A Private Stable Matching Algorithm

Philippe Golle

Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA 94304, USA
pgolle@parc.com

Abstract. Existing stable matching algorithms reveal the preferences of all participants, as well as the history of matches made and broken in the course of computing a stable match. This information leakage not only violates the privacy of participants, but also leaves matching algorithms vulnerable to manipulation [8, 10, 25].
To address these limitations, this paper proposes a *private* stable matching algorithm, based on the famous algorithm of Gale and Shapley [6]. Our private algorithm is run by a number of independent parties whom we call the Matching Authorities. As long as a majority of Matching Authorities are honest, our protocol correctly outputs a stable match, and reveals no other information than what can be learned from that match and from the preferences of participants controlled by the adversary. The security and privacy of our protocol are based on re-encryption mix networks and on an additively homomorphic semantically secure public-key encryption scheme such as Paillier.

1 Introduction

Stable matching algorithms are best explained with the terminology of marriage and are thus also known as stable marriage algorithms. Let us consider an equal number n of men and women. We assume that every man ranks the n women according to how desirable each is to him, without ties. Similarly, every woman ranks the n men according to how desirable each is to her, without ties.

A match is a bijection between men and women, or equivalently a set of n heterosexual monogamous marriages between the n men and the n women. Ideally, a perfect match would pair every man with the woman he likes best and vice versa. Clearly the preferences expressed by men and women rarely allow for a perfect match. For example, if two men rank the same woman first, one of them at least will have to settle for a less desirable partner.

A weaker requirement is to find a match that is, if not perfect, then at least *stable*. Consider a match in which a man A is married to a woman B and a man A' to a woman B'. If A prefers B' to his wife B, and B' prefers A to her husband A', A and B' both have an incentive to leave their partner and marry each other (the match is thus unstable). A *stable* match is a match such that there is no man and woman that both like each other better than their respective partners. When a match is stable, all couples are static: a man tempted to abandon his

wife for another woman he ranks higher will be rebuffed, since that woman ranks her partner higher than the new suitor.

A stable matching algorithm takes as inputs the preferences of men for women and of women for men, and outputs a stable matching between them. Efficient stable matching algorithms are well known [6] and have important real-world applications. They are used notably to assign graduating medical students to residency programs at hospitals in the US [15], Canada [3] and Scotland [23]: a stable match between students and hospitals is computed automatically based on the preferences of students for hospitals and of hospitals for students. In Norway and Singapore [25], stable matching algorithms are used to assign students to schools and universities. In fact, the use of stable matching algorithms is sufficiently widespread that there exist companies dedicated solely to administering matching programs. National Matching Services [14], for example, administers matching programs for psychology internships in the US and Canada [1], for articling positions with law firms in Alberta (Canada) and for many others. All told, stable matching algorithms impact the careers of tens of thousands of students and professionals annually.

Considering the stakes involved and the sensitive nature of the preferences expressed, stable matching algorithms should afford participants the maximum amount of privacy possible. Ideally, the algorithm should output a stable match without leaking any additional information. Unfortunately, existing stable matching algorithms fall far short of that ideal. They take as input the complete list of preferences of men and women (or students and hospitals), and reveal the complete history of engagements made and broken in the process of computing a stable match.

This information leakage violates the privacy of participants, and potentially exposes them to embarrassment or ridicule. Consider for example that no medical student would like it to be known that she was matched to her least favorite hospital. Worse still, the public disclosure of preferences leaves matching algorithms vulnerable to manipulation [8, 10, 25]: under certain circumstances, participants with knowledge of the preferences of other participants have incentives to alter their own true preference list (see Section 2 for detail).

In the absence of a better solution, these problems have up to now been weakly mitigated by the assumption that all participants trust a third party to receive their preference lists, run the algorithm and output a stable match without revealing any other information. This approach requires a considerable amount of trust in a single entity and runs counter to the security tenet that trust should be distributed. Reliance on a single trusted third party is particularly problematic when the third party must protect the interests of participants with unequal power and influence. The third party is at risk of corruption by the more powerful protocol participants (e.g. the hospitals) at the expense of the less powerful participants (e.g. the medical students). To afford equal privacy protection to all participants, we propose a private stable matching algorithm.

Our private stable matching algorithm is based on the famous algorithm of Gale and Shapley [6], which we review in Section 2. The private algorithm is run

by a number of independent parties whom we call the Matching Authorities. As long as a majority of Matching Authorities are honest, our protocol correctly outputs a stable match, and reveals no other information than what can be learned from that match and from the preferences of participants controlled by the adversary. The security and privacy of our protocol are based on re-encryption mix networks and on an additively homomorphic semantically secure public-key encryption scheme such as Paillier.

1.1 Financial Applications of Stable Matching Algorithms

The stable matching problem describes a two-sided, one-to-one matching market. The most famous examples of such markets are college admissions and entry-level labor markets. But examples of two-sided markets go far beyond labor markets [20, 21]. Among others examples, two-sided markets have been used to model the matching of venture capitalists and companies in capital markets [24] and the matching of suppliers and consumers in supply chain networks [17]. Aside from matching markets, the use of stable matching algorithms has recently been proposed to determine stable winner allocations in certain types of multi-unit or combinatorial auctions [2].

2 Gale-Shapley Stable Matching Algorithm

There exist several formulations of the stable matching problem, all closely related. In this section and the rest of the paper, we consider a model of one-to-one matchings (i.e. no polygamy), with complete preference lists (i.e every man ranks all women and every woman ranks all men). The results of this paper can easily be adapted to other models. For example, the many-to-one model (in which one hospital has internship slots for multiple students) reduces to the one-to-one model by cloning an appropriate number of times the participants who accept multiple partners.

