Monitoring Remote Data Problem

Naoshi Sakamoto

***Abstract*—On administrating a computer system, its configuration data is important. Thus, the system usually treats to protect the configuration data. Moreover, recently, computer systems that consist of large scale network, such as distributed systems, cloud systems and so on, have been developed. For these systems, it is an important issue to administrate the configuration data of each computer of the system. On the other hand, in general, a computer system becomes more complicated because of not only its scale but also its improvement of the services. Then, this causes increase in size of its configuration data.**

In this paper, we introduce the k-monitoring remote data problem where the remote data rarely happens to be changed. The data is expected to be changed in at most *k* bits. We assume that a remote computer and a local computer are connected by network, and share the same data of length *N* at the beginning. We also assume that some of bits of the remote data are expected to be changed, after a while. Then, the problem is finding the protocol between the computers so that the local computer outputs the each position of the changed bits.

In this paper, we propose the protocols within the communication bits of length $\Theta(\log N)$ for 1- and 2-Monitoring Remote Data Protocol.

***Index Terms*—Humming distance, communication complexity, redundant coding.**

I. Introduction

On administrating a computer system, its configuration data is important. For example, when we would administrate many computers at a computer center, we should keep each registry, the configuration data of an OS such as MS Windows, of the computers identical. While each data is basically kept identical for all of the computers, since system updates cause the change of the configuration data, we should assume that the configuration data takes arbitrary value.

On the other hand, recently, computer systems that consist of a large scale network, such as distributed systems, cloud systems and so on, have been developed. These systems consist of a lot of computers. Nevertheless, each computer of a system does not usually have the different role. Specially, in order that the system has the redundancy and the scalability, it usually consists of computers with almost the same configuration. Thus, it is important for the administrators to keep the configuration data of each computer identical.

Moreover, in general, a computer system becomes more complicated because of not only its scale but also its improvement of services. This is caused by improvement of performance, abstraction, and virtualization of the component (containing OSes) of the computer systems. This enables the system administration to increase flexibility and the amount of information. Thus, this also occurs to increase the amount of the configuration data.

In this paper, we introduce the monitoring remote data problem where the remote data rarely happens to be changed. We assume that a remote computer and a local computer are connected by network, and share the same data of length N at the beginning. We also assume that some of bits of the remote data are expected to be changed, after a while. Then, it is the problem to find the protocol between the computers so that the local computer outputs the each position of changed bits.

First, we can consider the trivial solution that (1) the remote computer sends the all configuration data to the local computer, (2) then the local computer receives the remote configuration data, and computes the differences. (3) The local computer outputs each position of differences between the remote data and the local data. Thus, the problem can be solved if the protocol is allowed to communicate messages up to $O(N)$ bits, where N is the length of the data that both sides own. However, do there exist any protocols that communicate messages in at most $o(N)$ bits?

Secondly, we can consider to apply a hash function such as MD5 [1] to create the message. A hash function is a many-one function from a bit stream of an arbitrary length to a bit stream of the fixed length. It has the property that it is difficult to compute the original bit stream from a function value. Moreover, the function value is expected to be changed whenever the original bit stream is slightly changed even if the original bit stream is long. Thus, we can consider an algorithm that for given a hash function value from the remote side, the local computer seeks difference by searching all slightly changed data where the hush function value matches the received value. However, since the length of the hash function value is limited, some two slightly different changes for a sufficient long bit stream must yield the same hash function value. Then, so far as applying a hash function, we cannot determine each position of the changed bits for a sufficient long bit stream.

Now, in this study, in order to formulate the monitoring remote data problem, we propose the following conditions:

1) The remote computer and the local computer has the same data at the beginning;
2) The remote data might be changed, but the degree of change is limited;
3) The length of communication messages is limited to $o(N)$ where the length of the data is $N;$
4) Finally, the local computer outputs each position of the changes of the remote data.

From the point of view that the data is slightly changed,

Manuscript received September 10, 2014; revised November 18, 2014.

N. Sakamoto is with the Department of Information and Communication Engineering, Tokyo Denki University, Tokyo, Japan (e-mail: sakamoto@c.dendai.ac.jp.)

our study is resembling to the theory of error-correction code, and the theory of redundant array. However, our study is different from these since our study assumes that the length of data is not limited.

