Improved Efficiency for Private Stable Matching ⋆

Matthew Franklin, Mark Gondree, and Payman Mohassel

Department of Computer Science
University of California, Davis
{franklin, gondree, mohassel}@cs.ucdavis.edu

Abstract. At Financial Crypto 2006, Golle presented a novel framework for the privacy preserving computation of a stable matching (stable marriage). We show that the communication complexity of Golle's main protocol is substantially greater than what was claimed in that paper, in part due to surprising pathological behavior of Golle's variant of the Gale-Shapley stable matching algorithm. We also develop new protocols in Golle's basic framework with greatly reduced communication complexity.

Keywords: stable matching, stable marriage, Gale-Shapley, privacy-preserving protocols, secure multiparty computation, passive adversaries.

1 Introduction

Efficient stable matching (stable marriage) algorithms are used in a wide variety of practical settings, including the well-known example of matching U. S. medical school graduates to hospitals, for their residencies. Gusfield and Irving [14] have written a good overview of stable matching algorithms.

Golle [12] argues persuasively that efficient privacy-preserving protocols for stable matching could have great practical benefit. In fact, Golle goes on to develop just such a protocol. We find the basic framework of his approach to be quite appealing, and worthy of further examination. In his framework, some number of honest-but-curious "matching authorities" (MAs) receive encrypted preference lists from the participants, and then execute a variant of the classic Gale-Shapley algorithm that has been specially tuned for concealment. The main cryptographic tools in Golle's protocol are threshold homomorphic encryption and re-encryption mixnets.

Our first contribution is to show that Golle's protocol has substantially greater communication complexity than what was reported in the original paper. For example, the total communication is $O(tn^5)$ ciphertexts instead of $O(n^3)$ ciphertexts as reported (where the number of matching authorities is t, and the

⋆ This is the full version of an article [8] to appear in *CT-RSA 2007*.

number of participants is $O(n)$). This is due in part to a surprising anomaly in Golle's variant of the Gale-Shapley algorithm that requires it to run for more rounds than Golle's analysis suggests (quadratic rather than linear in the number of participants).

Our second contribution is to design new privacy preserving protocols in Golle's basic framework with reduced communication complexity (under similar cryptographic assumptions). Our protocol in Section 4.2 has improved communication complexity when there are an arbitrary number t of matching authorities. Our protocol in Section 5 reduces the communication complexity even further when there are exactly two matching authorities. One way we achieve our improved efficiency is by designing our own variant of Gale-Shapley that is "tuned for concealment" but with better convergence properties than Golle's variant. The following table summarizes the efficiency of these new protocols and our new analysis of Golle's protocol.

Section	Protocol	MAs	Total Work	Total Communication	Round Complexity
4.1	Golle's	t	$O(n^5)$	$O(tn^5)$	$\widetilde{O}(n^3)$
4.2	Ours	t	$O(n^4\sqrt{\log n})$	$O(tn^3)$	$\widetilde{O}(n^2)$
5	Ours	2	$O(n^4)$	$O(n^2)$	$\widetilde{O}(n^2)$

The organization of the rest of the paper is as follows. Preliminary notions (models, definitions, primitives) are given in Section 2. In Section 3, we discuss the Gale-Shapley stable matching algorithm, with a careful analysis of Golle's variant and our new variant. In Section 4, we present and analyze Golle's protocol and our protocol for the case with multiple matching authorities. In Section 5, we describe a protocol for the case of just two matching authorities. Conclusions are presented in Section 6.

2 Preliminaries

2.1 The Stable Marriage Problem

We consider the formulation of the stable matching problem with complete preference lists (every man ranks every woman, and vice-versa) and one-to-one matchings. Due to its simplicity, and the fact that other variants of the stable matching problem can be reduced to this formulation, it is a particularly attractive version with which to work. The problem is as follows. Consider two sets, one of n men and one of n women. Every man ranks the n women, and every woman ranks the n men. A matching, or *marriage*, is a bijection between the sets. A matching is *stable* if there is no unmatched man and woman who rank

each other higher than their own spouses. The stable marriage problem is, given the preference lists of n men and n women, to find a stable marriage (there is always one, and there may be several). In Section 3, we discuss algorithms for the stable marriage problem.

2.2 Models and Definitions

We adopt the same network model as Golle [12]. At the start of the protocol, each player sends a single encrypted message (derived from his or her preferences) to two or more *matching authorities* (MA's). These matching authorities execute a synchronous protocol among themselves to compute the stable matching.

For simplicity (and fairness of comparison), our security model is the same as that considered by Golle [12]. Specifically, we consider a *passive* adversary, meaning an adversary with passive control over any of the players (men and women) and passive control over all but one of the matching authorities. More precisely, our security guarantees hold for any adversary in the intersection of the adversarial models of the primitives we use. Our guarantees are in relation to the following security notion, due to Golle [12]: a protocol is a *private stable matching protocol* if it outputs a stable match and reveals no other information to a passive adversary than what she can learn from the matching and from the preferences of the participants she controls.

In all our constructions, we compose protocols that are private with respect to passive adversaries, and make use of composition theorems that guarantee security under composition. These theorems enable us to claim our protocols private against a passive adversary as long as our subprotocols are private. Please see [3, 10] for more details.

Encryption. Unless otherwise stated, we let E denote the encryption function for a threshold public-key encryption scheme that is additively homomorphic, such as a threshold version [7, 6] of the Paillier encryption scheme [23]. The matching authorities are the joint holders of the decryption key, such that only a quorum of all parties can decrypt.

