SCIS 2011 The 2011 Symposium on Cryptography and Information Security
Kokura, Japan, Jan. 25-28, 2011
The Institute of Electronics, Information and Communication Engineers

A Simplified Private Stable Matching Algorithm

Hermanto * Yoshifumi Manabe † Tatsuaki Okamoto †

Abstract— Stable matching algorithms are best known for their use in assigning graduating medical students to their first hospital appointments. They are also widely used in the matching of groups of men and women, employers and companies. At CT-RSA 2007, Matthew Franklin, Mark Gondree, and Payman Mohassel presented an improved protocol of Golle's private stable matching algorithm for the privacy preserving computation of Gale-Shapley stable matching algorithm.

In this paper, we prove that when the number of *matching authorities* (MAs) is more than two, the number of rounds of computation required for stable matching can be reduced to half, which is nearly the same number of rounds needed for the Gale-Shapley stable matching algorithm.

Keywords: Stable matching, privacy-preserving protocols, secure multiparty computation, passive adversaries.

1 Introduction

Stable matching algorithms are most commonly explained using the state of bijection between two groups of people, regardless of whether the groups are of men and women, employers, companies, or medical school graduates being appointed to hospitals. In this paper, we consider one-to-one matching algorithm to solve a model of one-to-one matchings. Throughout this paper, we denote men as one group and women as the other group.

Consider that there are n men and n women who try to find their best matches. It would be perfect for each person in both groups to find naturally their best partner without having any conflict with the other people, however, this is a very rare case. There is a probability that two men or more will rank the same woman as their best mate and vice versa.

David Gale and Lloyd Shapley [GS62] presented a novel framework for solving the stable matching problem. The algorithm itself is very simple, by using the lists of preference from men and women to find a stable matching result. However, in the Gale-Shapley stable matching algorithm [GS62], the history of the engagements during the execution of the algorithm is known to everyone, which negates the privacy of each rejected person and leaves the history open to all participants. This leads that the stable matching algorithm is vulnerable to manipulation [GI89]. Under certain circumstances, participants with the knowledge of the preference lists of other participants have incentives to change their own true preference list. In order to output a stable matching result without revealing any information, we need a secure protocol that shows only the matching result to all participants, which is called a private stable matching algorithm. By adopting the Gale-Shapley algorithm [GS62], Golle [Go06] and Matthew Franklin, Mark Gondree, and Payman Mohassel [FGM07] presented their own variant of a private stable matching algorithm. Both of these protocols use the Paillier encryption scheme [Pa99] and re-encryption mix networks as their cryptographic building blocks. The main difference between these two protocols is the number of fake participants introduced in the protocols. We will discuss these protocols in Section 2.

In our study, we note that Matthew Franklin, Mark Gondree, and Payman Mohassel protocol [FGM07] can be simplified in some way so that the number of rounds of computation for stable matching is reduced by half, which is almost the same as the number of rounds needed for the original Gale-Shapley stable matching algorithm [GS62].

2 Stable Matching Algorithms

In this section, we explain briefly the Gale-Shapley algorithm [GS62], and review the private stable matching algorithms presented by Golle [Go06] and Matthew Franklin, Mark Gondree, and Payman Mohassel [FGM07]. All of these stable matching algorithms use preference lists (every man ranks every woman, and every woman ranks every man) as input and give a stable matching result (there is always one, and there may be several).

Let $A_1,\ldots,A_n$ denote n men and $B_1,\ldots,B_n$ denote n women. The men rank the women from most to least desired, and vice versa. If man A_1 has a preference list $(B_1, B_2, \ldots, B_3)$, it means that man A_1 likes woman B_1 the most, while B_3 is the least favorite woman. A matching is called *stable*, when there is no unmatched man and woman that like each other better than their own current match.

* Department of Social Informatics, Graduate School of Informatics, Kyoto Univ.
† NTT Labs, Nippon Telegraph and Telephone Corporation.

2.1 The Gale-Shapley Algorithm

The Gale-Shapley algorithm [GS62] was originally known for solving the stable matching problem. In this algorithm, men and women play different roles. Simply, the algorithm can be expressed as "proposals" from men to women.

