

# Simplified Verifiable Re-encryption Mix-nets

Shinsuke Tamura\*, Shuji Taniguchi

Graduate School of Engineering, University of Fukui, Fukui, Japan

\*Corresponding author: [tamura@u-fukui.ac.jp](mailto:tamura@u-fukui.ac.jp)

Received December 25, 2012; Revised February 19, 2013; Accepted March 07, 2013

**Abstract** Under the assumption that numbers of data that are encrypted and decrypted are sufficiently large and final decryption results of individual data can be publicly disclosed, a simplified mechanism for implementing re-encryption type verifiable mix-nets is proposed. Different from already proposed mechanisms, in which mix-servers prove their honest encryptions while concealing their encryption parameters, mix-servers in the proposed scheme simply disclose their aggregate encryption parameter values. As a consequence anyone can verify encryption results without interacting with mix-servers. Also, its primary verification procedures examine only aggregate behavior of mix-servers, in other words, it does not examine correct encryptions of individual data. Therefore computation volumes required for mix-servers to prove their correct behaviors are reduced substantially. In addition, the proposed scheme can cope with various attacks from malicious entities more effectively than optimistic verifiable mix-nets that also examine only aggregate behaviors of mix-nets.

**Keywords:** anonymous communication, privacy, e-voting systems, e-poll systems

## 1. Introduction

Verifiable mix-net  $M$  consists of multiple mutually independent mix-servers that shuffle, encrypt and decrypt given data so that no one can know correspondences between inputs and outputs of  $M$  while ensuring the honest handlings of the data, and it plays important roles in e-voting systems, i.e. when  $M$  shuffles, encrypts and decrypts votes put in it, voters can conceal their votes from others so that they can preserve their privacies and protect themselves from coercers. In addition participants of elections can convince themselves that individual votes are correctly counted [13,14,15]. Same kind of situations occur also in other important applications e.g. in e-poll systems, and various schemes had been proposed to implement verifiable mix-nets [5-10,12]. However, to convince participants honest behaviors of mix-nets while preserving privacies of data owners many existing mechanisms rely on zero knowledge proofs (ZKPs) [2,3] that require numbers of interactions between mix-nets and verifiers so that mix-servers can conceal their encryption parameters. Also in cases where participants cannot completely trust verification results made by other entities, mix-nets must prove their honesties to participants individually. In addition, in some schemes behaviors of mix-servers about honest handlings of data must be examined individually, and they are not efficient enough to handle large volume of data.

To make verifiable mix-nets practical enough for even large-scale applications, this paper proposes a simplified mechanism for ensuring honest behaviors of mix-nets under the assumption that numbers of data are sufficiently large and final decryption results of individual data can be disclosed (this assumption is satisfied in e-voting systems,

i.e. in e-voting systems individual votes are finally disclosed so that participants can know election winners). In the proposed mechanism, to prove honest behaviors of mix-nets, individual mix-servers simply disclose their encryption parameters in their aggregated forms. Also, the mechanism examines only aggregated behaviors of mix-servers; in other words, it does not examine individual data. Therefore, complicated verification processes can be removed, and more importantly, because mix-servers do not need to participate in verification processes anyone can easily verify behaviors of mix-nets while examining them by itself without interacting with mix-nets. Then the mechanism works efficiently even in environments where participants cannot trust verification results made by other entities. In addition, when compared with optimistic verifiable mix-nets that also examine only aggregated behaviors of mix-servers [7], the proposed mechanism handles various attacks from malicious entities more effectively.

## 2. Re-encryption Mix-nets

Re-encryption type mix-net  $M$  consists of multiple mutually independent mix-servers  $S_1, S_2, \dots, S_Q$  that are arrayed in re-encryption and decryption stages as shown in Figure 1, so that entities  $P_1, P_2, \dots, P_N$  that put their data  $D_1, D_2, \dots, D_N$  can conceal correspondences between the data and outputs of  $M$  from anyone including even  $P_1, P_2, \dots, P_N$  themselves. In the figure,  $\{y_q = g^{x_q \bmod p}, x_q\}$  is mix-server  $S_q$ 's encryption and decryption key pair of an ElGamal encryption function, where integers  $g$  and  $p$  are publicly known and common to all data, and encryption key  $y_q$  is publicly disclosed but decryption key  $x_q$  is the secret of  $S_q$ .  $y_1 y_2 \dots y_Q = g^{x_1 + x_2 + \dots + x_Q \bmod p} = g^{x^* \bmod p} = y^*$  is a common public encryption key.

Firstly in the re-encryption stage, each  $P_j$  encrypts its data  $D_j$  by publicly known common encryption key  $y^*$  while using its secret integer  $a_j$ , i.e.  $D_j$  is encrypted to  $\{g^{a_j} \bmod p, D_j y^{*a_j} \bmod p\}$ . Then, mix-servers  $S_1, S_2, \dots, S_Q$  repeatedly re-encrypt  $\{g^{a_j} \bmod p, D_j y^{*a_j} \bmod p\}$  to  $\{g^{a_j+k_{1j}+k_{2j}+\dots+k_{Qj}} \bmod p, D_j y^{*a_j+k_{1j}+k_{2j}+\dots+k_{Qj}} \bmod p = D_j y^{*k_j^{*(Q)}} \bmod p\}$ , while generating their secret integers  $k_{1j}, k_{2j}, \dots, k_{Qj}$ . Namely,  $S_1$  calculates  $\{g^{a_j+k_{1j}} \bmod p = g^{k_j^{*(1)}} \bmod p, D_j y^{*a_j+k_{1j}} \bmod p = D_j y^{*k_j^{*(1)}} \bmod p\}$  based on  $\{g^{a_j} \bmod p, D_j y^{*a_j} \bmod p\}$  received from  $P_j$  and forwards it to  $S_2$  while shuffling it with other re-encryption results,  $S_2$  calculates  $\{g^{a_j+k_{1j}+k_{2j}} \bmod p = g^{k_j^{*(2)}} \bmod p, D_j y^{*a_j+k_{1j}+k_{2j}} \bmod p = D_j y^{*k_j^{*(2)}} \bmod p\}$  from  $\{g^{k_j^{*(1)}} \bmod p, D_j y^{*k_j^{*(1)}} \bmod p\}$  and forwards it to  $S_3$  of course while shuffling it with other results, and finally  $S_Q$  calculates  $\{g^{k_j^{*(Q)}} \bmod p, D_j y^{*k_j^{*(Q)}} \bmod p\}$  to put the result in the 1st bulletin board (BB). Where, BB is a memory of which contents can be read by anyone at any time. In the remainder, notation (mod  $p$ ) is omitted.