Notation. We use the following asymptotic notation: $o(f)$ denotes that the asymptotic upper bound f is not tight; $\Omega(f)$ denotes that the asymptotic lower bound f is tight; and $\widetilde{O}(f)$ denotes the asymptotic upper bound $O(f)$, ignoring $\mathrm{polylog}(f)$ factors. We denote the XOR operation between two equal-length bit strings a, b by $a \oplus b$. In Section 2.3 below, unless otherwise noted, "poly-log complexity" is in reference to the security parameter for each primitive.

2.3 Primitives

Re-encryption Mix Network. In our application, when we say the authorities *mix* some (Paillier) ciphertexts, we mean the authorities run a re-encryption mix network [22, 15], permuting the ciphertexts according to a secret permutation known to none of the individual authorities. Since we consider a passive adversary, n ciphertexts can be mixed by t mixing authorities in constant round and $O(n)$ time, taking advantage of parallel mixing techniques [13]. The total communication complexity of the parallel mixnet is, like a serial mixnet, $O(tn)$ ciphertexts.

Private Oblivious Equality Test. Let $E(m_1), E(m_2)$ be two Paillier ciphertexts. Define $\mathsf{EQTEST}(E(m_1), E(m_2)) = b$ where $b = 1$ if $m_1 = m_2$ and $b = 0$ otherwise. EQTEST is a (chooser private) oblivious test of plaintext equality [16, 20] if it reveals the output to the joint holders of the decryption keys, without revealing any other information to any other parties.

MPC Private Equality Test. Let $E(m_1), E(m_2)$ be two Paillier ciphertexts. Define $\mathsf{EEQTEST}(E(m_1), E(m_2)) = E(b)$ where $b = 1$ if $m_1 = m_2$ and $b = 0$ otherwise. $\mathsf{EEQTEST}$ is the secure multiparty computation of the equality test if our parties learn the output $E(b)$, but no additional information about the plaintexts m_1, m_2. [4, 17] both give constant-round protocols with poly-log communication complexity for computing this function, either of which are adaptable to our setting (*i.e.*, threshold, additively homomorphic ciphertexts).

Private Oblivious Value Comparison. Let $E(m_1), E(m_2)$ be two Paillier ciphertexts. Define $\mathsf{COMPARE}(E(m_1), E(m_2)) = b$ where $b = 1$ if $m_1 < m_2$ and $b = 0$ otherwise. For our purposes, we have $0 \leq m_1, m_2 \leq n$. 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 $\mathsf{EQTEST}(E_i, E(0))$. 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 $E(m_1), E(m_2)$ be two Paillier ciphertexts. Define $\mathsf{ECOMPARE}(E(m_1), E(m_2)) = E(b)$ where $b = 1$ if $m_1 < m_2$ and $b = 0$ otherwise. $\mathsf{ECOMPARE}$ is the secure multiparty computation of the less-than function if our parties learn the output $E(b)$, but no additional information about the plaintexts m_1, m_2. [17, 5] both give constant-round protocols with poly-log communication complexity for computing this function, either of which are adaptable to our setting.

Private Reduction of a Secret Modulo a Public Integer. Let $E(a)$ be a Paillier ciphertext, and q be an integer. Define $\mathsf{MOD}(E(a), q) = E(a \bmod q)$. MOD is the secure multiparty computation of the modular function if our parties learn the output, but no additional information about the plaintext integer a. [1, 17] both give protocols with poly-log communication complexity for computing this function, either of which are adaptable to our setting. The former has a poly-log round complexity, the latter is constant-round.

SPIR with sublinear communication complexity. Let δ be a database with N elements, indexed $\{0, \ldots, N-1\}$. Let $\mathsf{SPIR}^{\delta}_{m}(\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$ represent Stern's symmetric private information retrieval protocol [24]. As in any PIR protocol, a chooser holds a secret index i and, at the end of the protocol, the chooser learns the element of the database at index i, while the database learns nothing about which index was accessed. Additionally, the chooser learns nothing about any of the other database elements (thus, symmetry). In $\mathsf{SPIR}^{\delta}_{m}$, the index i is encoded following a standard trick, due to Kushilevitz and Ostrovsky [18]. The database is imagined as a series of m sized buckets (the first m entries 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 $b_{i,k} = E(0)$ for all $k \neq j$. Define $\mathbf{b}_1 = (b_{1,1}, \ldots, b_{1,m})$. 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. With $m = N^{1/\ell}$ and $\ell = O(\sqrt{\log N})$, the protocol has total computational complexity $O(N\sqrt{\log N})$ and total communication complexity $2^{O(\sqrt{\log N})}$. 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 $\mathsf{SPIR}^{\delta}_{m}$.

Private table read/write protocols. Initially, party A holds i_A and $L_A[1...n]$, and party B holds i_B and $L_B[1...n]$. In other words, parties share the index $i = i_A \oplus i_B$, and the array L such that $L[j] = L_A[j] \oplus L_B[j]$ for $1 \leq j \leq n$. For our applications, we assume that the indices and elements of the array are k-bit integers, for some k. Naor and Nissim [21] design protocols for both reading and writing to the table in this setting, requiring $O(\mathrm{polylog}(n))$ communication. Their protocols use an *oblivious transfer* protocol as their main building block. The read protocol returns $R \oplus L[i]$ to A and R to B, where R is a random k-bit integer. In the oblivious write protocol (writing a shared value v), each party will obtain new shares of L such that $L[i] = v$.