Men and women are divided into two groups, the *engaged* group and the $free$ group. All men and women start from the free group. Whenever there is any man in the free group, one of them is randomly selected. Then, the selected free man proposes to the woman whom he likes the most and has never proposed to before. Suppose the selected free man A proposes to B (whom he never proposed before and he likes the most). In this state there are two cases. The first case is when the woman B is free. In this case, A and B are automatically paired. The second case is when the woman B is already engaged to another man A'. In this case, using the woman B's preference list, the man who proposed to her A and the man who is currently engaged to her A' are compared. When A has a higher rank than A', B engages with A, while A' is divorced and grouped into the free group. On the other hand, if A has a lower rank than A', B will continue to be engaged to A', while A is considered to be rejected and regrouped again into the free group. Next, another man from the free group is randomly chosen and matched by the same way. This manner of matching repeats until all men and women are paired and no person remains in the free group. The total round of matching is at most $n^2 - n + 1$ [GI89].

2.2 Golle's Private Stable Matching Algorithm

In order to retain the privacy of the participants' information, this algorithm added independent parties called Matching Authorities (MAs). By setting the *MAs* as *honest but curious* parties, participants obtain the stable matching result without knowing any other information. Fake men are added to the other real men and women in the matching algorithm. The total communication complexity of this private stable matching is $O(n^3)$.

2.3 Improved Efficiency for Golle's Private Stable Matching

Matthew Franklin, Mark Gondree, and Payman Mohassel showed that communication complexity of Golle's [Go06] main protocol is $O(n^5)$. In addition, they also introduced their own variant protocol in which the communication complexity is reduced. As participants, fake women are added to Golle's private stable matching algorithm [Go06] (real men, real women, and fake men). The computation complexity of their private stable matching algorithm is $O(n^4\sqrt{\log n})$, while the rounds of computation are $2n^2$.

3 Preliminaries

3.1 Models and Definition

We adopt the same network model as Matthew Franklin, Mark Gondree, and Payman Mohassel [FGM07]. Once the protocol starts, all participants send their encrypted preference lists to one of the matching authority referred to DBMA, the role of which is to save the encrypted preference lists. All MAs other than the DBMA execute a synchronous protocol among themselves to compute a stable matching.

On the other hand, our security model is the same as that used by Golle [Go06]. A stable matching protocol is *secure* if it outputs a stable matching and reveals no other information to a passive adversary than what the adversary can learn from the matching and from the preferences of the participants the adversary controls.

Encryption

We let E denote the encryption function for a threshold public-key encryption scheme that is additively homomorphic, which is a threshold version [DJ01] [FPS01] of the Paillier encryption scheme [Pa99]. Only when a quorum of all MAs is reached, the decryption can be executed.

Notation

Term $O(f)$ denotes the asymptotic upper bound of f that is not tight; $\omega(f)$ denotes the asymptotic lower bound of f that is tight. In Section 3.2 below, unless noted, "poly-log complexity" is in reference to the security parameter for each primitive.

3.2 Primitives

Re-encryption Mix Network

In this paper, when the authorities *mix* some ciphertexts (Paillier), it means that the authorities run a re-encryption mix network [Ne01] [JJR02], permuting the ciphertexts according to a secret permutation such that no individual authority knows. As we take a passive adversary, n ciphertexts can be mixed by t mixing authorities in constant rounds and $O(n)$ time, taking advantage of parallel mixing techniques [GJ04]. The total communication complexity of the parallel mixnet is, $O(tn)$ ciphertexts.

Private Oblivious Equality Test

Let $E(m_1)$ and $E(m_2)$ be two Paillier ciphertexts. Define $\text{EQTEST}(E(m_1), E(m_2)) = b$ where $b = 1$ if $m_1 = m_2$ and $b = 0$ otherwise. EQTEST is a (chooser private) oblivious test of plaintext equality [JS99] [Li03] if it reveals the output to the MAs, without revealing any other information to any other parties.

MPC Private Equality Test

Let $E(m_1)$ and $E(m_2)$ be two Paillier ciphertexts. Define $\text{EEQTEST}(E(m_1), E(m_2)) = E(b)$ where $b = 1$ if $m_1 = m_2$ and $b = 0$ otherwise. EEQTEST is the

secure multiparty computation of the equality test if the parties learn the output $E(b)$, but no additional information regarding the plaintexts m_1 and m_2. [Kl05] [CD01] both give constant-round protocols with poly-log communication complexity for computing this function.

