RESEARCH ARTICLE OPEN ACCESS

An Anonymous System using Single Sign-on Protocol

Manabu Okamoto*
**Kanagawa Institute of Technology, Shimo-ogino1030 Atsugi Kanagawa Japan*

ABSTRACT
Anonymity is essential in various online scenarios, such as a questionnaire system for lecture evaluation in a university. In such a scenario, only a person who attended the lecture concerned should be able to access the system and fill out the questionnaire. Further, anonymity is very important because only anonymous users will answer the questions honestly. Making users who participate from the beginning anonymous is relatively easy. In contrast, making users who participate in the middle anonymous is very difficult. In this paper we propose a new anonymous system based on single sign-on (SSO)—an authentication process that allows a user to access multiple services with one set of login credentials. The proposed system makes all users, both from the beginning and those who participate in the middle, anonymous.

***Keywords*:** Questionnaire, security, single sign-on, user anonymity.

I. INTRODUCTION

Anonymous systems are required in a variety of situations, such as elections [1], whistle-blowing, questionnaires, and bulletin board systems (BBS). Without anonymity these processes may not be accomplished satisfactorily.

For example, when students are asked to complete questionnaires about lectures at a university, no one (including teachers, students, and service managers) should know who submitted what answer. Without anonymity many students would not be comfortable giving an honest evaluation as they may fear retribution from the lecturer concerned. Thus, without anonymity all users may refrain from giving bad evaluations and so the results obtained may not be an accurate reflection of the quality of the lecture. Paper questionnaires allow students to give honest answers as they do not have to submit their IDs, names, and affiliations. However, paper questionnaires are labor-intensive, time-consuming, and costly. Further, filling out a questionnaire at an appointed location and at an appointed time can be a bit onerous for students.

Questionnaires on the web enable students to complete questionnaires simply by launching a browser anywhere in their own time until the deadline. In addition, the aggregation process is much easier because the answers can be tabulated electronically. Naturally, anonymity is also important on the web. Moreover, considering rules such as one person being able to vote only once and students being able to change their votes as many times as they desire before the deadline, user account management is also necessary.

In this paper, we propose a new anonymous system that can be used for anonymous questionnaire systems and other user account systems that require anonymity. The proposed method utilizes a single sign-on (SSO) mechanism—which allows a user to access multiple services with one set of login credentials—to manage users' accounts and make all users anonymous. The proposed method enables both users who participate from the beginning of a process and those who participate in the middle anonymous again easily.

II. ASSUMPTIONS MADE IN THE PROPOSED SYSTEM

We assume the following conditions in the proposed system:

- Only users who have proper access rights can use the system.
- One person can make only one submission in one term.
- There is a deadline. However, users can change the content of their answers such as a vote or an answer to a questionnaire any number of times prior to the deadline. In other words, users can manage their accounts on the system and save their answers in the middle of writing to their account. In addition, the system manager can manage all user accounts.
- The system has anonymity. At no time can anyone, including the system manager, ascertain what content is submitted by any specific user.
- However, if any problem arises, then the source should be traceable.

For example, only students who registered for a particular lecture at a university should be able to answer questionnaires related to that lecture. Each student can submit only one questionnaire for each lecture. Further, until the deadline, which may be the next lecture, students can save their partially completed questionnaires to their user account.

Furthermore, no one, including users and system managers, can know who completed any particular questionnaire. We propose a new method that satisfies these conditions.

III. RELATED WORK

Various types of anonymity techniques exist. They include proxy servers, mix-net, and onion routers. Proxy servers [2] serve as hubs through which internet requests are processed. Client computers that utilize proxy servers send requests to the proxy server, which then processes the request and returns the requested information to the client. Proxy servers act as intermediaries between clients and the web servers on the internet. They can filter web content, circumvent restrictions such as parental blocks, and also provide anonymity for users on the internet. Web servers recognize requests as coming from the proxy server and do not know the client itself.

Mix-nets [3–5] provide anonymous channels. They provide anonymity and privacy by shuffling information sent from clients. Anonymity is kept unless all servers conspire. Mix-nets are used for electronic voting. Multiple mix-net servers are aligned and each voter encrypts his/her vote with the public key of each mix-net server in sequence and sends it to the first mix-net server. Then, each mix-net server decrypts the vote and mixes it with other votes. In the end, no one can trace the votes that arrive at the tallying server through all the mix-net servers.

