Information Security and Computer Fraud, 2014, Vol. 2, No. 3, 39-47
Available online at http://pubs.sciepub.com/iscf/2/3/2
© Science and Education Publishing
DOI:10.12691/iscf-2-3-2

Linear Mix-net and a Scheme for Collecting Data from Anonymous Data Holders

Shinsuke Tamura*, Shuji Taniguchi

School of Engineering, University of Fukui, Fukui, Japan
*Corresponding author: tamura@dance.plala.or.jp

Received October 13, 2014; Revised November 01, 2014; Accepted November 04, 2014

Abstract To make e-society, e-governance and cloud computing systems be utilized more widely, this paper proposes a scheme to collect attribute values belong to same data holders and calculate functions of them without knowing correspondences between the attribute values and their holders or links among attribute values of same holders. Different from most of other schemes the proposed scheme is based on linear Mix-net that exploits secret key encryption functions such as linear equation based (LE-based) and multidimensional array based (MA-based) ones, therefore it can handle real numbers that appear in many important business and engineering applications efficiently in the same way as integers. In addition, anonymous tag based credentials used in the scheme ensure the correctness of calculation results. Although the scheme can calculate only linear combinations of attribute values when LE-based encryption functions are used, if they are replaced with MA-based ones, it can calculate also general polynomial functions of attribute values.

Keywords: *E-society, E-governance, cloud computing, privacy, homomorphic encryption functions, anonymous tag based credentials*

Cite This Article: Shinsuke Tamura, and Shuji Taniguchi, “Linear Mix-net and a Scheme for Collecting Data from Anonymous Data Holders.” *Information Security and Computer Fraud*, vol. 2, no. 3 (2014): 39-47. doi: 10.12691/iscf-2-3-2.

1. Introduction

Data collection systems are ones that collect attribute values belong to same data holders and calculate functions of them, and by using these systems, government agencies can quickly and correctly calculate taxes of citizens for example. However, many people do not want such systems because their all information are linked and may be used for other purposes. Same difficulties exist in many applications in e-society and e-governance systems. To make e-society, e-governance and cloud computing systems be utilized more widely, this paper proposes a scheme that enables entities (e.g. servers in cloud computing systems) to collect attribute values belong to same data holders and calculate functions of them without knowing links between attribute values and their holders or among attribute values of same holders.

The above scheme can be developed by using Mix-nets [2,4,5] as shown in Figure 1 that consist of data holders, authority *A*, and mix-servers M_1, M_2, ---, M_N in the encryption and the decryption stages. Here, each data holder P owns its attribute values $X_P(1)$, $X_P(2)$, ---, $X_P(Q)$, and although each $X_P(q)$ is disclosed by some reasons (e.g. if $X_P(q)$ is the amount of P’s deposit in a bank P must disclose it to the bank) P wants to conceal the fact that $X_P(q)$ belongs to it from others including *A* and mix-servers (actually, P must conceal also links among $X_P(1)$, ---, $X_P(Q)$ because they are good clues to identify P). On the other hand, *A* needs to calculate functions of attribute values of same data holders and let data holders take actions according to the calculation results (e.g. pay taxes).

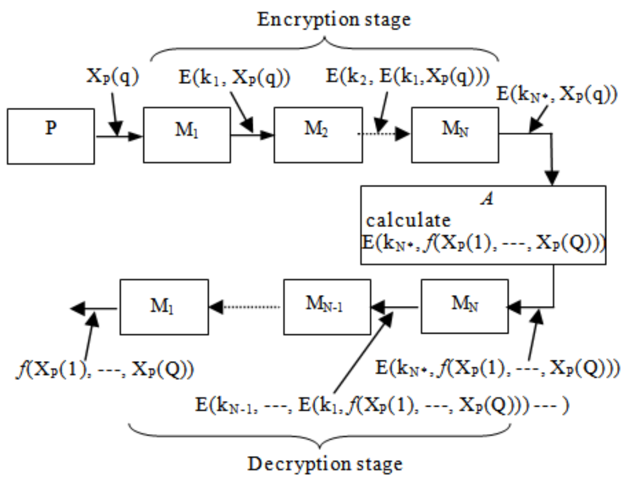


Figure 1. Configuration of a data collection system

To satisfy these requirements, mix-servers M_1, ---, M_N in the encryption and the decryption stages repeatedly encrypt individual attribute values and decrypt encrypted function values respectively while disclosing their encryption and decryption results publicly so that they can convince others of their correct encryptions and decryptions. An important things here are firstly encryption functions are probabilistic (this means they generate different encryption forms even for same plain texts) and secondly each M_h shuffles its encryption or decryption results before it discloses them. Therefore no one can know the correspondence between attribute values received by M_1 in the encryption stage and final decryption results calculated by M_1 in the decryption stage unless all mix-servers conspire.

In detail, each data holder P informs 1st mix-server M_1 in the encryption stage of its q-th attribute value $X_P(q)$ without disclosing its identity, and provided that k_h is an encryption key of M_h's encryption function $E(k_h, x)$ for each h, M_1, ---, M_N repeatedly encrypt each $X_P(q)$ to $E(k_{N*}, X_P(q)) = E(k_N, E(k_{N-1}, \text{---} E(k_1, X_P(q)) \text{---}))$ while shuffling their encryption results. After that, authority *A* collects $E(k_{N*}, X_P(1))$, $E(k_{N*}, X_P(2))$, ---, $E(k_{N*}, X_P(Q))$ that correspond to anonymous data holder P, and calculates $E(k_{N*}, f(X_P(1), \text{---}, X_P(Q)))$, an encryption form of function value $f(X_P(1), \text{---}, X_P(Q))$. Then, M_N, M_{N-1}, ---, M_1 in the decryption stage repeatedly decrypt each $E(k_{N*}, f(X_P(1), \text{---}, X_P(Q)))$ to $f(X_P(1), \text{---}, X_P(Q))$ so that P that owns $X_P(1)$, ---, $X_P(Q)$ can know value $f(X_P(1), \text{---}, X_P(Q))$, and finally *A* asks individual anonymous data holders to take actions according to their function values.

But to implement this scheme the following difficulties must be removed. The 1st difficulty is about the anonymity of data holders, i.e. each data holder P must convince 1st mix-server M_1 of its eligibility without disclosing its identity. About the 2nd and the 3rd difficulties, authority *A* must collect P's encrypted attribute values $E(k_{N*}, X_P(1))$, ---, $E(k_{N*}, X_P(Q))$ from all encryption results in the encryption stage, on the other hand, P must identify its function value $f(X_P(1), \text{---}, X_P(Q))$ from all decryption results in the decryption stage. The 4th and the 5th difficulties relate to mechanisms to calculate encryption form $E(k_{N*}, f(X_P(1), \text{---}, X_P(Q)))$ and to ensure mix-servers' correct handlings of attribute values. Namely, each encryption function $E(k_h, x)$ must have specific features to enable *A* to calculate $E(k_{N*}, f(X_P(1), \text{---}, X_P(Q)))$ from $E(k_{N*}\ X_P(1))$, ---, $E(k_{N*}, X_P(Q))$. Also, even if some entities behave dishonestly, *A* and data holders must detect incorrect calculation results and identify entities liable for them to re-calculate correct values.

Among these difficulties, the 1st difficulty can be removed by anonymous authentication schemes [6,7,12], in addition some of them can handle also the 2nd, the 3rd and the 5th difficulties. But currently available schemes are not practical for handling the 4th difficulty. Namely, although simple ElGamal re-encryption schemes [4] enable authority *A* to calculate encrypted weighted sum of attribute values $E(k_{N*}, a_1X_P(1)+ \text{---} +a_QX_P(Q))$ as $E(k_{N*}, a_1X_P(1))+ \text{---} +E(k_{N*}, a_QX_P(Q))$ and recent fully homomorphic encryption functions [8,9] may enable *A* to calculate arbitrary function $E(k_{N*}, f(X_P(1), \text{---}, X_P(Q)))$ as $f(E(k_{N*}, X_P(1)), \text{---}, E(k_{N*}, X_P(Q)))$, they are designed for handling integer values. Therefore they are not practical to handle real number attribute values that appear in many important business and engineering applications.

The scheme proposed in this paper exploits linear equation based (LE-based) encryption functions [10] to enable authority *A* to efficiently calculate weighted sums of real number attribute values. *A* can calculate also their general polynomial functions, when LE-based encryption functions are replaced with multidimensional array based (MA-based) ones [10] which are both additive and multiplicative.

2. LE-based Encryption Functions

A linear equation based (LE-based) encryption function considers information as integers or real numbers, and encrypts an (H+G)-dimensional vector of integers or real numbers $X = \{x_1, x_2, \text{---}, x_H, r_1, r_2, \text{---}, r_G\}$ to (H+G)-dimensional vector $X_* = \{x_{*1}, x_{*2}, \text{---}, x_{*H+G}\}$ by using secret (H+G)x(H+G)-dimensional coefficient matrix $Q = \{q_{ij}\}$, i.e. $x_{*i} = q_{i1}x_1+q_{i2}x_2+ \text{---} +q_{iH}x_H+q_{i(H+1)}r_1+ \text{---} +q_{i(H+G)}r_G$ for each i [10]. Where, each x_j constitutes a real term that corresponds to information to be encrypted, on the other hand, each r_h is a random number secret of the information holder and constitutes a dummy term.

Then, for an entity that does not know matrix Q it is difficult to calculate *X* from X_*; but when Q is known, anyone can calculate *X* from X_* by solving the linear equations provided that Q has its inverse. Therefore, coefficient matrices Q and Q^{-1} work as an encryption and a decryption keys. Here, it is apparent that encryption function $E(Q, X)$ is additive, i.e. provided that *s* and *t* are real numbers and *sX* represents the product of scalar number *s* and vector *X*, when *X* and *Y* are encrypted to, $E(Q, X)$ and $E(Q, Y)$, $sE(Q, X)+tE(Q, Y)$ is decrypted to $sX+tY$. Also, because each x_j, r_h and elements of matrix Q are not limited to integers, LE-based encryption functions can handle real numbers in totally the same way as integers.