Yao's garbled circuit protocol. *Yao's garbled circuit* protocol [25] is the first general purpose secure two-party protocol. In this protocol, parties compute a

functionality using the circuit for that functionality. Please see [19] for a detailed description of Yao's protocol. The protocol runs in a constant number of rounds, and has a communication and computation complexity that is linear in the size of the circuit. We use Yao's garbled circuit to design portions of our protocols. Therefore, we sometimes need to switch from a different setting to Yao's garbled circuit setting, and back to the original one. Specifically, in Section 5, we need to switch to Yao's garbled setting from a setting where inputs are shared using XOR sharing, and have the parties share the final output of the circuit using an XOR sharing. This can be done by adding small additional circuitry to the original circuit. This additional circuitry will not affect the complexity of the circuit size or the protocol.

3 Stable Marriage Algorithms

In this section, we will first briefly describe the Gale-Shapley algorithm, and then take a closer look at Golle's variant of Gale-Shapley. We explain why the complexity of this variant is a factor of n more than what was claimed in [12]. Finally, we design our own variant of the Gale-Shapley algorithm, in which we avoid the factor of n increase in the complexity while preserving the useful properties we need for a secure implementation. This new variant is what we use in Sections 4 and 5 to design more efficient *private stable matching protocols*.

3.1 The Gale-Shapley algorithm

We review the well-known algorithm of Gale and Shapley [9], not only because of its general importance but because the private stable matching protocols presented later are, in fact, simulations of variants of the Gale-Shapley algorithm.

The Gale-Shapley algorithm considers a series of proposals made by men, round-by-round. Whenever a proposal is accepted, the couple is considered engaged. If a man is not engaged, he is considered free. The algorithm proceeds as follows. If there are any free men, select one at random (call him A). A proposes to the woman he ranks highest among those to whom he has not yet proposed (call her B). If B is free, she accepts and the pair are considered engaged. If B is engaged to some A' and she ranks A' ahead of A, then B and A' remain engaged and A remains free. If B is engaged to A' and she ranks A' below A, then B and A become engaged and A' becomes free.

After $O(n^2)$ proposals, all participants will be engaged and we will have found a stable marriage. In fact, the marriage we find is men-optimal. Due to symmetry, it is clear we could run the algorithm to find a marriage that is women-optimal. For more on the Gale-Shapley algorithm, the interested reader

is referred to the treatment of the subject by Gusfield and Irving [14]. One important note, however, is that by observing the proposals, acceptances, and rejections round-by-round, one can (for some problem instances) reconstruct the entire preference lists of all participants.

3.2 Golle's variant of Gale-Shapley

In this section we describe Golle's variant of the classic Gale-Shapley algorithm, explain its suitability for implementation as a private stable matching protocol, and present our new complexity analysis. Consider Gale-Shapley's algorithm where there are n real women $B_1, \ldots, B_n$, and n real men $A_1, \ldots, A_n$. In Golle's variant, n "fake" men, $A_{n+1}, \ldots, A_{2n}$ are introduced (no fake women are defined). The preferences of fake men are not important, and can be chosen arbitrarily. The preferences of women need to be augmented to account for the fake men introduced. It is only important that each woman ranks the fake men lower than any real ones.

In what follows, let $\mathcal{F}_k$ and $\mathcal{E}_k$ denote the sets of free and engaged men in round k of the algorithm, respectively. Initially, all the real men are free and all the fake men are engaged (in an arbitrary way). The algorithm proceeds as follows:

For $k = 1$ to R:

While $\mathcal{F}_k$ is non-empty:
- Randomly select a man A from $\mathcal{F}_k$.
- A proposes to woman B, the woman he ranks highest among the women to whom he has never proposed before.
- Note that woman B is always already engaged to some man, A'. This is resolved in the following manner.
  * If B ranks A higher than A', she breaks her engagement to A' and becomes engaged to A. Man A is removed from set $\mathcal{F}_k$ and added to $\mathcal{E}_k$, whereas man A' is removed from $\mathcal{E}_k$ and added to $\mathcal{F}_{k+1}$.
  * If B ranks A behind A', she stays engaged to A'. Man A is removed from set $\mathcal{F}_k$ and added to set $\mathcal{F}_{k+1}$.
- When $\mathcal{F}_k$ is empty, let $\mathcal{E}_{k+1} = \mathcal{E}_k$.

Note that the above algorithm has some nice properties for designing a secure stable matching protocol. For instance, all n women are always engaged to some man. During round k, the number of engaged men is always $|\mathcal{E}_k| = n$. 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.

The algorithm, ends after R iterations. In [12], it is claimed that $R = n$ is enough to reach a stable matching. We observe that this is not the case, and for some inputs, $\Omega(n^2)$ iterations are necessary to achieve a stable matching. This implies that proposition 1, as stated in [12], is incorrect. A problem instance explored in Gusfield and Irving [14, pg15] is one such counterexample, and is presented in Appendix A.

The intuition behind this inefficiency is that, for some inputs, there may be many rounds where most of the proposals are made by fake men. Such proposals do not help the real men move forward in their preference lists, and hence do not help them reach a stable matching. This observation implies a factor of n increase in Golle's algorithm, and the same increase in the communication complexity of his privacy preserving stable matching protocol.