Onion routers [6] also use encryption and shuffle to provide anonymity. An onion network encapsulates messages in layers of encryption like layers of an onion. The encrypted data are transmitted through onion routers, with each decrypting the encryption and peeling away a layer to find the next destination. The final layer is decrypted when the message arrives at its destination. The sender remains anonymous because each router knows only the location of the preceding and next router. However, when a user wants to use this anonymous channel, he/she has to select onion routers, obtain their public keys, and make a capsule.

An anonymous channel is indispensable for our system, especially at the network layer. For example, in the lecture questionnaire scenario, if a respondent is traced by IP address then an anonymity problem arises. However, when users use an anonymous channel conventionally, it is very burdensome because of the need to, for example, select routers, acquire keys, and encrypt messages.

However, the necessity of anonymity on a network is lower for a lecture questionnaire than for a general political election. Further, many university networks utilize DHCP for students, which provides "negative anonymity." DHCP servers provide users with IP addresses. However, the same user may get different IP addresses from time to time. Thus, such a user cannot be traced by IP address. The manager of a DHCP server can obtain the MAC address according to the IP address, but tracing a user via MAC address is still difficult. In many cases, the manager of a university network does not manage the MAC addresses of all students. This situation, which we term "negative anonymity," is sufficient anonymity for a lecture questionnaire.

In this paper, we do not refer to the anonymous channel. We assume that anonymity at the network layer is achieved via a technique such as onion router, mix-net, or "negative anonymity."

IV. ISSUES WITH ANONYMOUS SYSTEMS

In this paper, we deal with anonymity not at the network layer but at the application layer. We discuss this issue in this section. It is very easy to make users anonymous in web systems. Anonymity can be easily achieved by web systems managers generating many IDs and passwords for web systems, printing them on paper, placing each set in an envelope, shuffling the envelopes, and then distributing them to users. Because all the envelopes would look the same, no one would be able to trace an individual user's ID and password.

However, in such a system, anonymity problems would arise when users enter in the middle of the process. If only one person joined in the middle, the manager would make one ID and one password but would not be able to shuffle this set with others in order to guarantee anonymity. Consequently, anyone could guess that the person with the new ID and password is a new participant. Further, rectifying this situation by generating new IDs and passwords for all users would be burdensome.

We call this situation "participant in the middle issue." We need to make users including participants in the middle anonymous again efficiently. In this paper, we propose a new method that can re-anonymize users including participants in the middle easily using SSO.

V. SINGLE SIGN-ON (SSO)

In this section, we explain the concept of SSO. SSO is a mechanism whereby a single user authentication action enables the user to access multiple web services without needing to enter multiple sets of credentials. OpenID [7] is an open SSO standard. When a user uses a service provider (called a relying party (RP) in the OpenID glossary), the RP redirects the user to an OpenID provider (OP) and the OP authenticates the user and brings him/her back to the RP with an authentication response. The RP then confirms the response from the OP and authenticates the user. Fig. 1 shows a basic OpenID authentication sequence.

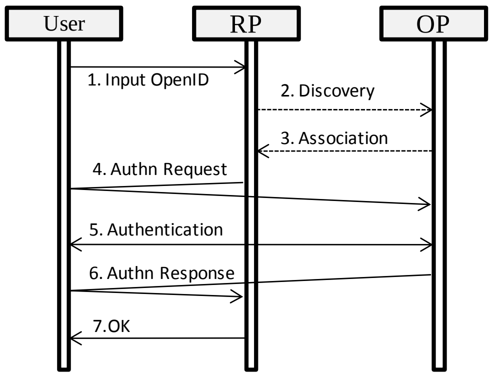

Fig. 1. OpenID authentication sequence.

SSO facilitates the use of multiple RPs without the need to input ID and password in each RP; instead certification is received from one OP. We use SSO on multi-layer RPs to re-anonymize users. OP is used as an identity server, one RP is used as a web service server that provides service for users such as lecture questionnaires, and other RPs are used as background servers for anonymity.