We review the famous stable matching algorithm of Gale and Shapley [6]. In this algorithm, men and women play different roles. Arbitrarily, we present a matching algorithm in which men propose to women (these roles can naturally be reversed). The algorithm takes as input the lists of preferences of men and women. Throughout the algorithm, men and women are divided into two groups: those that are *engaged*, and those that are *free* (i.e. not yet or no longer engaged). Initially, all men and all women are free.

As long as the group of free men is non-empty, the algorithm selects at random one man A from the group of free men. Man A proposes to the woman whom he ranks highest among the women to whom he has never proposed before (let's call this woman B). One of three things may happen:

- B is free. In this case, A and B are engaged to each other and both move to the engaged group.

– B is already engaged to A' and ranks A ahead of A'. In this case, B breaks her engagement to A' and instead gets engaged to A. A and B join the engaged group, whereas A' goes back to the group of free men.
– B is already engaged to A' and ranks A' ahead of A. In this case, B stays engaged to A' and A stays in the group of free men.

Properties and limitations. Let n denote the number of men and women. The algorithm terminates in at most n^2 steps and outputs a match that is stable (see [6] for more detail). This "men-propose" algorithm is men-optimal [19]: the optimal strategy for men is to reveal their true preference lists, as long as all other participants also reveal their true preferences. Women on the other hand have incentives to falsify their preferences [25] in men-propose algorithms, assuming they have full knowledge of the preference lists of all participants. This attack is of real practical concern. In fact, the Gale-Shapley algorithm gives women all the knowledge they need to manipulate the algorithm, since it exposes the complete preference lists of men and women, together with the entire history of engagements made and broken.

Private Gale-Shapley with Secure Multiparty Computation. Generic secure multiparty computation techniques [26, 9] allow n men and n women to compute privately the outcome of the Gale-Shapley algorithm. However, these generic techniques are ill-suited for this purpose:

– **Generic protocols incur high computation and communication costs.** It is hard to estimate precisely the number of gates required to build a circuit that implements the (randomized) Gale-Shapley algorithm. A lower bound is $O(n^2 \log(n))$ gates to perform n^2 comparisons between values of $\log(n)$ bits. With n players, this gives a lower bound on the computational and communication cost of the protocol of $O(n^3 \log(n))$ against passive adversaries. Against an active adversary corrupting less than $n/2$ of the players, the lower bound on the computational and communication cost is $O(n^4 \log(n))$ using the most efficient multiparty computation protocol [4]. In contrast, our protocol incurs a computational and communication cost of $O(n^3)$.
– **Generic protocols are impractical.** The process of building a circuit that implements Gale-Shapley is difficult and error-prone. In contrast, our protocol relies on standard cryptographic components (mix networks) with known efficient implementations.

Efficient private variant of Gale-Shapley. In this paper, we propose an efficient private variant of the Gale-Shapley matching algorithm that is based on mix networks rather than generic secure multiparty computations. A private variant of Gale-Shapley must address two main problems. The first problem is to redesign the algorithm so as to hide the history of engagements made and broken, the number of participants free or engaged at any given point, as well as any other information about the internal state of the algorithm. We propose a solution to this problem in Section 4. The second problem is that the preferences

of participants must be encrypted. We solve this problem in Section 5 and 6. Finally, we present a complete private stable matching algorithm in Section 7 and analyze its properties in Section 8.

3 Model and Definitions

Our algorithm is run jointly by a number of independent parties whom we call matching authorities. The matching authorities collectively run a number of distributed cryptographic protocols, such as distributed key generation, re-encryption mix networks, oblivious tests of plaintext equalities, etc. These protocols serve as building blocks for our private stable matching algorithm and are described in Section 5.

The security and privacy of our stable matching algorithm reduces to the security and privacy of the underlying cryptographic building blocks. We can thus define our adversarial model loosely as the intersection of the adversarial models of the building blocks. For simplicity, we present our results assuming a "honest-but-curious" adversary. More precisely, we consider a static adversary who has passive control over up to all the participants (men and women), and passive control over up to all but one of the matching authorities, as is commonly assumed in the literature on mix networks. Our techniques can easily be extended to accommodate active adversaries, as discussed in Section 8.1.

Definition 1. (Private stable matching algorithm) *An algorithm for computing a stable match is private if it outputs a stable match and reveals no other information to the adversary than what the adversary can learn from that match and from the preferences of the participants it controls.*

4 Hiding the Internal State of the Algorithm

We propose a variant of the Gale-Shapley algorithm that hides its internal state variables, such as the number of men and women free and engaged at any given time, or the history of engagements made and broken. The algorithm described here will not become private until it is combined in Section 7 with the techniques of Sections 5 and 6. It is presented here in non-private form to simplify the understanding of later sections. As before, the algorithm takes as input the lists of preferences of n men and n women and outputs a stable match between them.

Rankings. Let $A_1, \ldots, A_n$ denote n men and $B_1, \ldots, B_n$ denote n women. Every man ranks the women from most to least desired. Thus, a man assigns rank 0 to the woman he likes best, rank 1 to his second place favorite, and so on all the way to rank $n-1$ to the woman he likes the least (rankings do not allow for ties). Similarly, every woman assigns ranks to men from 0 (most favorite man) to $n-1$ (least favorite man). Being ranked *ahead* of someone means being assigned a lower rank, and thus being preferred to that other person. Being ranked *behind* someone means being assigned a higher rank, and thus being less desired than that other person.