In this paper, we propose the protocols for the problem within the communication bits of length $\Theta(\log N)$ with the assumption that the number of changed bits is limited within at most either one or two.

The structure of this paper is the following: In Section II, we introduce the related works. In Section III, we define some notations. Then, Section IV, we propose several theories and protocols for 1 bit and 2 bit difference. In Section V we conclude and propose a conjecture.

II. Related Works

Yao proposed the notion of communication complexity [2]. This investigates the complexity of functions by comparing the length of required communication messages to compute the function. Assume that the person P_1 has an integer $i\ (0 \le i \le n-1)$ and the person P_2 has another integer $j\ (0 \le i \le n-1)$. For given a function $f : \mathbb{N} \times \mathbb{N} \to \{0,1\}$, this study investigates the amount of bits for communication between P_1 and P_2 that function f requires to compute the value $f(i,j)$. One of what he showed is that the identification function requires at least $\log_2 n - 2$ bits. Note that it is trivial to compute the identification function within $\lceil \log_2 n \rceil$ bits message (for example, P_1 sends i itself to P_2).

The study of communication complexity has no restriction for the input data except for the assumption that the input data belongs to the fixed set. Moreover, the study aims to compare the complexity between functions by the amount of communication bits. On the other hand, our study considers the amount of communication bits to compute each position of difference between two data by assuming the strong correlation between them. That is, our study is different from the one of communication complexity in the point of the restriction for the input data.

By considering sharing the same data as a redundant storage, and the detection of the difference of the data as error-correcting, we can say that our study would concern with the study of data redundancy such as RAID5 and RAID6 [3], [4]. RAID5 and RAID6 are the storage virtualization technology to store every data of a few bit into separate several storages by dividing information with a certain method. RAID5 requires three storages, and RAID6 requires four storages. Then, RAID5 can correct data even if one storage fails, and RAID6 can also correct data even if at most two storages fail. The method to dividing information applies the theory of Galois field. Though the model of this problem is different from the one of our study, this study contributes to the basic idea of our algorithm.

III. Preliminary

Let N be the bit length of the data. P_r denotes a remote player and P_l denotes a local player. Let $\mathbf{r} = (r_1, r_2, \ldots, r_N)$ be the bit stream that P_r owns, and $\mathbf{l} = (l_1, l_2, \ldots, l_N)$ be the bit stream that P_l owns, where $\mathbf{r}, \mathbf{l} \in \{0,1\}^N$. $H(\mathbf{r}, \mathbf{l})$ denotes the Humming distance between $\mathbf{r}$ and $\mathbf{l}$.

For a bit stream $\mathbf{x} \in \{0,1\}^N$, $index(\mathbf{x}) = \{i \mid x_i = 1\}$ denotes the set of positions of one. For $1 \le i \le N$, $\bar{\imath}$ denotes the binary representation of i with length $\lceil \log_2 N \rceil$. $\langle N, k \rangle$ denotes $(1^k, 2^k, \ldots, N^k)$. $\mathbf{0} = (0, 0, \ldots, 0)$ denotes the bit stream of 0 with length N.

For natural number m, $\mathbb{Z}/m\mathbb{Z}$ denotes a group with order m. $\oplus$ denotes its operator and 0 denotes its identity. $GF(q)$ denotes the Galois field with order of q. $+$, $\cdot$, 0, and 1 denotes the addition symbol, the multiplication symbol, the zero element symbol, and the unity element symbol, respectively. Let $|S|$ be the order of S.

Definition 1. Let player P_r have a bit stream $\mathbf{r}$ with bit length N, and player P_l have a bit stream $\mathbf{l}$ with bit length N. k-Monitoring Remote Data Problem is the problem that P_l outputs each position of different between $\mathbf{r}$ and $\mathbf{l}$ by communicating each other, when $H(\mathbf{r}, \mathbf{l})$ is at most k.

IV. Protocols

A. Lower Bound

Theorem 1. k-Monitoring Remote Data Problem requires communication message of more than $\Omega(\log_2 N)$ bits.

