A switching closure test to analyze cryptosystems

(Extended abstract)

Hikaru Morita   Kazuo Ohta   Shoji Miyaguchi

NTT Laboratories

NTT R&D Center (Room 309A), 1-2356 Take Yokosuka Kanagawa 238-03 Japan

Phone: +81-468-59-2514   FAX: +81-468-59-3858   E-mail: morita@sucaba.ntt.jp

Abstract

The closure test MCT (meet-in-the-middle closure test) was introduced to analyze the algebraic properties of cryptosystems [KaRiSh]. Since MCT needs a large amount of memory, it is hard to implement with an ordinary meet-in-the-middle method. As a feasible version of MCT, this paper presents a switching closure test SCT which based on a new memoryless meet-in-the-middle method. To achieve the memoryless method, appropriate techniques, such as expansion of cycling detection methods for one function into a method for two functions and an efficient intersection search method that uses only a small amount of memory, are used in an extremely effective manner.

1. Introduction

There are two approaches in cryptography to analyzing the security levels of cryptosystems. The first is to develop unique attacks for each cryptosystem. This utilizes the idiosyncrasies of each cryptosystem. The second approach is to analyze cryptosystems to find their features such as algebraic or statistical structures; the cryptosystems are regarded as black-box functions. The latter approach is important because you can accumulate knowledge of cryptosystems with a common framework.

Even if you find the statistical structure of a cryptosystem, you do not automatically know of a useful method to attack the cryptosystem. To find the algebraic structure of cryptosystems in general, Kaliski et al. [KaRiSh] proposed two closure tests: CCT (cycling closure test) and MCT (meet-in-the-middle closure test). These tests can detect features such as algebraic closure. Moreover, Kaliski et al. also proposed two cryptattack methods based on the algebraic features. CCT experiments performed by Kaliski et al. detected that DES is not closed.

Generally, both CCT and MCT can determine if a cryptosystem is closed or not. If a cryptosystem is closed, they give the same results "Fail", which means the cryptosystem might be breakable. However, if a cryptosystem is not closed, you cannot be sure that they will give the same results "Pass" because it isn't known whether CCT and MCT can detect the same algebraic structure or not. Our interest was that MCT might prove to be a fertile avenue for cryptographic research. MCT offers the possibility of extracting information from a not-closed cryptosystem that would allow the cryptosystem to be broken. Moreover, a cryptosystem may fail under MCT even if it passes CCT.

CCT needs (operation) time of $O\left(\sqrt{|K|}\right)$ and a small amount of (memory) space where $|K|$ is the size of key space K of the cryptosystem. However, MCT needs space and time of $O\left(\sqrt{|K|}\right)$. No feasible method to achieve the meet-in-the-middle strategy, which MCT uses, has been proposed because the strategy needs a lot of memory.

In this paper, we present a switching closure test (SCT) as a feasible version of MCT by using a memoryless method. Up to now, though the memoryless method has been applied to collision search [QuDe], this paper applies it for the first time to a closure test for cryptosystems. Section 2 gives the background of our research. Section 3 shows the procedures of SCT. Section 4 describes a feasibility study that confirms SCT performance. The paper is concluded in Section 5.

2. Background

2.1 What is a closure structure?

The below explanation of closure structure follows [KaRiSh].

Let us denote a cryptosystem $\Pi = (K, M, C, E)$, where K, M, C are key space, message space, and ciphertext space, respectively and $E: K \times M \to C$ is a transformation such that, for $k \in K$, the mapping $E_k = E(k,\cdot)$ is invertible.

[Definition 1]

$\Pi = (K, M, C, E)$ is closed if and only if its set of encryption transformations, $\{E_k | k \in K\}$ is closed under functional composition, i.e., for every two keys $i, j \in K$ there exists a key $k \in K$ such that $E_j E_i = E_k$.

Kaliski et al. presented two closure tests, MCT and CCT, for determining if a cryptosystem is closed or not. If the cryptosystem passes either test, then it is not closed with high probability. Though CCT was implemented in [KaRiSh], no feasible method has been presented to achieve MCT, because it uses $O\left(\sqrt{|K|}\right)$ operation time and $O\left(\sqrt{|K|}\right)$ memory space.