Notations. The preference of man A_i is a vector $\boldsymbol{a}_i = (r_{i,1}, \ldots, r_{i,n})$, where $r_{i,j} \in \{0, n-1\}$ is the rank of woman B_j for man A_i. Similarly, the preference of woman B_j is a vector $\boldsymbol{b}_j = (s_{j,1}, \ldots, s_{j,n})$, where $s_{j,i} \in \{0, \ldots, n-1\}$ is the rank of man A_i for woman B_j. The algorithm takes as inputs the vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ and $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$.

Preprocessing. The first step of the algorithm consists of introducing an additional n "fake" men, denoted $A_{n+1}, \ldots, A_{2n}$ (no fake women are defined). The preferences of fake men for women are unimportant to the algorithm. Arbitrarily, we let $\boldsymbol{a}_i = (0, 1, \ldots, n-1)$ for $i = n+1, \ldots, 2n$. The preferences $\boldsymbol{b}_j$ of women must be augmented to reflect the addition of the fake men. As long as women rank all fake men behind all real men, their preferences are unimportant to the algorithm. Arbitrarily, we let every woman B_j assign rank $s_{j,i} = i - 1$ to man A_i for $i = n+1, \ldots, 2n$. We keep the notation $\boldsymbol{b}_j$ for the vector of $2n$ elements that encodes the augmented preference of woman B_j. After this preprocessing step, the algorithm has $2n$ vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{2n}$ (each vector contains n elements that express the rankings assigned by one man to the n women) and n vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ (each vector contains $2n$ elements that express the rankings assigned by one woman to the $2n$ men). Note that the introduction of fake men, and the corresponding update of preferences is done entirely by the algorithm without any involvement from real men or real women.

Computing a stable match. The algorithm proceeds in n rounds. We let $\mathcal{E}_k$ denote the set of engaged men and $\mathcal{F}_k$ denote the set of free men at the beginning of round $k = 1, \ldots, n+1$ (there are only n rounds; with a slight abuse of notation, we let $\mathcal{F}_{n+1}$ and $\mathcal{E}_{n+1}$ denote the set of free and engaged men at the *end* of the last round). Initially, all real men are free $\mathcal{F}_1 = \{A_1, \ldots, A_n\}$, and all fake men are engaged $\mathcal{E}_1 = \{A_{n+1}, \ldots, A_{2n}\}$. Arbitrarily, we let fake man A_{n+i} be initially engaged to women B_i. The other sets are initially empty: $\mathcal{E}_k = \mathcal{F}_k = \emptyset$ for $k > 1$. The algorithm executes the following routine for $k = 1, \ldots, n$:

- While the set $\mathcal{F}_k$ is non-empty, select at random one man (denoted A_i) from $\mathcal{F}_k$. A_i proposes to the woman whom he ranks highest among the women to whom he has never proposed before (let's call this woman B_j). Note that women B_j is always already engaged to a man $A_{i'}$, for some $i' \neq i$. One of two things may happen:
  - If B_j ranks A_i ahead of $A_{i'}$, B_j breaks her engagement to $A_{i'}$ and becomes engaged to A_i. Man A_i is removed from the set $\mathcal{F}_k$ and added to $\mathcal{E}_k$, whereas man $A_{i'}$ is removed from $\mathcal{E}_k$ and added to $\mathcal{F}_{k+1}$.
  - If B_j ranks A_i behind $A_{i'}$, she stays engaged to $A_{i'}$. Man A_i is removed from set $\mathcal{F}_k$ and added to set $\mathcal{F}_{k+1}$.
- When $\mathcal{F}_k$ is empty, we define $\mathcal{E}_{k+1} = \mathcal{E}_k$.

The algorithm ends after n rounds and outputs the set $\mathcal{E}_{n+1}$ of engaged men and their current partners.

Invariants. Note that this algorithm preserves certain invariants. All n women are always engaged to some man. During round k, the number of engaged men is always exactly $|\mathcal{E}_k| = n$. Engaged men do not move progressively from set $\mathcal{E}_k$ to set $\mathcal{E}_{k+1}$ during round k, but rather they move all at once at the end of round k. Every time a new proposal is made, the cardinality of $\mathcal{F}_k$ decreases by one, the cardinality of $\mathcal{F}_{k+1}$ increases by one and the cardinality of $\mathcal{E}_k$ is unchanged, irrespective of whether a woman changes partner or not.

Proposition 1. *This algorithm outputs a stable match between the real men $A_1, \ldots, A_n$ and the n women $B_1, \ldots, B_n$.*

Proof. We must prove that the match is stable and involves only real men (no fake men). The proof that the final match is stable is exactly similar to that given for the original Gale-Shapley algorithm in [6].

The proof that the final match involves only real men is by contradiction. We observe first that once a woman is engaged to a real man, she will stay engaged to real men in subsequent rounds, since all women rank all real men ahead of all fake men. Now assume that a fake man A_i is engaged to a woman B_j when the algorithm ends after n rounds. This implies that B_j was never engaged to a real man. Since there are only n women, there must be at least one real man $A_{i'}$ who remains free at the end of the protocol. Now the free real man $A_{i'}$ must have proposed to all n women, B_j included, and must have been rejected by all. But B_j, who was always engaged to fake men, could not reject $A_{i'}$ without breaking the assumption that all women prefer real men to fake men. □

5 Cryptographic Building Blocks

Our private stable matching algorithm uses cryptographic building blocks which we now describe briefly. These building blocks are all standard distributed cryptographic algorithms run jointly by the matching authorities.

