HOW TO FIND A FIXED POINT IN SHUFFLE EFFICIENTLY

Mikako Kageshima[1] and ManabuOkamoto[1]

[1] Kanagawa Institute of Technology, Kanagawa, Japan
manabu@nw.kanagawa-it.ac.jp

ABSTRACT

In electronic voting or whistle blowing, anonymity is necessary. Shuffling is a network security technique that makes the information sender anonymous. We use the concept of shuffling in internet-based lotteries, mental poker, E-commerce systems, and Mix-Net. However, if the shuffling is unjust, the anonymity, privacy, or fairness may be compromised. In this paper, we propose the method for confirming fair mixing by finding a fixed point in the mix system and we can keep the details on 'how to shuffle' secret. This method requires only two steps and is efficient.

KEYWORDS

Network Security, Protocol, Privacy, Mix

1. INTRODUCTION

Mix-Net [1][2] is a key technique for maintaining anonymity and privacy in electronic voting[3][4]. Mix-Net servers shuffle encrypted messages in order to maintain the privacy of voters. Therefore, 'how to shuffle' is an important subject. However, thus far, 'how to shuffle' is in a black-box wherein it is not discussed whether the shuffle was mixed fairly.

We use the concept of shuffling in internet-based lotteries [5], mental poker [6], E-commerce systems [7], and Mix-Net. However, if the shuffling is unjust, it is reflected on the outcome, and the anonymity, privacy, or fairness may be compromised. In this study, we will propose a method to confirm fair mixing of information. In order to confirm this condition, we will attempt to find fixed points in the mix. If there are no fixed points in the mix, we can declare that the mix was performed well and fairly.

A method to confirm fair mixing was proposed by us in [8]. However, this method required many procedural steps to achieve the correct result. Additionally, the mix would not be fair if these steps were performed only once; hence, the same steps had to be performed repeatedly. This proved to be cumbersome.

In this study, we propose the complete method for confirming fair mixing by finding a fixed point in the mix system and we can keep the details on 'how to shuffle' secret. This method requires only two steps and is efficient. This method needs to be performed only once to get correct results.

2. FAIR MIXING

Mix-Net [1] is a technique for maintaining privacy in electronic voting. Mix-Net is used for making the identity of voters anonymous. Mix-Net servers shuffle information so that its users become anonymous. Plural Mix-Net servers shuffle encrypted messages in order to maintain the privacy of voters, and they are safe unless all servers conspire.

How the shuffle is performed is the key to anonymity. If Mix-Net servers do not shuffle well, the user identities are not anonymous. However, thus far, it has not been discussed whether the algorithm mixes fairly.

In this paper, we will propose a method for confirming fair mixing. To confirm this condition, we will attempt to find fixed points in the mix. If there are no fixed points in the mix, we can declare that the mix was performed well and fairly.

What is "mix well"? What is "fixed point"?

Imagine a card game and a shuffle by the dealer.

Assume that before the mix, the cards are

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}. \tag{1}$$

After the cards are shuffled by the dealer, they are

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}, \tag{2}$$

this shuffle is 'not mixed at all'. If we have

$$\{1, 3, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}, \tag{3}$$

this shuffle is also 'not mixed well'. In the shuffle above, only 2 and 3 are swapped. If we have

$$\{1, 3, 2, 7, 6, 5, 8, 13, 12, 9, 10, 11, 4\}, \tag{4}$$

this shuffle is almost mixed well but it is 'not mixed completely.' In this shuffle, 1 is in the same position before and after the shuffle. This 1 is known as the "fixed point." This example contains only one fixed point.

We can characterize a mix as 'not mixed completely' when there is even a single fixed point. Conversely, we define that when there are no fixed points in the shuffle, the mix is 'mixed well completely.'

In this paper, we will propose a method to confirm whether a mix is 'mixed well completely.' In the method, we ascertain whether there are any fixed points in the mix. As described above, when we have no fixed point, the mix is 'mixed well completely.'

3. RELATED WORK AND OUR GOAL

Thus far, 'how to shuffle well' has not been discussed. First, in [8], we had proposed a similar method. However, in that system, we have to do test repeatedly because it may be wrong if only once. When the test result is "we have fixed point," there may actually be no fixed point. The probability of this error is 1/N, when N is number of cards. To reduce this probability, we need to repeat the test. Hence, this method is not efficient.