VI. THE PROPOSED METHOD

In this section, we discuss our proposed anonymous system. First, we describe the various entities in our system and our assumptions.

6.1 User (participant)

Users have their own accounts in the system (which we call SP in this paper) and identity provider (OP in this paper) and can use the system anonymously. A user can submit his/her answer only once to the system through the web. No one, including managers of the system, can trace the user from the answer submitted. Users can save and rewrite their answers at any time until the deadline. They can also opt not to submit any answers, and begin participating in the middle of the process. For example, in the lecture questionnaire at a university students can register for the lecture in the middle of the term.

6.2 Service provider (SP)

The SP provides the web service system for users anonymously. For example, for the lecture questionnaire, the SP provides questions for users and request and collates the answers. The SP needs to make the same number of user accounts as number of users. However, the SP cannot connect an account to a real user. For example, the SP cannot know who uses account "id00001" but the user who has account "id00001" can use the system anonymously. Any user who has an account on the SP can save answers in the middle of writing up to the deadline. Further, we assume that the SP cannot change the answers submitted by users, increase or decrease the number of answers, or cut and ignore answers. Only a malicious SP would be able to carry out such actions. To prevent unfair actions we can use existing security technology such as electronic sign or blind signatures [8] and checking via a trusted third party. (However, such details are beyond the scope of this paper.) In addition, the content of the system, such as multiple-choice questions or free descriptive, is independent of our method. Further, in this method, the SP also acts as the RP in the OpenID protocol (Fig. 1). Specifically, the SP obtains identity information with credentials from other servers and authenticate users.

6.3 Relying party (RP)

We use the same term "RP" as in the OpenID glossary. Our RP is used as an account relay server to realize anonymity. Users do not need to be aware of these RPs. RPs connect SP accounts to OP accounts anonymously. RPs are different from SPs and OPs and act as trusted third parties. Multiple RPs are needed, which we denote RPi (i = 1, 2, …, M-1, M).

6.4 OpenID provider (OP)

OP is used as identity providers to provide user authentication results and user IDs to RPs. We use the same term "OP" as in the OpenID glossary. First, the user accesses the OP and inputs his/her ID/password. Op is also different from SPs and RPs and act as trusted third parties. An OP knows which OP account is used by whom. For example, an OP may know that Alice uses the account with ID "alice_0123." However, the OP cannot know the answer Alice submitted to the SP. When the SP needs N users the OP has to have the same N number of accounts. The OP can provide the identity of users with multiple SPs, in which case it may have more accounts than one SP has.

6.5 Preliminary preparation in the proposed method

In the proposed method, SPs, Ops, and all RPs need to make accounts for users. We assume that there are N users and M RPs. Thus, the OP makes N accounts, X_1, X_2, …, X_N. The SP also makes N accounts, U_1, U_2, …, U_N. RP$_i$ also makes N accounts, a_1^i, a_2^i, …, a_N^i.

The OP accounts, X_1, X_2, …, X_N can be connected to real users. For example, the OP knows that the user who has account X_i is Mr. Smith. These are not necessarily anonymous.

After making the user accounts, the OP notifies RP$_1$ about them. RP$_1$ gets the account information from the OP and links accounts {X_1, X_2, …, X_N} of OP and accounts {a_1^1, a_2^1, …, a_N^1} of

RP$_1$. These connections must be kept secret.

RP$_1$ then notifies RP$_2$ about accounts $\{a_1^1, a_2^1, \ldots, a_N^1\}$ and RP$_2$ links RP$_1$ accounts with accounts of RP$_2$, $\{a_1^2, a_2^2, \ldots, a_N^2\}$. Next, the RPs carry out the same actions. Fig. 2 shows the account linkage. To simplify the explanation, "straight" links are used in the figure, but all RPs and SPs actually make random linkages freely, such as $[a_7^1 - a_3^2]$.

As a result, all accounts, $\{X_1, X_2, \ldots, X_N\}$, $\{U_1, U_2, \ldots, U_N\}$, and $\{a_1^i, a_2^i, \ldots, a_N^i\}_{i=1\ldots M}$ are connected to each other, as shown in Fig. 2. However, these connections are known only by servers that have made connections.