Proof. For a bit stream $\boldsymbol{l}$, consider a set $R = \{\mathbf{r} | H(\mathbf{r}, \mathbf{l}) \le k$. The number of elements of R is given by (1):

$$|R| = \textstyle\sum_{i=0}^{k} \binom{N}{i}. \tag{1}$$

Whichever $\mathbf{r}$ is taken from R, P_l must output the different value. Thus, P_l must communicate the enough amount of information to distinguish each element of R with P_r. Then, we have the minimum amount of communication bits by (2).

$$\begin{aligned} \lceil \log_2 |R| \rceil &\ge \left\lceil \log_2 \binom{N}{k} \right\rceil \\ &\ge \lceil \log_2 N \rceil. \end{aligned} \tag{2}$$

Therefore, the minimum amount of communication bits is $\Omega(\log_2 N)$.

B. A Protocol Computing the Difference of One Bit

In this section, for the remote data $\mathbf{r}$ and the local data $\mathbf{l}$, we assume that the humming distance between $\mathbf{r}$ and $\mathbf{l}$ is at most one.

Let's consider a finite undirected graph such that its vertex set consists of the whole bit streams of the length N, and each pair of vertices is connected if the humming distance of the pair is equal to one. This graph is called a Boolean cube (Fig. 1). According to the definition, each degree of the vertices is N. Now, we can consider the label of each edge that indicates the position of the changed bit. Then, we can easily know that for every vertex, each label of edges connected to the vertex is different each other, and formed of the numbers from 1 to N (Fig. 2).

Now, we propose a protocol such that the players do not communicate each other, but simply the remote player P_r computes a value of a function and sends the value to the local player P_l only. Let $c(\cdot)$ be the function that P_r computes.

1) Protocol 1

1) At the remote side, P_r computes $m = c(\mathbf{r})$, then sends m to P_l;

2) At the local side, P_l receives m, computes the position of the changed bit by using m and $\mathbf{l}$, then outputs it.

Now, for $\mathbf{0}$, the bit stream of length N where all bits are 0, consider $c(\mathbf{0})$. Moreover, for any bit stream $\mathbf{r}$, consider the representation of $c(\mathbf{r})$ as the path from $\mathbf{0}$ to $\mathbf{r}$ on the Boolean cube. Then, we consider the transformation for the value of c. Let s_j $(1 \le j \le N)$ be the transformation between $c(\mathbf{x})$ and $c(\boldsymbol{y})$ where $\boldsymbol{x}$ and $\boldsymbol{y}$ differ only at the j-th bit, and let S be the set of all s_j $(1 \le j \le N)$ and the identity transformation. Then, we can represent $c(\mathbf{r})$ as a sequence of $s_{r_1}, \dots, s_{r_n}$, where $r_1, \dots, r_n$ are the positions of one in $\mathbf{r}$ and $c(\mathbf{r}) = s_{r_n}(\dots(s_{r_1}(c(\mathbf{0}))))$ holds.

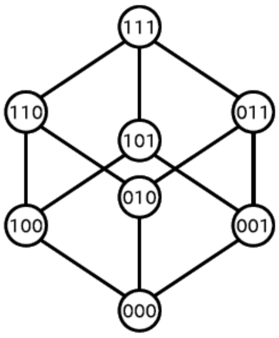

Fig. 1. Boolean cube.

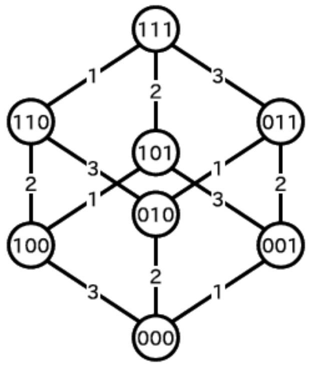

Fig. 2. Assignment of the edge label.

Lemma 2. S is a finite abelian group.

Proof. Let e be the identity transformation of S. Consider about the composition of the elements of S. The composition has the following properties:

1) It is equivalent to invert i-th bit and j-th bit before invert k-th bit and to invert i-th bit before invert and j-th bit and k-th bit;
2) It is equivalent to apply the same transformation twice and to apply nothing.
3) It is equivalent to invert i-th bit before invert j-th bit and to invert j-th bit before invert i-th bit.