Private Oblivious Value Comparison

Let $E(m_1)$ and $E(m_2)$ be two Paillier ciphertexts. Define COMPARE$(E(m_1), E(m_2)) = b$ where $b = 1$ if $m_1 < m_2$ and $b = 0$ otherwise. For our purposes, we have $0 \leq m_1$ and $m_2 \leq n$. Golle instantiates this primitive by preparing $n-1$ ciphertexts, $D_1, \ldots, D_{n-1}$ where $D_i = E(m_1 - m_2 - i)$; mixing these n ciphertexts; and finally running n parallel instances of EQTEST$(E_i, E(0))$. If $m_1 < m_2$ then, for some $0 < i \leq n$, one of these instances returns 1. Otherwise, all instances return 0.

MPC Private Value Comparison

Let $E(m_1)$ and $E(m_2)$ be two Paillier ciphertexts. Define ECOMPARE$(E(m_1), E(m_2)) = E(b)$ where $b = 1$ if $m_1 < m_2$ and $b = 0$ otherwise. ECOMPARE is the secure multiparty computation of the less-than function if parties learn the output $E(b)$, but no additional information regarding the plaintexts m_1 and m_2. [Kl05] [DFNT05] both give constant-round protocols with poly-log communication complexity for computing this function.

Private Reduction of a Secret Modulo a Public Integer

Let $E(a)$ be a Paillier ciphertext, and q be an integer. Define MOD$(E(a), q) = E(a \mod q)$. MOD is the secure multiparty computation of the modular function if our parties learn the output, but no additional information regarding integer a. [Kl05], [ACS02] both give protocols with poly-log communication complexity for computing this function. The former has a poly-log round complexity, and the latter is constant-round.

Strong Private Information Retrieval (SPIR) with Sublinear Communication Complexity

Let δ be a database with N elements, and indexed $\{0, \ldots, N-1\}$. Let $SPIR_m^\delta$ $(b_1, \ldots, b_l)$ represent Stern's symmetric private information retrieval protocol [St98]. As in any PIR protocol. a chooser holds a secret index i, while the database learns nothing about which index was accessed. Furthermore, the chooser knows nothing regarding the other database elements.

In $SPIR_m^\delta$, index i is encoded by following a trick, developed by Kushilevitz and Ostrovsky [KO97]. The database is described as a series of m sized buckets (the first m entry is in the first bucket, and so on). If element i is the jth element in one of these buckets, then $b_{1,j} = E(1)$ and $b_{1,k} = E(0)$ for all k $\neq$ j. Define $b_1 = (b_{1,1}, \ldots, b_{1,m})$. We recurse, imagining the collection of former buckets as, themselves, a series of m sized buckets. The output of the protocol must be decrypted by the chooser l times, to recover the element at index i. Since we consider passive adversaries, we do not include Stern's interactive zero-knowledge proofs showing the indices are well-formed as a part of $SPIR_m^\delta$. With $m = N^{1/l}$ and $l = O(\sqrt{\log N})$, the protocol has a total communication complexity $l = O(N\sqrt{\log N})$ and the total communication complexity $2^{O(\sqrt{\log N}}$.

4 New Private Stable Matching Algorithm

4.1 Franklin-Gondree-Mohassel Algorithm

In this algorithm, the notations for A means men, for B means women, and real groups are represented by 1,..., n. Fake groups are represented by $n+1$,..., $2n$. Thus, we have: real men(A_1,..., A_n), real women(B_1,..., B_n), fake men(A_{n+1},...,A_{2n}), and fake women(B_{n+1},..., B_{2n}). As the initial setup, preference lists are generated as below.

	Preference lists (For $x \geq 1$)
A_x	(Real preference list),(B_{n+1},..., B_{2n} in any order)
B_x	(Real preference list),(A_{n+1},..., A_{2n} in any order)
A_{n+x}	(B_{n+2},..., B_{2n} in any order), (B_{n+1}),(B_1,..., B_n in any order)
B_{n+x}	(A_{n+1},..., A_{2n} in any order), (A_1,..., A_n in any order)

Algorithm

Initialization :
1. $\mathbb{F}_1 = \{A_1\}$ (man A_1 is free).
2. Real men $\{A_2, \ldots, A_n\}$ are engaged to fake women $\{B_{n+2}, \ldots, B_{2n}\}$, respectively.
3. Fake men $\{A_{n+1} \ldots, A_{2n}\}$ are engaged to fake women $\{B_1, \ldots, B_n\}$, respectively.