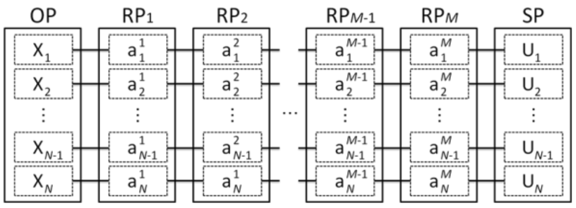

Fig. 2. Account linkage.

The OP can connect account X_i to a real user but through the accounts in the RPs SP cannot connect account U_i to a real user. Further, the RPs in the middle also cannot know who is using account a_i^j. The OP also cannot know who is using the final account U_i. In order to trace a real user from U_i, the RPs, OP, and SP have to conspire together and provide linkage information.

6.6 Basic actions

The basic actions in this proposed system are as follows. A user accesses the SP to use the anonymous service via a browser. The user presses the "START" button on the top page of the SP that then redirects the user to the RP$_M$ with an OpenID authentication request. Next, RP gets the authentication request and remakes it as an OpenID authentication request by the RP and sends it to the next RP. The other RPs carry out the same action. Finally, the user is taken to the OP. This action follows sequence No. 4 of the OpenID protocol in Fig. 1. These actions by the RP are carried out automatically. The user need not to be aware of these RP actions. Thus, after the user presses the "START" button on SP, the user looks at the top page of the OP.

The OP shows the login page to the user. For example, the OP may request ID/password from the user on the login page. For example, in the university lecture questionnaire example, the student affairs office manages the OP and it knows whether the user has registered for the lecture. However, the actual system that carries out the lecture questionnaire is SP, and the OP cannot know the result. The OP only manages the user's identity.

After the OP authenticates the user it provides authentication result of the user with RP$_1$ according to OpenID protocol sequence (sequence No. 6 in Fig. 1). The user is redirected to RP$_1$ with this authentication response automatically. An OpenID authentication response includes the identity of the user, timestamp, sequence number and signature of the authenticator and so on. RP$_1$ gets and confirms the OpenID authentication response from the OP and find the account of the user on RP$_1$ connected to the account of the user on OP. In Fig. 2, when RP$_1$ receives the authentication result with the account of X_1 on OP and find account a_1^1 which is bound to X_1 in RP$_1$.

Next, RP$_1$ reforms the authentication result with the account of a_1^1 on RP$_1$ as RP$_1$'s OpenID authentication response and sends it to the next RP, namely, RP$_2$. In fact, the user is taken back to RP$_2$ automatically. The other RPs all carry out the same action. Each RP reforms the authentication result with the account of the user on the RP as the RP's OpenID authentication response and sends it to the next RP. Finally, the SP gets the authentication response from RP$_M$ and finds the user account in $\{U_1, U_2, \ldots, U_N\}$. The account identity of the user changes through all the RPs and the SP cannot know who uses U_i. This anonymous system is similar to mix-net. To trace the user all RPs and OP have to reveal all connections of all users' accounts. If only one RP conceal the linkage of an account it cannot be traced.

Fig. 3 shows sequences of basic actions. In the figure, sequence (1) shows that the user accesses the SP, presses the "START" button, and is redirected to RP$_M$ with an authentication request. Sequence (2) shows that the RP in the middle relays the authentication request. Sequence (3) shows that the OP authenticates the user. Sequence (4) shows that RPs relay the OpenID authentication response. Finally, in sequence (5), the SP gets the authentication response of the user and identifies the user on the SP's account.

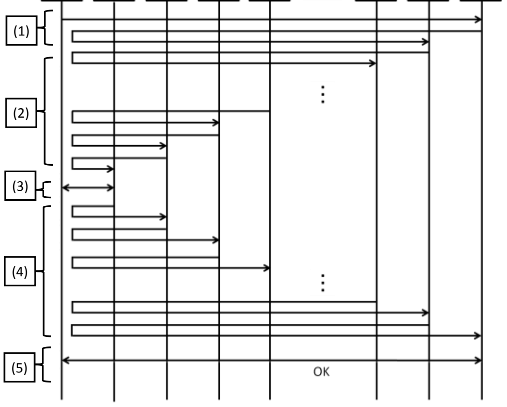

Figure 3. Sequences of basic actions.