However the above additive feature introduces a serious drawback, i.e. LE-based encryption functions are weak against plain text attacks. When mutually independent (H+G)-dimensional vectors A_{1*}, A_{2*}, ---, $A_{(H+G)*}$ are known as encryption forms of known vectors A_1, A_2, ---, A_{H+G}, because arbitrarily given (H+G)-dimensional vector X_* is represented as $X_* = g_1A_{1*}+ \text{----} +g_{H+G}A_{(H+G)*}$, X_* can be easily decrypted to $X = g_1A_1+ \text{----} +g_{H+G}A_{H+G}$ without knowing coefficient matrix Q. Therefore, LE-based encryption functions must be used in applications where data are encrypted and decrypted by same entities, i.e. in these applications entities that encrypt information do not need to disclose at least dummy term values to others in their plain forms and plain text attacks become difficult (it must be noted that by various reasons real part values must be disclosed in their plain forms in many applications). Because entities that encrypt and decrypt data are same, a fact that lengths of encryption keys are prone to being long is not a disadvantage either.

LE-based encryption functions can be intensified further by inserting secret dummy elements at random positions in encrypted vectors, i.e. elements of encryption form $\{x_{*1}, x_{*2}\text{---}, x_{*H+G}\}$ and secret dummy vector $\{w_{*1}, w_{*2}, \text{---}, w_{*L}\}$ are merged while being shuffled to constitute (H+G+L)-dimensional encryption form $\{X_*'\} = \{w_{*1}, w_{*2}, x_{*3}\ w_{*3}, x_{*1}\text{---}, w_{*4}\}$ for example. As a consequence, positions where x_{*1}, $x_{*2}\text{---}$, x_{*G+H} are located in $\{X_*'\}$ must be determined in order to decrypt $\{X_*'\}$. When L-dummy elements $\{w_{*1}, w_{*2}\text{---}, w_{*L}\}$ are added to (H+G)-dimensional vector $\{x_{*1}, x_{*2}\text{---}, x_{*H+G}\}$, ${}_{H+G+L}P_{H+G}$ number of possibilities must be examined to remove the dummy elements, and when (H+G) and L are set to 50, ${}_{H+G+L}P_{H+G}$ is ${}_{100}P_{50} > 2^{500}$.

On the other hand, solving linear equations is not difficult when the coefficient matrix is given. For example, LU-decomposition [3] solves linear equations with sufficient performance in terms of both computation speed and accuracy. Computation speed is fast enough compared with that of modern asymmetric key encryption functions such as RSA even the dimensions of coefficient matrices are more than 100, also computation errors are small enough.

However, it is still easy to generate consistent encryption forms even without knowing encryption keys. By linearly combining known encryption forms, anyone can generate consistent encryption forms of variety of data without knowing the key as same as man in the middle attacks in environments where public key encryption functions are used. But many mechanisms are available to remove these threats, e.g. implicit transaction links (ITLs) [10] enable entities to detect forged encryption forms without examining individual forms.

About the verifiability, although LE-based encryption functions are secret key based, correctness of E(Q, x) can be verified by using the additive property as follow. Conceptually, entity V that verifies E(Q, x^*), encryption form of x, generates arbitrary vectors {E(Q, T_1), ---, E(Q, T_m)} as a set of encryption forms of test values, and asks P, which had calculated E(Q, x^*), to decrypt them to {T_1, ---, T_m}. After that, V calculates $\underline{X}$ = w_0E(Q, x^*)+w_1E(Q, T_1)+ --- +w_mE(Q, T_m) while generating random numbers w_0, w_1, ----, w_m secret from P, and asks P to decrypt $\underline{X}$. Then, because E(Q, x) is additive $\underline{X}$ must be decrypted to X = $w_0x+w_1T_1$+ ---- +w_mT_m if E(Q, x^*) and E(k, T_1), ---, E(k, T_m) are correct. But if they are incorrect, P that does not know w_0, w_1, ---, w_m cannot calculate X from $\underline{X}$.

Here, actually $\underline{X}$, E(Q, x^*), E(Q, T_1), ---, E(Q, T_m) in the above are vectors, therefore P obtains multiple relations about (m+1)-variables w_0, w_1, ---- w_m, which may enable P to calculate w_0, w_2, ---- w_m when m is small. But a slight extension disables P to calculate w_0, w_1, ---- w_m even when m is small. This means P does not need to disclose numbers of plain and encryption forms pairs of test values. P does not need to disclose dummy terms of vector X and each test vector T_j either; in other words, verification of dummy terms has no meaning for V because they can have any values without making real terms inconsistent. Then, encryption function E(Q, x) can be protected from plain text attacks even in environments where correctness of numbers of encryption forms are verified.

3. Linear Mix-net

A scheme that enables authority A to calculate linear combinations of attribute values belong to same data holders without knowing correspondences between attribute values and their holders can be developed by linear Mix-nets as below [11]. As mentioned before, one of advantages of linear Mix-nets is they use LE-based or MA-based encryption functions and can handle real numbers and integers efficiently totally in the same way.

Here, implementation of a re-encryption scheme based on LE-based encryption functions E(k_1, x), E(k_2, x), ---, E(k_N, x) is straightforward, i.e. 1st mix-server M_1 encrypts real number or integer x to z_1-dimensional vector {$\underline{x}_1(1)$, $\underline{x}_1(2)$, ---, $\underline{x}_1(z_1)$} based on its secret coefficient matrix {q_1(i, j)} and dummy terms, and merges it and dummy elements {$\underline{y}_1(z_1+1)$, $\underline{y}_1(z_1+2)$, ---, $\underline{y}_1(z^*_1)$} to construct z^*_1-dimensional vector E(k_1, x) = {$x_1(1)$, $x_1(2)$, ---, $x_1(z^*_1)$}. Then 2nd mix-server M_2 adds dummy terms to {$x_1(1)$, ---, $x_1(z^*_1)$} to construct z_2-dimensional ($z_2 > z^*_1$) vector {$x_1(1)$, ---, $x_1(z^*_1)$, $x_1(z^*_1+1)$, ---, $x_1(z_2)$}, encrypts it to {$\underline{x}_2(1)$, $\underline{x}_2(2)$, ---, $\underline{x}_2(z_2)$} by calculating each $\underline{x}_2$(s) as a linear combination of $x_1(1)$, ---, $x_1(z_2)$ while using secret coefficient matrix {q_2(i, j)}, and merges it and dummy elements {$\underline{y}_2(z_2+1)$, $\underline{y}_2(z_2+2)$, ---, $\underline{y}_2(z^*_2)$} to construct z^*_2-dimensional vector E(k_2, E(k_1, x)) = {$x_2(1)$, $x_2(2)$, ---, $x_2(z^*_2)$}. Remaining mix-servers behave in the same way.

In the remainder, a linear Mix-net is configured based on LE-based encryption functions, and notation E(k_{h*}, x) is used to represent re-encryption form E(k_h, E(k_{h-1}, --- E(k_1, x) ---)).

3.1. Configuration of LE-based Linear Mix-net

In LE-based linear Mix-net, mix-servers are arrayed also in the verification stage and numbers of mix-servers in the encryption and the decryption stages are not equal, i.e. it consists of data holders, authority A, mix-servers M_1, ---, M_T in the encryption and the verification stages and M_1, ---, M_N (N < T) in the decryption stage as shown in Figure 2. As same as in Figure 1 mix-servers in the encryption stage repeatedly encrypt individual attribute values, and based on the encryption results, authority A calculates encrypted weighted sums of individual data holders' attribute values to be repeatedly decrypted by mix-servers in the decryption stage. But different form Figure 1 mix-servers in the encryption stage encrypt single attribute value X_P(q) into multiple different forms, also before entering the decryption stage mix-servers in the verification stage decrypt individual encrypted attribute values to convince others of their honest encryptions.

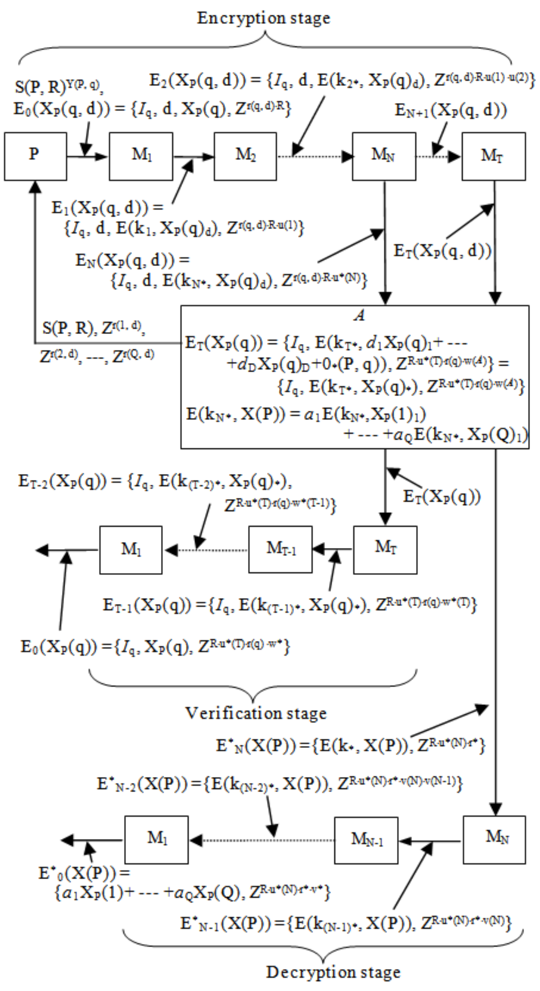
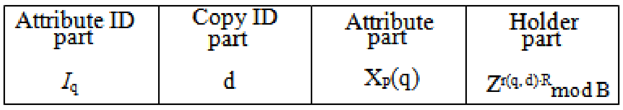