Claim. The algorithm of Section 3.2 (with $R = n^2$) produces a stable matching among the n men and n women. That is, the algorithm is correct. This claim replaces Proposition 1 of [12].

Proof. After $n^2 - n + 1$ proposals from real men, we will have a stable marriage (from the same counting argument used to show the correctness of the Gale-Shapley algorithm). Until a stable marriage is reached, some real man will be free. So, after n^2 rounds, real men will have made at least n^2 proposals and, thus, the algorithm outputs a stable marriage. The minimal number of rounds required for correctness is less, but is $\Omega(n^2)$ (see Appendix A).

3.3 Our new variant of Gale-Shapley

Here, we describe our variant of Gale-Shapley which improves on the complexity of Golle's variant by a factor of n and which also has nice properties for designing a private matching protocol.

Once again, as with Gale-Shapley's algorithm, there are n real men and n real women with their corresponding preference lists. We add n fake men and n fake women to this setting (note that Golle's variant did not include fake women). Thus, we have: real men $\{A_1, \ldots, A_n\}$, fake men $\{A_{n+1}, \ldots, A_{2n}\}$, real women $\{B_1, \ldots, B_n\}$, and fake women $\{B_{n+1}, \ldots, B_{2n}\}$. Preference lists are adjusted in the following way.

	Preference lists
Real men	([actual preference list], [$B_{n+2}, \ldots, B_{2n}$, in any order])
Fake men	([$B_{n+2}, \ldots, B_{2n}$, in any order], B_{n+1}, [$B_1, \ldots, B_n$, in any order])
Real women	([actual preference list], [$A_{n+1}, \ldots, A_{2n}$, in any order])
Fake women	([$A_{n+1}, \ldots, A_{2n}$, in any order], [$A_1, \ldots, A_n$, in any order])

As before, set $\mathcal{F}_k$ contains the free men in round k. The algorithm works as follows.

Initialization:

- $\mathcal{F}_1 = \{A_1\}$ (man A_1 is free).
- $\{A_2, \ldots, A_n\}$ are engaged to $\{B_{n+2}, \ldots, B_{2n}\}$, respectively.
- $\{A_{n+1}, \ldots, A_{2n}\}$ are engaged to $\{B_1, \ldots, B_n\}$, respectively.

For $k = 1$ to R:

- The free man A in $\mathcal{F}_k$ proposes to B, the next woman in his preference list to whom he has not yet proposed.
- Let A' denote the man to whom B is already engaged.
  * If B ranks A higher than A', she breaks her engagement to A' and becomes engaged to A. Let $\mathcal{F}_{k+1} = \{A'\}$.
  * If B ranks A lower than A', she stays engaged to A'. Let $\mathcal{F}_{k+1} = \{A\}$.

Note that in each iteration, exactly one proposal is made. The number of free men in each round is $|\mathcal{F}_k| = 1$. As we will show next, the above algorithm reaches a stable matching in at most $2n^2$ iterations. In the matching reached, all the real men are engaged to real women and all the fake men to fake women.

Claim. Once a fake man proposes to fake woman B_{n+1}, we have reached a stable matching. In this stable matching, real men are engaged to real women and fake men are engaged to fake women.

Proof. Note that woman B_{n+1} is the n^{th} preference of all the fake men. Therefore, when a fake man proposes to B_{n+1}, he has already proposed to the other $n-1$ fake women in his list and has been rejected by them at some point during the execution of the protocol. This implies that all the other $n-1$ fake women were or became engaged to other fake men (fake women rank fake men higher than real men). This, in turn, implies that all the real women are engaged to real men.

The argument for having reached a stable matching is similar to the one for the original Gale-Shapley algorithm. Particularly, lets assume that real man A prefers woman B to woman B', to whom he is engaged. Then, B must have rejected A at some point during the execution. But, this implies that B was or became engaged to a man she prefers to A. So B cannot prefer A to her current match. This further implies that there are no unstable matches.

It is easy to verify that before $2n^2$ proposals, at least one fake man will have proposed to B_{n+1}. Therefore, based on the above claim, we reach a stable matching in at most $R = 2n^2$ steps.

For the secure implementation of the above algorithm, it is useful to run the algorithm for the same number of rounds for all inputs (e.g. $R = 2n^2$). This will help us avoid leaking the number of proposals necessary to reach a stable matching for a specific input. But, note that in the above algorithm, once a fake man proposes to 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}. The algorithm runs exactly as before and once a fake man proposes to woman B_{n+1}, the same claims as above are true. The only advantage is that we will always have a free man who will propose next. This is a useful invariant for the secure implementation we give in Section 4.2.

4 Privacy preserving Stable Marriage Protocols

In this section, we present the privacy preserving implementation of the two Gale-Shapley variants presented earlier: one for Golle's (modified) variant in Section 4.1, and one for our new variant in Section 4.2.

4.1 Privacy preserving protocol for Golle's variant of Gale-Shapley

In this section, we briefly present the implementation of Golle's (modified) variant of Gale-Shapley as a private stable matching protocol. This section will also provide a basis for comparison with the secure variant in Section 4.2.

Protocols for the implementation. The following are used in the secure implementation of Golle's variant given at the end of this section. Many will be of use later, in Section 4.2. Slight modifications to some protocols were necessary due to the observations from Section 3.2.

Notation. Let $r_{i,j} \in \{0, \ldots, n-1\}$ be the rank given to real woman B_j by man A_i. Let $s_{j,i} \in \{0, \ldots, n-1\}$ be the rank given to real man A_i by woman B_j. Our convention is that the highest possible rank is 0, and the lowest is $n-1$.