MCT is based on the following property which is satisfied by closed cryptosystems.

[Property 1]

Let $\Pi = (K, M, C, E)$ satisfy $M = C$, the number of $\{E_k | k \in K\}$ be m, and $k \in K$ be any key. If $\Pi = (K, M, M, E)$ is closed, then there are exactly m pairs of keys (i, j) such that $E_j E_i = E_k$.

2.2 Meet-in-the-middle closure test (MCT) [KaRiSh]

MCT works as follows: given $\Pi = (K, M, M, E)$, pick any key $k \in K$ and search for keys $a, b \in K$ such that $E_k = E_b E_a$. To search for them, you use a standard "meet-in-the-middle" strategy. The MCT procedure is shown in Fig. 2.1.

Suppose that a pair (p, c) is given where $c = E_k(p)$. If Π is closed, then property 1 above means that there are m pairs (i, j) among all m^2 key pairs satisfying $E_j E_i = E_k$.

Therefore, the probability of a match between $\{E_i(p)\}$ and $\{E_j^{-1}(c)\}$ is $r_1 r_2 \frac{m}{m^2}$, where r_1 is the number of elements of $\{E_i(p)\}$ and r_2 is the number of elements of $\{E_j^{-1}(c)\}$. If Π is closed, $m = O(|K|)$, and $r_1 r_2 = O(m)$, then we can find a pair (i, j) such that $E_j E_i = E_k$ with high probability.

If there is no match in MCT under the condition $r_1 r_2 = O(m)$, you can conclude that the cryptosystem is not closed with confidence. Note that MCT requires r_1 operations of $E_i(p)$, r_2 operations of $E_j^{-1}(c)$ and at least r_1 or r_2 memory space. If $r_1 = r_2$, which is optimal from the standpoint of operation number, $r_1 = r_2 = O(\sqrt{m})$.

```
input: a cryptosystem $\Pi = (K, M, M, E)$ and integer control parameters $r_1$, $r_2$, $l$.
Step 1. Pick $k \in K$ and $p_1, p_2, \cdots, p_l \in M$ at random.
        For $i = 1$ to $l$, compute $c_i = E(k, p_i)$.
        Let $p = p_1$ and $c = c_1$.
Step 2. For $i = 1$ to $r_1$ and $j = 1$ to $r_2$, select $a_i, b_j \in K$ at random
        and compute $x_i = E(a_i, p)$ and $y_j = E^{-1}(b_j, c)$.
Step 3. Sort triples $(x_i, a_i, "A")$ for $1 \le i \le r_1$
        and $(y_j, b_j, "B")$ for $1 \le j \le r_2$ on the first components.
Step 4. For each "match" $x_i = y_j$ with $1 \le i \le r_1$ and $1 \le j \le r_2$,
        if $E_k = E_{b_j} E_{a_i}$, then return ("Match found").
        (To test if $E_k = E_{b_j} E_{a_i}$, statistically verify that $c_h = E(b_j, E(a_i, p_h))$ for all $1 \le h \le l$.)
Step 5. return("No match found").
```

Fig. 2.1 Meet-in-the-middle closure test (MCT) procedure

2.3 Related Work

The meet-in-the-middle strategy is effective in cryptography [DiHe, QuDe].

In [DiHe], this strategy is used to find a key pair (k_1, k_2) from a pair (p, c) in the double-encryption cryptosystem, where $c = E(k_2, E(k_1, p))$. If an exhaustive key search is used, $O(|K|^2)$ operations are needed. If a meet-in-the-middle strategy is applied, only $O(|K|)$ operations and $O(|K|)$ memory space are required.

Quisquater et al. developed the meet-in-the-middle technique with $O(\sqrt{|K|})$ operations without memory. The technique was applied to find collisions of hash functions [QuDe].