Let $\oplus$ denote the composition of transformations. Then, we can express the above properties as (3), (4), and (5) for $i, j, k \in S$:

$$(i \oplus j) \oplus k = i \oplus (j \oplus k), \tag{3}$$

$$k \oplus k = e, \tag{4}$$

$$i \oplus j = j \oplus i. \tag{5}$$

Thus, since we show that S has the associative law, the property that every element has an inverse element, and the commutative law, then S is a finite abelian group.

Lemma 3. If S contains the transformations for 1 bit difference only, the minimum group of S is isomorphic to $\mathbb{Z}/2^{\mathrm{m}}\mathbb{Z}$, where m is at least $\lceil \log_2(N+1) \rceil$.

Proof. Since S is a finite abelian group, S can be decomposed into the direct sum of cyclic groups of primepower order, according to the fundamental theorem.

Now, we assume that S is the minimum group, and S can be decomposed into $K \oplus \mathbb{Z}/p^k\mathbb{Z}$ where K is a group, p $(p \ge 3)$ is a prime, and $k \ge 1$. However, there exists only one element s in $\mathbb{Z}/p^k\mathbb{Z}$ such that $s \oplus s = e$ since p^k is an odd number. Then the number of elements s such that $s \oplus s = e$ in S and the number of elements s such that $s \oplus s = e$ in K are equal. Thus, by considering their orders, this implies $|S| > |K|$. However, this contradicts the assumption of the minimality. That is, if S is the minimum, then S is isomorphic to $\mathbb{Z}/2^m\mathbb{Z}$ for some m.

On the other hand, for any bit streams $\mathbf{r}$ and $\mathbf{l}$, every s must be contained by S, where $H(\mathbf{r}, \mathbf{l}) = 1$ and $s(\mathbf{r}) = \mathbf{l}$. Since the length of the bit stream is N, each transformation between the one bit difference should have the different representation.

Moreover, S must have the identity transformation. Thus, the order of S is more than $N + 1$. Therefore, since $\left|\frac{\mathbb{Z}}{2^m\mathbb{Z}}\right| \ge N + 1$ holds, we have $\lceil \log_2(N+1) \rceil \le m$.

Now, let us consider the way to compute $c(\cdot)$ concretely. We consider the sequence of elements of S that are corresponding to ones of the bit stream $\mathbf{r}$. Let the positions of one in $\mathbf{r}$ be r_1, ..., and r_n. Let s_{r_1}, ..., and s_{r_n} be the elements of S that invert r_1-th, ..., and r_n-th bit, respectively. Then, according to (6), we can obtain s_r in S.

$$\begin{aligned} c(\mathbf{r}) &= s_{r_n}(\dots(s_{r_1}(c(\mathbf{0})))) \\ &= (s_{r_n} \oplus \dots \oplus s_{r_1})(c(\mathbf{0})) \\ &= s_r(c(\mathbf{0})) \text{for some } s_r \in S. \end{aligned} \tag{6}$$

Moreover, for a bit stream $\mathbf{l}$ that $H(\mathbf{l}, \mathbf{r}) = 1$ holds, there exists a transformation $s \in S$ that satisfies (7) and we obtain $s_l \in S$.

$$\begin{aligned} c(\mathbf{l}) &= s(c(\mathbf{r})) \\ &= (s \oplus s_{r_n} \oplus \dots \oplus s_{r_1})(c(\mathbf{0})) \\ &= s_l(c(\mathbf{0})) \text{ for some } s_l \in S. \end{aligned} \tag{7}$$

Thus, since S is a finite abelian group, both compositions of transformations that yield $c(\mathbf{r})$ and $c(\mathbf{l})$ are corresponding to some element s_r and s_l of S, where $s_r = s \oplus s_l$. Then, when the remote player P_r computes and send s_r, then the local player P_l can receive s_r, compute s_l and $s_r \oplus s_l = s$, and output s. Even if both s_r and s_l might not be corresponding to the position of the bit changes, s must be corresponding to the position of the bit change, since it is guaranteed that s is corresponding to a single bit change. Now, we show an algorithm that computes $c(\mathbf{r})$ as (8):

$$c(\mathbf{r}) = \oplus_{i \in index\,(r)} \ \bar{\imath}. \tag{8}$$

Thus, we propose Protocol 2 as following:

2) Protocol 2

1) At the remote side, according to (8), P_r computes $m = c(\mathbf{r})$ and sends it to P_l;
2) At the local side, once P_l receives m, P_l computes $k = m \oplus c(\mathbf{l})$ by using m and (8);
3) If $k = 0$, P_l outputs "unchanged," otherwise, P_l outputs the position corresponding to k of the changed bit.

