C, Y can decrypt it by computing $M = D(K_{ry}, C)$ and can read it. Since K_{ry} is private for Y, no one else can perform this decryption process. Any one can send an encrypted message to Y but only Y can read it. Clearly, one requirement is that no one can figure out the private key from the corresponding public key.

In 1995 Lin, Chang and Lee [3] proposed a new public key cipher system based upon the Diophantine equations. In general, a diophantine equation [3] is defined as follows: We are given a polynomial equation $f(x_1, x_2, ..., x_n) = 0$ with integer coefficients and we are asked to find rational or integral solutions. For instance, consider the following equation [2]:

$$3x_1 + 4x_2 + 7x_3 + 5x_4 = 78.$$

The above equation is a diophantine equation if we have to find a non-negative solution for this equation. Another example of a diophantine equation is:

$$3x_1^3 + 4x_1x_2x_3 + 5x_4 = 105.$$

Diophantine equations are usually hard to solve.

To generate private keys of Lin, Chang and Lee's public key cipher system we should follow DK-conditions. We will describe DK-conditions in the next section.

3 DK-CONDITIONS

According to Lin-Chang-Lee's block cipher system private keys $(q_1, k_1), (q_2, k_2), ..., (q_n, k_n)$ must be chosen such that some specified conditions hold. Let w be some positive integer and the domain D be a set of positive integers in the range $[0, w]$. Let $w = 2^b - 1$, where b is some positive integer. Assume that a message M, is sent with length nb bits broken up into n pieces of submessages, namely $m_1, m_2, ..., m_n$. Each submessage is of length b bits. In other words, each submessage can be represented by a decimal number m_i and m_i in D. Suppose that n pairs of integers $(q_1, k_1), (q_2, k_2), ...$ and (q_n, k_n) are chosen such that the following conditions hold:

(1) q_i's are pairwise relative primes; i.e., $gcd(q_i, q_j) = 1$ for $i \neq j$, where gcd denotes greatest common divisor.
(2) $k_i > w$ for $i = 1, 2, ..., n$.
(3) $q_i > k_i w(q_i \bmod k_i)$ and $q_i \bmod k_i \neq 0$ for $i = 1, 2, ..., n$.

These n integer pairs (q_i, k_i)'s will be kept secret and used to decrypt messages. For convenience, the above three conditions are named the DK-conditions [3], since they are used as deciphering keys. In this paper we will deduce required algorithms to determine decryption keys with respect to DK-conditions.

4 ALGORITHMS TO COMPUTE DECRYPTION KEYS

In this section we describe the algorithms which are required to compute decryption keys according to the DK-conditions stated in previous section. From condition 1 q_i's must be pairwise relative primes i.e., $gcd(q_i, q_j) = 1$ for $i \neq j$. That means, to implement this condition we need algorithm to find out gcd of two numbers. For this we select extended Euclid's algorithm [7]. The algorithm is as follows.

Algorithm 4.1- *(Extended Euclid's algorithm) to determine gcd for two numbers*

Given two positive integers m and n, we compute their greatest common divisor d and two integers a and b, such that $d = am + bn$.

Step 1: Set $a_1 = 1, b = 1, a = 0, b_1 = 0, c = m, d = n$;
Step 2: Compute $q = quotient\ (c \div d)$,
$r = remainder\ (c \div d)$;
Step 3: While $r \neq 0$ do
begin
Set $c = d, d = r, t = a_1, a_1 = a, a = t - qa$,
$t = b_1, b_1 = b, b = t - qb$,
end;
Step 4: Compute $d = am + bn$, if $r = 0$;

Now we have to pick q_i's. To compute q_i's we must observe conditions 2 and 3 of DK-conditions. From condition 3 we notice that $q_i \bmod k_i \neq 0$ that means $\min\{q_i \bmod k_i\} = 1$, let $q_i \bmod k_i = 1$, then from condition 3 we get $q_i > k_i w$. We name this condition as critical condition, because q_i must be always greater than $k_i w$, otherwise condition 3 will not be satisfied. On the other hand from condition 2, we have $k_i > w$. Then we can write $\min\{k_i\} = w + 1$. Let $k_i = w + 1$, then the critical condition stands $q_i > (w + 1)w$, where $w = 2^b - 1$ and b is the desired block. Suppose that $b = 8$, then $w = 255$. Thus the value of $q_i > (255 + 1) \times 255 = 256 \times 255 = 65280$, which implies that q_i must be greater than 65280 for $b = 8$. Therefore we will take q_i's which will be greater than $w(w+1)$ and pairwise relatively primes. To determine pairwise relative primes we take an odd number, m_1 which is greater than $w(w + 1)$ and then take another odd number, m_2 by adding 2 to m_1. After this we check whether these two numbers are relatively primes or not. If these two numbers are relatively primes then take another odd number, m_3 by adding 2 to m_2. Now we have to determine whether m_3 is relatively prime to both m_1 and m_2. If the answer is yes, then we have three pairwise relative primes m_1, m_2 and m_3. Let us take a number, m_4 by adding 2 to m_3. Now if m_4 is not relatively prime to any number which are selected before, the m_4 will not be selected and will take another number m_5 by adding 2 to m_4. If m_5 is relatively prime to m_1, m_2 and m_3, then we will have four pairwise relative primes m_1, m_2, m_3, m_5 and so on. The algorithm by which we can compute pairwise relative prime numbers is as follows.