For k=1 to R:
a. Free man A_x in $\mathbb{F}_k$ proposes to B_y, the next woman in his preference list to whom he has not yet proposed.
b. Let A'_x denote the man to whom B_y is already engaged.
Case 1. If B_y ranks A_x higher than A'_x, she leaves A'_x and become engaged to A_x. Let $\mathbb{F}_{k+1} = \{A'_x\}$.
Case 2. If B_y ranks A_x lower than A'_x, she stays engaged to A'_x. Let $\mathbb{F}_{k+1} = \{A_x\}$.

In each round of matching, only one proposal is made. Therefore, the number of free men in each round $\mid \mathbb{F}_k \mid$ is 1. In [FGM07], it is claimed that once a fake man proposes to fake woman B_{n+1}, it means that a stable matching is reached. The algorithm reaches stable matching in at most $2n^2$ rounds where all real men are engaged to real women and all fake men to fake women. However, there are too many rounds where most of the proposals are made by fake men. Such proposals increase the number of rounds of matching.

4.2 Our Algorithm

As described in the Matthew Franklin, Mark Gondree, and Payman Mohassel algorithm [FGM07], there are

n real men, n real women, n fake men, and n fake women with their respective preference lists. In our algorithm, we use the same algorithm as the Matthew Franklin, Mark Gondree, and Payman Mohassel algorithm [FGM07] but a few changes are made in the preference lists. Our preference lists are described in the following table.

	Preference lists *(For $x \geq 1$; $y \geq 2$)*
A_x	(Real preference list), (B_{n+x})
B_x	(Real preference list), (A_{n+x})
A_{n+1}	(B_{n+2},..., B_{2n} in any order),(B_{n+1}), (B_1)
A_{n+y}	(B_{n+y}),(Fake women other than B_{n+y} in any order),(B_y)
B_{n+x}	(A_{n+x}),(Fake men other than A_{n+x} in any order),(A_x)

As you can see, the preferences for the fake groups are set differently, so that fake men and fake women rank the members with the same number as their best mate, except for A_{n+1}. For example, fake man A_{n+2} likes fake woman B_{n+2} the best, and fake woman B_{n+2} likes fake man A_{n+2} the best. Consecutively, fake man A_{n+3} likes fake woman B_{n+3} the best and fake woman B_{n+3} likes fake man A_{n+3} the best, and so on. The size of each preference list is reduced from $2n$ to $n+1$. When each of the real men is engaged to a real woman, and the fake men are engaged to the fake women, the matching is said to be in a stable state. As a comparison, our algorithm reaches a stable matching in at most $n^2 + n$ iterations.

Proposition. Once A_{n+1} proposes to B_{n+1}, it is the last round of matching.

Proof. The first point is that fake man A_{n+1} ranks fake woman B_{n+1} as his last preference among all the fake women while fake woman B_{n+1} ranks fake man A_{n+1} as her first preference. The other point is the preference lists of the fake men and fake women are fixed in a way that the first preference of each person is totally different, except for A_{n+1}. As a result, there is no conflict among fake men. For instance, as fake man A_{n+2} ranks fake woman B_{n+2} as his first preference, fake woman B_{n+2} also ranks fake man A_{n+2} as her first preference. By doing this, when fake men propose to the fake women, only one proposal is needed for fake men to get engaged, moreover, fake men will not be rejected by their first preference, except for fake man A_{n+1}.

The statement that the matching result is stable is exactly the same as that for the original Gale-Shapley algorithm [GS62]. Matching becomes stable when real men are engaged to real women, while fake men are engaged to fake women. Notice that once fake men other than A_{n+1} are engaged to fake women, they will stay engaged until the last round, leaving only real men to find their best mates. Nevertheless, let us assume that real man A_x is engaged to a fake woman B_{n+x} when the algorithm ends. This indicates that fake woman B_{n+x} was never proposed to any fake man. Considering that the number of fake men and fake women are the same, by the end of the matching process, all fake men must have proposed to the fake women including B_{n+x}. Especially in our proposed algorithm, at least fake man A_{n+1} must propose to fake woman B_{n+x} once and must have been rejected. So fake woman B_{n+x} prefers real man A_x to fake man A_{n+1}, which is contrary to the assumption that all fake women rank real men behind the fake men.