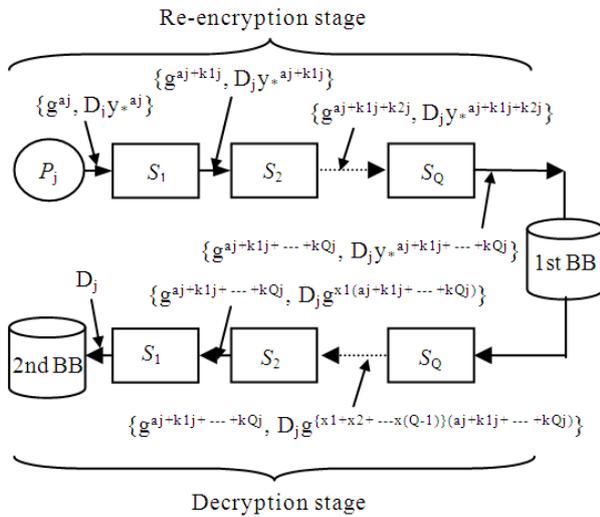


Figure 1. Re-encryption Mix-net

In the decryption stage, mix-servers  $S_Q, S_{Q-1}, \dots, S_1$  simply decrypts the re-encrypted form  $\{g^{k_j^{*(Q)}}, D_j y^{*k_j^{*(Q)}}\}$  to  $D_j$  by their secret decryption keys  $x_Q, x_{Q-1}, \dots, x_1$ , i.e. from  $\{g^{k_j^{*(Q)}}, D_j y^{*k_j^{*(Q)}}\}$ ,  $S_Q$  calculates  $\{g^{k_j^{*(Q)}}, D_j y^{*k_j^{*(Q)}}/g^{(k_j^{*(Q)})(x_Q)}\} = \{g^{k_j^{*(Q)}}, D_j g^{(x_1+\dots+x_{(Q-1)})k_j^{*(Q)}}\}$  by using its secret decryption key  $x_Q$ ,  $S_{Q-1}$  that knows its secret key  $x_{Q-1}$  calculates  $\{g^{k_j^{*(Q)}}, D_j g^{(x_1+\dots+x_{(Q-1)})k_j^{*(Q)}}/g^{(k_j^{*(Q)})(x_{Q-1})}\} = \{g^{k_j^{*(Q)}}, D_j g^{(x_1+\dots+x_{(Q-2)})k_j^{*(Q)}}\}$ , and  $S_{Q-2}, \dots, S_1$  repeat the same operations until  $S_1$  calculates  $D_j g^{(x_1)(k_j^{*(Q)})}/g^{(k_j^{*(Q)})(x_1)} = D_j$  to put the result in the 2nd BB.

Here, although it is assumed that individual decryption results are publicly disclosed in the 2nd BB, no one except  $P_j$  can know  $D_j$  that  $P_j$  owns unless all mix-servers conspire because anyone does not know all decryption keys  $x_1, x_2, \dots, x_Q$ , in addition, no one including  $P_j$  itself can know the correspondence between  $D_j$  owned by  $P_j$  and  $\{g^{k_j^{*(Q)}}, D_j y^{*k_j^{*(Q)}}\}$  in the 1st BB or decryption result  $D_j$  in the 2nd BB either except the case where  $D_j$  is unique to  $P_j$  because each  $S_q$  in the re-encryption stage shuffles  $\{g^{k_j^{*(q)}}, D_j y^{*k_j^{*(q)}}\}$  with other re-encryption results.

It must be noted that mix-servers in the decryption stage do not shuffle their decryption results, i.e. even if decryption results are shuffled anyone can identify the correspondence between input  $\{g^{k_j^{*(Q)}}, D_j g^{(x_1+\dots+x_{(Q-1)})k_j^{*(Q)}}\}$  and output  $\{g^{k_j^{*(Q)}}, D_j g^{(x_1+\dots+x_{(Q-1)})k_j^{*(Q)}}\}$  of  $S_q$  by tracing

$g^{k_j^{*(Q)}}$  included in them. Also, although mix-servers in Figure 1 are arrayed as  $S_Q, S_{Q-1}, \dots, S_1$  in the decryption stage, they can be arrayed in an arbitrary order because ElGamal encryption functions are commutative. Moreover,  $S_1, S_2, \dots, S_Q$  in the re-encryption stage can be replaced with any entities because anyone knows common encryption key  $y^*$ .

### 3. An Approach to Verifying Honest Behaviors of Mix-servers

A problem in re-encryption mix-nets is that because no one knows all decryption keys and each integer  $k_{qj}$  is a secret of mix-server  $S_q$  no single entity can notice even if mix-servers carry out their re-encryptions or decryptions dishonestly, despite that each  $S_q$  in the re-encryption or the decryption stage can easily forge consistent re-encryption or decryption forms arbitrarily (encryption keys  $y_1, y_2, \dots, y_Q$  and  $y^*$  are publicly known). But fortunately, this problem can be solved as follows [5-10]. Namely, if  $S_q$  had honestly calculated  $\{g^{k_j^{*(q)}}, D_j y^{*k_j^{*(q)}}\}$  from  $\{g^{k_j^{*(q-1)}}, D_j y^{*k_j^{*(q-1)}}\}$ ,  $\alpha = g^{k_j^{*(q)}/g^{k_j^{*(q-1)}}} = g^{k_{qj}}$  and  $\beta = D_j y^{*k_j^{*(q)}}/(D_j y^{*k_j^{*(q-1)}}) = y^{*k_{qj}}$  must be equal to  $g$  and  $y^*$  to the power of same  $S_q$ 's secret integer  $k_{qj}$  (in the remainder, pair  $\{\alpha, \beta\}$  with this property is called DDH pair). In other words, verifier  $V$  that verifies honest behaviors of mix-servers can confirm that  $S_q$  had honestly calculated re-encrypted form  $\{g^{k_j^{*(q)}}, D_j y^{*k_j^{*(q)}}\}$  by examining whether  $S_q$  knows  $k_{qj}$  that satisfies relations  $\alpha = g^{k_{qj}}$  and  $\beta = y^{*k_{qj}}$  or not.

Usually ZKP schemes are used to enable  $S_q$  to convince verifier  $V$  it knows  $k_{qj}$  that makes  $\{\alpha, \beta\}$  a DDH pair without disclosing  $k_{qj}$  itself, but ZKPs require numbers of challenges and responses between  $V$  and mix-servers which degrade performances of the mix-net. In addition,  $S_q$  must prove its correct operations to participants individually if the participants cannot trust  $V$ . To simplify verification procedures and remove necessity that  $S_q$  must prove its honesty to participants individually, each mix-server  $S_q$  in the proposed mechanism simply discloses its secrets  $k_{q1}, k_{q2}, \dots, k_{qN}$  in their aggregated form.

Firstly, it must be noted that if  $\{g^{k_{qj}}, y^{*k_{qj}}\}$  and  $\{g^{k_{qh}}, y^{*k_{qh}}\}$  are DDH pairs their product  $\{g^{k_{qj}+k_{qh}}, y^{*k_{qj}+k_{qh}}\}$  is also a DDH pair. Therefore product of all inputs and that of all outputs of mix-server  $S_q$  must constitute a DDH pair when  $S_q$  is honest. Then, as in optimistic verifiable mix-nets [7], provided that  $\kappa_q = k_{q1}+k_{q2}+\dots+k_{qN}$ , the proposed mechanism calculates the product of all inputs of  $S_q$ , i.e.  $\{g^{k_1^{*(q-1)}+k_2^{*(q-1)}+\dots+k_N^{*(q-1)}}, D_1 D_2 \dots D_N y^{*k_1^{*(q-1)}+k_2^{*(q-1)}+\dots+k_N^{*(q-1)}}\}$ , and that of all outputs of  $S_q$ , i.e.  $\{g^{k_1^{*(q)}+k_2^{*(q)}+\dots+k_N^{*(q)}}, D_1 D_2 \dots D_N y^{*k_1^{*(q)}+k_2^{*(q)}+\dots+k_N^{*(q)}}\}$ , to examine whether  $\alpha = g^{k_1^{*(q)}+k_2^{*(q)}+\dots+k_N^{*(q)}/g^{k_1^{*(q-1)}+k_2^{*(q-1)}+\dots+k_N^{*(q-1)}} = g^{\kappa_q}$  and  $\beta = D_1 D_2 \dots D_N y^{*k_1^{*(q)}+k_2^{*(q)}+\dots+k_N^{*(q)}}/D_1 D_2 \dots D_N y^{*k_1^{*(q-1)}+k_2^{*(q-1)}+\dots+k_N^{*(q-1)}} = y^{*\kappa_q}$  constitute a DDH pair or not, and when  $S_q$  is determined as dishonest, it is forced to re-encrypt its inputs again.