SP manages user accounts $\{X_1, X_2, \ldots, X_N\}$ but cannot know the linkage with the OP's accounts, $\{U_1, U_2, \ldots, U_N\}$. A user can use the account U_i on SP and then he/she can save the answer on the account until the deadline. The SP can tally the answer submitted by $\{U_1, U_2, \ldots, U_N\}$ easily and quickly.

6.7 Re-anonymizing all users

In this section, we describe how all users, including participators in the middle, are made anonymous again. In order to make all users anonymous again all RPs and SP shuffle the linkage with all new and old accounts.

In Fig. 4, there are N users at first and L additional users come. The SP or OP announces this addition for all RPs, and OP and all RPs and SP add L accounts but have not yet made any linkage each other.

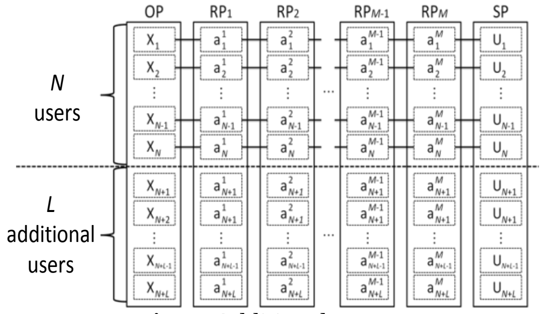

Fig. 4. Additional users.

Then, the OP and all RPs need to tell the next RP or SP about the new accounts made. First, the OP tells the next RP about the accounts, which then informs the next RP in line. All RPs or SP get information about new accounts from the previous RP and make linkages with the new accounts and shuffles them with all its accounts. Fig. 5 shows the shuffle action of RPs and makes new linkage between all accounts of OP and RPs and SP. As shown in Fig. 5, all servers mix the linkages of accounts with both new and old ones. This is a feature of our proposed method; after new participants enter the system, all linkages are mixed with both new and old ones. This helps the re-anonymization process.

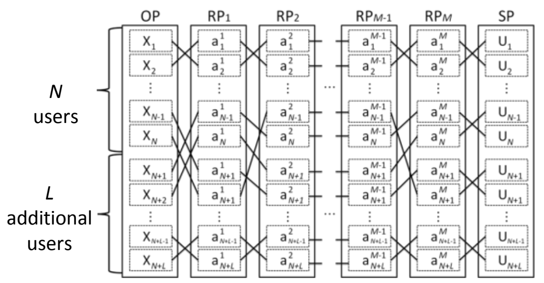

Fig. 5. Shuffle linkages of accounts.

However, the linkages can only be changed after the deadline because the user who uses account U_i is another person after these changes. Thus, we have to assume that our system needs not carry over any information about each account after the deadline. For example, in the lecture questionnaire scenario, users need to save and rewrite their answers but after the deadline users cannot change their answer. Further, the system does not need to save it on each user's account after the system gathers them and tallies them. Furthermore, the system manager can erase it before it starts the questionnaire for the next lecture. Thus, we have to add one condition to the system. We assume the following.

- In this system, no information on account is carried over after the deadline. Thus, after the deadline a user cannot see and change his/her own past answer.

By changing the linkage, all users, including participants in the middle, are made anonymous again. No one can guess who the newcomer is. For example, U_{N+1} is a new account but we cannot determine whether the user who uses account U_{N+1} is new or old. All linkage with all accounts, including both new and old, are shuffled on RPs and anyone cannot trace them. New account U_{N+1} may be used by an old user at that time.

To facilitate easy understanding, we describe a very simple example in Fig. 6. In this figure two users, Suzuki@OP and Sato@OP, are on OP. SP knows that there are two accounts on RP_1 (qwer@RP_1 and asdf@RP_1) but cannot know which is Suzuki@OP or Sato@OP. In this situation, newcomer Tanaka@SP participates in the system in the middle. Then, it is necessary that no one can guess which account is a newcomer.