Bids. Define the (free) *bid* for man A_i as $W_i = \langle E(i), \mathbf{a}_i, \mathbf{v}_i, \mathbf{q}_i, E(\rho)\rangle$, where $\mathbf{a}_i = (E(r_{i,1}), \ldots, E(r_{i,n}))$, $\mathbf{q}_i = (E(s_{1,i}), \ldots, E(s_{n,i}))$, and $\mathbf{v}_i = (E(1), \ldots, E(n))$. Initially, $\rho = 0$.

Engaged Bids. The *engaged bid* $\langle W_i, E(j), E(s_{j,i})\rangle$ denotes that man A_i is engaged to woman B_j. Let $\mathcal{F}_k$ and $\mathcal{E}_k$ denote the sets of free and engaged bids in round k of the algorithm, respectively. When we *mix the bids*, the t matching authorities mix each of these sets separately.

Input submission and Initialization. Each man A_i initially sends his preference list $\langle E(r_{1,i}), \ldots, E(r_{n,i})\rangle$ and each woman B_j sends her list $\langle E(s_{j,1}), \ldots, E(s_{j,n})\rangle$ to the matching authorities. The matching authorities jointly create the preferences for the fake men $A_{n+1}, \ldots, A_{2n}$ and augment the women's preference lists with the fake men. The matching authorities generate the $2n$ bids for $A_1, \ldots, A_{2n}$ and the n engaged bids to denote the engagement of A_{n+j} to woman B_j. We add the n engaged bids to $\mathcal{E}_1$, and the n free bids to $\mathcal{F}_1$.

Breaking an engagement. Let $\langle W_i, E(j), E(s_{j,i})\rangle$ 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 incrementing it by the value $1 - b$, where $E(b) = \mathsf{EEQTEST}(E(\rho), E(n-1))$, using Paillier's additive homomorphism (ie, we multiply $E(\rho)$ by $E(1)/E(b)$). That is, we obliviously increment the next desired rank ρ when it is less than $n - 1$ and, otherwise, we do not. This is a modification from the presentation in [12]. If we did not increment safely, the new n^2 loop bound generates the possibility that we may increment some man's ρ more than n times which would lead, in a sense, to a pointer error.

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

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

Summary of the privacy preserving implementation The following algorithm, with $R = n^2$, summarizes the secure implementation of Golle's variant

of Gale-Shapley. How to process the submitted inputs, initialize the data structures, find a conflicting bid, resolve the conflict, and break an engagement are explained in Section 4.1. We have, however, omitted the details of some steps, such as internal bid mixing, opening a bid , and some others. We refer the reader to Golle's paper [12] for those details we have omitted.

Briefly, when a bid W_i is "opened," the matching authorities jointly determine $E(j)$ (the woman at rank ρ on man A_i's preference list) and her preference $E(s_{j,i})$ for A_i, without learning anything about ρ, j or i.

Input submission and Initialization
For $k = 1$ to R:

While F_k is non-empty:

1. Randomly select a bid W_i from $\mathcal{F}_k$.
2. Open W_i to recover $E(j), E(s_{j,i})$
3. Form the engaged bid $\langle W_i, E(j), E(s_{j,i})\rangle$
4. Find the conflicting engaged bid, $\langle W_{i'}, Ej, E(s_{j,i'})\rangle$
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 $\mathcal{F}_{k+1}$
8. Add the winner to $\mathcal{E}_k$
9. Mix all the bids, and perform internal bid mixing
10. If $\mathcal{F}_k$ is empty, let $\mathcal{E}_{k+1} = \mathcal{E}_k$

Announce stable marriage

At step $k = n^2$, all data is discarded, save the ciphertext pairs $(E(i), E(j))$ from each engaged bid in $\mathcal{E}_{n^2}$. These are (publicly or privately) announced to each participant.

Complexity analysis. The work and communication complexity is dominated by running the 3 re-encryption mix networks in steps 4, 5, and 9 — specifically, the mixnet in step 9. This re-encryption mix network is run on $2n$ bids, n times each round (since $|\mathcal{F}_k| = n$ at the start of each round). Furthermore, each bid contains $O(n)$ ciphertexts. Thus, each of the t authorities does $O(n^5)$ total work. The total communication complexity is $O(tn^5)$ ciphertexts. This differs from Golle's $O(n^3)$ analysis in [12], which claims the bid size to be constant, claims $R = n$ instead of $R = n^2$, and omits t as a factor impacting the number of messages passed. For similar reasons, the round complexity is now $O(n^3\text{polylog}(n))$, instead of $O(n^2\text{polylog}(n))$ as claimed in [12].

Claim. The protocol of Section 4.1 is a private stable matching protocol, assuming Paillier encryption is semantically secure and the underlying re-encryption mix network is private. When t matching authorities participate, the protocol's total communication complexity is $O(tn^5)$ ciphertexts. This claim replaces Propositions 2 and 3 of [12].

Proof (sketch). The correctness of the algorithm from Claim 3.2 shows the protocol outputs a stable matching. To show the protocol is private, we direct the reader to the proof sketch of Proposition 3 in [12]; our modifications to the protocol do not impact the proof that a passive adversary learns no additional information during the protocol's execution. The complexity analysis is shown above.