Theorem 4. There exists a protocol solving 1-Monitoring Remote Data Problem, where the number of message bits is $O(\log_2 N)$, the time to compute the communication message is $O(N \log_2 N)$, and the time to compute the position of the changed bit is $O(N \log_2 N)$.

C. A Protocol Computing the Difference of at Most Two Bits

We know that $\mathbb{Z}/2^m\mathbb{Z}$ is isomorphic to the additive group of $GF(2^m)$. Now, we correspond the vector representation of elements of $GF(2^m)$ to the binary representation of natural numbers. Thus, we can denote the elements of $GF(2^m)$ as the natural numbers such as $0, 1, \dots, 2^m - 1$. We denote the inner product of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ consisting of elements of $GF(2)$ as $\mathbf{x} \cdot \mathbf{y}$. According to these definition, we can consider that $\mathbf{r}, \mathbf{l} \in GF(2)^N$ and $\langle N, k\rangle \in GF(2^m)^N$, where m satisfies $N \leq 2^m$.

Then we can rewrite protocol 2 as following:

1) Protocol 2'

1) P_r computes $m = \mathbf{r} \cdot \langle N, 1\rangle$, then send it to P_l;
2) Once P_l receives m, P_l computes $k = m + \mathbf{l} \cdot \langle N, 1\rangle$;
3) If $k = 0$, P_l outputs "unchanged," otherwise, P_l outputs the position corresponding to k of the changed bit.

For the remote data $\mathbf{r}$ and the local data $\mathbf{l}$, we propose Protocol 3 for 2-Monitoring Remote Data Problem.

2) Protocol 3

1) P_r computes $m_1 = \mathbf{r} \cdot \langle N, 1\rangle$, $m_2 = \mathbf{r} \cdot \langle N, 2\rangle$, and $m_3 = \mathbf{r} \cdot \langle N, 3\rangle$, then send them to P_l;
2) Once P_l receives m_1, m_2, and m_3, P_l computes $k_1 = m_1 + \mathbf{l} \cdot \langle N, 1\rangle$, $k_2 = m_2 + \mathbf{l} \cdot \langle N, 2\rangle$, and $k_3 = m_3 + \mathbf{l} \cdot \langle N, 3\rangle$;
3) By using k_1, k_2, and k_3, P_l outputs a message by examining the following conditions:
   a) If $k_1 = 0$, then P_l outputs "unchanged";
   b) If $k_1 \neq 0$ but $k_1 k_2 = k_3$, then P_l outputs the position corresponding to k_1; Note that this condition means $H(\mathbf{r}, \mathbf{l})$ is equal to one;
   c) If $k_1 \neq 0$ nor $k_1 k_2 \neq k_3$, then P_l seeks a, b in $GF(2^m)$ where (9) holds, then outputs the two positions corresponding to a and b.

$$\begin{cases} a + b = k_1, \\ a^2 + b^2 = k_2, \\ a^3 + b^3 = k_3. \end{cases} \tag{9}$$

Theorem 5. Protocol 3 can computes the positions of changed bits collectly.

Proof. Let us consider separately the cases of the value of the hamming distance:

In the case that $H(\mathbf{r}, \mathbf{l})$ is zero. that is, $\mathbf{r} = \mathbf{l}$:

According to (10), we can see that k_1 is always zero.

$$\begin{aligned} k_1 &= m_1 + \mathbf{l} \cdot \langle N, 1\rangle \\ &= \mathbf{r} \cdot \langle N, 1\rangle + \mathbf{l} \cdot \langle N, 1\rangle \\ &= (\mathbf{r} + \mathbf{l}) \cdot \langle N, 1\rangle \\ &= \mathbf{0} \cdot \langle N, 1\rangle \\ &= 0. \end{aligned} \tag{10}$$

Then, in Protocol 3, P_l outputs "unchanged" when condition 3a is examined.