Encryption stage
S(P, R)^{Y(P, q)},
E_0(X_P(q, d)) = {I_q, d, X_P(q), Z^{r(q, d)R}}
E_2(X_P(q, d)) = {I_q, d, E(k_{2*}, X_P(q)_d), Z^{r(q, d)·R·u(1)·u(2)}}
E_{N+1}(X_P(q, d))
P
M_1
M_2
M_N
M_T
E_1(X_P(q, d)) = {I_q, d, E(k_1, X_P(q)_d), Z^{r(q, d)·R·u(1)}}
E_N(X_P(q, d)) = {I_q, d, E(k_{N*}, X_P(q)_d), Z^{r(q, d)·R·u*(N)}}
E_T(X_P(q, d))
A
S(P, R), Z^{r(1, d)}, Z^{r(2, d)}, ---, Z^{r(Q, d)}
E_T(X_P(q)) = {I_q, E(k_{T*}, d_1X_P(q)_1+ --- +d_DX_P(q)_D+0*(P, q)), Z^{R·u*(T)·r(q)·w(A)}} = {I_q, E(k_{T*}, X_P(q)_*), Z^{R·u*(T)·r(q)·w(A)}}
E(k_{N*}, X(P)) = a_1E(k_{N*}, X_P(1)_1) + --- +a_QE(k_{N*}, X_P(Q)_1)
E_{T-2}(X_P(q)) = {I_q, E(k_{(T-2)*}, X_P(q)_*), Z^{R·u*(T)·r(q)·w*(T-1)}}
E_T(X_P(q))
M_1
M_{T-1}
M_T
E_{T-1}(X_P(q)) = {I_q, E(k_{(T-1)*}, X_P(q)_*), Z^{R·u*(T)·r(q)·w*(T)}}
E_0(X_P(q)) = {I_q, X_P(q), Z^{R·u*(T)·r(q)·w*}}
Verification stage
E*_N(X(P)) = {E(k_*, X(P)), Z^{R·u*(N)·r*}}
E*_{N-2}(X(P)) = {E(k_{(N-2)*}, X(P)), Z^{R·u*(N)·r*·v(N)·v(N-1)}}
M_1
M_{N-1}
M_N
E*_0(X(P)) = {a_1X_P(1)+ --- +a_QX_P(Q), Z^{R·u*(N)·r*·v*}}
E*_{N-1}(X(P)) = {E(k_{(N-1)*}, X(P)), Z^{R·u*(N)·r*·v(N)}}
Decryption stage

Figure 2. Configuration of LE-based linear Mix-net

Where, although each M_h in all stages discloses its encryption and decryption results publicly despite $E(k_h, x)$ is weak against plain text attacks to prove its correct handlings of attribute values, $E(k_h, x)$ is protected because M_h in each stage shuffles its encryption or decryption results. For an entity that does not know shuffling rules, every possible input and output values pair of M_h is a candidate of plain and encryption (or encryption and plain) forms pair of $E(k_h, x)$, and provided that Ω and Ψ are the number of data holders and the dimension of vector $E(k_h, x)$ respectively, ${}_{\Omega}P_{\Psi}$ number of possibilities must be examined for obtaining Ψ-mutually independent plain and encryption forms pairs (${}_{\Omega}P_{\Psi}$ is greater than 10^{1000} when $\Omega = 200$ and $\Psi = 100$).

Figure 3 shows the data structure of data holder P's q-th attribute value $X_P(q)$ that P puts in the encryption stage. Attribute ID and attribute parts correspond to q-th attribute name I_q (e.g. height of persons) and attribute value $X_P(q)$ itself (e.g. height of a particular person P). About copy ID part value d, P generates multiple copies for single $X_P(q)$ and d is the identifier of the d-th copy. The holder part value $Z^{r(q,\ d)\cdot R}{}_{\mathrm{mod}\ B}$ is calculated by P from publicly known integer $Z^{r(q,\ d)}{}_{\mathrm{mod}\ B}$ as a used seal of P's anonymous tag based credential S(P, R) [12] (B is a publicly known sufficiently large appropriate integer and notation $_{\mathrm{mod}\ B}$ is omitted in the remainder).



Attribute ID part	Copy ID part	Attribute part	Holder part
I_q	d	$X_P(q)$	$Z^{r(q,\ d)\cdot R}{}_{\mathrm{mod}\ B}$

Figure 3. Data structure of attribute values

In detail, for its q-th attribute value $X_P(q)$, P shows D-quadruplets $E_0(X_P(q, d)) = \{I_q, d, X_P(q), Z^{r(q,\ d)\cdot R}\}$ (d =, 1, 2, ---, D) to 1st mix-server M_1 to be encrypted repeatedly to quadruplets $E_1(X_P(q, d)) = \{I_q, d, E(k_1, X_P(q)_d), Z^{r(q,\ d)\cdot R\cdot u(1)}\}$, ---, $E_N(X_P(q, d)) = \{I_q, d, E(k_{N*}, X_P(q)_d), Z^{r(q,\ d)\cdot R\cdot u*(N)}\}$, ---, $E_T(X_P(q, d)) = \{I_q, d, E(k_{T*}, X_P(q)_d), Z^{r(q,\ d)\cdot R\cdot u*(T)}\}$, (d =, 1, 2, ---, D) by M_1, ---, M_N, ---, M_T in the encryption stage, where $E(k_{h*}, X_P(q)_d)$ represents a re-encryption form of $X_P(q)$ that is generated based on dummy terms and dummy elements corresponding to copy ID value d, therefore although $E(k_{h*}, X_P(q)_1)$, ---, $E(k_{h*}, X_P(q)_D)$ are decrypted to same value $X_P(q)$ they have different forms.

About the verification stage, authority *A* collects quadruplets $E_T(X_P(q, 1))$, ---, $E_T(X_P(q, D))$, which are calculated from $E_0(X_P(q, 1))$, ---, $E_0(X_P(q, D))$ in the encryption stage corresponding to attribute value $X_P(q)$, to construct single triplet $E_T(X_P(q)) = \{I_q, E(k_{T*}, X_P(q)_*), Z^{r(q)\cdot R\cdot u*(T)\cdot w(A)}\}$ that has the same structure as in Figure 3 except it does not include the copy ID part. Then, M_T, ---, M_1 decrypt $E_T(X_P(q))$ repeatedly to triplets $E_{T\text{-}1}(X_P(q))$, ---, $E_0(X_P(q))$. Important things are firstly $E(k_{T*}, X_P(q)_*)$ in triplet $E_T(X_P(q))$ is decrypted also to $X_P(q)$, and secondly, each mix-server M_h cannot identify the correspondence between $E_h(X_P(q, d))$ in the encryption stage and $E_h(X_P(q))$ in the verification stage for each d, in other words, attribute and holder parts values in triplets $E_T(X_P(q))$, ---, $E_1(X_P(q))$ have different forms from those in quadruplets $E_T(X_P(q, d))$, ---, $E_1(X_P(q, d))$.

Finally, the data structure of the weighted sum of P's attribute values $X(P) = a_1X_P(1)+ \text{---} +a_QX_P(Q)$ put in the decryption stage consists of attribute part and holder part values pair $E_N(X(P)) = \{E(k_{N*}, X(P)), Z^{R\cdot u*(N)\cdot r*}\}$. Its attribute part and holder part values are aggregations of those in 1st copies of re-encrypted quadruplets $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$ in the encryption stage, i.e. $E(k_{N*}, X(P)) = a_1E(k_{N*}, X_P(1)_1)+ \text{---} +a_QE(k_{N*}, X_P(Q)_1)$ and $Z^{R\cdot u*(N)\cdot r*} = Z^{R\cdot u*(N)\cdot r(1,\ 1)\cdot r(2,\ 1)\text{---}r(Q,\ 1)}$. Here, it must be noted that authority *A* generates $E_N(X(P))$ from $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$ instead of $E_T(X_P(1, 1))$, ---, $E_T(X_P(Q, 1))$.

Under the above settings, first 3 difficulties in Introduction are removed by anonymous credential S(P, R) and holder part values of quadruplets in the encryption stage and pairs in the decryption stage. In detail, to register itself as an authorized entity, each data holder P shows its exact identity to authority *A*, and *A* issues integers $Z^{r(1,\ d)}$, ---, $Z^{r(Q,\ d)}$ (d = 1, 2, ---, D) and anonymous credential S(P, R) that includes P's secret unique integer R to P. After that, P generates secret integer y(P, q), calculates $S(P, R)^{y(P,\ q)}$, and shows $S(P, R)^{y(P,\ q)}$ together with $X_P(q)$ to 1st mix-server M_1 in the encryption stage. At the same time, P calculates holder part value $Z^{r(q,\ d)\cdot R}$ from $Z^{r(q,\ d)}$ for each d as a used seal of S(P, R) to be incorporated in each quadruplet $E_0(X_P(q, d)) = \{I_q, d, X_P(q), Z^{r(q,\ d)\cdot R}\}$, where, anonymous credential S(P, R) forces P to honestly calculate holder part value $Z^{r(q,\ d)\cdot R}$ from $Z^{r(q,\ d)}$ by using secret integer R in S(P, R). About integers Z and r(q, d), Z is common to all attribute vales of all data holders and publicly known, on the other hand, r(q, d) is unique to q-th attribute values for each d and it is secret from all entities including *A* and mix-servers.

Then the 1st difficulty is removed, i.e. anonymous credential S(P, R) enables P to convince M_1 of its eligibility without disclosing its identity or secret integer R [12]. Here, P assigns different values to secret integers y(P, 1), ---, y(P, Q), also each holder part value $Z^{r(q,\ d)\cdot R}$ is constructed by integers Z, r(q, d) and R with the above properties. Therefore entities other than P cannot identify links among P's attribute values $X_P(1)$, ---, $X_P(Q)$ even if they examine credential forms $S(P, R)^{y(P,\ 1)}$, ---, $S(P, R)^{y(P,\ Q)}$ or used seals $Z^{r(1,\ d)\cdot R}$, ---, $Z^{r(Q,\ d)\cdot R}$. When difficulties of solving discrete logarithm problems are considered, to know that the above credential forms or used seals are calculated from same S(P, R) or R is computationally infeasible for entities that do not know y(P, q), R or r(q, d).

About the 2nd and the 3rd difficulties, holder part value $Z^{r(q,\ d)\cdot R}$ in initial quadruplet $E_0(X_P(q, d))$ is transformed to $Z^{r(q,\ d)\cdot R\cdot u(1)\cdot u(2)\text{---}u(N)} = Z^{r(q,\ d)\cdot R\cdot u*(N)}$ by M_1, ---, M_N in the encryption stage as a holder part value of re-encrypted quadruplet $E_N(X_P(q, d))$, and before entering the decryption stage, *A* asks mix-servers to calculate $(Z^{r(q,\ 1)\cdot R\cdot u*(N)})^{r(1,\ 1)\cdot r(2,\ 1)\text{---}r(q\text{-}1,\ 1)\cdot r(q+1,\ 1)\text{---}r(Q,\ 1)} = Z^{R\cdot u*(N)\cdot r(1,\ 1)\cdot r(2,\ 1)\text{---}r(Q,\ 1)} = Z^{R\cdot u*(N)\cdot r*}$ from $Z^{r(q,\ 1)\cdot R\cdot u*(N)}$ for each q. Therefore, value $Z^{R\cdot u*(N)\cdot r*}$ becomes common to P's all encrypted quadruplets $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$, and as a result, to calculate pair $E_N(X(P)) = \{E(k_*, X(P)), Z^{R\cdot u*(N)\cdot r*}\}$ *A* can collect $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$ despite that M_1, ---, M_N shuffle their encryption results. Here, u(h) is mix-server M_h's secret integer common to all attribute values of all data holders, and although no one knows each r(q, d) mix-servers can calculate $Z^{R\cdot u*(N)\cdot r*}$ from $Z^{r(q,\ 1)\cdot R\cdot u*(N)}$ as in Sec. 3.2.3.