Collision is not a problem for a hash function since collision is only based on the hash-function's randomness. On the other hand, if a pair of keys (k_1, k_2) is found for a cryptosystem by using MCT, it implies that the cryptosystem is vulnerable.

3. Switching closure test (SCT) procedure

This section introduces the memoryless procedure needed to make the ordinary meet-in-the-middle method feasible and shows how to apply the procedure to a closure test.

3.1 Memoryless meet-in-the-middle procedure

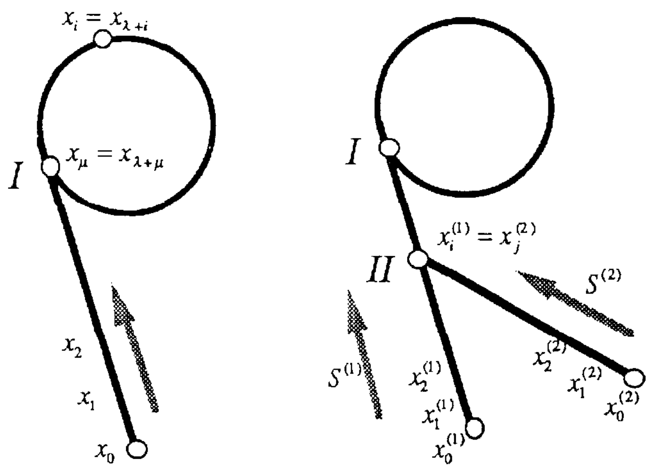
We propose the memoryless meet-in-the-middle procedure in Fig. 3.1. In the procedure, the period-length search required by Steps 2 and 6, is given in Paragraph (1). The intersection-point search of Steps 3 and 7 is described in Paragraph (2). If the random function h is defined as shown in Paragraph (3), the intersection-point search can be expanded into the collision-point search for a meet-in-the-middle method.

```
input: a function $h(x)$ switching $f(x)$ or $g(x)$ and an integer control parameter $t$.
Step 1. Pick $x_0^{(i)} \in M$ for $i = 1,2,\cdots,t$ at random.
Step 2. $i = 1$.
        Search period - length $\lambda$ for sequence $S^{(i)}$ started from $x_0^{(i)}$.
        (where $S^{(i)} = \{x_0^{(i)}, x_1^{(i)}, \cdots\}$)
Step 3. Search for an intersection point in the sequence $S^{(i)}$.
Step 4. If the intersection point found is the collision point in $S^{(i)}$ where $f(x)$ and $g(y)$ meet
        then return$(x, y)$.
Step 5. $i = i + 1$.
Step 6. Search period - length $\lambda$ for the sequence $S^{(i)}$ started from $x_0^{(i)}$
        and check that the new sequence $S^{(i)}$ meets the previous sequnces
        $\{S^{(1)}, S^{(2)}, S^{(3)}, \cdots, S^{(i-1)}\}$.
Step 7. Search for a new intersection point in $S^{(i)}$ or between $S^{(i)}$ and $S^{(j)}$ $(1 \le j \le i-1)$.
        If the intersection point found is the collision point in $S^{(i)}$ or between $S^{(i)}$ and $S^{(j)}$
        where $f(x)$ and $g(y)$ meet, then return$(x, y)$.
Step 8. Go to Step 5.
```

Fig. 3.1 Memoryless meet-in-the-middle procedure

(1) Period-length search [Kn, SeSzYa]

If the random function h generates values x, the "Birthday Paradox" states that the random sequence $S = \{x_0, x_1, \cdots\}$ has a period whose length is $O\left(\sqrt{|X|}\right)$, where $x_{i+1} = h(x_i)$ and $|X|$ shows the size of X space. When the sequence $\{x_0, x_1, \cdots\}$ is plotted on directed graphs, it can be drawn as shown in Fig.3.2 (a).



(a) Original sequence contains Point I **(b) Two sequences meet at Point II**

Fig. 3.2 Sequences on directed graphs

Methods to find period-length λ have been developed in the design of a random generator [Kn, SeSzYa]. The methods compare two values x_i and x_j $(i \neq j)$ to determine $x_i = x_{\lambda+i}$. We based our memoryless meet-in-the-middle procedure on these methods.