In the new scheme presented in this paper, the probability of an error approaches zero. Hence, we need to perform the procedure only once to confirm the shuffle. Thus, this method is simple and efficient.

Our goal is to obtain a method to confirm whether the mix is 'mixed well completely.' However, we can confirm only whether fixed points exist, and we can keep the details on 'how to shuffle' secret. In addition, we focus on only fixed points in the mix system. Hence, we will not delve into the details of the mix system in examples, such as a card game or a Mix-Net.

4. PROPOSED METHOD

We assume that a network consists of mix servers, which are placed in line, and these servers shuffle messages in turn. This system is the same as Mix-Net.

We denote the servers that shuffle as T_1, T_2,…,T_M, where M is the number of servers. Each mix server shuffles N messages from the previous server and sends them to the next server.

Each mix server T_l has three functions: (1) a permutation function π, (2) an encrypt function E, and (3) a decrypt function D. The permutation function is denoted as $\pi_l(i) = j$, which means that the mix servers T_l change position i for position j. The encrypt function is denoted as E (P:k), where P is plain text and the encrypt key is k. The decrypt function is denoted as D (C:k), where C is the encrypted message and the decrypt key is k. Additionally, the encrypt and decrypt function must be commutative.

Commutative means that for any plain text P and any keys k_1, k_2,

$$E\,(P{:}k_1,k_2) = E\,(P{:}k_2,k_1). \tag{5}$$

Both the encrypt key and the decrypt key must be secret.

We illustrate the whole system in Figure 1. In this figure, we have three mix servers and three messages. {a_1, a_2, a_3} is shuffled through all the servers and this mix system generates an output {a_1,a_3, a_2}. This system has one fixed point a_1. Hence, it is 'not mixed well.'

A 'tester' generates test values such as {a_1, a_2, a_3} and initiates the test, as shown in Figure 1.The tester also has an encrypt and decrypt function. These encrypt and decrypt functions must be commutative with each other as well as with the functions of the mix servers. Additionally, all the encrypt keys and decrypt keys of the tester must remain secret.

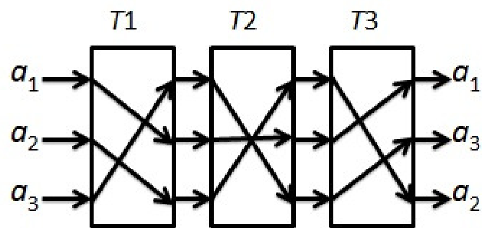

Figure 1. Mix system

We describe the procedure for confirming fixed points in this system. We require two turns to complete the procedure. The tester needs to generate N test values and Nencrypt and decrypt keys. We denote the encrypt keys of the tester with k_i^0(i=1,2,…,N).

The initial test value is defined as

$$J = \{a_1,a_2,\ldots,a_N\} \quad a_i : \text{any integer} \tag{6}$$

The tester encrypts each test value a_i with a corresponding key k_i^0 (i=1,2,…,N) and sends them to the mix server T_1.

Each mix server T_l receives the test values from the tester or the previous mix server, shuffles them, and encrypts each test value with a corresponding key in turn 1. Each mix server T_l has N encrypt and decrypt keys. They must be secret and the encrypt and decrypt functions must be commutative with each other. The resultant values of turn 1 are returned to the tester, which uses them as the input value in turn 2.

In turn 2, each mix server performs a similar procedure; each mix server shuffles and decrypts the test values. We proceed to describe these turns in detail.

< Turn 1 >

The tester encrypts each element of the initial test value J using the public key k_i^0. Subsequently, the tester sends the encrypted values to T_1.

$$\{E\,(a_1 : k_1^0),\, E\,(a_2{:}\, k_2^0),\ldots,\, E\,(a_N{:}\, k_N^0)\}. \tag{7}$$