In the same way, *A* can collect all quadruplets $E_T(X_P(q, 1))$, ---, $E_T(X_P(q, D))$ corresponding to $X_P(q)$ to construct triplet $E_T(X_P(q))$ to be decrypted in the verification stage.

Also, P can identify finally decrypted pair $E^*_0(X(P)) = \{X(P), Z^{R\cdot u*(N)\cdot r*\cdot v*}\}$ in the decryption stage based on its holder part value $Z^{R\cdot u*(N)\cdot r*\cdot v*}$, i.e. provided that mix-servers calculate $Z^{u*(N)\cdot r*\cdot v*}$ separately, only P that knows R can calculate $Z^{R\cdot u*(N)\cdot r*\cdot v*}$ in pair $E^*_0(X(P))$ from $Z^{u*(N)\cdot r*\cdot v*}$. Moreover although R is P's secret, P must calculate it honestly as a used seal of its credential S(P, R).

In the above, unique integer r(q, d) secret from all entities can be generated easily. In detail, each M_h generates its secret integer r(q, d; h) and calculates $Z^{r(q,\ d;\ 1)\cdot r(q,\ d;\ 2)---r(q,\ d;\ h-1)\cdot r(q,\ d;\ h)}$ from $Z^{r(q,\ d;\ 1)\cdot r(q,\ d;\ 2)---r(q,\ d;\ h-1)}$ received from M_{h-1} to forward the result to M_{h+1} so that finally M_T can calculate $Z^{r(q,\ d;\ 1)\cdot r(q,\ d;\ 2)---r(q,\ d;\ T)} = Z^{r(q,\ d)}$. Namely, no one knows all r(q, d; 1), ---, r(q, d; T) and calculating r(q, d) from $Z^{r(q,\ d)}$ is a discrete logarithm problem. Also uniqueness of r(q, d) can be maintained by discarding r(q, d) to replace it with new one when mix-servers had calculated same value $Z^{r(q,\ d)}$ before. Integers $u^*(N)$, v^* and r^* are generated in the same way. Therefore, no one can know values of r(q, d), $u^*(N)$, v^* or r^*, and as a result, anyone including P itself cannot examine holder part values to know the correspondence between P and encrypted quadruplet $E_N(X_P(q, d))$ or between finally decrypted pair $E^*_0(X(P))$ and each $E_N(X_P(q, d))$.

LE-based encryption functions and the verification stage remove the remaining difficulties. Firstly, encryption function $E(k_{N*}, x)$ is additive because each $E(k_h, x)$ is LE-based. Therefore, authority *A* can calculate encryption form $E(k_{N*}, a_1X_P(1)+a_2X_P(2)+ --- +a_QX_P(Q))$ from encrypted attribute values $E(k_{N*}, X_P(1)_1)$, $E(k_{N*}, X_P(2)_1)$, ---, $E(k_{N*}, X_P(Q)_1)$ as $E(k_{N*}, a_1X_P(1)+ --- +a_QX_P(Q)) = E(k_{N*}, X_P(1)_1)+ --- +E(k_{N*}, X_P(Q)_1)$. Namely, the 4th difficulty is removed if function $f(x_1, ---, x_Q)$ is a linear combination of attribute values $x_1, ---, x_Q$.

About the 5th difficulty, mix-servers in the encryption and the verification stages transform each attribute value in different ways, i.e. corresponding to same attribute value $X_P(q)$, mix-server M_h in the encryption stage calculates quadruplet $E_h(X_P(q, d))$, and M_{h+1} in the verification stage calculates triplet $E_h(X_P(q))$ so that no one can identify the correspondence between them. Therefore, if initial quadruplet $E_0(X_P(q, d))$ was dishonestly transformed to $E_h(X^*_P(q, d))$ by M_h in the encryption stage, even M_h in the verification stage cannot replace $E(k_{h*}, X^*_P(q)_*)$ in $E_h(X^*_P(q))$ with $E(k_{h*}, X_P(q)_*)$ that is finally decrypted to $X_P(q)$, because it does not know triplet $E_h(X^*_P(q))$ corresponding to $E_h(X^*_P(q, d))$. This means authority *A* can verify the correct encryption of each $E_0(X_P(q, d))$ by comparing $X_P(q)$ in it and $X^*_P(q)$ in decrypted triplet $E_0(X^*_P(q))$ in the verification stage, i.e. $E_N(X^*_P(q))$ is incorrect when $X_P(q) \neq X^*_P(q)$. Here, data holder P can identify triplet $E_0(X^*_P(q))$ corresponds to it in the same way as it finds pair $E^*_0(X(P))$ in the decryption stage.

By exploiting, integers $Z^{r(1,\ d)}$, ---, $Z^{r(Q,\ d)}$, verifiable features of LE-based encryption functions and features of anonymous tag based credentials, *A* and data holders also can detect dishonesties in the decryption stage, identify liable entities, and re-calculate correct results without knowing secrets of honest entities, i.e. the 5th difficulty is removed.

3.2. Behaviours of the LE-based Linear Mix-net

After quadruplets $E_0(X_P(q, d)) = \{I_q, d, X_P(q), Z^{r(q,\ d)\cdot R}\}$ (d = 1, 2, ---, D) were disclosed publicly corresponding to attribute value $X_P(q)$ of anonymous data holder P, individual mix-servers and authority *A* behave as below. In the remainder, $u_*(h)$, $v_*(h)$ and $w_*(h)$ represent products u(1)u(2)---u(h), v(N)v(N-1)---v(h) and w(*A*)w(N)w(N-1)---w(h), and as a special case $v_* = v_*(1)$ and $w_* = w_*(1)$. Where, u(h), v(h) and w(h) are integers common to all attribute values of all data holders and secrets of h-th mix-server M_h, and w(*A*) is a secret integer of authority *A* and common to all attribute values of all data holders.

3.2.1. Encryption Stage

1st mix-server M_1 in the encryption stage that picks disclosed $E_0(X_P(q, d)) = \{I_q, d, X_P(q), Z^{r(q,\ d)\cdot R}\}$ encrypts $X_P(q)$ to $E(k_1, X_P(q)_d)$ by encryption key k_1, calculates $M^{r(q,\ d)\cdot R\cdot u(1)}$ from $M^{r(q,\ d)\cdot R}$ by using secret integer u(1), and constructs quadruplet $E_1(X_P(q, d)) = \{I_q, d, E(k_1, X_P(q)_d), Z^{r(q,\ d)\cdot R\cdot u(1)}\}$. Other mix-servers behave in the same way, i.e. each M_h picks $E_{h-1}(X_P(q, d)) = \{I_q, d, E(k_{(h-1)*}, X_P(q)_d), Z^{r(q,\ d)\cdot R\cdot u*(h-1)}\}$ disclosed by M_{h-1}, encrypts $E(k_{(h-1)*}, X_P(q)_d)$ to $E(k_{h*}, X_P(q)_d)$ and calculates $Z^{r(q,\ d)\cdot R\cdot u*(h)}$ from $Z^{r(q,\ d)\cdot R\cdot u*(h-1)}$ by its secret key k_h and secret integer u(h), and constructs $E_h(X_P(q, d)) = \{I_q, d, E(k_{h*}, X_P(q)_d), Z^{r(q,\ d)\cdot R\cdot u*(h)}\}$ to disclose it publicly. As a result, $E_0(X_P(q, d))$ received by M_1 is finally transformed to $E_N(X_P(q, d))$ and $E_T(X_P(q, d))$ by M_N and M_T respectively, where each M_h shuffles its generating quadruplets of course.

Then, because no one knows all keys k_1, ---, k_N , ---, k_T nor integers u(1), ---, n(N), ---, u(T), anyone cannot link $E_0(X_P(q, d))$ to $E_N(X_P(q, d))$ or $E_T(X_P(q, d))$ unless all mix-servers conspire. Anyone except P cannot know links among P's attribute values $X_P(1)$, $X_P(2)$, ---, $X_P(Q)$ either. About encryption function $E(k_h, x)$, although each M_h shuffles its encryption results, anyone can obtain a plain and encryption forms pair of $E(k_h, x)$. Namely, if an entity calculates X_{h-1} and X_h as sums of all attribute part values in quadruplets that M_h receives and generates respectively, $\{X_{h-1}, X_h\}$ is a plain and encryption forms pair because $E(k_h, x)$ is additive. But, M_h can protect $E(k_h, x)$ from plaintext attacks because known pair is only $\{X_{h-1}, X_h\}$.

3.2.2. Verification Stage

After the encryption stage, authority *A* conducts the verification stage to examine whether individual quadruplets were honestly encrypted or not, and when dishonestly handled quadruplets are detected it asks mix-servers to carry out the encryption stage again while using new secret values including encryption keys. Also, *A* identifies dishonest mix-servers if necessary to replace them with new ones as will be discussed in Sec. 3.4.2. Therefore, the encryption stage eventually generates correct encryption results.

To examine individual quadruplets, firstly authority *A* collects $E_T(X_P(q, 1)) = \{I_q, 1, E(k_{T*}, X_P(q)_1), Z^{r(q,\ 1)\cdot R\cdot u*(T)}\}$, ---, $E_T(X_P(q, D)) = \{I_q, D, E(k_{T*}, X_P(q)_D), Z^{r(q,\ D)\cdot R\cdot u*(T)}\}$ corresponding to each attribute value $X_P(q)$ of each data holder P. Here, provided that r(q, $\underline{d}$; h) and r(q) represent products r(q, 1; h)·r(q, 2; h)---r(q, d-1; h)·r(q, d+1; h)---r(q, D; h) and r(q, 1)·r(q, 2)---r(q, D) respectively, each M_h receives $Z^{r(q,\ d)\cdot R\cdot u*(T)\cdot r(q,\ \underline{d};\ 1)\cdot r(q,\ \underline{d};\ 2)---\cdot r(q,\ \underline{d};\ h-1)}$ from M_{h-1} and calculates $Z^{r(q,\ d)\cdot R\cdot u*(T)\cdot r(q,\ \underline{d};\ 1)\cdot r(q,\ \underline{d};\ 2)---\cdot r(q,\ \underline{d};\ h-1)\cdot r(q,\ \underline{d};\ h)}$ by using its secret integer r(q, $\underline{d}$, h) to forward it to M_{h+1}. As a result, M_T calculates $Z^{r(q,\ d)\cdot R\cdot u*(T)\cdot r(q,\ \underline{d};\ 1)---\cdot r(q,\ \underline{d};\ T)} = Z^{r(q,\ d)\cdot R\cdot u*(T)\cdot r(q,\ 1)\cdot r(q,\ 2)---\cdot r(q,\ d-1)\cdot r(q,\ d+1)---r(q,\ D)} = Z^{r(q)\cdot R\cdot u*(T)}$. Therefore, *A* can collect $E_N(X_P(q, 1))$, ---, $E_N(X_P(q, D))$ that include $Z^{r(q)\cdot R\cdot u*(T)}$ as their holder part values.