Now all RPs make a new account and shuffles the linkages between both old and new accounts. Fig. 7 shows the linkages after shuffles. The SP gets two accounts, qwer@RP_1 and asdf@RP_1, which the SP previously knew and gets a new unknown account, zxcv@RP_1. After shuffles, no one can determine who uses the new zxcv@RP_1. After shuffles the new account may be used by old users Suzuki@OP or Satop@OP or may be used by a new user Tanaka@OP.

VII. ADVANTAGES AND SECURITY ANALYSIS

In this section, we discuss the advantages and security of our method.

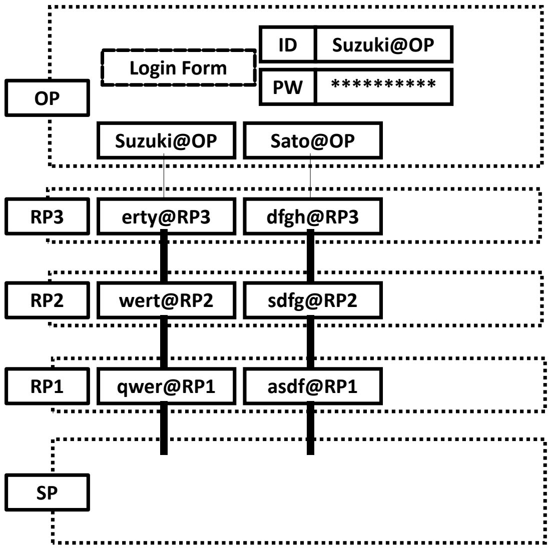

Fig. 6. Simple example.

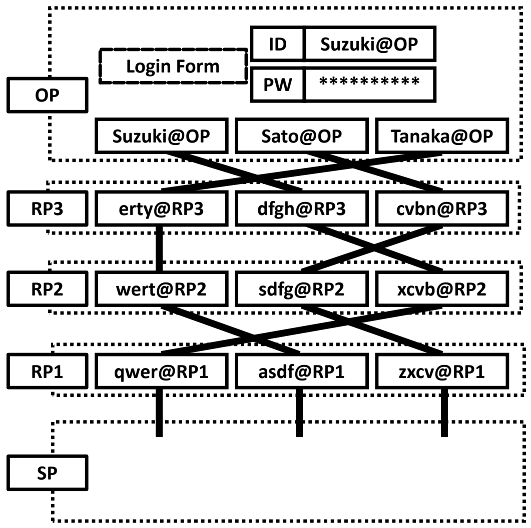

Fig. 7. After shuffles.

7.1 Unfair answer

Unfair answer refers to the submission of an answer more than twice or submission by an unauthorized person. However, this anonymous system manages all the users by accounts. The OP manages each account, authenticates users, and checks the right of users to submit. Further, no one without an account on the OP can login on the OP because they cannot be authenticated by the OP. Each account is allowed to submit an answer once. Consequently, no one can send answers more than twice.

7.2 Anonymity

No one can trace the user unless all RPs and the OP provide all linkage information associated with the identity of the users' accounts. If only one RP make the linkages of accounts on the RP secret, the trace line expires there. The more RPs there are, the more secure the system are. This logic is also the same as that used by mix-net.

When new participants arrive, all users are made anonymous again. Thus, we also cannot trace the newcomer. This point is a feature of our method.

Anonymity at the network layer needs to be carried out by a different technology. We assume that this system is used through the web and then we can trace the user by the IP address which the user had when he/she submitted. To solve this problem, we can use an anonymous channel such as onion router or mix-net. When we use these anonymous channels, we have to select the servers for the anonymous channel, get all the keys of those servers and encrypt the message with those keys. This is a burdensome process.

As already stated, we can use "negative anonymity." In the university, students can use DHCP and many students utilize the network at the same time. Thus, no one can know a user only by watching the IP address.

7.3 User management

SP can manage each user by the associated account. Thus, SP can confirm which account has not submitted the answer yet but SP cannot know who uses the account.

On each account, the associated user can write and save and rewrite his/her answer. Users can carry out these actions until the deadline. However, after the deadline a user cannot see and change his/her answer because when the re-anonymizing shuffle is done the account which he/she used are used by the other user.