In this algorithm, the proposals among real men and real women are at most $n^2 - n + 1$. This is the same as that for the Gale-Shapley stable matching algorithm [GS62] [GI89]. On the other hand, the preference lists of fake men and fake women are fixed as above, resulting in fake man A_{n+1} executing at most n proposals, while the other fake men take only one proposal. Thus, the total number of proposals needed for fake men and fake woman to finish is $2n - 1$. Therefore, during the last round of the proposals, fake man A_{n+1} will be proposing to B_{n+1} as his last preference, as in this turn, A_{n+1} have been rejected by all fake women except for B_{n+1}. This is the reason why, in the $n^2 + n$ rounds of matching, stable matching is reached.

We simply fixed the preference lists of fake men and fake woman in order to reduce the rounds of computation for stable matching. Note that MAs can jointly randomize the best mate for each fake man and woman to hide fake people's preference lists from any single matching authority.

For the secure implementation on the above algorithm, we need to run the algorithm for the same number of rounds for all inputs. By doing this, we will avoid leaking the number of proposals necessary to reach a stable matching for a specific input. But, note that once fake man A_{n+1} proposes to fake woman B_{n+1}, no free man will remain and the algorithm has to end. A simple fix is to add an extra fake man A_{2n+1}, and initially let him be engaged to woman B_{n+1}. We set $((E(2n+1), E(0)), E(n+1), E(0))$ be the engaged bid for fake woman B_{n+1} and fake man A_{2n+1}. Here we set the rank of fake man A_{2n+1} to 0 (as the most favorite man). By doing this, when A_{n+1} proposes to B_{n+1}, B_{n+1} will always prefer fake man A_{2n+1} to fake man A_{n+1}. The advantage is that there is always a free man who will propose next. This is a useful invariant for the secure implementation in Section 5.

4.3 Privacy Preserving Protocol

In this section, we present the privacy preserving implementation of the Gale-Shapley [GS62] variants, which we adopt the same model from Matthew Franklin, Mark Gondree, and Payman Mohassel private stable matching algorithm [FGM07].

Notation and Bids. Let $r_{i,j} \in [0, \ldots, n-1]$ be the rank given to woman B_j by man A_i. Let $s_{i,j} \in [0, \ldots, n-1]$ be the rank given to man A_i by woman B_j. In this convention, the highest possible rank is 0, and the lowest is $n-1$. Define the (free) bid for man A_i as $W_i = (E(i), E(\rho))$, where initially, $\rho = 0$. The *engaged* bid $(W_i, E(j), E(s_{j,i}))$ denotes that man A_i is

engaged to woman B_j. Let $\mathbb{F}_k$ and $\mathbb{E}_k$ denote the sets of free and engaged bids in round k of the algorithm, respectively.

Input Submission and Initialization. Each man sends his preference list a_i and each woman sends her preference list q_i to the MAs ($a_i = (E(r_{i,1}), \ldots, E(r_{i,n}))$, $q_i = (E(s_{1,i}), \ldots, E(s_{n,i}))$). The MAs generate the free bids for A_1 and the engaged bid for man A_i, for $i \neq 1$. Then the MAs jointly create the preferences for the fake men and fake women. The MAs here set the preference lists of fake man and fake woman according to the rule in Section 4.2. Let one matching authority collect and organize these lists, and call this authority database δ Let δ =[$(a_1, q_1, \ldots, a_{2n}, q_{2n})$]. Thus $\delta[4n(i-1)+(j-1)] = E(r_{i,j})$ and $\delta[4n(i-1)+(j-1)+n+1] = E(s_{j,i})$, for $i, j \leq n+1$.

Open a Bid. Given a free bid, we must recover $E(j)$ and $E(s_{j,i})$ ($E(j)$ refers to the encrypted index of the woman at rank ρ on man A_i's preference list). It happens that $E(j)$ is located at $\delta[4n(i-1)+(\rho-1)]$ and $E(s_{j,i})$ is located at $\delta[4n(i-1)+(j-1)+2n]$. We can calculate $E(4n(i-1)+\rho-1)$ using the Paillier addictive homomorphism, given $E(i)$ and $E(\rho)$). We can recover $E(j)$ by accessing the database at this secret index, using the protocol below. Similarly, given $E(j)$ we can calculate $E(4n(i-1)+(j-1)+2n)$ and, again recover $E(s_{j,i})$ by accessing the database at this secret index.