An important thing is although integers  $k_{q1}, k_{q2}, \dots, k_{qN}$  are secrets of  $S_q$ ,  $S_q$  can disclose their sum  $\kappa_q$  if  $N$  (the number of data) is sufficiently large. Namely, because  $S_q$  assigns different secret values to  $k_{q1}, k_{q2}, \dots, k_{qN}$ , it is computationally infeasible for entities other than  $S_q$  to know  $k_{qj}$  from  $\kappa_q$ . Then, each mix-server  $S_q$  in the proposed mechanism simply discloses sum  $\kappa_q$  when it had

re-encrypted all inputs. As a consequence, without the participation of  $S_q$ , anyone can convince itself that  $S_q$  is honest by calculating  $g^{kq}$  and  $y^{*kq}$  based on  $k_q$  that  $S_q$  had disclosed and by examining relations  $\alpha = g^{kq}$  and  $\beta = y^{*kq}$ . Here, the assumption that  $N$  is sufficiently large is essential, i.e. if  $N$  is small entities can easily identify correspondences between input  $\{g^{k^*(q-1)}, D_{jy^*}^{k^*(q-1)}\}$  and output  $\{g^{k^*(q)}, D_{jy^*}^{k^*(q)}\}$ , in other words, can know  $k_{q1}, k_{q2}, \dots, k_{qN}$  by calculating  $Y(j, h) = g^{kh^*(q)}/g^{k^*(q-1)}$  for each possible pair  $(j, h)$  to examine whether relation  $y^{*kq} = Y(1, h_1)Y(2, h_2) \dots Y(N, h_N)$  holds or not. But it is practically impossible when  $N$  is large enough, i.e. the number of combinations is  $N!$ .

In the above, if the product of all inputs and that of all outputs of whole mix-net  $M$  are examined, it is not necessary for  $V$  to verify mix-servers individually, but examinations of individual mix-servers make dishonest entity identification processes simple as will be discussed in Sec. 5.

## 4. Detecting Disrupted Data

It must be noted that if input and output pairs of mix-net  $M$  are not examined individually as in the previous section mix-server  $S_q$  can disrupt data without being detected by others. Namely, even if  $S_q$  re-encrypts 2 inputs  $\{g^{k^*(q-1)}, D_{jy^*}^{k^*(q-1)}\}$  and  $\{g^{kh^*(q-1)}, D_{hy^*}^{kh^*(q-1)}\}$ , which correspond to data  $D_j$  and  $D_h$  owned by entities  $P_j$  and  $P_h$ , to  $\{g^{k^*(q)}, UD_{jy^*}^{k^*(q)}\}$  and  $\{g^{kh^*(q)}, (D_h/U)y^{*kh^*(q)}\}$  while generating integer  $U$  arbitrarily, pair  $\{g^{k^*(q)+kh^*(q)}/g^{k^*(q-1)+kh^*(q-1)} = g^{kq+khq}, (UD_j)(D_h/U)y^{*k^*(q)+kh^*(q)}/D_jD_hy^{*k^*(q-1)+kh^*(q-1)} = y^{*kq+khq}\}$  is still a DDH pair. In the same way, even if  $S_q$  re-encrypts the above 2 inputs into  $\{g^{k^*(q)}, D_{jy^*}^{k^*(q-1)}y^{*kq}\}$  and  $\{g^{kh^*(q)}, D_{hy^*}^{kh^*(q-1)}y^{*kq}\}$ , pair  $\{g^{k^*(q)+kh^*(q)}/g^{k^*(q-1)+kh^*(q-1)} = g^{kq+khq}, D_jD_hy^{*k^*(q-1)+kh^*(q-1)+kq}/D_jD_hy^{*k^*(q-1)+kh^*(q-1)} = y^{*kq+khq}\}$  is still a DDH pair. These dishonesties of  $S_q$  can be detected when individual decryption results are disclosed in the 2nd BB, i.e. decryption results  $(UD_j), (D_h/U), D_{jy^*}^{kq+khq}$  or  $D_{hy^*}^{kq+khq}$  may become meaningless. But the probability that they have meanings is not 0. In addition more seriously,  $P_j$  and  $P_h$ , the owners of data  $D_j$  and  $D_h$ , may put meaningless data from the beginning. Therefore, a mechanism that enables any entity to determine whether the decryption results are disrupted by mix-servers or not is necessary.

In an optimistic verifiable mix-net [7], to detect disrupted decryption results, entity  $P_j$  encrypts  $D_j$  to  $\{g^{aj}, D_{jy^*}^{aj}\} = \{\omega, \zeta\}$  and calculates hash value  $h(\omega, \zeta) = H_j$  as a check-code for verifying validity of the decryption result, then constructs triple  $\Pi(D_j) = \langle \{g^{aj}, \omega y^{*aj}\}, \{g^{aj}, \zeta y^{*aj}\}, \{g^{aj}, H_j y^{*aj}\} \rangle$ , in other words, encrypts elements in triple  $\{\omega, \zeta, H_j\}$  individually. After that, mix-servers re-encrypts and decrypts  $\Pi(D_j)$  instead of  $\{\omega, \zeta\}$ , and they convince others that decrypted results are not disrupted by showing consistent check-code  $H_j$  in each decrypted form  $\Pi(D_j)$ , i.e.  $h(\omega, \zeta)$  is a one-way function and no one other than  $P_j$  can generate consistent triple  $\{\omega, \zeta, H_j\}$ . Then finally,  $\{\omega, \zeta\}$  is decrypted to  $D_j$ .

The proposed scheme exploits unknown random numbers generated before the re-encryption stage as shown in Figure 2. When compared with optimistic verifiable mix-nets, an advantage is a check-code attached to data  $D_j$  is calculated from  $D_j$  itself instead of its

encrypted form  $\{g^{aj}, D_{jy^*}^{aj}\}$ , and by this feature the proposed mechanism can easily avoid relation attacks [4, 11], which optimistic verifiable mix-nets cannot cope with well, as discussed in Sec. 6. Here, anyone can confirm the honest calculation of the check-code while not knowing  $D_j$  itself without using additional mechanisms such as ZKPs as will be discussed later.