Algorithm 4.2-*Pairwise relative primes selecting algorithm*
Here we will select pairwise relative prime numbers according to DK-conditions.
Step 1: Input block size, b;
Step 2: Compute $w = 2^b - 1$;
Step 3: Compute $x = w(w + 1)$;
Step 4: Pick a number $q > x$;

Step 5: Take a blank array $x[i], 1 \leq i \leq n$;
Step 6: Set $p = 1, x[p] = q$;
Step 7: While $p < n$ do

```
begin
    Set c = 0, q = q + 2;
        p = p + 1, x[p] = q
        i = 1,
    While c = 0 and i < p do
        begin
            Set i = i + 1,
            Compute d = gcd(x[i - 1], x[p])
            If d ≠ 1, then set c = 1,
        end,
    If c = 1, then set p = p - 1,
end;
```

Step 8: Output array $x[i], 1 \leq i \leq n$;

Now we will choose k_i's according to DK-conditions. k_i's must be selected with respect to conditions 2 and 3. The algorithm to choose k_i's is as follows.

Algorithm 4.3-*Algorithm for Choosing k_i's*
Step 1: Set $k = w, j = 1$;
Step 2: Input the array $x[i]$ output from algorithm 4.2, $1 \leq i \leq n$;
Step 3: Take a blank array $k[i], 1 \leq i \leq n$;
Step 4: While $j \leq n$ do

```
begin
    Set k = k + 1, i = j, ok = 0,
    Compute y = kw,
    While ok = 0 and i ≤ n do
        begin
            Compute r = remainder (x[i] ÷ k),
            If r ≠ 0, then
                begin
                    Compute z = yr,
                    if x[i] > z, then
                        begin
                            Set ok = 1, t = 0,
                                t = x[j], x[j] = x[i],
                                x[i] = t, k[j] = k,
                                j = j + 1,
                        end,
                end,
            i = i + 1,
        end,
end;
```

Step 5: Output $k[i]$ and rearranged $x[i], 1 \leq i \leq n$;

5 EXPERIMENTAL RESULTS

The time complexity needed to compute q_i's is proportional to n^2 as n increases and the time required to choose k_i's is proportional to n [3]. Now we draw the graph with execution times to compute q_i's that are taken for different number of data. FujitsuICL Pentium base Personal Computer is used to take execution time. From the graph of figure 1 we notice that the time for computing q_i's increases exponentially and from the graph of figure 2 it is clear that the time for computing k_i's increases as the number of k_i's increase.

6 CONCLUSION

Here we describe the generating procedure of private keys of a new public key cipher system based upon diophantine equations proposed by Lin, Chang and Lee. We also give a brief description of the public key cipher system and diophantine equation. Underlying conditions to compute the keys are also discussed. Some algorithms are encoded for computing the keys. We plot here the graph to see the trend of time to compute the keys. The time to compute q_i's increases exponentially, but the time to choose k_i's increases as the number of data increases. In the Appendix A we include 100 key-pairs that are selected during computation.

Here q_i's are pairwise relatively primes and as we know pairwise relatively prime numbers are widely used in cryptographic applications. But we compute q_i's and k_i's according to the conditions called DK-conditions. The experiment may be extended for computing very large pairwise relatively primes as well as their associate numbers, k_i's.

No. of Data	t(q)
100	28
500	725
1000	3075
2000	12468
3000	28243
4000	51108
5000	81603
6000	120058
7000	166074
8000	219488

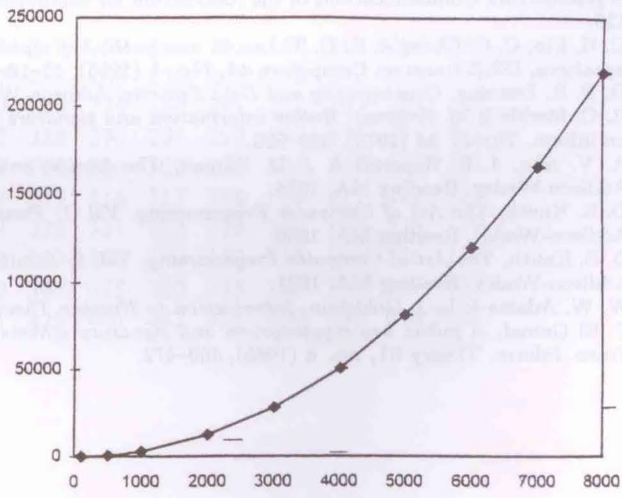

Fig. 1 Graph of execution time, $t(q)$ to compute q_i's (Here $t(q)$ is taken in millisecond)