Threshold Paillier encryption. The Paillier encryption scheme [18] allows for threshold encryption [5, 7]. In what follows, all ciphertexts are encrypted with a threshold version of Paillier. The matching authorities hold shares of the corresponding decryption key, such that a quorum consisting of all parties can decrypt.

Robust re-encryption mix network. A re-encryption mix network re-encrypts and permutes a number of input (Paillier) ciphertexts. In our application, the matching authorities play the role of mix servers. If we allow active adversaries (see Section 8.1), we must use robust re-encryption mixnets such as [11] or [16]. When we say the matching authorities "mix" a set of inputs according to a permutation π, we mean that they run the set of inputs through a mix network and we let π denote the global (secret) permutation (which is not known to the matching authorities).

Oblivious test of plaintext equality. Let $E(m_1)$ and $E(m_2)$ be two Paillier ciphertexts. An oblivious test of plaintext equality [12, 13] lets the joint holders of the decryption key determine whether $m_1 = m_2$ without revealing any other information (hence the name oblivious). We denote this protocol $\mathsf{EQTEST}(E(m_1), E(m_2))$. The protocol outputs either $m_1 = m_2$ or $m_1 \neq m_2$.

Repeated test of plaintext equality. The protocol $\mathsf{INDEX}(\boldsymbol{a}, E(\rho))$ takes as input a vector $\boldsymbol{a} = (E(a_1), \ldots, E(a_n))$ of n Paillier ciphertexts and an additional Paillier ciphertext $E(\rho)$ such that there exists one and only one value $i \in \{1, \ldots, n\}$ for which $\rho = a_i$. The protocol outputs the index i such that $a_i = \rho$. The protocol INDEX can be implemented with n instances of EQTEST.

Finding the larger of 2 plaintexts. Let $E(m_1)$ and $E(m_2)$ be two Paillier ciphertexts such that $m_1, m_2 \in \{0, \ldots, n-1\}$ and $m_1 \neq m_2$. We propose a protocol $\mathsf{COMPARE}(E(m_1), E(m_2))$ that outputs true if $m_1 > m_2$ and false otherwise, without leaking any other information. The protocol proceeds as follows. For $i = 1, \ldots, n-1$, the matching authorities compute ciphertext $D_i = E(m_1 - m_2 - i)$ using Paillier's additive homomorphism. Note that $m_1 > m_2$ if and only if one of the ciphertexts D_i is an encryption of 0. The matching authorities mix (i.e. re-encrypt and permute) the set of ciphertexts $D_1, \ldots, D_{n-1}$. Let $D'_1, \ldots, D'_{n-1}$ denote the mixed set. The matching authorities then compute $\mathsf{EQTEST}(D'_i, E(0))$ for $i = 1, \ldots, n-1$. If an equality is found, they output true, otherwise they output false.

6 Encrypting Preferences

Let E denote the encryption function for a threshold public-key encryption scheme with an additive homomorphism, such as for example a threshold version [5, 7] of the Paillier encryption scheme [18]. We assume that the matching authorities are the joint holders of the private decryption key.

Let $A_1, \ldots, A_m$ be m men and $B_1, \ldots, B_n$ be n women. As in Section 4, we let $r_{i,j} \in \{0, \ldots, n-1\}$ denote the rank of woman B_j for man A_i, and $s_{j,i} \in \{0, \ldots, m-1\}$ denote the rank of man A_i for woman B_j. We define $p_{i,j} = E(r_{i,j})$ and $\boldsymbol{a}_i = (p_{i,1}, \ldots, p_{i,n})$. Similarly, we define $q_{j,i} = E(s_{j,i})$ and $\boldsymbol{b}_j = (q_{j,1}, \ldots, q_{j,m})$.

6.1 Bid Creation

We define a "bid" as an encrypted representation of the preferences of one men for women, together with additional "book-keeping" information. For $i \in \{1, \ldots, m\}$, the bid W_i that represents the preferences of man A_i consists of $3n + 2$ Paillier ciphertexts defined as follows:

- An encryption $E(i)$ of the index i of man A_i.
- The vector $\boldsymbol{a}_i = (p_{i,1}, \ldots, p_{i,n})$.

- A vector $\boldsymbol{v}_i = (E(1), \ldots, E(n))$.
- The vector $\boldsymbol{q}_i = (q_{1,i}, \ldots, q_{n,i})$.
- A ciphertext $E(\rho)$, where ρ is the number of times the bid has been rejected. Initially $\rho = 0$.

The role of the ciphertext $E(i)$ is to maintain the association between bid W_i and the man A_i whose preferences the bid expresses. The vector $\boldsymbol{a}_i$ encodes the preferences of man A_i for women $B_1, \ldots, B_n$. As we shall see, the elements of $\boldsymbol{a}_i$ are permuted at random in the course of the private stable matching algorithm. Thus the need for the vector $\boldsymbol{v}_i$, whose role is to maintain the association between the rankings contained in $\boldsymbol{a}_i$ and the women these rankings pertain to: the element in position j of $\boldsymbol{v}_i$ is always an encryption of the index of the woman whose rank is given by the element in position j of $\boldsymbol{a}_i$. The vector $\boldsymbol{q}_i$ encodes the initial rank given to man A_i by women $B_1, \ldots, B_n$. Finally, the ciphertext $E(\rho)$ records the number of times that the bid has been rejected: the value ρ is updated every time an engagement is broken.