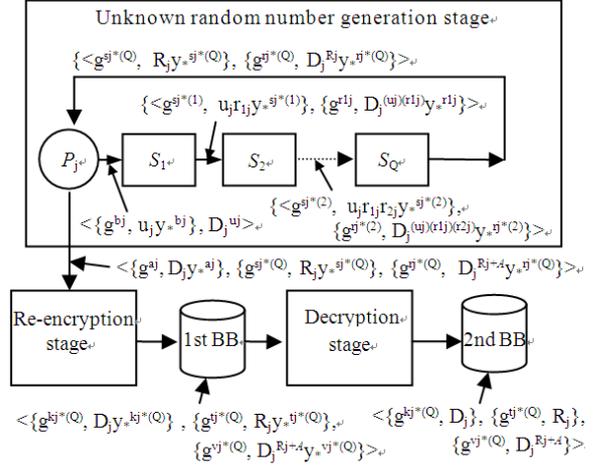


Figure 2. Unknown random number generation stage

Firstly, entity  $P_j$ , the owner of data  $D_j$ , generates its secret integers  $u_j$  and  $b_j$ , encrypts  $u_j$  to  $\{g^{bj}, u_j y^{*bj}\}$  and calculates  $D_j^{uj}$ , and mix-server  $S_1$  that receives pair  $\langle \{g^{bj}, u_j y^{*bj}\}, D_j^{uj} \rangle$  from  $P_j$  (it must be noted that under the strong RSA assumption it is computationally infeasible for  $S_1$  to know  $D_j$  or  $u_j$  from  $D_j^{uj}$ ) generates its secret integers  $r_{1j}$  and  $s_{1j}$  and calculates pair  $\langle \{g^{bj+s_{1j}}, u_j r_{1j} y^{*bj+s_{1j}} = u_j r_{1j} y^{*s_{1j}^{(1)}}\}, \{g^{r_{1j}}, (D_j^{uj} y^{*r_{1j}})^{s_{1j}} = D_j^{(uj)(r_{1j})} y^{*r_{1j} s_{1j}}\} \rangle$ . In the same way,  $S_q$  that receives pair  $\langle \{g^{bj+s_{1j}+\dots+s_{(q-1)j}}, (u_j r_{1j} \dots r_{(q-1)j}) y^{*bj+s_{1j}+\dots+s_{(q-1)j}} = (u_j r_{1j} \dots r_{(q-1)j}) y^{*s_{qj}^{(q-1)}}\}, \{g^{r_{1j}+\dots+r_{(q-1)j}}, (D_j^{(uj)(r_{1j}) \dots (r_{(q-1)j})})^{s_{qj}^{(q-1)}} = D_j^{(uj)(r_{1j}) \dots (r_{(q-1)j})} y^{*r_{1j} \dots r_{(q-1)j} s_{qj}^{(q-1)}} \rangle$  from  $S_{q-1}$  calculates  $\langle \{g^{bj+s_{1j}+\dots+s_{qj}}, (u_j r_{1j} \dots r_{(q-1)j}) y^{*bj+s_{1j}+\dots+s_{qj}} = (u_j r_{1j} \dots r_{(q-1)j}) y^{*s_{qj}^{(q)}}\}, \{g^{r_{1j}+\dots+r_{qj}}, (D_j^{(uj)(r_{1j}) \dots (r_{(q-1)j})})^{r_{qj}} = D_j^{(uj)(r_{1j}) \dots (r_{(q-1)j})} y^{*r_{1j} \dots r_{(q-1)j} r_{qj}} = D_j^{(uj)(r_{1j}) \dots (r_{qj})} y^{*r_{1j} \dots r_{(q-1)j} r_{qj}} \rangle$ , while generating its secret integers  $r_{qj}$  and  $s_{qj}$ , and last mix-server  $S_Q$  calculates pair  $\langle \{g^{s_Qj}, (u_j r_{1j} \dots r_{Q-1}j) y^{*s_Qj} = R_{jy^*}^{s_Qj}\}, \{g^{r_{1j}+\dots+r_{Qj}}, (D_j^{R_{jy^*}^{r_{1j}+\dots+r_{Qj}}})^{s_Qj} = D_j^{R_{jy^*}^{r_{1j}+\dots+r_{Qj}}}\} \rangle$ . Then, while using  $A$  that is a constant integer common and known to all participants,  $P_j$  puts triple  $\langle \{g^{aj}, D_{jy^*}^{aj}\}, \{g^{s_Qj}, R_{jy^*}^{s_Qj}\}, \{g^{r_{1j}+\dots+r_{Qj}}, D_j^{R_{jy^*}^{r_{1j}+\dots+r_{Qj}}}\} \rangle$  in mix-net  $M$  instead of  $\{g^{aj}, D_{jy^*}^{aj}\}$ , and based on integers  $k_{qj}, t_{qj}$  and  $v_{qj}$  which are secrets of each mix-server  $S_{qj}$ ,  $M$  re-encrypts it to triple  $\langle \{g^{aj+k_{1j}+\dots+k_{Qj}}, D_{jy^*}^{aj+k_{1j}+\dots+k_{Qj}} = D_{jy^*}^{k^*(q)}\}, \{g^{s_Qj+t_{1j}+\dots+t_{Qj}}, R_{jy^*}^{s_Qj+t_{1j}+\dots+t_{Qj}} = R_{jy^*}^{s_Qj+A}\}, \{g^{r_{1j}+\dots+r_{Qj}+v_{1j}+\dots+v_{Qj}}, D_j^{R_{jy^*}^{r_{1j}+\dots+r_{Qj}+v_{1j}+\dots+v_{Qj}}} = D_j^{R_{jy^*}^{R_{jy^*}^{r_{1j}+\dots+r_{Qj}+v_{1j}+\dots+v_{Qj}}}\} \rangle$  while shuffling it with other data. Therefore, when mix-servers in  $M$  behave honestly, the re-encrypted form of  $D_j$  is decrypted to triple  $\langle \{g^{k^*(q)}, D_j\}, \{g^{t^*(q)}, R_j\}, \{g^{v^*(q)}, D_j^{R_{jy^*}^{t^*(q)}}\} \rangle$ , i.e. regardless that  $D_j$  has meanings or not, the decrypted triple is valid only when  $D_j^{R_{jy^*}^{t^*(q)}}$  is calculated as  $D_j$  to the power of  $R_{jy^*}^{t^*(q)}$ .

As a consequence, to dishonestly re-encrypt the previous 2 triples that correspond to  $D_j$  and  $D_h$  to  $\langle \{A_1(j, q), A_2(j, q)\}, \{B_1(j, q), B_2(j, q)\}, \{C_1(j, q), C_2(j, q)\} \rangle$  and  $\langle \{A_1(h, q), A_2(h, q)\}, \{B_1(h, q), B_2(h, q)\}, \{C_1(h, q), C_2(h, q)\} \rangle$  without being detected by others, provided that they are finally decrypted to triples  $\langle \{A_1(j, Q), \Gamma_j\}, \{B_1(j, Q), \Omega_j\}, \{C_1(j, Q), \Phi_j\} \rangle$  and  $\langle \{A_1(h, Q), \Gamma_h\}, \{B_1(h, Q), \Omega_h\},$

$\{C_1(h, Q), \Phi_h\}$  respectively,  $S_q$  must define integers  $\Gamma_j$ ,  $\Omega_j$ ,  $\Phi_j$ ,  $\Gamma_h$ ,  $\Omega_h$  and  $\Phi_h$  so that relations  $\Gamma_j^{\Omega_j+A} = \Phi_j$  and  $\Gamma_h^{\Omega_h+A} = \Phi_h$  hold. But it is impossible unless  $S_q$  conspires with all other mix-servers and  $P_j$  and  $P_h$ . Namely, although  $S_q$  must know integers  $R_j = u_j r_{1j} r_{2j} \dots r_{Qj}$  and  $R_h = u_h r_{1h} r_{2h} \dots r_{Qh}$  to calculate  $\Omega_j$  and  $\Omega_h$  consistently,  $r_{zj}$ ,  $r_{zh}$ ,  $u_j$  and  $u_h$  are secrets of  $S_z$ ,  $P_j$  and  $P_h$  respectively. Here, it must be noted that the mechanism discussed in this section works when it is combined with the one discussed in Sec. 3, e.g. without the mechanism in Sec.3, mix-server  $S_q$  that knows encryption key  $y^*$  can re-encrypt the above triples that correspond to  $D_j$  and  $D_h$  at their will so that their final decryption results satisfy  $\Gamma_j^{\Omega_j+A} = \Phi_j$  and  $\Gamma_h^{\Omega_h+A} = \Phi_h$ .