No.of Data	t(k)
100	50
500	80
1000	220
2000	1040
3000	1060
4000	1150
5000	1920
6000	2080
7000	4070
8000	5430

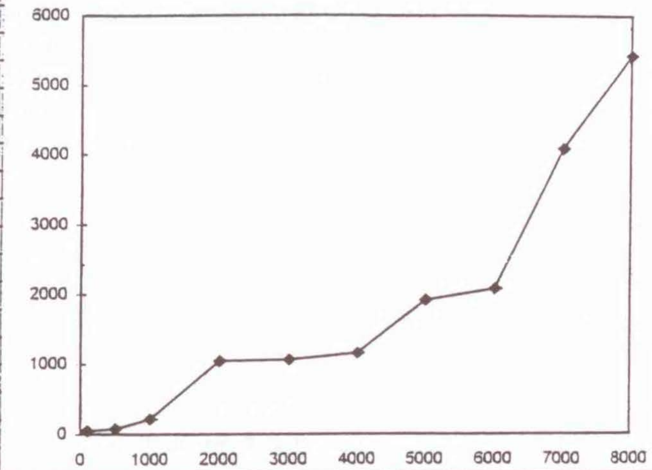

Fig. 2 Graph of execution time, $t(k)$ to choose k_i's (Here $t(k)$ is taken in microsecond)

REFERENCES

[1] W. Diffie & M. Hellman, *New directions in cryptography*, IEEE Trans. Inform. Theory **22** (1976), 644–654.

[2] R. L. Rivest, A. Shamir & L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the Association for computing Machinery **21, No. 2** (1978), 120–126.

[3] C. H. Lin, C. C. Chang & R. C. T. Lee, *A new public-key cipher system based upon the diophantine equations*, IEEE trans. on Computers **44, No. 1** (1995), 13–19.

[4] D. E. R. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading MA, 1982.

[5] R. C. Merkle & M. Hellman, *Hiding information and signature in trap-door knapsacks*, IEEE Trans. on inform. Theory **24** (1978), 525–530.

[6] A. V. Aho, J. E. Hopcroft & J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading MA, 1974.

[7] D. E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms, second edition*, Addison-Wesley, Reading MA, 1980.

[8] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, second edition*, Addison-Wesley, Reading MA, 1981.

[9] W. W. Adams & L. J. Goldstein, *Introduction to Number Theory*, Prentice-Hall, 1976.

[10] T. El Gamal, *A public key cryptosystem and signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory **31, no. 4** (1985), 469–472.

Appendix A

100 Pairwise relatively primes(q_i's)

1000193	1000001	1000009	1000007	1000231	1000157	1000057	1000199
1000033	1000117	1000169	1000183	1000177	1000151	1000081	999999
1000421	1000003	1000379	1000187	1000507	1000249	1000253	1000499
1000171	1000361	1000261	1000409	1000537	1000067	1000429	1000487
1000513	1000211	1000459	1000393	1000015	1000189	1000063	1000303
1000091	1000159	1000213	1000469	1000403	1000019	1000387	1000243
1000061	1000273	1000357	1000411	1000451	1000457	1000453	1000121
1000327	1000561	1000141	1000333	1000031	1000621	1000079	1000291
1000651	1000343	1000667	1000313	1000283	1000397	1000289	1000697
1000541	1000639	1000589	1000309	1000039	1000037	1000367	1000679
1000613	1000543	1000577	1000133	1000547	1000619	1000267	1000579
1000049	1000099	1000423	1000381	1000669	1000631	1000691	1000553
1000093	1000663	1000609	1000427				

100 data for k_i's which follows DK-condition with above 100 q_i's

256	257	258	259	260	261	262	263	264	265
266	267	268	269	270	271	272	273	274	275
276	277	278	279	280	281	282	283	284	285
286	287	288	290	291	292	293	294	295	297
298	299	300	304	305	306	308	309	310	312
313	314	315	316	317	318	320	321	322	323
325	327	328	331	332	333	334	335	336	339
340	342	343	346	349	350	352	355	356	357
358	359	362	375	380	385	388	391	397	401
408	411	428	441	442	455	490	491	615	3761