4.2 Privacy preserving protocol for our new variant of Gale-Shapley

To implement our new variant of Gale-Shapley securely, we must modify the initialization procedure of Golle's secure protocol to accommodate the addition of fake women. Furthermore, we present a new procedure to open a bid with the aid of a database that holds the participants' encrypted preference lists. By removing the preference lists from the bids themselves, we make our bids constant-sized. Now, we define a bid W_i for man A_i as $\langle E(i), E(\rho)\rangle$. We assume one of the t matching authorities plays the role of the database.

Protocols for the implementation. From Section 4.1, the definition for an engaged bid and the procedures for finding a conflict, breaking an engagement, and resolving a conflict remain the same for the secure implementation of the new variant of Gale-Shapley given at the end of this section. The following procedures are also used.

Input submission. As before, each woman sends her preference list $\mathbf{q}_i$. Similarly, each man submits a vector $\mathbf{a}_i$, but the vector encodes his preference list in a new way. Now, man A_i defines $\mathbf{a}_i = (E(a_{i,1}), \ldots, E(a_{i,n}))$, where $a_{i,j} \in \{1, \ldots, n\}$ is the index of the woman to whom he has given rank $j - 1$.

Initialization. The matching authorities generate the free bid for man A_1 and the engaged bids for man A_i, for $i \neq 1$. The preference lists for the $n + 1$ fake men and n fake women are generated, and the preference lists for the real men and women are augmented, according to the rules above. Let one matching authority collect and organize these lists, and call this authority the database δ. Let $\delta = [(\mathbf{a}_1, \mathbf{q}_1), \ldots, (\mathbf{a}_{2n}, \mathbf{q}_{2n})]$. Thus $\delta[4n(i-1) + (j-1)] = E(a_{i,j})$ and $\delta[4n(i-1) + (j-1) + 2n] = E(s_{j,i})$, for $i, j \leq 2n$.

Open a bid. Given a free bid $\langle E(i), E(\rho)\rangle$, we must recover $E(j)$ (the encrypted index of the woman at rank ρ on man A_i's preference list) and $E(s_{j,i})$. 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 additive homomorphism, given $E(i)$ and $E(\rho)$. We can recover $E(j)$ by accessing the database at this secret index, using the protocols 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 the database at a secret index. Given $E(x)$, we can generate a series of indices $\mathbf{b}_1, \ldots, \mathbf{b}_\ell$ which singulate the element at index x using the index conversion procedure below, without learning anything about index x. Then let $y = \mathsf{SPIR}^{\delta}_{m}(\mathbf{b}_1, \ldots, \mathbf{b}_\ell)$. We jointly decrypt y, ℓ times, to recover $\delta[x]$. For the values of m and ℓ indicated in Section 2.3, this joint decryption takes $O(\sqrt{\log n})$ rounds, passing a $2^{O(\sqrt{\log n})}/2^i$ size message between t authorities on round i, yielding a total communication complexity of $o(tn)$. After this procedure, the entire database should re-encrypt all of its entries. The total computational complexity of this database access is $O(n^2\sqrt{\log n})$.

Secure index conversion. Given $E(x)$, we can securely calculate the indices $\mathbf{b}_1, \ldots, \mathbf{b}_\ell$ that are used as input to the protocol $\mathsf{SPIR}^{\delta}_{m}$. Recall that $\mathbf{b}_k = (b_{k,1}, \ldots, b_{k,m})$ 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

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

Thus, $\mathbf{b}_k$ can be calculated using MOD, $\mathsf{EEQTEST}$, and the vectors $\mathbf{b}_j$ for $j < k$ calculated in earlier rounds. Each round, this procedure takes polylog work with polylog communication complexity. The procedure ends after $\ell = O(\sqrt{\log n})$ rounds.

Full privacy preserving implementation The secure implementation of our new variant of Gale-Shapley is assembled using the protocols indicated above, according to the algorithm below.

Input submission and Initialization
For $k = 1$ to $2n^2$:

1. Select the single free bid W_i from $\mathcal{F}_k$.
2. Open W_i to recover $E(j), E(s_{j,i})$
3. Form the engaged bid $\langle W_i, E(j), E(s_{j,i})\rangle$
4. Find the conflicting engaged bid, $\langle W_{i'}, Ej, E(s_{j,i'})\rangle$
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 $\mathcal{F}_{k+1}$
8. Add the winner to $\mathcal{E}_k$
9. Mix the engaged bids
10. Let $\mathcal{E}_{k+1} = \mathcal{E}_k$

Announce stable marriage

Complexity analysis. As in Golle's, the communication complexity here is dominated by the re-encryption mix networks run in steps 4, 5, and 9 — specifically, the mixnets run in steps 4 and 9. These mixnets runs on $O(n)$ ciphertexts each round. The total communication complexity is $O(tn^3)$. Accessing the database, however, dominates the computational cost, when $t \leq n$. Each step k, the database access takes $O(n^2\sqrt{\log n})$ work. The total computational complexity is $O(n^4\sqrt{\log n})$. The round complexity is $O(n^2\mathrm{polylog}(n))$.

Claim. The algorithm of Section 4.2 is a private stable matching protocol, assuming Paillier encryption is semantically secure and the underlying re-encryption mix network is private.

We note the above claim can be proven by a hybrid argument similar to that of the proof of Proposition 3 in [12], or using the composition theorems mentioned in Section 2.2. Due to length constraints (and the lack of novelty in applying either of these proof techniques), we omit a full proof.