But 2 problems must be taken into account. The 1st problem is even if mix-servers in the re-encryption stage had honestly re-encrypted their all inputs, each mix-server in the decryption stage can forge its output triples at their will in ways that they are finally decrypted to consistent triples because encryption keys  $y_1, y_2, \dots, y_Q$  and  $y^*$  are public known. For example,  $S_q$  that receives triple  $\langle \{g^{k_j^*(Q)}, D_j g^{(x_1 + \dots + x_Q)k_j^*(Q)}\}, \{g^{t_j^*(Q)}, R_j g^{(x_1 + \dots + x_Q)t_j^*(Q)}\}, \{g^{v_j^*(Q)}, D_j^{R_j+A} g^{(x_1 + \dots + x_Q)v_j^*(Q)}\} \rangle$  from  $S_{q+1}$  can decrypt it to  $\langle \{g^{z_1}, \Gamma_j (y_1 y_2 \dots y_{q-1})^{z_1} = \Gamma_j g^{(x_1 + \dots + x_{(q-1)})z_1}\}, \{g^{z_2}, \Omega_j g^{(x_1 + \dots + x_{(q-1)})z_2}\}, \{g^{z_3}, \Phi_j g^{(x_1 + \dots + x_{(q-1)})z_3}\} \rangle$  while arbitrarily defining integers  $z_1, z_2, z_3, \Gamma_j, \Omega_j$  and  $\Phi_j$  so that relation  $\Gamma_j^{\Omega_j+A} = \Phi_j$  holds. Also, if last mix-server  $S_Q$  that is conspiring with 1st mix-server  $S_1$  replaces one of re-encrypted result in the 1st BB with  $\Theta(D_j) = \langle \{g^{aj+\gamma}, D_j y_*^{aj+\gamma}\}, \{g^{sj^*(Q)+\delta}, R_j y_*^{sj^*(Q)+\delta}\}, \{g^{tj^*(Q)+\epsilon}, D_j^{R_j+A} y_*^{tj^*(Q)+\epsilon}\} \rangle$ , because it is decrypted to consistent value  $D_j$  and mix-servers in the decryption stage do not shuffle their decryption results,  $S_Q$  can know  $D_j$ , the secret of  $P_j$  ( $S_Q$  can easily calculate  $\Theta(D_j)$  if  $S_1$  informs it of  $\langle \{g^{aj}, D_j y_*^{aj}\}, \{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}, \{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\} \rangle$  put by  $P_j$ ). Here, it may be possible to make each encryption key  $y_q$  as  $S_q$ 's secret, but common encryption key  $y^*$  must be publicly disclosed, and at least 1st mix-server  $S_Q$  in the decryption stage can dishonestly forge its decryption results at its will as above. Therefore, honest behaviors of the decryption stage also must be verified.

Correct decryptions of data in the decryption stage also can be verified in the same way as in Sec. 3. Firstly  $S_1$  a representative of mix-servers calculates  $g^{a_1} g^{a_2} \dots g^{a_N} = g^{a_1+a_2+\dots+a_N}$  based on inputs from  $P_1, P_2, \dots, P_N$ , and asks each  $S_q$  to calculate and disclose  $G_q = g^{(a_1+a_2+\dots+a_N)x_q} = y_q^{a_1+a_2+\dots+a_N}$ . After that verifier  $V$  calculates  $G_q y_q^{k_1+k_2+\dots+k_N} = y_q^{a_1+a_2+\dots+a_N+k_1+k_2+\dots+k_N} = y_q^{k_1^*(Q)+k_2^*(Q)+\dots+k_N^*(Q)}$  for each  $q$ , where  $V$  can know each sum  $k_q = k_{q1}+k_{q2}+\dots+k_{qN}$  because it is already disclosed in the re-encryption stage by  $S_q$ , also entities other than  $P_j$  cannot know  $y_*^{aj}$  from  $g^{a_1+a_2+\dots+a_N}$ . Then, it is straightforward for anyone to determine whether  $\phi_{q-1} = D_1 D_2 \dots D_N g^{(x_1 + \dots + x_{(q-1)}) (k_1^*(Q)+k_2^*(Q)+\dots+k_N^*(Q))}$ , a product of all decryption results of  $S_q$ , is honestly calculated from  $\phi_q = D_1 D_2 \dots D_N g^{(x_1 + \dots + x_q) (k_1^*(Q)+k_2^*(Q)+\dots+k_N^*(Q))}$ , a product of all inputs of  $S_q$ , or not. Namely, it is enough to confirm relations  $\phi_q/\phi_{q-1} = g^{x_q (k_1^*(Q)+k_2^*(Q)+\dots+k_N^*(Q))} = G_q y_q^{k_1+k_2+\dots+k_N}$  and  $g^{k_1^*(Q)+k_2^*(Q)+\dots+k_N^*(Q)} = G_q g^{k_1+k_2+\dots+k_N}$ . In the same way, anyone easily can verify decryption results of  $\{g^{t_j^*(Q)}, R_j g^{(x_1 + \dots + x_Q)t_j^*(Q)}\}$  and  $\{g^{v_j^*(Q)}, D_j^{R_j+A} g^{(x_1 + \dots + x_Q)v_j^*(Q)}\}$ , and when  $S_q$  is dishonest, it is forced to decrypt its inputs again as in the re-encryption stage.

$S_q$  in the decryption stage also can dishonestly decrypt 2 triples  $\langle \{g^{k_j^*(Q)}, D_j g^{(x_1 + \dots + x_Q)k_j^*(Q)}\}, \{g^{t_j^*(Q)}, R_j g^{(x_1 + \dots + x_Q)t_j^*(Q)}\}, \{g^{v_j^*(Q)}, D_j^{R_j+A} g^{(x_1 + \dots + x_Q)v_j^*(Q)}\} \rangle$  and  $\langle \{g^{k_h^*(Q)}, D_h g^{(x_1 + \dots + x_Q)k_h^*(Q)}\}, \{g^{t_h^*(Q)}, R_h g^{(x_1 + \dots + x_Q)t_h^*(Q)}\}, \{g^{v_h^*(Q)}, D_h^{R_h+A} g^{(x_1 + \dots + x_Q)v_h^*(Q)}\} \rangle$  to  $\langle \{g^{k_j^*(Q)}, \Gamma_j(q-1)\}, \{g^{t_j^*(Q)}, \Omega_j(q-1)\}, \{g^{v_j^*(Q)}, \Phi_j(q-1)\} \rangle$  and  $\langle \{g^{k_h^*(Q)}, \Gamma_h(q-1)\}, \{g^{t_h^*(Q)}, \Omega_h(q-1)\}, \{g^{v_h^*(Q)}, \Phi_h(q-1)\} \rangle$  that are finally decrypted into  $\langle \{g^{k_j^*(Q)}, \Gamma_j\}, \{g^{t_j^*(Q)}, \Omega_j\}, \{g^{v_j^*(Q)}, \Phi_j\} \rangle$  and  $\langle \{g^{k_h^*(Q)}, \Gamma_h\}, \{g^{t_h^*(Q)}, \Omega_h\}, \{g^{v_h^*(Q)}, \Phi_h\} \rangle$  while not being detected by the above mechanism. But anyone can detect this dishonesty because  $S_q$  cannot calculate  $\Gamma_j(q-1), \Omega_j(q-1), \Phi_j(q-1), \Gamma_h(q-1), \Omega_h(q-1)$  or  $\Phi_h(q-1)$  so that the final decryption results satisfy relations  $\Gamma_j^{\Omega_j+A} = \Phi_j$  and  $\Gamma_h^{\Omega_h+A} = \Phi_h$ .