T_1 shuffles these values and encrypts them using the secret encrypt keys $\{k_i^1\}$ (i=1,2,…,N). Then, T_1 send the resultant values to T_2. The values that T_1 will send are

$$\{\ldots, E\,(a_i{:}k_i^0, k^1_{(\pi 1(i))}, \ldots)\}. \qquad (8)$$

T_2 also shuffles these test values and encrypts them using the secret keys $\{k_i^2\}$. Then, T_2 send the final values to T_3. They are

$$\{\ldots, E\,(a_i{:}k_i^0, k^1_{(\pi 1(i))}, k^2_{\pi 2(\pi 1(i))}, \ldots)\}. \qquad (9)$$

The mix servers that follow perform the same procedure. The values sent by the last server T_M back to T_1 is

$$Y_1 = \{\ldots, E\,(a_i{:}k_i^0, k^1_{(\pi 1(i))}, k^2_{\pi 2(\pi 1(i))}, \ldots, k^M_{\pi M\,(\pi M-1\,(\pi M-2\,(\ldots(\pi 3(\pi 2(\pi 1(i)\ldots\,)}), \ldots)\}. \qquad (10)$$

Y_1 is the result of turn 1.

< Turn 2>

The procedure followed in turn 2 is similar to that in turn 1.In turn 1, we use the initial test value J; however, in turn 2, we use Y_1 as the initial value. In turn 1, we encrypted the elements; in turn 2, we decrypt these elements.

T_M sends Y_1 to T_1. T_1 shuffles Y_1 in the same manner as that in turn 1 and decrypts each element of Y_1. Subsequently, T_1 send them to T_2.

The mix servers that follow perform the same action. They shuffle and decrypt the elements. The last server T_M sends the decrypted elements back to the tester. The tester also decrypts these to obtain Y_2. Y_2 is the result of turn 2.

The tester can compare J with Y_2 and find fixed points. If we have the same value at its original position, it is a fixed point. When we have

$$J = \{a_1, a_2, \ldots, a_N\} \qquad (11)$$

$$Y_2 = \{y_{21}, y_{22}, \ldots, y_{2N}\}, \qquad (12)$$

and there exists i such that

$$a_i = y_{2i}, \qquad (13)$$

then this is a fixed point. Thus, the test is completed.

If an element is a fixed point, that element must go through the same route, both in turn 1 and in turn 2. In this case, the encrypt keys, which the mix servers use in turn 1, match all the decrypt keys, which mix servers use in turn 2. Thus, the fixed points are restored to their initial values.

5. Sample Example

We describe this process using a simple example wherein the number of mix servers and the number of messages for shuffling are manageably small. We have three mix servers and three messages, similar to the example in Figure 1. In this example, we have one fixed point, which is the first element.

< Turn 1 >

1) The tester generates the initial test value J.

$J = \{a_1, a_2, a_3\}$ (14)

2) Each element of J is encrypted with $\{k_i^0\}$ (i = 1, 2, 3) by the tester, and these elements are sent to T_1. $\{k_i^0\}$ are the encrypt keys of the tester.

$E\,(a_1{:}k_1^0)$ (15-1)

$E\,(a_2{:}k_2^0)$ (15-2)

$E\,(a_3{:}k_3^0)$ (15-3)

3) T_1 receives the test values from the tester and shuffles them using π_1. $\pi_1$is a function to shuffle elements.

$\pi_1(1) = 2$ (16-1)

$\pi_1(2) = 3$ (16-2)

$\pi_1(3) = 1$ (16-3)

4) Each shuffled element is encrypted with $\{k_i^1\}$ by T_1, and the results are sent to T_2. $\{k_i^1\}$ are the encrypt keys of T_1. Note that, after the shuffle, the positions of the elements are changed.

$E\,(a_3{:}k_3^0,k_1^1)$ (17-3)

$E\,(a_1{:}k_1^0,k_2^1)$ (17-1)

$E\,(a_2{:}k_2^0,k_3^1)$ (17-2)

5) T_2 shuffles the encrypted elements using π_2. $\pi_2$is a function to shuffle elements.

$\pi_2(1) = 3$ (18-1)

$\pi_2(2) = 2$ (18-2)

$\pi_2(3) = 1$ (18-3)

6) Each shuffled element is encrypted with $\{k_i^2\}$ by T_2, and this is sent to T_3. $\{k_i^2\}$ are the encrypt keys of T_2.