5 An efficient private stable matching protocol for $t = 2$ MAs

In this section, we take a closer look at the case where there are only two Matching Authorities (MAs). We design a secure protocol for this case with $O(n^2\mathrm{polylog}(n))$ communication complexity. This is a factor of n more efficient than our protocols for the general case. We generalize this protocol for the setting with multiple pairs of MAs in Appendix B.

We base our secure implementation on our variant of Gale-Shapley from Section 3.3. To do so, we use rather different techniques from those we used in

the general case. As before, each participant sends shares of his/her input to the two MAs. The rest of the protocol is performed by the two MAs without help from the participants. The final matching is then revealed to the participants. Before we proceed with the details of the protocol, let us define the data structures that are shared by the MAs. For simplicity, in what follows we label men and women using only their index.

$A[i][j] = a_{i,j}$,	the identity of the woman ranked $j-1$ by man A_i.
$B[j][i] = s_{j,i} + 1$,	where $s_{j,i} \in [0, n-1]$ is the rank given to man A_i by woman B_j.
$P[i] = \rho_i + 1$,	where $\rho_i \in [0, n-1]$ is the rank of the woman to whom man A_i will propose next.
$E[j] \in \{1, \ldots, n\}$,	the identity of the man engaged to woman B_j.

Using the above data structures, we can rewrite our variant of the Gale-Shapley algorithm (from Section 3.3), after the initialization stage, as follows.

For $k = 1$ to $2n^2$

1. Remove i from $\mathcal{F}_k$
2. Let $p = P[i]$
3. Let $j = A[i][p]$ (the index of the woman to whom A_i proposes)
4. Let $i' = E[j]$ (the index of the man currently engaged to her)
5. Let $s_{j,i} = B[j][i]$ and $s_{j,i'} = B[j][i']$ (her rankings for A_i and $A_{i'}$)
6. If $s_{j,i'} > s_{j,i}$, then swap the labels i, i'
7. $E[j] \leftarrow i'$ (store the "winner" as her husband)
8. $p' = P[i]$
9. $P[i] \leftarrow p' + 1$ (increment the "loser")
10. $\mathcal{F}_{k+1} = \{i\}$ (free the "loser")

At the end of the protocol, $E[j]$ stores the index of the man to whom woman B_j is married. The MAs can privately (or publicly) announce to each participant shares of his/her partner. Now, we explain how to implement the above algorithm securely. Note that all the data structures and intermediate values in the algorithm are shared between the two MAs. To be compatible with the private table access primitives we use, we employ a simple XOR sharing scheme: to share a k-bit integer a between the MAs, a participant sends a random k-bit r to one MA and sends $a \oplus r$ to the other MA.

In steps 2–5 and 7–9, MAs need to privately read and write to a table. We can use the techniques of Naor and Nissim [21] to implement these steps securely (see Section 2.3 for more detail), with $O(\mathrm{polylog}(n))$ communication. We can do step 6 (compare two integers and potentially swapping them) and

step 9 (computing shares of $\rho + 1$ from shares of ρ) by switching to Yao's garbled circuit protocol and then switching back to the initial setting (see Section 2.3 for more detail). The circuit for performing such computations is of size $O(\mathrm{polylog}(n))$. This leads to a protocol with $O(n^2\mathrm{polylog}(n))$ communication between the MAs.

According to the composition theorems with respect to passive adversaries (see Section 2.2), the above protocol is privacy-preserving as long as the underlying subprotocols (private table read/write protocols and Yao's garbled circuit protocol) are secure against passive adversaries.

6 Conclusion

In conclusion, we have given new protocols (and new analyses) for stable matching with security against a passive adversary. Our protocols can be adapted to provide security against an active adversary using standard compilation techniques [11, 10]. An interesting open question is to gain some protection against malicious faults without a huge increase in communication complexity. One intriguing possibility along these lines would be to exploit a kind of "double entry bookkeeping" in our variant of Gale-Shapley. If all women rank the fake man A_{2n+1} last, then the protocol outputs a stable marriage among the n real men and n real women, *and* among the n fake men and n fake women. By examining the fake marriages and the preference lists of the fake players, evidence of misbehavior might be suggested (e.g., if the fake marriages are not stable).

References

1. Joy Algesheimer, Jan Camenisch, and Victor Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology – Proceedings of CRYPTO'02*, number 2442 in Lecture Notes in Computer Science, pages 417–432, 2002.
2. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *Proceedings of the 22nd ACM Symposium on Theory of Computing*, pages 503–513, 1990.
3. Ran Canetti. Security and composition of multiparty cryptographic protocols. In *Journal of Cryptology*, volume 13, pages 143–202, 2000.
4. Ronald Cramer and Ivan Damgård. Secure distributed linear algebra in a constant number of rounds. In *Advances in Cryptology – Proceedings of CRYPTO'01*, number 2139 in Lecture Notes in Computer Science, pages 119–136, 2001.
5. Ivan Damgård, 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. `http://eprint.iacr.org/`.
6. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