Free and engaged bids. A bid by itself, as defined above, is called a free bid because it is not paired up with a woman. A bid paired up with a woman is called an engaged bid. More precisely, an engaged bid is a triplet $(W_i,\ E(j),\ q_{j,i})$, where:

- $W_i = [E(i),\ \boldsymbol{a}_i,\ \boldsymbol{v}_i,\ \boldsymbol{q}_i,\ E(\rho)]$ is the bid of man A_i
- $E(j)$ is an encryption of the index $j \in \{1, \ldots, n\}$ of a woman B_j
- $q_{j,i}$ is an encryption of the rank given to man A_i by woman B_j

Breaking an engagement. Let $(W_i,\ E(j),\ q_{j,i})$ be an engaged bid. If this bid loses woman B_j to another bid, we update it as follows. First, we strip the triplet of the values $E(j)$ and $q_{j,i}$, keeping only the free bid W_i. Next, we increment the counter ρ in W_i by one, using Paillier's additive homomorphism (i.e. we multiply $E(\rho)$ by $E(1)$ to obtain $E(\rho + 1)$).

6.2 Bid Mixing

The Paillier cryptosystem allows for semantically secure re-encryption of ciphertexts. Since bids (both free and engaged) are made up of Paillier ciphertexts, they can be re-encrypted, and in particular they can be mixed with a re-encryption mix network. We consider two types of mixing for bids: "external" mixing and "internal" mixing.

External bid mixing. External mixing takes as input a set of bids, either all free or all engaged, and mixes them in a way that hides the order of the bids but preserves the internal position of ciphertexts *within* a bid. External mixing considers bids as atomic elements and preserves their internal integrity. More precisely, let us consider an initial ordering of k free bids $W_1, \ldots, W_k$ and let σ be a permutation on k elements. The external mixing operation re-encrypts all

the Paillier ciphertexts in all the bids (preserving the order of ciphertexts within each bid) and outputs $W_{\sigma(1)}, \ldots, W_{\sigma(k)}$. A set of engaged bids can be mixed externally in exactly the same way. In this paper, free and engaged bids are never mixed externally together (since free bids are made of $3n+2$ ciphertexts and engaged bids of $3n+4$, they would not blend together). Intuitively, external bid mixing hides which bid encodes the preferences of which man.

Internal bid mixing. Internal mixing takes as input a set of bids that may contain both free and engaged bids. These bids are mixed "internally" in a way that hides the order of a subset of the ciphertexts within the bids but preserves the order of the bids themselves. More precisely, let us consider a set of k bids and let π be a permutation on n elements. The bids in the set are processed one by one, and output in the same order as they were given as input. A free bid is processed as follows. Let $W_i = [E(i),\ \boldsymbol{a}_i,\ \boldsymbol{v}_i,\ \boldsymbol{q}_i,\ E(\rho)]$ be a free bid. We define the internally permuted bid as $\pi(W_i) = [E(i),\ \pi(\boldsymbol{a}_i),\ \pi(\boldsymbol{v}_i),\ \pi(\boldsymbol{q}_i),\ E(\rho)]$, where the permuted vectors $\pi(\boldsymbol{a}_i),\ \pi(\boldsymbol{v}_i)$ and $\pi(\boldsymbol{q}_i)$ are defined as follows:

> Let $\boldsymbol{a}_i = (p_{i,1}, \ldots, p_{i,n})$. Let $p'_{i,1}, \ldots, p'_{i,n}$ be re-encryptions of the ciphertexts $p_{i,1}, \ldots, p_{i,n}$. We let $\pi(\boldsymbol{a}_i) = \left(p'_{i,\pi(1)}, \ldots, p'_{i,\pi(n)}\right)$. The vectors $\pi(\boldsymbol{v}_i)$ and $\pi(\boldsymbol{q}_i)$ are defined in exactly the same way.

Engaged bids are processed in the same way. Let $(W_i,\ E(j),\ q_{j,i})$ be an engaged bid. We define the corresponding internally permuted engaged bid as $(\pi(W_i),\ E(j),\ q_{j,i})$.

Note that the same internal permutation π is applied to all the bids in the set. Note also that, as always in mix networks, the global permutation π is in fact the combination of permutations chosen by all the matching authorities, so that the matching authorities themselves do not know π (unless they all collude). Intuitively, internal mixing hides which woman a particular ciphertext pertains to.

6.3 Conflicts Between Bids

Opening a free bid. Let $\pi(W_i) = [E(i),\ \pi(\boldsymbol{a}_i),\ \pi(\boldsymbol{v}_i),\ \pi(\boldsymbol{q}_i),\ E(\rho)]$ be a free bid that has been internally permuted by a permutation π on n elements. Since π is the result of one (or several) internal bid mixing operations, it is not known to the matching authorities. Let j be the index of the woman B_j assigned rank ρ by that bid. Opening W_i means determining $E(j)$ and $q_{j,i} = E(s_{j,i})$ without learning anything else about the bid. Note that opening a bid would be trivial if the permutation π were known. Without knowledge of π, the matching authorities open a bid as follows. The matching authorities jointly compute $\alpha = \mathsf{INDEX}(\pi(\boldsymbol{a}_i), E(\rho))$. Since the same permutation π is applied to $\boldsymbol{a}_i, \boldsymbol{v}_i$ and $\boldsymbol{q}_i$, the element in position α of $\pi(\boldsymbol{v}_i)$ is $E(j)$ and the element in position α of $\pi(\boldsymbol{q}_i)$ is $q_{j,i} = E(s_{j,i})$.