After that, each M_h ($h \le Y < N$) calculates $d_{h1}E(k_{T*}, X_P(q)_1)$+ --- $+d_{hD}E(k_{T*}, X_P(q)_D)+E(k_{T*}, 0_h(P, q)) = E(k_{T*}, d_{h1}X_P(q)_1$+ --- $+d_{hD}X_P(q)_D+0_h(P, q)) = E(k_{T*}, X_P(q; h))$, and *A* calculates $\{E(k_{T*}, X_P(q; 1))$+ --- $+E(k_{T*}, X_P(q; Y))\}/Y = E(k_{T*}, X_P(q)_*)$. Here, d_{h1}, d_{h2}, ---, d_{hD} are real numbers secrets of M_h and relation d_{h1}+ --- $+d_{hD} = 1$ holds, and $E(k_{T*}, 0_h(P, q))$ is M_h's secret encryption form of value 0. Therefore each $E(k_{T*}, X_P(q; h))$ is decrypted to $X_P(q)$. In addition, $E(k_{T*}, X_P(q)_*)$ is also represented as $E(k_{T*}, X_P(q)_*) = E(k_{T*}, d_1X_P(q)_1$+ --- $+d_DX_P(q)_D+0_*(P, q))$ for some real number coefficients d_1, ---, d_D that satisfy d_1+ --- $+d_D = 1$ (where $E(k_{T*}, 0_*(P, q)) = \{E(k_{T*}, 0_1(P, q)$+ --- $+0_Y(P, q))\}/Y$), and this means $E(k_{T*}, X_P(q)_*)$ is also decrypted to $X_P(q)$. But no one knows coefficients d_1, ---, d_D or encryption form $E(k_{T*}, 0_*(P, q))$ because d_{h1}, d_{h2}, ---, d_{hD} and $E(k_{T*}, 0_h(P, q))$ are known only to M_h.

About encryption form $E(k_{T*}, 0_h(P, q))$, each M_h can generate it by choosing data holders $P_{(h1)}$, ---, $P_{(hS)}$, attribute IDs $I_{q(h1)}$, ---, $I_{q(hS)}$ and copy IDs pairs $\{d_{(h1)}, d_{\underline{(h1)}}\}$, ---, $\{d_{(hS)}, d_{\underline{(hS)}}\}$ arbitrarily as its secrets and linearly combining $\{E(k_{T*}, X_{P(hs)}(q_{(hs)})_{d(hs)})-E(k_{T*}, X_{P(hs)}(q_{(hs)})_{d\underline{(hs)}})\}$ by its secret coefficients, i.e. $E(k_{T*}, X_{P(hs)}(q_{(hs)})_{d(hs)})$ and $E(k_{T*}, X_{P(hs)}(q_{(hs)})_{d\underline{(hs)}})$ are encryption forms of same value $X_{P(hs)}(q_{(hs)})$. M_h also can identify pair $\{E(k_{T*}, X_{P(hs)}(q_{(hs)})_{d(hs)}), E(k_{T*}, X_{P(hs)}(q_{(hs)})_{d\underline{(hs)}})\}$ because they are accompanied by same holder part value $Z^{r(q(hs))\cdot R(hs)\cdot u*(T)}$.

Then *A* generates secret integer w(*A*), and constructs triplet $E_T(X_P(q)) = \{I_q, E(k_{T*}, X_P(q)_*), Z^{r(q)\cdot R\cdot u*(T)\cdot w(A)}\}$ to disclose it publicly, and mix-servers M_T, ---, M_1 in the verification stage repeatedly decrypt $E_T(X_P(q))$ to $E_0(X_P(q)) = \{I_q, X_P(q), Z^{r(q)\cdot R\cdot u*(T)\cdot w*}\}$. In detail, each M_h picks $E_h(X_P(q)) = \{I_q, E(k_{h*}, X_P(q)_*), Z^{r(q)\cdot R\cdot u*(T)\cdot w*(h+1)}\}$ disclosed by M_{h+1}, decrypts $E(k_{h*}, X_P(q)_*)$ to $E(k_{(h-1)*}, X_P(q)_*)$ calculates $Z^{r(q)\cdot R\cdot u*(T)\cdot w*(h)} = Z^{r(q)\cdot R\cdot u*(T)\cdot w*(h+1)\cdot w(h)}$ by using key k_h^{-1} and integer w(h), constructs triplet $E_{h-1}(X_P(q)) = \{I_q, E(k_{(h-1)*}, X_P(q)_*), Z^{r(q)\cdot R\cdot u*(T)\cdot w*(h)}\}$, and discloses it to be picked by M_{h-1}. As a consequence, M_1 generates decrypted triplet $E_0(X_P(q)) = \{I_q, X_P(q), Z^{r(q)\cdot R\cdot u*(T)\cdot w*}\}$.

Here, although *A* and M_N, ---, M_1 shuffle their calculation results, links among copies of quadruplets $E_T(X_P(q, 1))$, ---, $E_T(X_P(q, D))$ are revealed. Nevertheless, P can preserve its privacy, i.e. they are encryption forms of same attribute value $X_P(q)$. Also, mix-servers can protect their encryption functions from plain text attacks despite anyone can obtain plain and encryption forms pair $\{0, E(k_{T*}, 0)\}$ as above, i.e. no one knows its dummy term values.

<Detecting dishonesties in the encryption stage>

In the above, because no one knows coefficients d_1, ---, d_D, encryption form $E(k_{h*}, 0_*(P, q))$ or integer $w_*(h)$, anyone cannot link triplet $E_h(X_P(q)) = \{I_q, E(k_{h*}, X_P(q)_*), Z^{r(q)\cdot R\cdot u*(T)\cdot w*(h+1)}\}$ to corresponding quadruplet $E_h(X_P(q, d)) = \{I_q, d, E(k_{h*}, X_P(q)_d), Z^{r(q, d)\cdot R\cdot u*(h)}\}$. This means if initial quadruplet $E_0(X_P(q, d))$ is dishonestly encrypted to $E_T(X^*_P(q, d))$ in the encryption stage, any M_h in the verification stage cannot modify corresponding triplet $E_h(X^*_P(q))$ to $E_h(X_P(q))$ so that it is finally decrypted to $E_0(X_P(q)) = \{I_q, X_P(q), Z^{r(q)\cdot R\cdot u*(T)\cdot w*}\}$ (actually, 1st mix-server M_1 can do as will be discussed later).

Authority *A* detects dishonesties in the encryption stage by using this property. In detail, firstly *A* requests mix-servers to calculate integer $Z^{r(q)\cdot u*(T)\cdot w*}$ for each q to disclose it publicly. After that for each q, each data holder P calculates used seals $Z^{r(q, 1)\cdot R}$ and $Z^{r(q)\cdot R\cdot u*(T)\cdot w*}$ of its credential S(P, R) from $Z^{r(q, 1)}$ and $Z^{r(q)\cdot u*(T)\cdot w*}$, and based on $Z^{r(q, 1)\cdot R}$ and $Z^{r(q)\cdot R\cdot u*(T)\cdot w*}$ finds $E_0(X_P(q, 1))$, i.e. 1st copy of the initial quadruplet corresponding to attribute value $X_P(q)$, and triplets $E_0(X^*_P(q))$ in the verification stage.

Then, P shows initial quadruplet and decrypted triplet pair $<E_0(X_P(q, 1)), E_0(X^*_P(1))>$ to *A* while convincing *A* of its ownership of the pair by used seals $Z^{r(q, 1)\cdot R}$ and $Z^{r(q)\cdot R\cdot u*(T)\cdot w*}$, and *A* determines pair $<E_0(X_P(q, 1)) = \{I_q, 1, X_P(q), Z^{r(q, 1)\cdot R}\}, E_0(X^*_P(q)) = \{I_q, X^*_P(q), Z^{r(q)\cdot R\cdot u*(T)\cdot w*}\}>$ is inconsistent, i.e. at least one copy $E_0(X_P(q, d))$ was dishonestly handled, when $X_P(q) \neq X^*_P(q)$. Here, correctness of $X_P(q)$ in $E_0(X_P(q, 1))$ is ensured because P shows $X_P(q)$ to M_1 in its plain form. About privacy preservation, P must report pairs separately and without disclosing its identity of course.

A also can force all data holders to report their all pairs, i.e. when no one appears as the holder of pair $<E_0(X_P(q, 1)), E_0(X_P(q))>$, it asks all data holders to calculate used seals of their credentials from $Z^{r(q, 1)}$ and $Z^{r(q)\cdot u*(T)\cdot w*}$ while disclosing their identities as will be discussed in Sec. 3.4.3. In a case where mix-servers transform holder part value $Z^{r(q, d)\cdot R}$ in $E_0(X_P(q, d))$ to an invalid value P cannot identify its decrypted triplet in the verification stage, but even in this case *A* can detect the dishonestly handled quadruplet as $E_0(X_P(q, 1))$ that is not paired with any triplet.