Access Database at Secret Index. Given $E(x)$, we can generate a series of indices $b_1, \ldots, b_l$ which singulate the element at index x using the index conversion procedure below, without learning anything regarding x. Then let $y = SPIR_m^{\delta}(b_1, \ldots, b_l)$. We jointly decrypt y, l times, to recover$\delta[x]$.

Breaking an Engagement. Let $(W_i, E(j), E(s_{j,i}))$ be an engaged bid. We break this engagement by discarding $E(j), E(s_{j,i})$ and keeping W_i. We also "safely" update $E(\rho)$ by increasing it by the value 1-b, where $E(b)$ = EEQTEST$(E(\rho), E(n))$, using Paillier's addictive homomorphism (i.e. multiply $E(\rho)$ by $E(1)$ to obtain $E(\rho+1)$). That is , we clearly increase the next desired rank ρ when it is less than n ; otherwise, we do not. This is a modification from the presentation in [Go06]. If we did not increase safely, the new n^2 loop bound generates the possibility that we may increase some men's ρ more than n times which would lead to an error.

Find a Conflicting Bid. Given a newly created engaged bid $(W_i, E(j), E(s_{j,i}))$, there will be exactly one existing engaged bid that conflicts $(W_{i'}, E(j'), E(s_{j',i'}))$ $\in \mathbb{E}_k$ where $j = j'$. We can find this by preparing set $\{E(j') \mid (W_{i'}, E(j'), E(s_{j',i'})) \in \mathbb{E}_k\}$, mixing the $2n-1$ ciphertexts in this set, and then performing $2n-1$ parallel instances of EQTEST$(E(j), E(j'))$ for each $E(j')$ in the mixed set.

Resolve a Conflict. Given two random conflicting engaged bids, $(W_i, E(j), E(s_{j,i}))$ and $(W_{i'}, E(j), E(s_{j,i'}))$, we determine the "winner" and "loser" of the conflict by performing the following. Jointly compute $b =$ COMPARE $(E(s_{j,i}), E(s_{j,i'}))$. If $b = 1$ then woman j prefers man i' over man i and, we call the first engaged bid the "loser". Otherwise, we call the second engaged bid the "loser"and the remaining bid the "winner".

Secure Index Conversion. Given $E(x)$, we can securely calculate the indices $b_1, \ldots, b_l$ that are used as input to the protocol $SPIR_m^{\delta}$. Recall that b_k =($b_{k,1}$, $\ldots$, $b_{k,m}$) is the encryption of an m-length bit-string of Hamming weight 1, selecting the appropriate item from each m sized bucket at step k. If we consider the buckets to be arranged consecutively (the first m elements in the first bucket, and so on) then $b_{k,j}$=$E(c_{k,j})$ where

$$c_{k,j} = (x \mod m^k \stackrel{?}{=} (j-1)m^{k+1} + \sum_{h=1}^{k-1} \sum_{i=1}^{m} (i-1) c_{h,i} m^{h-1})$$

Thus b_k can be calculated using MOD, EEQTEST and the vectors b_j for $j < k$ calculated in the earlier rounds. In each round, this procedure takes polylog work with polylog communication complexity.

Full Privacy Preserving Implementation. The secure implementation of our new variant of Gale-Shapley [GS62] is assembled using the protocols indicated above, according to the algorithm below.

Input submission and initialization
For k = 1 to $n^2 + n$; perform the following.
1. Select the single free bid W_i from $\mathbb{F}_k$.
2. Open W_i to recover $E(j)$ and $E(s_{j,i})$ from the database.
3. Create engaged bid $(W_i, E(j), E(s_{j,i}))$.
4. Find conflicting engaged bid $(W_{i'}, E(j), E(s_{j,i'}))$.
5. Mix these two engaged bids.
6. Resolve the conflict to find the "winner" and "loser".
7. Break the engagement for the loser and add this free bid to $\mathbb{F}_{k+1}$.
8. Add the winner to $\mathbb{E}_k$.
9. Mix the engaged bids.
10. Let $\mathbb{E}_{k+1} = \mathbb{E}_k$.
MAs jointly decrypt the all engaged bids in $\mathbb{E}_k$ then announce the matching result.