Detecting a conflict. Let $\pi(W_i)$ be a free bid, and let $\left(\pi(W_{i'}),\ E(j'),\ q_{j',i'}\right)$ be an engaged bid, both internally permuted according to the same permutation π on n elements (we assume again that π is not known to the matching authorities). Let $E(j)$ and $q_{j,i}$ be the ciphertexts obtained when the free bid $\pi(W_i)$ is opened. Detecting a conflict between these two bids means determining whether $j = j'$, without learning anything else about the bids. To do so, the matching authorities jointly compute $\mathsf{EQTEST}(E(j), E(j'))$. The bids conflict if and only if EQTEST returns an equality.

Resolving a conflict. Let $\pi(W_i)$ be a free bid that opens up to $E(j), q_{j,i}$ and conflicts with an engaged bid $\left(\pi(W_{i'}),\ E(j),\ q_{j,i'}\right)$ for woman B_j. Resolving the conflict means outputting a new free bid and a new engaged bid such that:

- if B_j ranks A_i ahead of $A_{i'}$, the free bid is a re-encryption of $W_{i'}$ and the engaged bid is a re-encryption of $(W_i,\ E(j),\ q_{j,i})$
- if B_j ranks A_i behind $A_{i'}$, the free bid is a re-encryption of W_i and the engaged bid is a re-encryption of $(W_{i'},\ E(j),\ q_{j,i'})$

without revealing anything else about the bids (in particular the protocol does not reveal which bid wins the contested woman). To resolve the conflict, the matching authorities first create an engaged bid $\left(\pi(W_i),\ E(j),\ q_{j,i}\right)$ out of the free bid $\pi(W_i)$. The two engaged bids are then mixed externally. Let $q'_{j,i'}$ and $q'_{j,i}$ denote the re-encrypted and permuted images of $q_{j,i'}$ and $q_{j,i}$. The matching authorities jointly compute $\mathsf{COMPARE}(q'_{j,i'}, q'_{j,i})$. The result of this comparison determines (privately) which bid stays engaged, and which is stripped of B_j to make a free bid.

7 Private Stable Matching Algorithm

We describe a private algorithm for finding a stable matching in which men propose to women. The algorithm follows the general structure of the algorithm described in Section 4, but operates on encrypted bids to preserve privacy. The algorithm is run by a number of matching authorities. We use the notations defined in Section 6.

Setup. In a setup step, the matching authorities jointly generate the public/private key pair for a threshold public-key encryption scheme E with an additive homomorphism. For example, E may be a threshold version [5, 7] of the Paillier encryption scheme [18].

Input submission. As before, we let $r_{i,j} \in \{0, \ldots, n-1\}$ denote the rank of woman B_j for man A_i, and $s_{j,i} \in \{0, \ldots, n-1\}$ denote the rank of man A_i for woman B_j. Every man A_i submits a vector of n Paillier ciphertexts

$$\boldsymbol{a}_i = (p_{i,1}, \ldots, p_{i,n}),$$

where $p_{i,j} = E(r_{i,j})$, and every woman B_i similarly submits a vector of n Paillier ciphertexts

$$\boldsymbol{b}_j = (q_{j,1}, \ldots, q_{j,n}),$$

where $q_{j,i} = E(s_{j,i})$.

Addition of fake men. The matching authorities define an additional n fake men $A_{n+1}, \ldots, A_{2n}$ as described in Section 4. Specifically, the matching authorities define $r_{i,j} = j - i + n \mod (n-1)$ for $i \in \{n+1, \ldots, 2n\}$ and $j \in \{1, \ldots, n\}$ and compute the corresponding vectors $\boldsymbol{a}_i = (p_{i,1}, \ldots, p_{i,n})$ for $i = n+1, \ldots, 2n$, where $p_{i,j} = E(r_{i,j})$. The matching authorities also define $s_{j,i} = i - 1$ for $j \in \{1, n\}$ and $i \in \{n+1, 2n\}$ and augment the vectors $\boldsymbol{b}_j$ with these new values (we keep the notation $\boldsymbol{b}_j$ for the augmented vectors): $\boldsymbol{b}_j = (q_{j,1}, \ldots, q_{j,2n})$. After this preprocessing step, the matching authorities have $2n$ vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{2n}$ (each vector contains n ciphertexts that express the rankings assigned by one man to the n women) and n vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ (each vector contains $2n$ ciphertexts that express the rankings assigned by one woman to the $2n$ men).

Bid creation. The matching authorities create $2n$ bids $W_1, \ldots, W_{2n}$, where W_i encodes the preferences of man A_i. Bid W_i is defined as follows (see Section 6.1):

$$W_i = [E(i),\ \boldsymbol{a}_i,\ \boldsymbol{v}_i,\ \boldsymbol{q}_i,\ E(0)]$$

Throughout the algorithm, bids are divided into free bids and engaged bids. Initially, the n bids corresponding to real men are free: $\mathcal{F}_1 = (W_1, \ldots, W_n)$, whereas the n bids corresponding to the fake men are engaged: $\mathcal{E}_1 = (W_{n+1}, \ldots, W_{2n})$. More precisely, man W_{n+j} is paired with woman B_j. For $j = 1, \ldots, n$ the engaged bid of (fake) man A_{n+j} is thus defined as:

$$\Big(W_{n+j},\ E(j),\ q_{j,n+j}\Big)$$

Initial mixing. The sets $\mathcal{E}_1$ and $\mathcal{F}_1$ are each independently mixed externally by the matching authorities. Next, the matching authorities mix internally the set $\mathcal{E}_1 \cup \mathcal{F}_1$.