$E\,(a_2{:}k_2^0,k_3^1,k_1^2)$ (19-1)

$E\,(a_1{:}k_1^0,k_2^1,k_2^2)$ (19-2)

$E\,(a_3{:}k_3^0,k_1^1,k_3^2)$ (19-3)

7) T_3 shuffles the encrypted elements using π_3. $\pi_3$is a function to shuffle elements.

$\pi_3(1) = 3$ (20-1)

$\pi_3(2) = 1$ (20-2)

$\pi_3(3) = 2$ (20-3)

8) Each shuffled element is encrypted with $\{k_i^3\}$ by T_3. $\{k_i^3\}$ are the encrypt keys of T_3.

$E\,(a_1{:}k_1^0,k_2^1,k_2^2,k_1^3)$ (21-1)

$E\,(a_3{:}k_3^0,k_1^1,k_3^2,k_2^3)$ (21-3)

$E\,(a_2{:}k_2^0,k_3^1,k_1^2,k_3^3)$ (21-2)

Turn 1 is completed and the values obtained above are the results of turn 1.

< Turn 2 >

Turn 2 is similar to turn 1.We use an initial test value in turn 1; however, in turn 2 we use the resultant values of turn 1 as the initial test value. In turn 1, we encrypt the elements; in turn 2, we decrypt these elements.

1) T_3 sends the results of turn 1 to T_1.

2) T_1 shuffles them using

$\pi_1(1) = 2$ (22-1)

$\pi_1(2) = 3$ (22-2)

$\pi_1(3) = 1$ (22-3)

3) Each shuffled element is decrypted with a decrypt key corresponding to the encrypt key $\{k_i^1\}$. Here,for convenience in explanation, we use the same term $\left\{k_i^{j}\right\}$ to denote the encrypt and the decrypt keys. The following results are sent to T_2.

$D\,(E\,(a_3{:}k_3^0,k_1^1,k_3^2,k_2^3){:}k_3^1\,)$ (23-1)

$D\,(E\,(a_1{:}k_1^0,\cancel{k}_2^1,k_2^2,k_1^3){:}\cancel{k}_2^1\,)$ (23-2)

$D\,(E\,(a_2{:}k_2^0,k_3^1,k_1^2,k_3^3){:}k_1^1\,)$ (23-3)

4) T_2 shuffles them using

$\pi_2(1) = 3$ (24-1)

$\pi_2(2) = 2$ (24-2)

$\pi_2(3) = 1$ (24-3)

5) Each shuffled element is decrypted with $\{k_i^2\}$ by $T_2$and the following results are sent to T_3.

$D\,(E\,(a_2{:}k_2^0,k_3^1,k_1^2,k_3^3){:}k_1^1,k_3^2\,)$ (25-1)

$D\,(E\,(a_1{:}k_1^0,\cancel{k}_2^1,\cancel{k}_2^2,k_1^3){:}\cancel{k}_2^1,\cancel{k}_2^2\,)$ (25-2)

$D\,(E\,(a_3{:}k_3^0,k_1^1,k_3^2,k_2^3){:}k_3^1,k_1^2\,)$ (25-3)

6) T_2 shuffles them using

$\pi_3(1) = 3$ (26-1)

$\pi_3(2) = 1$ (26-2)

$\pi_3(3) = 2$ (26-3)

7) Each shuffled element is decrypted with $\{k_i^3\}$ by $T_3$and the following results are sent back to the tester.

$D\,(E\,(a_1{:}k_1^0,\cancel{k}_2^1,\cancel{k}_2^2,\cancel{k}_1^3){:}\cancel{k}_2^1,\cancel{k}_2^2,\cancel{k}_1^3)$ (27-1)

$D\,(E\,(a_3{:}k_3^0,k_1^1,k_3^2,k_2^3){:}k_3^1,k_1^2,k_3^3)$ (27-2)

$D\,(E\,(a_2{:}k_2^0,k_3^1,k_1^2,k_3^3){:}k_1^1,k_3^2,k_2^3)$ (27-3)

8) Each shuffled element is decrypted with $\{k_i^0\}$ by the tester.