(2) Intersection search

There are two kind of intersection points as shown in Fig.3.2. Point I is represented by

$$x_\mu = h(x_{\mu-1}), \quad x_{\lambda+\mu} = h(x_{\lambda+\mu-1}),$$
$$\text{where } x_\mu = x_{\lambda+\mu}, \quad x_{\mu-1} \neq x_{\lambda+\mu-1}.$$

Point I is shown as the junction between a loop and a leader in Fig.3.2 (a). Point II in Fig.3.2 (b) is represented by $x_i^{(1)} = h(x_{i-1}^{(1)})$ and $x_j^{(2)} = h(x_{j-1}^{(2)})$ where $x_i^{(1)} = x_j^{(2)}$ and $x_{i-1}^{(1)} \neq x_{j-1}^{(2)}$ where the superscripts distinguish values of Sequences $S^{(1)}$ and $S^{(2)}$. To find Point II, sequences starting from two different initial values $x_0^{(1)}$ and $x_0^{(2)}$ are needed. It is known that graphs of the random function h tend to have one giant loop (or component) and a few large leaders (or trees) [FlOd].

In brief, to find Point I, after the period-length λ is determined, pairs, (x_0, x_λ), $(x_1, x_{\lambda+1})$, $\cdots$, $(x_i, x_{\lambda+i})$ are compared until $x_\mu = x_{\lambda+\mu}$ is found. However, this procedure is inefficient because you need μ comparisons and 2μ operations of the function h. Therefore, to overcome this inefficiency, we have developed techniques that require only $O(\log_2(\mu))$ operations. The techniques will be presented in the full paper version. The same approach is used to find Point II.

(3) Switching function

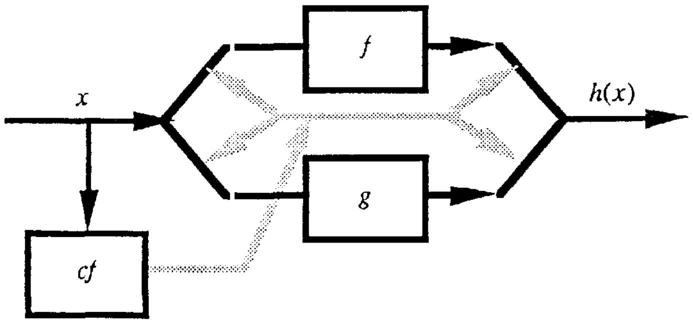


Fig. 3.3 Block diagram of *h(x)* function

To find the match between $f(x)$ and $g(x)$ functions as the collision of the meet-in-the-middle procedure, let's define $h(x)$ as a switching-type function of $f(x)$ or $g(x)$ conditionally defined as

$$h(x) = \begin{cases} f(x) & \text{if } cf(x) \text{ is true,} \\ g(x) & \text{if } cf(x) \text{ is false,} \end{cases}$$

where $cf(x)$ is a conditional function which generates true or false with 50-% probability for each value x.

The intersection points found in Paragraph (2) are expected to contain collision points between f and f with 25 %, between g and g with 25 %, and between f and g with 50 %. The intersection points between f and g mean the collision points between two different functions f and g. Consequently, the meet-in-the-middle procedure succeeds for two different functions.

3.2 Closure test for cryptography functions

(1) How to apply to a closure test

The SCT procedure is presented in Fig. 3.4. If SCT detects an x and y pair that satisfies $E(x,p) = E^{-1}(y,c)$ for any (p,c), the cryptosystem fails in SCT. On the other hand, if SCT doesn't detect any x and y pair several times, the cryptosystem passes.

```
input:  functions $f(x) \equiv E(x,p)$ and $g(x) \equiv E^{-1}(x,c)$ and integer control parameter $l,t$.
Step 1. Pick $k \in K$ and $p_1, p_2, \cdots, p_l \in M$ at random.
        For $i = 1$ to $l$, compute $c_i = E(k, p_i)$.
        Set $p = p_1, c = c_1$.
Step 2. Call Memoryless Meet - in - the - middle procedure $(f, g, t)$,
        then get $(x, y)$ at the collision point.
Step 3. If $E_k = E_y E_x$, then return ("Match found")
        (To test if $E_k = E_y E_x$, statistically verify $c_h = E(y, E(x, p_h))$ for all $1 \le h \le l$.)
Step 4. return("No match found").
```