In the case that $H(\mathbf{r}, \mathbf{l})$ is one:

Assume that the bit streams differ at, say, j-th bit. Then, we have (11).

$$\begin{aligned} k_1 &= (\mathbf{r} + \mathbf{l}) \cdot \langle N, 1\rangle = j \\ k_2 &= (\mathbf{r} + \mathbf{l}) \cdot \langle N, 2\rangle = j^2, \\ k_3 &= (\mathbf{r} + \mathbf{l}) \cdot \langle N, 3\rangle = j^3. \end{aligned} \tag{11}$$

In this case, since j is not zero, Condition 3a does not hold. On the other hand, since $k_1 k_2 = j \cdot j^2 = j^3 = k_3$ holds, Condition 3b still holds. Then, in Protocol 3, P_l can detect the position of the changed bit as $k_1 = j$ collectly.

In the case that $H(\mathbf{r}, \mathbf{l})$ is two:

Assume that the bit streams differ at, say, i-th and j-th bits, where $i \neq j$ nor $ij \neq 0$.

Then, we have (12).

$$\begin{aligned} k_1 &= (r + l) \cdot \langle N, 1\rangle = i + j, \\ k_2 &= (r + l) \cdot \langle N, 2\rangle = i^2 + j^2, \\ k_3 &= (r + l) \cdot \langle N, 3\rangle = i^3 + j^3. \end{aligned} \tag{12}$$

Since $i \neq j$ implies $i + j \neq 0$, Condition 3a does not hold.

On the other hand, Condition 3b yields (13).

$$\begin{aligned} k_1 k_2 - k_3 &= (i + j) \cdot (i^2 + j^2) - (i^3 + j^3) \\ &= i^2 j + i j^2 = ij(i + j). \end{aligned} \tag{13}$$

In order that (13) is equal to 0, $i + j$ must be zero, while $ij \neq 0$. This implies that i must be equal to j. However, this contradicts the assumption. Therefore, Condition 3b does not hold ether.

Hence, P_l enters Condition 3c, and seeks a and b where (9) holds. Assume that there exists a pair of solution, say, i and j.

Now, we show that the different elements, say, a and b from i and j can not satisfy (9).

From $i + j = a + b = k_1$, we find that b is $i + a + j$.

$$\begin{aligned} &(i^3 + j^3) - (a^3 + b^3) \\ &= i^3 + j^3 + a^3 + (i + j + a)^3 \\ &= i^3 + j^3 + a^3 + i^3 + j^3 + a^3 \\ &\quad + i^2 j + i j^2 + a i^2 + a^2 i + a^2 j + a j^2 \\ &= (i + a)(a + j)(i + j) \end{aligned} \tag{14}$$

Then, if two pair of elements a and b satisfy (9), $i^3 + j^3$ and $a^3 + b^3$ must equal. However, since (14) must be zero, a should be equal to either i or j, while $i \neq j$. However, by using k_1, if $a = i$ then $b = j$, otherwise, if $a = j$ then $b = i$. This contradicts the assumption. That is, if there is a solution, the solution is unique.

Therefore, in every case, Protocol 3 can determine the position of the changed bits.

Next, we propose an effective algorithm that seeks a and b where (9) holds. According to Theorem 5, we have already known that we can find a and b by searching all exhaustively.

Here, we propose a probabilistic algorithm with polynomial time for $\log N$.

Algorithm 4

1) Compute $s = k_2 - k_3/k_1$ over $GF(2^m)$;
2) Solve the quadratic equation (15) over $GF(2^m)$ by using Itoh's algorithm [5];

$$x^2 + k_1 x + s = 0. \tag{15}$$

3) The solution, say, $x = a, b$ is corresponding to the positions of the changed bit.

Note that we quote Itoh's algorithm [5] as the following:

1) input: $f(x) = x^2 + ax + b$ over $GF(2^m)$;
2) Choose $c \in GF(2^m)$ at random($c \neq 0$);
3) if $Tr(ca) = 0$ then return to 2), where $Tr(x) = x + x^2 \dots + x^{2m-1}$;
4) Compute $g(x) = GCD(f(c^{-1}x, Tr(x))$;
5) output $c^{-1}d$ where $g(x) = x - d$.