As the 2nd problem to be considered, data owner  $P_j$  itself may disrupt its data to impute the liability to mix-net  $M$ , and when  $P_j$  or  $S_q$  in the random number generation stage calculates  $\{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}$  or  $\{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\}$  in triple  $\langle \{g^{aj}, D_j y_*^{aj}\}, \{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}, \{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\} \rangle$  dishonestly, the decryption result of the triple becomes inconsistent even each mix-server in the re-encryption and the decryption stages had behaved honestly. The proposed mechanism copes with this problem by identifying entities that had behaved dishonestly as discussed in the next section, i.e. although  $P_j$  or  $S_q$  in the random number generation stage can behave dishonestly, entities liable for the dishonesties are finally revealed and they are forced to reprocess the data that correspond to the dishonesties.

Then, each mix-server can prove its honest behaviors to others while maintaining its decryption key  $x_j$  and integers  $k_{qj}, s_{qj}, t_{qj}$  and  $v_{qj}$  as its secrets. Also, although  $P_j$  in the above discloses not only  $\{g^{aj}, D_j y_*^{aj}\}$  but also  $\{g^{bj}, u_j y_*^{bj}\}, D_j^{uj}, \{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}$  and  $\{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\}$ , apparently no one other than  $P_j$  can know that  $P_j$  owns  $D_j$ . In addition even  $P_j$  itself cannot identify the correspondence between triple  $\langle \{g^{aj}, D_j y_*^{aj}\}, \{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}, \{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\} \rangle$  it had put in mix-net  $M$  and triple  $\langle \{g^{k_j^*(Q)}, D_j\}, \{g^{t_j^*(Q)}, R_j\}, \{g^{v_j^*(Q)}, D_j^{R_j+A}\} \rangle$  that  $M$  calculated except cases where  $D_j$  is unique to  $P_j$ , because no one knows integer  $R_j$ .

## 5. Identifications of Entities Liable for Data Disruptions

Entities that had behaved dishonestly can be identified easily as follows. Firstly, anyone can detect that some entity is dishonest if  $P_j$  or  $S_q$  had behaved dishonestly. namely, if  $P_j$  or  $S_q$  constructs  $\{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}$  or  $\{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\}$  dishonestly in the unknown random number generation stage, mix-net  $M$  decrypts input  $\langle \{g^{aj}, D_j y_*^{aj}\}, \{g^{sj^*(Q)}, R_j y_*^{sj^*(Q)}\}, \{g^{tj^*(Q)}, D_j^{R_j+A} y_*^{tj^*(Q)}\} \rangle$  to an inconsistent value, i.e. final decryption result  $\Psi = \langle \{g^{k_j^*(Q)}, \Gamma_j\}, \{y_*^{tj^*(Q)}, \Omega_j\}, \{g^{v_j^*(Q)}, \Phi_j\} \rangle$  does not satisfy relation  $\Gamma_j^{\Omega_j+A} = \Phi_j$ , and if  $S_q$  in the re-encryption stage re-encrypts  $\langle \{g^{k_j^*(q-1)}, D_j y_*^{k_j^*(q-1)}\}, \{g^{t_j^*(q-1)}, R_j y_*^{t_j^*(q-1)}\}, \{g^{v_j^*(q-1)}, D_j^{R_j+A} y_*^{v_j^*(q-1)}\} \rangle$  or that in the decryption stage decrypts  $\langle \{g^{k_j^*(Q)}, D_j g^{(x_1+x_2+\dots+x_Q)k_j^*(Q)}\}, \{g^{t_j^*(Q)}, R_j g^{(x_1+x_2+\dots+x_Q)t_j^*(Q)}\}, \{g^{v_j^*(Q)}, D_j^{R_j+A} g^{(x_1+x_2+\dots+x_Q)v_j^*(Q)}\} \rangle$  dishonestly, product of  $S_q$ 's inputs and that of outputs constitute inconsistent pairs or final decryption result  $\Psi$  becomes inconsistent. Here, in cases where mix-servers generate inconsistent input output

pairs, dishonest mix-servers are forced to repeat their re-encryptions or decryptions until the pairs become consistent; therefore all dishonesties are detected as inconsistent decryption results in the 2nd BB. Then, although a part of procedures require participations of mix-servers, once inconsistent decryption results are detected, verifier  $V$  can easily identify dishonest mix-servers as follows.

Firstly provided that decryption result  $\Psi$  in the 2nd BB is determined as inconsistent,  $V$  can identify dishonest entities by examining behaviors of mix-servers about only data that correspond to  $\Psi$ . Dishonest mix-servers in the decryption stage can be identified based on Diffie-Hellman scheme [1], namely, for each  $S_q$  in the decryption stage, verifier  $V$  calculates triples  $\langle D_j g^{(x_1+x_2+\dots+x_q)k_j^*(Q)} / D_j g^{\{x_1+x_2+\dots+x(q-1)k_j^*(Q)} = \eta_{qj}$ ,  $R_j g^{(x_1+x_2+\dots+x_q)t_j^*(Q)} / R_j g^{\{x_1+x_2+\dots+x(q-1)t_j^*(Q)} = \mu_{qj}$ ,  $D_j^{R_j+A} g^{(x_1+x_2+\dots+x_q)v_j^*(Q)} = \sigma_{qj}$  and  $\langle g^{k_j^*(Q)w_1}, g^{t_j^*(Q)w_2}, g^{v_j^*(Q)w_3} \rangle$  while generating its secret integers  $w_1$ ,  $w_2$  and  $w_3$ . After that  $S_q$  calculates  $(g^{k_j^*(Q)w_1})^{x_q}$ ,  $(g^{t_j^*(Q)w_2})^{x_q}$  and  $(g^{v_j^*(Q)w_3})^{x_q}$  by using its secret key  $x_q$ , and  $V$  determines  $S_q$  is honest when relations  $(\eta_{qj})^{w_1} = (g^{k_j^*(Q)w_1})^{x_q}$ ,  $(\mu_{qj})^{w_2} = (g^{t_j^*(Q)w_2})^{x_q}$  and  $(\sigma_{qj})^{w_3} = (g^{v_j^*(Q)w_3})^{x_q}$  hold. If  $S_q$  that knows  $x_q$  is honest it is easy to calculate  $(g^{k_j^*(Q)w_1})^{x_q}$ ,  $(g^{t_j^*(Q)w_2})^{x_q}$  and  $(g^{v_j^*(Q)w_3})^{x_q}$  consistently. On the other hand under the strong RSA assumption it is computationally infeasible to calculate them consistently without knowing  $w_1$ ,  $w_2$ ,  $w_3$ . By the same reason,  $V$  cannot know  $x_q$  from  $(g^{k_j^*(Q)w_1})^{x_q}$ ,  $(g^{t_j^*(Q)w_2})^{x_q}$  or  $(g^{v_j^*(Q)w_3})^{x_q}$  either.