Fig. 3.4 Switching closure test (SCT) procedure

Since we want to find key pairs (k_1, k_2) instead of the real secret key k for any plaintext and ciphertext pair (p,c) where $c = E(k_2, E(k_1, p))$ and $c = E(k,p)$, a meet-in-the-middle method is used to find pairs of x and y which make $E(x,p) = E^{-1}(y,c)$. In the SCT procedure in Fig. 3.4, SCT verifies that $E(x,p) = E^{-1}(y,c)$ for any (p,c). To simplify the explanation, we omit functions F_E and F_D from Fig. 3.4. More precisely, the f and g functions are defined as shown in Fig. 3.5.

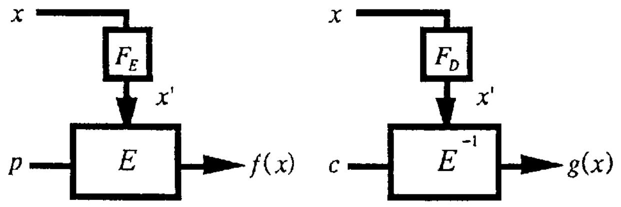


Fig. 3.5 Definitions of f and g

(2) Strategies & probability to detect closure structure

We have devised two strategies to increase the intersection points in SCT after the first period-length and intersection-point searches. In Strategy #1, an intersection search is carried out from another start point for the same plaintext and ciphertext pair. Strategy #2 is to carry out an intersection search for another plaintext and ciphertext pair. Therefore, Strategy #2 employs both period-length and intersection searches and always finds the Point I in Fig. 3.2.

If a cryptosystem is closed, their probability of success can be assessed from the analysis given in Section 2.2. If, finally, r_1 values of f and r_2 values of g have been generated by the h function, their probability is approximately given by

$$1-\exp\left(-\frac{r_1 r_2}{m}\right),$$

where there are m kinds of independent transformations. In Strategy #1, if r_1' values of f and r_2' values of g newly appear, its probability becomes

$$1-\exp\left(-\frac{(r_1+r_1')(r_2+r_2')}{m}\right),$$

because Strategy #1 replaces r_1 and r_2 with (r_1+r_1') and (r_2+r_2'), respectively. On the other hand, in Strategy #2, if r_1' and r_2' values also newly appear, its probability becomes

$$1-\exp\left(-\frac{r_1 r_2+r_1' r_2'}{m}\right),$$

because the first and second searches are independent.

Strategy #2 needs more operations than #1. Strategy #1 has a higher probability of success than #2 if r_1' and r_2' have the same values for both strategies. However, Strategy #2 can be efficiently implemented on parallel processing hardware. Therefore, Strategy #2 is better for hardware implementations.

4. Feasibility study

This section explains the SCT's detectability of closure property for both symmetric and asymmetric ciphers. Moreover, we will show that SCT yields a known-plaintext attack against closed cryptosystems. In the case of a symmetric cipher, if you find the key pair (k_1,k_2) from (p,c) using SCT, where $c=E(k,p)$ and $E_k=E_{k_2}E_{k_1}$, then you can decipher any ciphertext c' by $E^{-1}(k_1,E^{-1}(k_2,c'))$ without a secret key k. However, is it impossible to attack asymmetric cryptosystems using SCT? In Section 4.2 we attack a small model of an RSA cryptosystem.