$D\,(E\,(a_1{:}\cancel{k}_1^0,\cancel{k}_2^1,\cancel{k}_2^2,\cancel{k}_1^3){:}\cancel{k}_2^1,\cancel{k}_2^2,\cancel{k}_1^3,\cancel{k}_1^0) = a_1$ (28-1)

$D\,(E\,(a_3{:}\cancel{k}_3^0,k_1^1,k_3^2,k_2^3){:}k_3^1,k_1^2,k_3^3,\cancel{k}_3^0)$ (28-2)

$D\,(E\,(a_2{:}\cancel{k}_2^0,k_3^1,k_1^2,k_3^3){:}k_1^1,k_3^2,k_2^3,\cancel{k}_2^0)$ (28-3)

9) The tester obtains the result of the decryption. The tester can compare (14) with (28-1,2,3) and find fixed points. If we have the same value at its original position, it is a fixed point. In

this example, we have one fixed point, which is the first element a_1. Figure 2 illustrates this example.

If an element is a fixed point, that element must go through the same route, both in turn 1 and in turn 2. In this case, element a_1 goes through the same route in turn 1 and in turn 2.As all the encrypt keys $\{k_1^0, k_2^1, k_2^2, k_1^3\}$, which the mix servers use in turn 1, match all the decrypt keys$\{k_1^0, k_2^1, k_2^2, k_1^3\}$, which mix servers use in turn 2,the fixed point is restored to its initial values.

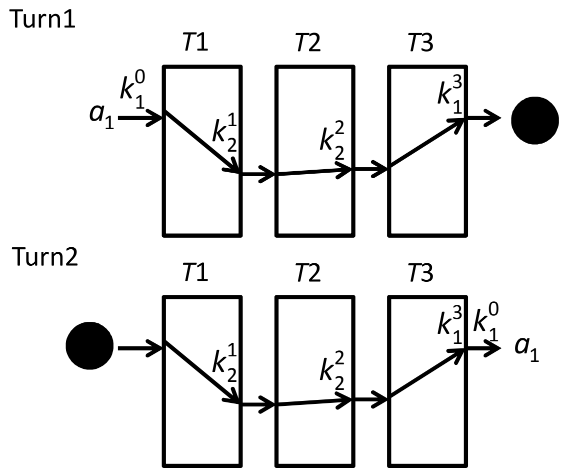

Figure 2. Example

6. Security Analysis

In this chapter, we discuss the security analysis of our scheme. However, we only focus on attacks that attempt to infer the path of the messages using the test results. We do not include other types of attacks, such as the simple sabotage without the profit, in our discussion.

First, we assume that all mix servers are fair. The attacker (may include a tester) attempts to infer the permutation of each mix server using only the result value. If the tester is a attacker, the tester can use any initial value to perpetrate the attack.

The attacker attempts to chase the test value by comparing the input with the output of the mix servers in order to infer the permutation. However, the encrypt keys, with which the input values are encrypted, are not known by the perpetrator as the keys are not public. This implies that an attacker cannot chase an element using only the test values as these values are encrypted at the output.

If the difference between the maximum and the minimum value of the input is large, for example {10000000, 2, 3,…}, the old scheme [8] is problematic. However, in the new scheme proposed in this paper, we encrypt the values. When we encrypt them, we use 'modulo' like RSA and the difference in the sizes of the test values is rendered meaningless.

Next, we assume an attack by only one malicious mix server. However, a single mix server can hardly attack alone because the other mix servers shuffle and encrypt values using individual secret keys, hence the values cannot be chased.

Finally, we assume that the fair mix server *S*, between unjust mix servers *P* and *Q*, is attacked, and the attackers conspire to infer the permutation of the fair server *S*.

In turn 1, *P* send test value *U* to *S*, and *Q* obtains an output *V* from *S*. They use this *V* in turn 2. *P* sends *V* to *S* as the input in turn 2 and *Q* obtains output *W* from *S*. The result of the test in the complete mix system becomes erroneous because the malicious servers discard the real values. Figure 3 shows this attack.

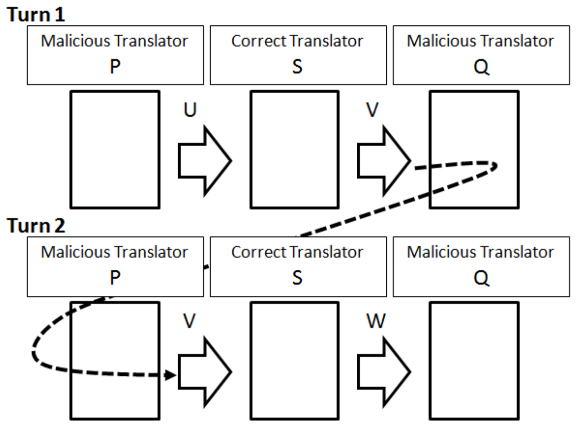

Figure 3.Attack from both side.