Computing a stable match. As in Section 4, the core of our private stable matching algorithm proceeds in n rounds. We let $\mathcal{E}_k$ denote the set of engaged bids and $\mathcal{F}_k$ denote the set of free bids at the beginning of round $k = 1, \ldots, n+1$. The algorithm executes the following routine for $k = 1, \ldots, n$:

While the set $\mathcal{F}_k$ is non-empty, select at random one free bid (denoted W_i) from $\mathcal{F}_k$. Then:

1. The matching authorities jointly open up bid W_i, and learn $E(j)$ and $q_{j,i} = E(s_{j,i})$.

2. There is always exactly one engaged bid in $\mathcal{E}_k$ that conflicts with W_i. The matching authorities jointly find that engaged bid using (at most $|\mathcal{E}_k| = n$ times) the conflict detection protocol described in Section 6.3. Let's call the conflicting engaged bid $(W_{i'}, E(j), q_{j,i'})$.
3. Using the conflict resolution protocol of Section 6.3, the matching authorities resolve the conflict. The conflict resolution protocol does not reveal which bid wins but it ensures that one bid (either W_i or $W_{i'}$) is added to $\mathcal{E}_k$ and the other to $\mathcal{F}_{k+1}$. For clarity, we explain what happens behind the scene:
   - If W_i wins, it becomes an engaged bid $(W_i, E(j), E(s_{j,i}))$ and is moved from the set $\mathcal{F}_k$ to the set $\mathcal{E}_k$. The engagement of bid $(W_{i'}, E(j), E(s_{j,i'}))$ is broken (see Section 6.1) and the newly free bid $W_{i'}$ moves from the set $\mathcal{E}_k$ to $\mathcal{F}_{k+1}$.
   - If W_i loses, it remains free and moves from $\mathcal{F}_k$ to $\mathcal{F}_{k+1}$. The engaged bid $(W_{i'}, E(j), E(s_{j,i}))$ stays in the set $\mathcal{E}_k$.
4. The set $\mathcal{E}_k$ is mixed externally. All bids in the sets $\mathcal{E}_k \cup \mathcal{F}_k \cup \mathcal{F}_{k+1}$ are then mixed internally.

At the end of the round (when the set $\mathcal{F}_k$ is empty), we define $\mathcal{E}_{k+1} = \mathcal{E}_k$. The sets $\mathcal{E}_{k+1}$ and $\mathcal{F}_{k+1}$ are independently mixed externally. The set $\mathcal{E}_{k+1} \cup \mathcal{F}_{k+1}$ is then mixed internally.

Bid decryption and final output. After n rounds, the final set $\mathcal{E}_{n+1}$ consists of n engaged bids of the form $(W_i, E(j), E(s_{j,i}))$, where $W_i = [E(i),\ \boldsymbol{a}_i,\ \boldsymbol{v}_i,\ \boldsymbol{q}_i,\ E(\rho)]$. At this point, the matching authorities retain only two ciphertexts from an engaged bid: $E(i)$ and $E(j)$. The matching authorities thus obtain n pairs of the form $(E(i); E(j))$. These pairs $(E(i); E(j))$ are (externally) mixed by the matching authorities, then jointly decrypted. The decryption of pair $(E(i); E(j))$ reveals than man A_i is paired with woman B_j.

8 Properties

Proposition 2. *The algorithm of Section 7 terminates after n rounds and outputs a stable matching between n real men and n real women. The computational cost of the algorithm is dominated by the cost of running $3n^2$ re-encryption mix networks on at most $2n$ Paillier ciphertexts. The corresponding communication cost is $O(n^3)$.*

Since we assume an honest-but-curious passive adversary, the proof of correctness follows directly from Proposition 1. The computational cost is dominated by the cost of re-encryption mix networks. For every element in $\mathcal{F}_k$ in every round k, the matching authorities must run 3 re-encryption mix networks: one to resolve the conflict between bids, one for external mixing and one for internal mixing. The overall computational cost is thus $O(n^3)$ modular exponentiations. This is a substantial cost, but not unreasonable considering that stable matching algorithms are typically run off-line and that low latency is not a requirement. In practice, stable matching algorithms involving up to a few thousands of participants could be run privately within a day on commodity hardware.

Proposition 3. *The algorithm of Section 7 is private according to Definition 1, assuming Paillier encryption is semantically secure and the underlying re-encryption mix network is private.*

Proof (Sketch). In the execution of the protocol, the matching authorities compute and output intermediate values (Paillier ciphertexts, modular integers and boolean values), then finally a stable match. We prove that a passive adversary cannot distinguish between the sequence of intermediate values produced by the protocol, and a random sequence of intermediate values drawn from an appropriate probability distribution. The proof is by contradiction. If an adversary $\mathcal{A}$ can distinguish with non-negligible advantage the output of the algorithm from random, then by a standard hybrid argument, there exists one intermediate value V that $\mathcal{A}$ can distinguish from random.

If V is a Paillier ciphertext, we can use $\mathcal{A}$ to break the semantic security of Paillier encryption, contradicting our assumption about Paillier's security.

If V is a modular integer or a boolean value, the value of V depends on the internal or external permutation applied by the matching authorities immediately before computing V. Thus if $\mathcal{A}$ can distinguish between different values of V, we can use $\mathcal{A}$ to distinguish between the outputs produced by a re-encryption mix-network using different permutations, breaking the assumption that the mix network is private. □

8.1 Active Adversaries

We have assumed a passive adversary throughout, but our techniques can be extended to accommodate active adversaries at the cost of additional proofs of correct execution. We consider here an active adversary who has static control over up to all the participants (men and women), and static control over up to a strict minority of matching authorities. We must augment the private stable matching algorithms of Section 7 with proofs of correct protocol execution by participants and matching authorities. These proofs are verified by the matching authorities (a strict majority of whom is assumed honest).