In the above, mix-servers M_1, ---, M_T can calculate $Z^{r(q)\cdot u*(T)\cdot w*}$ as same as $Z^{r(q)\cdot R\cdot u*(T)}$ at the beginning of this subsection. Namely provided that r(q; h) represents product r(q, 1; h)·r(q, 2; h)---r(q, D; h), each M_h calculates $Z^{\{r(q;\ 1)\cdot r(q;\ 2)----r(q;\ h-1)\cdot r(q;\ h)\}\cdot\{u(1)\cdot u(2)----u(h-1)\cdot u(h)\}\cdot\{w(A)\cdot w(1)\cdot w(2)----w(h-1)\cdot w(h)\}}$ from $Z^{\{r(q;\ 1)----r(q;\ h-1)\}\cdot\{u(1)----u(h-1)\}\cdot\{w(A)\cdot w(1)----w(h-1)\}}$ given by M_{h-1} to forward it to M_{h+1} so that finally M_T calculates $Z^{\{r(q;\ 1)----r(q;\ T)\}\cdot\{u(1)----u(T)\}\cdot\{w(A)\cdot w(1)----w(T)\}} = Z^{r(q)\cdot u*(T)\cdot w*}$. Here, although P knows integer R, it cannot identify triplet $E_h(X_P(q)) = \{I_q, E(k_{h*}, X_P(q)_*), Z^{r(q)\cdot R\cdot u*(T)\cdot w*(h+1)}\}$ based on $Z^{\{r(q;\ 1)----r(q;\ h)\}\cdot\{u(1)----u(h)\}\cdot\{w(A)\cdot w(1)----w(h)\}}$ disclosed by M_h for $h > 1$, because r(q; 1)----r(q; h)u(1)----u(h)w(*A*)w(1)----w(h) and $r(q)u_*(T)w_*(h+1)$ are different.

About privacy of data holder P, P is anonymous, and because it reports pairs $<E_0(X_P(1, 1)), E_0(X_P(1))>$, ---, $<E_0(X_P(Q, 1)), E_0(X_P(Q))>$ separately, anyone other than P cannot know links among $X_P(1)$, ---, $X_P(Q)$. Although $E_0(X_P(q))$ includes plain attribute value $X_P(q)$, $X_P(q)$ is publicly known from the beginning.

<Dishonest 1st mix-server M_1>

1st mix-server M_1 in the encryption stage can encrypt $E_0(X_P(q, d))$ dishonestly while making triplet $E_0(X_P(q))$ in the verification stage include consistent attribute value $X_P(q)$. For example, provided that $X^*_P(q)$ is a unique value, even if M_1 dishonestly encrypted $E_0(X_P(q, d)) = \{I_q, d, X_P(q), Z^{r(q, d)\cdot R}\}$ to $E_1(X^*_P(q, d)) = \{I_q, d, E(k_1, X^*_P(q)), Z^{r(q, d)\cdot R\cdot u(1)}\}$ in the encryption stage for each d, because $X^*_P(q)$ is unique it can identify incorrect triplet $E_0(X^*_P(q)) = \{I_q, X^*_P(q), Z^{r(q)\cdot R\cdot u*(T)\cdot w*}\}$ in the verification stage and replace $X^*_P(q)$ in it with $X_P(q)$ to produce correct decrypted triplet $E_0(X_P(q)) = \{I_q, X_P(q), Z^{r(q)\cdot R\cdot u*(T)\cdot w*}\}$.

To disable above dishonesties, in other words, to convince others that M_1 is honest, after completing the verification stage, *A* discloses its secret integer w(*A*) and asks mix-servers M_1, M_2, ---, M_Y (Y < N) to disclose their encryption keys k_1, k_2, ---, k_Y, and secret integers u(1), u(2), ----, u(Y), w(1), W(2), ---, w(Y). Namely, by the disclosed information, anyone can confirm M_1 is honest. On the other hand, encryption keys k_{Y+1}, ---, k_N, ---, k_T and integers u(Y+1), ---, u(N), ---, u(T), w(Y+1), ---, w(N), ---, w(T) are still secrets of M_{Y+1}, ---, M_T, therefore secrets of honest data holders can be preserved. Also, *A* and mix-servers replace secret integers w(*A*), u(1), ---, u(T), w(1), ---, w(T) and encryption keys k_1, ---, k_T with new ones for handling new sets of attribute values.

About last mix-server M_T, if it conspires with authority *A*, it also can encrypt $E_{T-1}(X_P(q, d))$ to $E_T(X^*_P(q, d))$ dishonestly and decrypt $E_T(X^*_P(q))$ to $E_{T-1}(X_P(q))$ that is finally decrypted to correct value $X_P(q)$, but incorrect $E_T(X^*_P(q, d))$ does not affect the final calculation results because M_{N+1}, M_{N+2}, ---, M_T are not involved in the decryption stage.

3.2.3. Calculating Encrypted Weighted Sums of Attribute Values

To calculate encrypted weighted sums of attribute values corresponding to individual data holders, provided that $r_q(h)$ and r_* represent products {r(1, 1; h)·r(2, 1; h)---r(q-1, 1; h)·r(q+1, 1; h)---r(Q, 1; h)} and {r(1, 1)·r(2, 1)---r(Q, 1)} respectively, authority *A* asks mix-servers M_1, ---, M_T to transform the holder part value of each quadruplet $E_N(X_P(q, 1)) = \{I_q, 1, E(k_{N*}, X_P(q)_1), Z^{r(q, 1)\cdot R\cdot u*(N)}\}$ from $Z^{r(q, 1)\cdot R\cdot u*(N)}$ to $Z^{R\cdot u*(N)\cdot r*}$, i.e. each M_h calculates $Z^{r(q, 1)\cdot R\cdot u*(N)\cdot rq(1)\cdot rq(2)---rq(h-1)\cdot rq(h)}$ from $Z^{r(q, 1)\cdot R\cdot u*(N)\cdot rq(1)\cdot rq(2)--- rq(h-1)}$ received from M_{h-1} to forward the result to M_{h+1}. Therefore, finally M_T calculates $Z^{r(q, 1)\cdot R\cdot u*(N)\cdot rq(1)---rq(T)} = Z^{R\cdot u*(N)\cdot r(1, 1)---r(Q, 1)} = Z^{R\cdot u*(N)\cdot r*}$, and same value $Z^{R\cdot u*(N)\cdot r*}$ is assigned to P's quadruplets $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$ as their holder part values. Then, *A* can collect $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$ from all quadruplets disclosed by M_N to calculate the encrypted weighted sum of P's attribute values as $E(k_{N*}, X(P)) = E(k_{N*}, a_1X_P(1)_1+ --- +a_QX_P(Q)_1) = a_1E(k_{N*}, X_P(1)_1)+ --- +a_QE(k_{N*}, X_P(Q)_1)$ and to construct pair $E^*_N(X(P)) = \{E(k_{N*}, X(P)), Z^{R\cdot u*\cdot r*}\}$.

3.2.4. Decryption Stage

In the decryption stage, mix-servers M_N, M_{N-1}, ---, M_1 repeatedly decrypt pair $E^*_N(X(P))$ to $E^*_{N-1}(X(P)) = \{E(k_{(N-1)*}, X(P)), Z^{R\cdot u*(N)\cdot r*\cdot v(N)}\}$, $E^*_{N-2}(X(P)) = \{E(k_{(N-2)*}, X(P)), Z^{R\cdot u*(N)\cdot r*\cdot v*(N-1)}\}$, ---, $E^*_0(X(P)) = \{X(P), Z^{R\cdot u*(N)\cdot r*\cdot v*}\}$. Here, although each M_h shuffles its decryption results *A* can verify correct decryptions of $E^*_N(X(P))$ partially without knowing secrets of mix-servers by exploiting additive feature of each $E(k_h, x)$. Namely, if *A* calculates sums of attribute part values in all pairs received and generated by M_h as $\underline{X}_h$ and $\underline{X}_{h-1}$ respectively, both $\underline{X}_h$ and $\underline{X}_{h-1}$ must be encryption forms of $\underline{X}$, weighted sum of all attribute values of all data holders. This means $\{\underline{X}_{h-1}, \underline{X}_h\}$ is a plain and encryption forms pair of $E(k_h, x)$, and additive (as a result verifiable) feature of $E(k_h, x)$ enables *A* to force mix-servers to decrypt pairs so that sums of their decrypted attribute part values coincide with $\underline{X}$ even if they decrypt individual pairs dishonestly.

About encryption function $E(k_h, x)$ of each mix-server M_h, because M_h shuffles its decryption results, no one other than M_h can identify plain and encryption forms pair $\{E(k_{(h-1)*}, X(P)), E(k_{h*}, X(P))\}$ except the above pair $\{\underline{X}_{h-1}, \underline{X}_h\}$. Therefore, M_h can protect $E(k_h, x)$ from plain text attacks also in the decryption stage.

Nevertheless, data holder P can find its pair $E^*_0(X(P)) = \{X(P), Z^{R\cdot u*(N)\cdot r*\cdot v*}\}$ to take actions for the pair. Firstly, *A* asks mix-servers to calculate $Z^{u*(N)\cdot r*\cdot v*}$ in the same way as in Sec. 3.2.3. Then, each P calculates $(Z^{u*(N)\cdot r*\cdot v*})^R = Z^{R\cdot u*(N)\cdot r*\cdot v*}$ from $Z^{u*(N)\cdot r*\cdot v*}$ based on integer R in its credential S(P, R), finds pair $E^*_0(X(P))$ according to $Z^{R\cdot u*(N)\cdot r*\cdot v*}$, and convinces *A* of its ownership of $E^*_0(X(P))$. Where, although P is anonymous, *A* can confirm P's ownership of pair $E^*_0(X(P))$, because P must calculate $Z^{R\cdot u*(N)\cdot r*\cdot v*}$ honestly as a used seal of S(P, R).

3.3. Detecting Dishonesties

In the encryption stage, data holder P puts its initial quadruplet $E_0(X_P(q, d)) = \{I_q, d, X_P(q), Z^{r(q, d)\cdot R}\}$ honestly, because P shows $X_P(q)$ in its plain form and it calculates $Z^{r(q, d)\cdot R}$ from $Z^{r(q, d)}$ as a used seal of its credential. This means P cannot behave dishonestly in any stage. Also, the verification stage ensures that mix-servers in the encryption stage eventually generate correct encryption results. Therefore, remaining dishonesties to be detected are ones in the decryption stage, i.e. each mix-server M_h may replace elements of pair $E^*_h(X(P))$ with those of other pair $E^*_h(X(P_*))$ and may simply decrypt $E^*_h(X(P))$ incorrectly.