4.1 Caesar cipher as symmetric cipher

The definition of the Caesar cipher is given as shown;

$$E(k,p)=p+k \bmod n,$$
$$E^{-1}(k,c)=c-k \bmod n.$$

We define functions, f, g, cf, respectively as;

$$f(x) \quad = p+x \bmod n,$$
$$g(x) \quad = c-x \bmod n,$$
$$cf(x) \quad = (\text{true if } x \text{ is "odd", false if } x \text{ is "even"}).$$

At first, $(p,c)=(5,2)$, and $n=11$ are given. Though a secret key $k=8$ is kept secret, Fig. 4.1 is drawn by SCT. If you start from $x_0^{(1)}=1$ or $x_0^{(2)}=0$, you find a loop with $\lambda=3$ or $\lambda=2$ without the collision point. If you start from $x_0^{(3)}=4$, you find a loop with $\lambda=4$ with the collision point $f(9)=g(10)$. Since you can verify $f(9)=g(10)$ for all other plaintext and ciphertext pairs $(p,c)=(6,3),(7,4),(8,5),\cdots$, the Caesar cipher fails in SCT.

The Caesar cipher is shown to be broken in SCT because plaintext values p' can be calculated from ciphertext values c' by using $p' = E^{-1}(9, E^{-1}(10, c'))$.

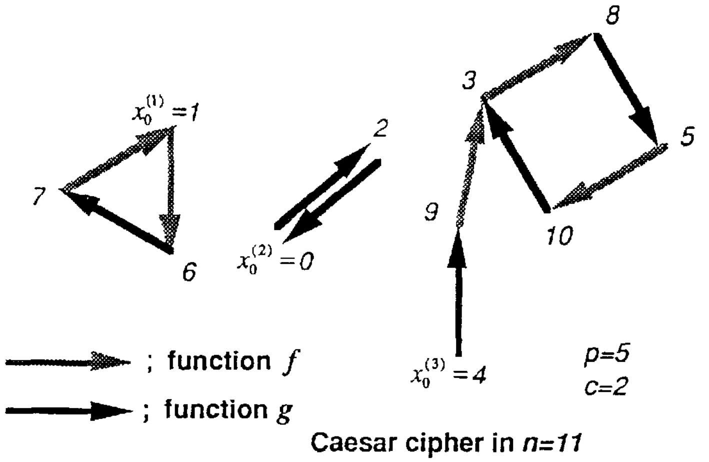


Fig. 4.1 Caesar cipher

4.2 RSA cryptosystem as asymmetric cipher

This section presents a feasibility study for a small version of RSA [RiShAd]. In this model, the relation of parameters is:

$$E(k_e, p) = p^{k_e} \bmod n,$$

$$E^{-1}(k_d, c) = c^{k_d} \bmod n$$

$$(k_e k_d = 1 \bmod L, n = pq, L = \mathrm{LCM}(p-1, q-1)).$$

Thus, we can define functions, f, g, cf, respectively as;

$$f(x) = p^x \bmod n,$$

$$g(x) = c^x \bmod n,$$

$$cf(x) = (\text{true if } x > \frac{n}{2}, \text{ false if } x \le \frac{n}{2}).$$

At first, $n = 33$, $k_e = 3$, and $(p,c) = (13,19)$ are given. Though secret values $(k_d = 7, L = 10)$ are kept secret, Fig. 4.2 is drawn by SCT. If you search from the initial value 18, you find period-length $\lambda = 5$ and a collision of $f(18) = f(28)$. Since SCT cannot find the collision of f and g, you know that the collision is wrong.

In Strategy #1, the intersection search is carried out on the same plane from another start point 12. Then, you can find the collision of $f(19) = g(13)$ which is right for all pairs (p,c). On the other hand, in Strategy #2, the intersection search is carried out on another plane for $(p,c) = (4,31)$ from another start point 22. You find the right collision of $f(22) = g(4)$. Thus, RSA fails in SCT.