Provided that  $D_I(j, q)$  and  $D_O(j, q)$  are input and output pair of mix-server  $S_q$  in the re-encryption stage that corresponds to inconsistent decryption result  $\Psi$ , dishonest entities in the re-encryption stage are identified as follows. Here last mix-server  $S_Q$  in the re-encryption stage can disclose its secret integers  $k_{Qj}$ ,  $t_{Qj}$ , and  $v_{Qj}$  it had used for re-encrypting  $D_I(j, Q)$  while maintaining confidentiality of individual data, i.e. no one can know other data, also integers  $k_{qj}$ ,  $t_{qj}$ , and  $v_{qj}$  are still secrets of  $S_q$  for  $q < Q$ . Therefore,  $V$  can easily determine whether  $S_Q$  is honest or not, i.e.  $S_Q$  is honest when  $D_I(j, Q)$  is certainly an output that  $S_{Q-1}$  had generated and  $D_I(j, Q)$  is successfully transformed to  $D_O(j, Q)$  by  $k_{Qj}$ ,  $t_{Qj}$ ,  $v_{Qj}$  disclosed by  $S_Q$  and public key  $y^*$ .  $V$  can determine whether  $S_q$  ( $q < Q$ ) is honest or not in the same way based on integers  $k_{qj}$ ,  $t_{qj}$  and  $v_{qj}$  disclosed by  $S_q$ . Also, even when  $V$  determines that  $S_1$  is dishonest, confidentiality of the data is still maintained because  $D_I(j, 1)$  is still encrypted by its owner  $P_j$ .

When all mix-servers in the re-encryption stage are honest,  $\Psi = \langle \{g^{k_j^*(Q)}, \Gamma_j\}, \{y_*^{t_j^*(Q)}, \Omega_j\}, \{g^{v_j^*(Q)}, \Phi_j\} \rangle$  is successfully corresponded to input  $\langle \{g^{a_j}, D_j y_*^{a_j}\}, \{g^{s_j^*(Q)}, R_j y_*^{s_j^*(Q)}\}, \{g^{r_j^*(Q)}, D_j^{R_j+A} y_*^{r_j^*(Q)}\} \rangle$  put by  $P_j$ , and to identify dishonest entities in the unknown random number generation stage,  $V$  asks  $S_Q$  again to disclose its input and output pair  $\langle G(j, Q-1), H(j, Q-1) \rangle$ ,  $\langle \{g^{s_j^*(Q)}, R_j y_*^{s_j^*(Q)}\}, \{g^{r_j^*(Q)}, D_j^{R_j} y_*^{r_j^*(Q)}\} \rangle$  in the unknown random number generation stage with its secret integers  $r_{Qj}$  and  $s_{Qj}$ , and examines whether  $\langle G(j, Q-1), H(j, Q-1) \rangle$  is a pair that was certainly calculated by  $S_{Q-1}$  and its re-encrypted form coincides with  $\langle \{g^{s_j^*(Q)}, R_j y_*^{s_j^*(Q)}\}, \{g^{r_j^*(Q)}, D_j^{R_j} y_*^{r_j^*(Q)}\} \rangle$  or not when integers  $r_{Qj}$  and  $s_{Qj}$  are applied. In the same way, each  $S_q$  discloses its input and output pair  $\langle G(j, q-1), H(j, q-1) \rangle$ ,  $\langle G(j, q), H(j, q) \rangle$  with its secret integers  $r_{qj}$  and  $s_{qj}$  when  $S_{q+1}$  is determined as honest, and  $S_{q^*}$  is identified as

dishonest when its input and output pair  $\langle G(j, q^*-1), H(j, q^*-1) \rangle$ ,  $\langle G(j, q^*), H(j, q^*) \rangle$  is inconsistent. When all mix-servers are determined as honest,  $V$  determines that  $P_j$  had put inconsistent triple in  $M$  from the beginning. About disclosures of  $r_{qj}$  and  $s_{qj}$ ,  $S_q$  defines different values for different data; therefore these disclosures bring any inconvenience to honest data owners or mix-servers.

In the above, when dishonest servers are identified in the decryption or the re-encryption stage, data  $D_j$  owned by  $P_j$  and that was re-encrypted or decrypted dishonestly can be reprocessed while preserving privacy of  $P_j$ , e.g. if  $S_q$  in the re-encryption stage is identified as a dishonest mix-server, verifier  $V$  simply asks  $S_q, S_{q+1}, \dots, S_Q$  in the re-encryption stage and  $S_Q, S_{Q-1}, \dots, S_1$  in the decryption stage to carry out their re-encryptions or decryptions again. But when dishonest mix-servers are those in the unknown random number generation stage or they include 1st mix-server  $S_1$  in the re-encryption stage, although  $P_j$  can preserve its privacy if  $D_j$  is not reprocessed, if  $D_j$  is reprocessed anyone can know the correspondence between  $P_j$  and its data  $D_j$ , because  $D_j$  is newly disclosed in the 2nd BB as the reprocessed data. However, in many cases mix-servers do not behave dishonestly because mix-servers are usually configured by authorities that cannot continue their businesses when their dishonesties are revealed, therefore  $P_j$  can preserve its privacy when it is honest. In other words, mechanisms discussed in this paper are prepared mainly for protecting mix-net  $M$  from dishonest entities that claim  $M$  is dishonest while generating inconsistent data by themselves, and usually the above problem is not serious.

## 6. Performance of the Proposed Mechanism

### 6.1. Numbers of Interactions

Regarding computation volumes, schemes based on approaches adopted in optimistic verifiable mix-nets also convince verifiers that given pairs of data constitute DDH pairs without examining the pairs individually [6,7], i.e. each mix-server  $S_q$  and verifier  $V$  exchange information about aggregated values of inputs and outputs of  $S_q$ , and consequently, computation volumes required for verifications do not depend on the numbers of data. But  $V$  and mix-server  $M$  must interact with each other while exchanging volume of data that is proportional to the number of mix-servers.

A distinctive advantage of the proposed mechanism compared with them is once a value about aggregated encryption parameters of each mix-server  $S_q$  (i.e.  $\kappa_q = k_{q1} + k_{q2} + \dots + k_{qN}$ ) is disclosed anyone can verify validities of given pairs by itself without any interaction with mix-servers. This means that mix-net  $M$  can convince all relevant entities that the pairs are valid without additional computations even in environments where trustworthy verifiers cannot be assumed. Different from the proposed mechanism, if entities in an optimistic verifiable mix-net cannot trust  $V$  as their representative, the mix-net must carry out same calculations for each entity individually while interacting with it.

As an exception, although procedures in the proposed mechanism for identifying entities liable for disrupted

decryption results require interactions between verifiers and mix-servers, these procedures are carried out only for inconsistent data when they are detected. Regarding consistencies of individual decryption results, different from verifications of DDH pairs, each decryption result must be examined individually, as a consequence the volume of computations for verifying  $M$ 's behaviors increases. But consistency of each decryption result  $\langle \{g^{kj^{*(Q)}}, D_j\}, \{g^{tj^{*(Q)}}, R_j\}, \{g^{vj^{*(Q)}}, D_j^{Rj+A}\} \rangle$  can be verified without participations of mix-servers as same as in DDH pair verifications, i.e. anyone can determine whether  $D_j$ ,  $R_j$  and  $D_j^{Rj+A}$  are consistent or not by calculating  $D_j^{Rj+A}$  by itself based on  $D_j$  and  $R_j$  disclosed in the 2nd BB. Therefore, verifications of individual decryption results do not require any computation for  $M$ . Also they can be verified in parallel by multiple entities without waiting responses from mix-servers, and the performance of the proposed mechanism can be maintained.

The unknown random number generation stage that does not exist in already existing verifiable mix-nets is also a cause of performance degradation, but when the fact that any entity can verify correct behaviors of mix-nets by itself without interacting with mix-servers is considered many applications can accept this inconvenience.