Provided that volume of duties P must accomplish increases (or decreases) when weighted sum of P's attribute values X(P) increases, authority *A* can easily detect above dishonesties as below. Namely as mentioned in the previous subsection, each mix-server M_h in the decryption stage must decrypt each pair $E^*_h(X(P))$ it receives from M_{h+1} so that $\underline{X}_{h-1}$, sum of attribute part values in all pairs it generates, is finally decrypted to $\underline{X}$, the weighted sum of attribute values of all data holders. Therefore, if dishonesties bring any benefit to a data holder some other data holder necessarily suffers loss. For example, if $X(P_*)$ is the amount data holder P_* must pay, when $X(P_*)$ becomes less than the actual value, for at least one other data holder P, weighted sum of its attribute values X(P) becomes larger than the actual value. As a result, P claims decrypted pair $E^*_0(X(P))$ is incorrect. It is also possible to endow each P with the ability to automatically notice *A* that $E^*_0(X(P))$ is incorrect by distributing adequate computer programs to data holders.

Also, each data holder P must appear to take actions for $E^*_0(X(P))$ as will be discussed in Sec. 3.4.3, although X(P) in it may not be correct. Here, M_h may transform a holder part value of $E^*_h(X(P)) = \{E(k_{h*}, X(P)), Z^{R\cdot u*(N)\cdot r*\cdot v(h+1)}\}$ in the decryption stage to an invalid value or to the one corresponding to data holder P_* different from P. But decrypted pairs accompanied by invalid holder part values are detected as the ones of which holders cannot be identified by the procedure in Sec. 3.4.3. In the latter case, some data holders claim that weighted sums of their attribute values are incorrect.

3.4. Identifying Dishonest Entities

3.4.1. Dishonest Mix-servers in the Decryption Stage

When authority *A* determines some decrypted pairs include invalid holder part values or some data holders claim that decrypted pairs corresponding to them are incorrect, *A* can identify liable entities and re-calculate correct pairs without knowing secrets of honest entities. In the following, pair $E^*_0(X^*(P)) = \{X^*(P), Z^{R\cdot u^*(N)\cdot r^*\cdot v^*}\}$ in the decryption stage is assumed incorrect, i.e. it is a pair that includes an invalid holder part value or that is claimed as inconsistent by anonymous data holder P.

To identify mix-servers liable for each incorrect pair $E^*_0(X^*(P))$, authority *A* requests mix-servers M_1, ---, M_N to prove their correct decryptions. In detail, M_1 finds pair $E^*_1(X^*(P)) = \{E(k_1, X^*(P)), Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(2)}\}$ that corresponds to $E^*_0(X^*(P))$, and *A* confirms that M_1 certainly had received $E^*_1(X^*(P))$ from M_2 in the decryption stage and $E^*_0(X^*(P))$ is the correct decryption form of $E^*_1(X^*(P))$. In the same way, each M_h finds pair $E^*_h(X^*(P)) = \{E(k_{h^*}, X^*(P)), Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}\}$ corresponds to $E^*_{h-1}(X^*(P)) = \{E(k_{(h-1)^*}, X^*(P)), Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}\}$ forwarded by M_{h-1}, and *A* confirms M_h certainly had received $E^*_h(X^*(P))$ from M_{h+1} and $E^*_{h-1}(X^*(P))$ is the correct decryption form of $E^*_h(X^*(P))$. Namely, M_h is dishonest when $E^*_h(X(P))$ that is decrypted to $E^*_{h-1}(X^*(P))$ does not exist.

Here, *A* can examine the consistency of pair $<E^*_h(X^*(P)), E^*_{h-1}(X^*(P))>$ without knowing secrets of M_h. Namely, consistency of attribute part values $E(k_{h^*}, X^*(P))$ and $E(k_{(h-1)^*}, X^*(P))$ can be verified because $E(k_h, x)$ is additive and verifiable (moreover, encryption keys of M_1, ---, M_Y are disclosed as in Sec. 3.2.2). About holder part values, pair $\{Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}, Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}\}$ is consistent if $Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}$ is calculated from $Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}$ by using M_h's secret integer v(h) that is common to all attribute values of all data holders. Therefore, when *A* calculates product of holder part values in all pairs M_h had generated in the decryption stage as $Z_*^{R^*\cdot u^*(N)\cdot r^*\cdot v^*(h)}$ for each h and defines $Z_-^{\underline{R}\cdot u^*(N)\cdot r^*\cdot v^*(h)} = Z_*^{R^*\cdot u^*(N)\cdot r^*\cdot v^*(h)}/Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}$, along the scheme of Diffie and Hellman [1], it can determine pair $\{Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}, Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}\}$ is consistent if $Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}$ and $Z_-^{\underline{R}\cdot u^*(N)\cdot r^*\cdot v^*(h)}$ are calculated as $(Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)})^{v(h)}$ and $(Z_-^{\underline{R}\cdot u^*(N)\cdot r^*\cdot v^*(h+1)})^{v(h)}$ by same unknown integer v(h).

In the following notations $Z_P(h+1)$, $Z_-(h+1)$, $Z_P(h)$ and $Z_-(h)$ represent $Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}$, $Z_-^{\underline{R}\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}$, $Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}$ and $Z_-^{\underline{R}\cdot u^*(N)\cdot r^*\cdot v^*(h)}$ respectively, therefore pair $\{Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}, Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h)}\}$ is consistent if relations $Z_P(h+1)^{v(h)} = Z_P(h)$ and $Z_-(h+1)^{v(h)} = Z_-(h)$ hold for same unknown v(h). Also it must be noted that $Z_P(h+1)Z_-(h+1) = Z_*^{R^*\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}$. Then, to confirm relations $Z_P(h+1)^{v(h)} = Z_P(h)$ and $Z_-(h+1)^{v(h)} = Z_-(h)$, *A* generates its secret integers δ_1, δ_2, δ_3, calculates pairs $\{Z_P(h+1)^{\delta 1}, Z_P(h)^{\delta 1}\}$, $\{Z_-(h+1)^{\delta 2}, Z_-(h)^{\delta 2}\}$, $\{(Z_P(h+1)Z_-(h+1))^{\delta 3}, (Z_P(h)Z_-(h))^{\delta 3}\}$, and shows $Z_P(h+1)^{\delta 1}$, $Z_-(h+1)^{\delta 2}$, $(Z_P(h+1)Z_-(h+1))^{\delta 3}$ to M_h. After that M_h calculates $(Z_P(h+1)^{\delta 1})^{v(h)}$, $(Z_-(h+1)^{\delta 2})^{v(h)}$, $\{(Z_P(h+1)Z_-(h+1))^{\delta 3}\}^{v(h)}$ from them and its secret integer v(h), and finally *A* confirms that $Z_P(h)$ and $Z_-(h)$ were calculated by same v(h) if relations $(Z_P(h+1)^{\delta 1})^{v(h)} = Z_P(h)^{\delta 1}$, $(Z_-(h+1)^{\delta 2})^{v(h)} = Z_-(h)^{\delta 2}$ and $\{(Z_P(h+1)Z_-(h+1))^{\delta 3}\}^{v(h)} = \{Z_P(h)Z_-(h)\}^{\delta 3}$ hold.

In the above, even if M_h had calculated $Z_P(h)$ and $Z_-(h)$ as $Z_P(h+1)^{\lambda}$ and $Z_-(h+1)^{v(h)}$ respectively while using different integers λ and v(h), it still can satisfy relations $(Z_P(h+1)^{\delta 1})^{\lambda} = Z_P(h)^{\delta 1}$ and $(Z_-(h+1)^{\delta 2})^{v(h)} = Z_-(h)^{\delta 2}$ despite it does not know δ_1 or δ_2. But according to the difficulty of solving discrete logarithm problems, it cannot find integer λ_* that satisfies relation $\{(Z_P(h+1)(Z_-(h+1))^{\delta 3}\}^{\lambda^*} = (Z_P(h)Z_-(h))^{\delta 3} = (Z_P(h+1)^{\lambda}Z_-(h+1)^{v(h)})^{\delta 3}$, because it cannot represent $Z_P(h+1)$ or $Z_-(h+1)$ as a function of $Z_-(h+1)$ or $Z_P(h+1)$ respectively.

Then, once M_h was identified as dishonest, M_h, ---, M_1 must honestly decrypt $E^*_h(X(P))$ to generate correct decrypted pair $E^*_0(X(P)) = \{X(P), Z^{R\cdot u^*(N)\cdot r^*\cdot v^*}\}$ (because *A* can verify correct decryption of M_h as above). About data holder P, it can conceal the correspondence between X(P) and it because it is still anonymous. But it must be noted that to protect $E(k_h, x)$ from plain text attacks (in other words not to disclose many plain and encryption forms pairs) *A* can examine only predefined number of incorrect decrypted pairs even if many decrypted pairs are determined as incorrect as will be discussed in Sec. 3.4.2. In addition, in cases where last mix-server M_N is dishonest, incorrect pair $<E^*_N(X(P)), E^*_{N-1}(X^*(P))>$ is disclosed, and corresponding data holder P may obtain plain and encryption forms pairs $\{X_P(1), E(k_{N^*}, X_P(1)_1)\}$, ---, $\{X_P(Q), E(k_{N^*}, X_P(Q)_1)\}$, i.e. P knows its attribute values $X_P(1)$, ---, $X_P(Q)$ and can know encrypted quadruplets $E_N(X_P(1, 1))$, ---, $E_N(X_P(Q, 1))$ as *A* collects them to calculate $E^*_N(X(P))$. Despite $E(k_1, x)$, ---, $E(k_N, x)$ are weak against plain text attacks, still they can be protected because the number of disclosed pairs is not large except cases where many attribute values are assigned to each data holder. But when individual data holders have many attribute values *A* must carry out all stages again as below.

Namely, when *A* cannot identify dishonest mix-servers among M_1, M_2, ---, M_U ($U < N$), it conducts all stages again from the encryption stage while arraying mix-servers in the different order so that M_{U+1}, M_{U+2}, ---, M_N are allocated before M_1, M_2, ---, M_U in the encryption stage. Then, if M_N behaves dishonestly again *A* can identify it without worrying about the disclosure of numbers of pairs $\{X_P(1), E(k_{N^*}, X_P(1)_1)\}$, ---, $\{X_P(Q), E(k_{N^*}, X_P(Q)_1)\}$. Fortunately, usually authority *A* or mix-servers do not behave dishonestly, because they are not anonymous, their dishonesties are necessarily revealed and they cannot continue their businesses after their dishonesties are revealed. This means that in actual applications authority *A* can conduct all stages again from the start without degrading the performance. Another fortunate thing is data holders are not required to put their attribute values again to re-conduct individual stages.