5 Security Definition

Note that neither each participants (men and women) nor matching authority can retrieve any information regarding the preference lists and the history of engagements. This protocol is said to be secure if there is no polynomial time adversary that can break privacy with a non-negligible advantage.

Definition

If the advantage of the adversary $Adv(\tau)$ is negligible then this stable matching algorithm is secure.

$$Adv(\tau) = Pr[b' = b] - 1/2$$

First of all, adversary creates two challenge preference lists as p_{m0}, p_{m1} where p_{m0} and p_{m1} each represent one preference list ($p_{m0} = (\ldots, n_1 \ldots, n_2, \ldots)$, and $p_{m1} = (\ldots, n_2, \ldots, n_1, \ldots)$). We notice that n_1 and n_2 written in above represents ranks of two participants in a person which are flipped. The secure games described above must be done with *condition*, that the flipping of n_1 and n_2 does not affect the result of the stable matching algorithm.

Then the challenger chooses one bit $b \in \{0,1\}$, encrypt p_{mb} and executes stable matching algorithm. Adversary outputs guess of b'. Adversary wins if $b = b'$.

6 Conclusion

We changed the preference lists from the network model presented by Matthew Franklin, Mark Gondree, and Payman Mohassel [FGM07], and reduced the number of rounds of computation from $2n^2$ to $n^2 + n$. In future work, we will try to compose a private stable matching algorithm by using BGN encryption instead of the Paillier encryption.

References

[DJ01] Ivan Damgard and Mads Jurik. A generalization, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*, LNCS Vol. 1992 pp.119-136, 2001.

[Go06] Phillipe Golle. A private stable matching algorithm. *Financial Crypto*, LNCS Vol. 4107, pp.65-80, 2006.

[GS62] David Gale and Lloyd Shapley. College Admissions and the Stability of Marriage. *American Mathematical Monthly*, Vol. 69, pp.:9-15, 1962.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *19th ACM Symposium on Theory of Computing*, pp.218-229, 1987.

[GI89] Dan Gusfield and Robert Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, pp.8-18, 1989.

[FPS01] Pierre-Alain Fouque, G.Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Crypto*, LNCS Vol. 1962, pp.90-104, 2001.

[JJR02] Markus Jakobsson, Ari Juels, and Ron Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of USENIX'02*, pp.339-353, 2002.

[FGM07] Matthew Franklin, Mark Gondree, and Payman Mohassel. Improved efficiency for private stable matching. *The Cryptographers' Track at the RSA Conference*, LNCS Vol 4377, pp.163-177, 2007.

[JS99] Markus Jakobsson and Claus Peter Schnorr. Efficient oblivious proofs of correct exponentiation. In *Communications and Multimedia Security*, pp.71-86, 1999.

[Kl05] Eike Klitz. Unconditionally secure constant round multi-party computation for equality, comparison, bits and exponentiation. Cryptology ePrint Archieve, Report 2005/066, 2005.

[Li03] Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *ASIACRYPT 2003*, LNCS Vol. 2894 pp.416-433, 2003.

[Pa99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, LNCS Vol. 1592 pp.223-238, 1999.

[Ne01] C.Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Computer and Communications Security*, pp116-125, 2001.

[GJ04] Phillippe Golle and Ari Juels. Parallel mixing. In *11th ACM Computer and Communications Security*, pp.220-226, 2004.

[CD01] Ronald Cramer and Ivan Damgard. Secure distributed linear algebra in a constant number of rounds. In *CRYPTO'01*, LNCS Vol. 2139 pp.143-202, 2001.

[DFNT05] Ivan Damgard, Matthias Fitzi, Jesper Buus Nielsen, and Tomas Toft. How to split a shared secret into shared bits in constant-round. Cryptology ePrint Archive, Report 2005/140, 2005.

[ACS02] Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *CRYPTO'02*, LNCS Vol. 2442 pp.417-432, 2002.

[St98] Julien P. Stern. A new efficient all-or-nothing disclosure of secrets protocol. In *ASIACRYPT'98*, LNCS Vol. 1514 pp.357-371, 1998.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Foundations of Computer Science*, pp.364-373, 1997.