The participants need only prove to the matching authorities that the preference vectors they submit ($\boldsymbol{a}_i$ for man A_i and $\boldsymbol{b}_j$ for woman B_j) follow the protocol specifications, i.e. are Paillier encryptions of a permutation of the set $\{0, \ldots, n-1\}$. We use non-interactive zero-knowledge (NIZK) proofs that the decryption $E^{-1}(C)$ of a Paillier ciphertext C lies within a given plaintext set $\{0, \ldots, n-1\}$. For Paillier encryption, these proofs reduce to proving knowledge of the root of the randomization factor [5]. These proofs can also be combined conjunctively and disjunctively using standard techniques [22]. We can thus prove that a vector $\boldsymbol{a}_i = (E(r_1), \ldots, E(r_n))$ is well-formed with the following NIZK proof: $\bigwedge_{j \in \{0,\ldots,n-1\}} \Big(\bigvee_{i \in \{1,\ldots,n\}} (E^{-1}(E(r_i)) = j) \Big)$.

The correct behavior of matching authorities must itself be verified. The building blocks of Section 5 all accept variants that are secure against active adversaries. As usual, a matching authority caught not following the protocol is excluded from future computations and replaced by a new authority.

9 Conclusion

We have proposed a private stable matching algorithm based on a variant of the Gale-Shapley algorithm. Assuming a majority of honest matching authorities, our protocol correctly outputs a stable match, and reveals no other information than what can be learned from that match and from the preferences of participants controlled by the adversary. We have proved the security and privacy of our protocol based on assumptions about standard distributed cryptographic protocols. Our protocol is practical and we hope that it will be used to offer greater privacy to the tens of thousands of students and professionals whose careers are affected every year by matching algorithms.

References

1. Association of Psychology Postdoctoral and Internship Centers. `http://www.appic.org/match/`
2. C. Bandela, Y. Chen, A. Kahng, I. Mandoiu and A. Zelikovsky. Multiple-object XOR auctions with buyer preferences and seller priorities. In *Competitive Bidding and Auctions*, K.K. Lai and S. Wang, ed. Kluwer Academic Publishers.
3. Canadian Resident Matching Service (CaRMS). `http://www.carms.ca/jsp/main.jsp`
4. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Proc. of Eurocrypt'99*, pp. 311–326.
5. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proc. of Public Key Cryptography 2001*, pp. 119–136.
6. D. Gale and H. S. Shapley. College Admissions and the Stability of Marriage. American Mathematical Monthly, 1962.
7. P.-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Proceedings of Financial Cryptography 2000*, pp. 90–104, 2000.
8. D. Gale and M. Sotomayor. Ms Machiavelli and the Stable Matching Problem. In *American Mathematical Monthly*, 92, pp. 261–268, 1985.
9. O. Goldreich, S. Micali and A. Widgerson. How to play any mental game. In *STOC'87*, pp. 218–229. ACM, 1987.
10. D. Gusfield and R. Irving. The Stable Marriage Problem: Structure and Algorithms. MIT Press.
11. M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proc. of USENIX'02*, pp. 339–353.
12. M. Jakobsson and C. Schnorr. Efficient Oblivious Proofs of Correct Exponentiation. In *Proc. of CMS 99*.
13. H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Proc. of Asiacrypt 2003*, pp. 416–433. LNCS 2894.
14. National Matching Services Inc. `http://www.natmatch.com/`
15. National Resident Matching Program (NRMP). `http://www.nrmp.org/`
16. A. Neff. A verifiable secret shuffle and its application to e-voting. In *Proc. of ACM CCS '01*, pp. 116–125.
17. M. Ostrovsky. Stability in supply chain networks. Available on the web at `economics.uchicago.edu/download/Supply\%20Chains\%20-\%20December\%2012.pdf`
18. P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proc. of Eurocrypt 1999*, pp. 223–238. LNCS 1592, Springer Verlag.
19. A. Roth. The Economics of Matching: Stability and Incentives. In *Mathematics of Operations Research*, 7, pp. 617–628, 1982.
20. A. Roth and M. Sotomayor. Two-sided matching: a study in game-theoretic modeling and analysis. Econometric Society Monograph Series (1990). New York: Cambridge University Press.
21. Al Roth's game theory, experimental economics, and market design page. Bibliography of two-sided matching. On the web at `http://kuznets.fas.harvard.edu/~aroth/bib.html#matchbib`
22. A. D. Santis, G. D. Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of szk. In *Proc. of the IEEE FOCS 1994*, pages 454–465, 1994.
23. Scottish PRHO Allocation (SPA) scheme. `http://www.nes.scot.nhs.uk/spa/`
24. M. Soerensen. How smart is smart money? An empirical two-sided matching model of venture capital. Available on the web at `http://finance.wharton.upenn.edu/department/Seminar/2004SpringRecruiting/Micro/SorensenPaper-micro-012204.pdf`
25. C.-P. Teo, J. Sethuraman and W.-P. Tan. Gale-Shapley stable marriage problem revisited: strategic issues and applications. In *Proc. of IPCO '99: the 7th Conference on Integer Programming and Combinatorial Optimisation*, pp. 429–438. LNCS 1610.
26. A. C. Yao. Protocols for secure computations. In *FOCS'82*, pp. 160–164. IEEE Computer Society, 1982.