3.4.2. Dishonest Mix-servers in the encryption and the verification stages

As in Sec. 3.2.2, authority *A* conducts the encryption stage again when initial quadruplet $E_0(X_P(q, 1))$ is not paired with any triplet in the verification stage or some data holder P claims pair $<E_0(X_P(q, 1)), E_0(X^*_P(q))>$ is inconsistent. But if *A* wants to remove dishonest mix-servers or replace them with new ones, it must identify dishonest mix-servers. *A* can identify dishonest mix-servers in the encryption and the verification stages without knowing secrets of honest entities as below.

Provided that pair $<E_0(X_P(q, 1)), E_0(X^*_P(q))>$ is inconsistent or $E_0(X_P(q, 1))$ is not accompanied by any triplet, to identify dishonest mix-servers, firstly 1st mix-server M_1 encrypts $E_0(X_P(q, 1))$, ---, $E_0(X_P(q, D))$ to $E_1(X_P(q, 1))$, ---, $E_1(X_P(q, D))$ by using secret parameters that it had used in the encryption stage. In the same way, each M_h encrypts quadruplets $E_{h-1}(X_P(q, 1))$, ---, $E_{h-1}(X_P(q, D)$ forwarded by M_{h-1} to $E_h(X_P(q, 1))$, ---, $E_h(X_P(q, D))$ to forward the result to M_{h+1}, and M_h is determined as dishonest if it cannot show consistent pair $<E_{h-1}(X_P(q, d)), E_h(X_P(q, d))>$.

Authority *A* identifies dishonest mix-servers in the verification stage in the same way. Here, *A* can verify M_h's correct encryption of $E_{h-1}(X_P(q, d))$ and correct decryption of $E_h(X^*{}_P(q))$ without knowing secrets of M_h as same as in the decryption stage. But when the number of inconsistent pairs is large, many plain and encryption forms pairs are disclosed. Therefore, *A* examines only predefined number of inconsistent pairs even if many pairs are inconsistent as same as in Sec. 3.4.1. Namely, after identifying dishonest mix-servers based on the limited number pairs, it conducts the encryption and the verification stages again, and identifies remaining dishonest mix-servers if exist. Here, re-executions of the stages do not degrade the performance in actual applications because usually *A* or mix-servers do not behave dishonestly as discussed previously.

3.4.3. Data Holders without Responses

In the decryption stage, authority *A* can force each anonymous data holder P to honestly report decrypted pair $E^*{}_0(X(P)) = \{X(P), Z^{R\cdot u^*(N)\cdot r^*\cdot v^*}\}$ corresponds to it as below. *A* in the verification stage also can force each P to report pair $<E_0(X_P(q, 1)), E_0(X_P(q))>$ honestly in the same way.

When no one appears as the holder of pair $E^*{}_0(X(P))$, conceptually, *A* asks all registered data holders to calculate used seals of their credentials from value $Z^{u^*(N)\cdot r^*\cdot v^*}$, which is calculated by M_1, ---, M_T as same as $Z^{R\cdot u^*(N)\cdot r^*}$ in Sec. 3.2.3, while disclosing their identities, and identifies P that calculates $(Z^{u^*(N)\cdot r^*\cdot v^*})^R$ as the holder. Namely, only P that knows R in credential S(P, R) can calculate holder part value $Z^{R\cdot u^*(N)\cdot r^*\cdot v^*}$ in $E^*{}_0(X(P))$ from $Z^{u^*(N)\cdot r^*\cdot v^*}$ and P must calculate it honestly.

But if honest data holder P_j calculates $(Z^{u^*(T)\cdot r^*\cdot v^*})^{R(j)}$, because P_j is disclosing its identity *A* can know that pair $E^*{}_0(X(P_j)) = \{X(P_j), Z^{R(j)\cdot u^*(N)\cdot r^*\cdot v^*}\}$ belongs to P_j despite P_j is honest. Therefore instead of $(Z^{u^*(N)\cdot r^*\cdot v^*})^{R(j)}$, P_j calculates used seal $(Z^{u^*(N)\cdot r^*\cdot v^*\cdot \mu})^{R(j)}$ while generating its secret integer μ. In detail, P_j calculates pair $\{Z_*{}^{\mu} = Z^{u^*(N)\cdot r^*\cdot v^*\cdot \mu}, Z_R{}^{\mu} = Z^{R\cdot u^*(N)\cdot r^*\cdot v^*\cdot \mu}\}$ from $Z_* = Z^{u^*(N)\cdot r^*\cdot v^*}$ and $Z_R = Z^{R\cdot u^*(N)\cdot r^*\cdot v^*}$, and calculates used seal $(Z_*{}^{\mu})^{R(j)}$ to show it with pair$\{Z_*{}^{\mu}, Z_R{}^{\mu}\}$, after that *A* compares $(Z_*{}^{\mu})^{R(j)}$ and $Z_R{}^{\mu}$. Then, P_j can conceal the correspondence between it and $E^*{}_0(X(P_j))$ because P_j did not calculate $(Z_*{}^{\mu})^{R(j)}$ before.

Here, *A* can confirm that P_j used same μ for calculating $Z_*{}^{\mu}$ and $Z_R{}^{\mu}$ without knowing μ as same as in Sec. 3.4.1 [12]. But it must be noted that different from pair $\{Z^{R\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}, Z_{-}^{\underline{R}\cdot u^*(N)\cdot r^*\cdot v^*(h+1)}\}$ in Sec. 3.4.1, P that knows its secret integer R can calculate Z_R in pair $\{Z_*, Z_R\}$ as a function of Z_*, i.e. $Z_R = Z_*{}^R$. Therefore, when P defines integers μ_* and μ_2 arbitrarily, calculates μ_1 as $\mu_1 = (R+1)\mu_*-R\mu_2$ and reports pair $\{Z_*{}^{\mu 1}, Z_R{}^{\mu 2}\}$ instead of $\{Z_*{}^{\mu}, Z_R{}^{\mu}\}$, for $Z_*{}^{\delta 1}$, $Z_R{}^{\delta 2}$ and $(Z_*Z_R)^{\delta 3}$ that *A* calculates by using its secret integers δ_1, δ_2, δ_3, P can show consistent values $(Z_*{}^{\delta 1})^{\mu 1}$, $(Z_R{}^{\delta 2})^{\mu 2}$ and $\{(Z_*Z_R)^{\delta 3}\}^{\mu^*}$. Namely, $(Z_*{}^{\delta 1})^{\mu 1} = Z_*{}^{\mu 1\cdot \delta 1}$, $(Z_R{}^{\delta 2})^{\mu 2} = Z_R{}^{\mu 2\cdot \delta 2}$ and $\{(Z_*Z_R)^{\delta 3}\}^{\mu^*} = \{(Z_*{}^{R+1})^{\delta 3}\}^{\mu^*} = (Z_*{}^{\mu 1+R\cdot \mu 2})^{\delta 3} = (Z_*{}^{\mu 1}Z_R{}^{\mu 2})^{\delta 3}$.

To disable P to use relation $Z_R = Z_*{}^R$, *A* defines integer β and examines consistencies of 2 pairs $\{\beta^{\mu}, Z_*{}^{\mu}\}$ and $\{\beta^{\mu}, Z_R{}^{\mu}\}$. Then, because P cannot represent β or Z_* as a function of Z_* or β, P must calculate β^{μ} and $Z_*{}^{\mu}$ by using same integer μ. In the same way, P must calculate β^{μ} and $Z_R{}^{\mu}$ by using same integer μ, as a consequence, *A* can convince itself that $Z_*{}^{\mu}$ and $Z_R{}^{\mu}$ were calculated by same integer μ.

4. Conclusion

As above, linear Mix-nets based on LE-based encryption functions can calculate linear combinations of real numbers owned by same data holders while preserving privacies of data holders and protecting encryption functions from plain text attacks. Here, it is apparent that linear Mix-nets can have totally the same features even if MA-based encryption functions are used instead of LE-based ones. Therefore, when LE-based encryption functions are replaced with MA-based ones which are both additive and multiplicative, they can calculate also general polynomial functions of attribute values.

About dishonesties of relevant entities, proposed linear Mix-net successfully detects inconsistent encryption and decryption results, and identifies dishonest entities to generate correct results without disclosing secrets of honest entities. An advantage in handling dishonesties is data holders cannot behave dishonestly. Therefore, together with the fact that authority *A* or mix-servers do not behave dishonestly usually (because they are not anonymous and their dishonesties are necessarily revealed), procedures including re-executions of stages for identifying dishonest entities and re-calculating correct encryption and decryption results do not degrade the performance in actual applications.

References

[1] Diffie, W. and Hellman, M. E., "New directions in cryptography," *IEEE Trans. On Information Theory*, IT-22 (6). 644-654. 1976.

[2] Chaum, D., "Untraceable electronic mail, return address and digital pseudonyms," *Communications of the ACM*, 24 (2). 84-88. 1981.

[3] Cormen, T., Leiserson, C., Rivest, R. and Stein, C., *Introduction to algorithms*, MIT Press and McGraw-Hill, 2001.

[4] Lee, B., Boyd, C., Dawson, E., Kim, K., Yang, J., and Yoo, S., "Providing receipt-freeness in Mixnet-based voting protocols," *Proc. of the ICISC '03*, 261-274. 2003.

[5] Golle, P. and Jakobsson, M., "Reusable anonymous return channels," *Proc. of the 2003 ACM Workshop on Privacy in the Electronic Society*, 94-100. 2003.

[6] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A. and Shacham, H., "Randomizable proofs and delegatable anonymous credentials," *Proc. of the 29th Annual International Cryptology Conference on Advances in Cryptology*, 108-125. 2009.

[7] Shahandashti, S. F. and Safavi-Naini, R., "Threshold attribute-based signatures and their application to anonymous credential systems," *Proc. of the 2nd International Conference on Cryptology in Africa: Progress in Cryptology*, 198-216. 2009.

[8] Gentry, C., "Fully homomorphic encryption using ideal lattices," *Proc. of Symposium on theory of computing –STOC 2009*, 169-178. 2009.

[9] Chung, K., Kalai, Y. and Vadhan, S., "Improved Delegation of Computation Using Fully Homomorphic Encryption," *CRYPT 2010, LNCS 6223*, 483-501. 2010.

[10] Tamura, S., *Anonymous Security Systems and Applications: Requirements and Solutions*, Information Science Reference, 2012.

[11] Tamura, S. and Taniguchi, S., "A scheme for collecting anonymous data," *Proc. of IEEE-ICIT2013*, 1210-1215. 2013.

[12] Tamura, S. and Taniguchi, S., "Enhanced Anonymous Tag Based Credentials," *Information Security and Computer Fraud*, 2 (1). 10-20. 2014.