P and *Q* infer the permutation of *S* by comparing *U* with *W*. If there is a value at the same position between *U* and *W*, the malicious servers can identify the fixed point of the permutation of the mix server *S*. *P* and *Q* cannot know the permutation of the entire mix system; however, this is a leak of a subset of the permutation in the mix system. This results in a breach of anonymity. We can prevent this by checking whether the output *V* and input *V* on *S* are the same. But unjust mix servers *P* and *Q* generate *V'* from *V* by encryption, and different values can be generated for the same input value. *V'* can be used for injustice. The fixed points can be identified by obtaining *W'* from *V'* via server *S* and converting it back to *W*.

To prevent this type of attack, we can add a new process. All mix servers $T_1, T_2,\ldots,T_M$ have keys p_i and q_i of commutative function such that when a is plain text

$$D\,(E\,(a{:}p_1, p_2, p_3,\ldots, p_M){:}\; q_1, q_2, q_3,\ldots, q_M) = a. \qquad (29)$$

However, basically, $D\,(E\,(a{:}p_i){:}q_i)$is not equal to a. In other words, key p_i is not corresponding to key q_i.

We can create these keys in a manner such that on public encrypt systems *E* and *D*, when the public keys are$\{p_1, p_2, p_3,\ldots,p_M\}$, the private keys are $\{q_1, q_2, q_3,\ldots, q_M\}$.

In turn 1, each mix server T_i encrypts values using p_i, and in turn 2 they are decrypted using q_i. By the addition of this procedure in the example shown in figure 3, malicious mix servers *P* and *Q* cannot find the fixed points because all $\{p_1, p_2, p_3,\ldots p_M\}$ and $\{q_1,q_2,q_3,\ldots q_M\}$ are necessary to erase all encryption. If at least two or more mix servers do not join in a malicious attack, injustice does not occur.

7. CONCLUSION

In this paper, we proposed a method to find fixed points in a mix system without reviewing shuffling. The probability of an error using this method approaches zero. Hence, the procedure needs to be performed only once to obtain optimal results.

REFERENCES

[1] D. Chaum, (1981) "Untraceable Electronic Mail, Return Address, and Digital Pseudonyms," Communications of the ACM, Vol. 24, No. 2, pp.84-88, 2.

[2] Kun Peng, (2011) "Failure of a Mix Network", International Journal of Network Security & Its Applications (IJNSA), Vol. 3, No. 1, 1.

[3] M. Okamoto & Y. Tanaka, (2006) "Secret Information Distribution," The Transactions of the Institute of Electronics, Information and Communication Engineers, A J89-A (8), pp.662-670.

[4] M. Okamoto, (2011) "A Multiple Ballots Election Scheme Using Anonymous Distribution," International Journal of Network Security & Its Applications (IJNSA), Vol. 3, No. 5, 9.

[5] B. Schneier, (1996) "Applied Cryptography," John Wiley & Sons.

[6] A. Shimir, R. Rivest, & L. Adleman, (1981) "Mental Poker," The Mathematical Gardner, Wadsworth International, California, pp.37-43.

[7] H. Jayasree& A. Damodaram, (2012) "A Novel Fair Anonymous Contract Signing Protocol for E-Commerce Applications," International Journal of Network Security & Its Applications (IJNSA), Vol. 4, No. 59.

[8] M. Kageshima, M. Okamoto, (2011) "How to find a Fixed point in a Mix System," TENCON 2011-2011 IEEE Region 10 Conference, Bali, Indonesia11.

Authors

MikakoKageshima

She is a student at Kanagawa Institute of Technology.

Manabu Okamoto

He received B.S. and M.S. degrees in Mathematics from Waseda University in 1995 and 1997, respectively. In 2010, he received a doctoral degree in the field of Global Information and Telecommunication from Waseda University. He is now an Associate Professor at Kanagawa Institute of Technology.