## 6.2. Performance against Attacks

It is reported that optimistic verifiable mix-nets are weak against relational attacks [11]. Namely, if 2-entities  $P_h$  and  $P_m$  find  $\Pi(D_j) = \langle \{g^{0j}, \omega y^{0j}\}, \{g^{\tau j}, \zeta y^{*\tau j}\}, \{g^{\lambda j}, H_j y^{*\lambda j}\} \rangle$  that  $P_j$  had put in mix-net  $M$  (where,  $\omega = g^{aj}$  and  $\zeta = D_j y^{*aj}$ ), and generate triples  $\langle \{g^{0h}, (g^{0j})^\gamma y^{*0h}\}, \{g^{\tau h}, (\omega y^{*0j})^\gamma y^{*\tau h}\}, \{g^{\lambda h}, h((g^{0j})^\gamma, (\omega y^{*0j})^\gamma) y^{*\lambda h}\} \rangle$  and  $\langle \{g^{0m}, (g^{0j})^\delta y^{*0m}\}, \{g^{\tau m}, (\omega y^{*0j})^\delta y^{*\tau m}\}, \{g^{\lambda m}, h((g^{0j})^\delta, (\omega y^{*0j})^\delta) y^{*\lambda m}\} \rangle$ , respectively to put them in  $M$ , they are finally decrypted to consistent triples  $\langle (g^{0j})^\gamma, (\omega y^{*0j})^\gamma, h((g^{0j})^\gamma, (\omega y^{*0j})^\gamma) \rangle$  and  $\langle (g^{0j})^\delta, (\omega y^{*0j})^\delta, h((g^{0j})^\delta, (\omega y^{*0j})^\delta) \rangle$ , therefore  $P_h$  and  $P_m$  can identify their data by examining all decryption results to find pair  $g^{0j\gamma}$  and  $g^{0j\delta}$  that satisfies relation  $(g^{0j\gamma})^\delta = (g^{0j\delta})^\gamma$ . In addition,  $\Pi(D_j)$  is decrypted to  $\langle \omega, \zeta, H_j \rangle$ , and identified triple  $\langle (g^{0j})^\gamma, (\omega y^{*0j})^\gamma, h((g^{0j})^\gamma, (\omega y^{*0j})^\gamma) \rangle$  is decrypted to  $\omega^\gamma$ . Then,  $P_h$  and  $P_m$  can know  $P_j$ 's data  $D_j$ , i.e.  $D_j$  is the decrypted form of triple  $\langle \omega, \zeta, H_j \rangle$  that has value  $\omega = (\omega^\gamma)^{1/\gamma}$  as its 1st element.

Here, the reason why optimistic verifiable mix-nets are weak against relational attacks is hash value  $H_j$  in the above is calculated based on the encrypted form of original data  $D_j$ . Namely, entities other than  $P_j$  that do not know the decrypted value of  $\Pi(D_j)$  also can generate consistent triples from it. Different from verifiable mix-nets, entities in the proposed mechanism cannot generate consistent encrypted forms without knowing their decrypted forms; therefore relational attacks can be avoided.

## 7. Conclusion

A simplified scheme of verifiable mix-nets is proposed. In environments where the number of data is sufficiently large and decryption results of individual data, i.e. outputs of mix-nets, can be publicly disclosed, the proposed scheme works well without using time consuming ZKPs. More importantly, entities can verify encryption and

decryption results by themselves without interacting with mix-servers. The scheme also can effectively protect itself from various attacks. Its primary verification procedures do not examine correct re-encryptions or decryptions of individual data, and although consistency of decrypted data must be examined individually, they can be examined by anyone without communicating with mix-servers, i.e. a set of decryption results can be examined by multiple entities in parallel. Therefore highly efficient e-voting and e-poll systems, in which decrypted votes and opinions can be publicly disclosed, can be developed based on the proposed mix-nets while preserving privacies of participants and ensuring honest behaviors of authorities.

As future works, inconveniences that privacy of data owners may be invaded when dishonest entities are identified in the unknown random number generation stage must be removed although usually these inconveniences occur when data owners themselves are dishonest. A simple solution is to let data owners access mix-nets anonymously (in the above, data owners are assumed that they access mix-nets while revealing their identities). But this solution cannot disable owners of the disrupted data themselves to know correspondences between them and their decrypted data in the 2nd BB because the owners know their inputs that are finally corresponded to the disrupted outputs. Also, data owner  $P_j$  can know the correspondence between it and its data  $D_j$  in the 2nd BB when  $D_j$  is unique to  $P_j$  as mentioned in previous sections. Therefore the proposed scheme cannot work perfectly in e-voting systems. Namely, if  $D_j$  is a vote of  $P_j$  in an e-voting system, an entity, which is forcing  $P_j$  to abstain from voting by casting an unique invalid vote, can easily confirm whether  $P_j$  behaved as it had asked or not by examining the 2nd BB. To make e-voting systems more secure, schemes for convincing voters that encrypted  $D_j$  is a vote for a non-winner (a candidate that cannot acquire enough number of votes) so that election authorities do not need to decrypt unique votes [15] must be developed.

## References

- [1] Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Trans. On Information Theory, IT-22(6), 644-654, 1976.
- [2] M. Blum, P. Feldman and S. Micali, "Non-interactive Zero-knowledge and Its Applications," Proc. of the 20th Annual ACM Symposium on Theory of Computing, 103-112, 1988.
- [3] S. Goldwasser, S. Micali and C. Rackoff, "The Knowledge Complexity of Interactive Proof System," SIAM Journal on Computing, 18(1), 291-304, 1989.
- [4] B. Pfitzmann, "Breaking an Efficient Anonymous Channel," Eurocrypt'95, LNCS 950, 332-340, 1995.
- [5] M. Abe, "Universally Verifiable Mix-Net with Verification Work Independent of the Number of Mix-Servers," IEICE Trans. Fundamentals, E83-A(7), 1431-1440, 2000.
- [6] D. Boneh and P. Golle, "Almost Entirely Correct Mixing with Applications to Voting," ACM Conference on Computer and Communication Security, 68-77, 2002.
- [7] P. Golle, S. Zhong, D. Boneh, M. Jakobsson and A. Juels, "Optimistic Mixing for Exit-Polls," Asiacypt 2002, 451-465, 2002.
- [8] M. Jakobson, A. Juels and R. Rivest, "Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking," USENIX Security '02, 339-353, 2002.
- [9] L. Nguen, R. Dafavi-Naini and K. Kurosawa, "Verifiable Shuffles: A Formal Model and a Paillier-based Efficient Construction with Provable Security," PKC 2004, LNCS 2248, 61-75, 2004.2002.

- [10] J. Furukawa, "Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability," PKC 2004, LNCS 2248, 319-332, 2004.
- [11] D. Wikstrom, "Five Practical Attacks for Optimistic Mixing for Exit-Polls," Proceedings of SAC 2003, 160-175, 2004.
- [12] K. Sampigethaya and R. Poovendran, "A Framework and Taxonomy for Comparison of Electronic Voting Schemes," Elsevier Computers and Security, 25, 137-153, 2006.
- [13] S. Weber, "A Coercion-Resistant Cryptographic Voting Protocol - Evaluation and Prototype Implementation," Diploma thesis, Darmstadt University of Technology; 2006.
- [14] K. A. Md Rokibul, S. Tamura, S. Taniguchi and T. Yanase, "An Anonymous Voting Scheme based on Confirmation Numbers," IEEJTrans. EIS. 130(11), 2065-2073, 2010.
- [15] S. Tamura, "Anonymous Security Systems and Applications: Requirements and Solutions," Information Science Reference, 2012.