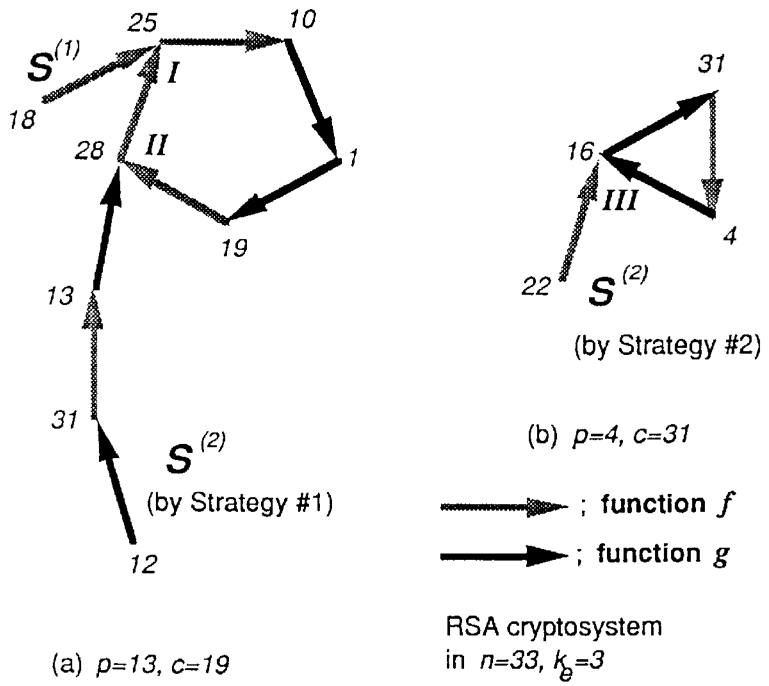


Fig. 4.2 Small model of RSA

Though RSA is an asymmetric cipher, it has one idiosyncrasy that allows it to be broken. If the right collision of $f(x) = g(y)$ is found, $x = k_e y \bmod L$ must be right. Therefore, since you can know a multiple of L, the composite number n can be factorized by using the multiple [Mi]. In the above example, $22 = 3 \times 4 \bmod 10$. However, since commercial RSA schemes use n greater than 2^{500}, SCT would need more than 2^{250} operations; RSA is not menaced by SCT.

5. Conclusion

A switching closure test SCT based on a new memoryless meet-in-the-middle procedure has been proposed as a feasible version of MCT. To achieve the memoryless procedure, several techniques, the most important of which are expansion of cycling detection methods for one function into a method for two functions and an efficient intersection search by using a small amount of memory, are effectively used. Moreover, feasibility studies using a Caesar cipher and a small model of an RSA cryptosystem have been presented.

We intend to apply SCT to various kinds of cryptosystems.

References

[DiHe] W. Diffie and M. E. Hellman: "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer, **10**, 6, pp.74-84, June 1977.

[FlOd] P. Flajolet and A. M. Odlyzko: "Random Mapping Statistics," Advances in Cryptology-EUROCRYPT'89, Proceedings, pp.329-354, Springer-Verlag, 1990.

[KaRiSh] B. S. Kaliski Jr., R. L. Rivest, and A. T. Sherman: "Is the Data Encryption Standard a Group? (Results of Cycling Experiments on DES)," J. Cryptology, **1**, 1, pp.3-36, 1988.

[Kn] D. E. Knuth: Exercise 3.1 No. 7, "The Art of Computer Programming 2nd ed. (Seminumerical Algorithms)," Addison-Wesley, 1981.

[Mi] G. L. Miller: "Riemann's hypothesis and tests for primality," J.Computer and System Science, **13**, pp.300-317, 1976.

[QuDe] J.-J. Quisquater and J.-P. Delescaille: "How Easy is Collision Search. New Results and Applications to DES," Advances in Cryptology-CRYPTO'89, Proceedings, pp.408-413, Springer-Verlag, 1990.

[RiShAd] R. L. Rivest, A. Shamir, and L. Adleman: "A Method of Obtaining Digital Signatures and Public Key Cryptosystems," Comm. of ACM, pp.120-126, Feb. 1978.

[SeSzYa] R. Sedgewick, T. G. Szymanski, and A. C. Yao: "The Complexity of Finding Cycles in Periodic Functions," SIAM J. Comp., **11**, 2, pp.376-390, May 1982.