Rabin and Itoh developed algorithms to solve quadratic equation over finite fields [5], [6]. Notice that Itoh's algorithm is some constant rate faster than Rabin's one.

Lemma 6. Algorithm 4 can computes a and b collectly in average polynomial time for *m*.

Proof. Let the solution be, say, *i* and *j*.

Then, we know that *s* is *ij*, according to (16).

$$\begin{aligned} s &= k_2 - k_3/k_1 \\ &= (i^2 + j^2) - (i^3 + j^3)/(i + j) \\ &= (i^2 + j^2) - (i^2 + ij + j^2) \\ &= ij. \end{aligned} \tag{16}$$

Then, (15) can be factorized as (17).

$$\begin{aligned} x^2 + k_1 x + s &= x^2 + (i + j)x + ij \\ &= (x + i)(x + j). \end{aligned} \tag{17}$$

Thus, we can see that (15) is the quadratic equation that has the solution as the positions of the changed bits.

Since *s* can be found in polynomial time for *m* and Itoh's algorithm can be proceed in average polynomial time for *m*, then algorithm 4 can be proceeded in average polynomial time for *m*.

Theorem 7. There exists a protocol solving 2-Monitoring Remote Data Problem, where the number of message bits is $O(\log_2 N)$, the time to compute the communication message is $O(N \log_2 N)$, and the average time to compute the position of the changed bit is $O(N \log_2 N)$.

V. Conclusion

In the area of Communication Complexity, the problem whether two data that are placed separately are the same or not is studied. In this study, we propose *k*-Monitoring Remote Data Problem where two input data differ in at most *k* bits and a player can use one data. Then, we propose a protocols for 1-Monitoring Remote Data Problem where the number of message bits is $O(\log_2 N)$, the time to compute the communication message is $O(N \log_2 N)$, and the time to compute the position of the changed bit is $O(N \log_2 N)$. Moreover, we propose a protocols for 2-Monitoring Remote Data Problem where the number of message bits is $O(\log_2 N)$, the time to compute the communication message is $O(N \log_2 N)$, and the average time to compute the position of the changed bit is $O(N \log_2 N)$.

Now we propose the following conjecture:

Conjecture 8. There exists a protocol solving k-Monitoring Remote Data Problem, where the number of message bits is $O(\log_2 N)$, the time to compute the communication message is $O(N \log_2 N)$, and the average time to compute the position of the changed bit is $O(N \log_2 N)$.

References

[1] R. Rivest. (Apr. 1992). The MD4 Message-Digest Algorithm. RFC 1320 (Informational), Internet Engineering Task Force. [Online]. Available: http://www.ietf.org/rfc/rfc1320.txt

[2] A. C.-C. Yao, "Some complexity questions related to distributive computing (preliminary report)," in *Proc. the Eleventh Annual ACM Symposium on Theory of Computing*, New York, NY, USA: ACM, 1979, pp. 209-213.

[3] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," in *Proc.* the 1988 ACM SIGMOD International Conference on Management of Data, New York, NY, USA: ACM, 1988, pp. 109-116.

[4] B. Dawkins and A. Jones. Common RAID Disk Data Format Specification, SNIA Technical Position. [Online]. Available: http://www.snia.org/sites/default/files/SNIA DDF Technical Position v2.0.pdf

[5] T. Itoh, "Efficient probabilistic algorithm for solving quadratic equations over finite fields," *Electronics Letters*, vol. 23, pp. 869-870, 1987.

[6] M. O. Rabin and M. O. Rabin, "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, vol. 9, pp. 273-280, 1979.

Naoshi Sakamoto was born in Japan in 1964. He graduated from the University of Electro-Communication in 1987, and received his M.S. and D.S. degrees from Tokyo Institute of Technology in 1989 and 2001, respectively.

From 1992 to 1997, he was a research assistant of Computer Center of Hitotsubashi University. From 1997 to 2001, he was a research assistant at Tokyo Institute of Technology. Since 2001, he has been an associate professor at Tokyo Denki University.

He got the 1999 IEICE excellent paper award. His research interests are distributed algorithms, randomized algorithms, and complexity theory.