7.4 Security on the network

In this method, we assume that an SSO protocol such as OpenID s used. OpenID assumes that SSL is used for all web access. Thus, nobody except a client and a server (SP, OP, or RP) can see the content of The HTTPS transmission. Further, the OpenID protocol adds the response with electronic signature. Thus, no one can change it on the way. Moreover, the OpenID protocol also can also add a timestamp and sequence number and then replay attach cannot work.

7.5 Efficiency

All users on all servers, both old and new, can be made anonymous again by remaking all the accounts of all users instead of our method. However, to remake all the accounts of all the users, all servers have to participate. For example, if RP_1 remakes all the accounts of all RP_1's users and provide information of new accounts to RP_2, then RP2 has to remake new RP's account or make new linkages according to the new accounts of RP_1. If RP_2 also remakes all user accounts on RP_2 then next RP_3 has to do so.

However, in our method, some SPs can skip the re-anonymizing action. In particular, even when no participants enter in the middle, for the sake of more security it is desired to re-anonymize after every deadline. By re-anonymizing every term, the possibility that the user is traced is reduced. At that time, in our method it is sufficient that only some of the RPs exchange the linkage of the accounts. Some RPs skip any actions during re-anonymization. All users use different accounts after change of linkage on some of RPs. So in view of the amount of work, our system is significant.

In our method, almost all web actions use OpenID. We can use a free module for our system and it is economical and efficient. Thus, we need little money to create the system.

7.6 Future work

In our method, multiple RPs is needed for secure anonymity. The more RPs there are, the more secure our system is. Users are not conscious of the RPs because each RP redirects the user to the next RP or OP automatically, and the user does not see the screen of the RP. Users look at a screen with only the SP and the OP. However, because users need to go through all RPs, much time is required when we use many RPs. In a simple system, one or two RPs is sufficient. With less than three RPs, the access time between SP and OP through all RPs is short.

This paper is primarily theoretical; however, we conducted a simple examination. We used XAMPP [9] (Apache 2.4.4 and PHP 5.4.19) for the web server (SP, OP, and RPs) and PHP OpenID Library [10] for the SSO engine and a laptop computer (Windows 7 32-bit, Intel Core 2.4 GHz CPU, 4 GB RAM) for server (OP, SP, and RPs) and client machines. We used 10 testers and three RPs. Users completed the system in this examination in three seconds in every action. However, this test is very simple; more tests are required with more RPs and testers over a longer time period in future work.

Further, in this paper we used OpenID as the SSO protocol but we may be able to use SAML [11] instead.

VIII. CONCLUSION

In this paper, we proposed a new anonymous system using the SSO protocol OpenID. In the proposed method, both users who participate from the beginning and in the middle are made anonymous by shuffling all linkages of all accounts created. This method can be used not only for questionnaires but also for simple e-voting, whistle-blowing, anonymous BBS, and so on.

REFERENCES

[1] Fujioka, T. Okamoto, and K. Ohta, A practical secret voting scheme for large scale elections, Advances in Cryptology - Auscrypt 1992, Lecture Notes in Computer Science, 718, 1992, 244-251.
[2] What is Proxy, http:/ /whatis. techtarget. com/definition/proxy-server
[3] D. Chaum, Untraceable electronic mail, return address, and digital pseudonyms, Communications of the ACM, 24(2), 1981, 84-88.
[4] M. Abe, Universally verifiable mix-net with verification work independent of the number of mix-servers, Advances in Cryptology - Eurocrypt 1998, Lecture Notes in Computer Science, 1403, 1998, 437–447.
[5] M. Abe, Mix-networks on permutation networks, Advances in Cryptology - Asiacrypt 1999, Lecture Notes in Computer Science, 1716, 1999, 258–273.
[6] P.F. Syverson, D.M. Goldschlag, and M.G. Reed, Anonymous connections and onion routing, IEEE Journal on Selected Areas in Communications, 16(4), 1998, 482-494.
[7] OpenID Foundation, http://openid.net/.
[8] D. Chaum, Blind signatures for untraceable payments, Advances in Cryptology - Crypto 1982, Plenum Press, 1983, 199-203.
[9] XAMPP, https:/ /www. Apachefriends .org/ jp/ index.html
[10] PHP OpenID Library, http:/ /www. openidenabled.com/php-openid
[11] SAML, https: // www.oasis-open.org.

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