7. Pierre-Alain Fouque, G Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *Financial Crypto (FC '00)*, 2000.
8. Matthew Franklin, Mark Gondree, and Payman Mohassel. Improved efficiency for private stable matching. In *The Cryptographer's Track at RSA Conference (CT-RSA)*, 2007.
9. David Gale and Lloyd Stowell Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.
10. Oded Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.
11. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 218–229, 1987.
12. Philippe Golle. A private stable matching algorithm. In *Financial Crypto (FC '06)*, 2006.
13. Philippe Golle and Ari Juels. Parallel mixing. In *ACM Conference on Computer and Communications Security'04*, pages 220–226, 2004.
14. Dan Gusfield and Robert Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.
15. Markus Jakobsson, Ari Juels, and Ron Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of USENIX'02*, pages 339–353, 2002.
16. Markus Jakobsson and Claus Peter Schnorr. Efficient oblivious proofs of correct exponentiation. In *Communications and Multimedia Security*, pages 71–86, 1999.
17. Eike Kiltz. Unconditionally secure constant round multi-party computation for equality, comparison, bits and exponentiation. Cryptology ePrint Archive, Report 2005/066, 2005. `http://eprint.iacr.org/`.
18. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, pages 364–373, 1997.
19. Yehuda Lindell and Benny Pinkas. A proof of Yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175, 2004. `http://eprint.iacr.org/`.
20. Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology – ASIACRYPT 2003*, number 2894 in Lecture Notes in Computer Science, pages 416–433, 2003.
21. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *Proceedings of the 33rd ACM Symposium on Theory of Computing*, pages 590–599, 2001.
22. C. Andrew Neff. A verifiable secret shuffe and its application to e-voting. In *ACM Conference on Computer and Communications Security'01*, pages 116–125, 2001.
23. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – Proceedings of Eurocrypt'99*, number 1592 in Lecture Notes in Computer Science, pages 223–238, 1999.
24. Julien P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *Advances in Cryptology – ASIACRYPT'98*, number 1514 in Lecture Notes in Computer Science, pages 357–371, 1998.
25. Andrew C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Symposium on Foundations of Computer Science*, pages 162–167, 1986.

A A case where [12] requires more than n iterations

Here we give an example input for which Golle's variant of Gale-Shapley requires $\Omega(n^2)$ iterations of the main loop to reach a stable matching. Consider the following preference lists for the real men and women (the preference lists of the n fake men are as indicated in [12]):

A_1	$[1, 2, \ldots, n-1, n]$	B_1	$[2, 3, \ldots, n, 1]$
A_2	$[2, 3, \ldots, 1, n]$	B_2	$[3, 4, \ldots, 1, 2]$
	$\ldots$		$\ldots$
A_{n-1}	$[n-1, 1, \ldots, n-2, n]$	B_{n-1}	$[n, 1, \ldots, n-2, n-1]$
A_n	$[1, 2, \ldots, n-1, n]$	B_n	$[1, 2, \ldots, n-1, n]$

After the first iteration of the main loop, $n-1$ real men are engaged and 1 real man remains free. This implies that $\mathcal{F}_2$ will include $n-1$ fake men. In the next $n^2 - 2n + 2$ iterations, one real man will get engaged and another will become free. In other words, for $2 \leq k \leq n^2 - 2n + 2$, only one real man will propose in each iteration and all the other proposals are made by fake men. Thus, $\Omega(n^2)$ iterations of the main loop is necessary to reach a stable matching.

B A generalization of Section 5 to multiple pairs of MAs

We briefly describe a simple way of generalizing the protocol for the case of two matching authorities to a communication network with an arbitrary number of pairs of authorities.

Setting. We will have $t = 2t'$ MAs. The MAs are paired up such that there are t' pairs. Each participant (man or woman) is assigned to exactly one pair of MAs. This assignment can be chosen by the participants, by the MAs, or can be a predetermined assignment. A participant sends shares of his/her preference list to the pair to whom he/she is assigned, and no others. We assume that the assignments are publicly revealed, or are at least known to all the MAs.

The data structures $A[\,][\,], B[\,][\,], E[\,], P[\,]$ are split between the pairs such that each pair only holds shares of the portion of each data structure corresponding to the participants assigned to him. The exception is the free man in $\mathcal{F}_k$. The index of this free man is shared between the t MAs, so that each MA has a share. Hence, we have two different types of sharing: one between a pair of MAs, and one between all t MAs. We can use XOR sharing for both cases. We also add special dummy locations to each data structure. For instance, we add the dummy row $A[2n+2][\,]$ to matrix A. Similarly, we add $B[2n+2][\,], E[2n+2]$, and $P[2n+2]$ as dummy locations. The value stored in these dummy locations is 0, and will stay 0 throughout the protocol.

Secure Implementation. Consider the algorithm described in Section 5. We will show how to perform the steps of this algorithm securely, using a communication network made of t' pairs of MAs:

In Step 1, each MAs removes her share of the element i in $\mathcal{F}_k$. MAs securely compute a multiparty circuit that takes the t shares of i as input and computes

the t shares as the output. The circuit returns shares of the value $(2n + 2)$ for $t' - 1$ pairs of MAs, and returns shares of the adjusted index[1] of i for the two MAs who were assigned the man A_i. A similar circuit protocol is run at the end of steps 2–4 and 8, to return shares of the adjusted index for the following step.

In steps 2–5 and 8, each pair of MAs will try to read from their portion of the data structure using the shares of the adjusted index they received as part of a multiparty circuit protocol. Only one pair will return shares of the actual value, while all the other pairs will return shares of 0 (which they obliviously read from a dummy location). This requires $O(\text{polylog}(n))$ communication between each pair of MAs using the private table read protocol, leading to a communication complexity of $O(t\text{polylog}(n))$.

In steps 7 and 9, each pair will try to write to their local data structure. Only one pair will write a correct value to its actual location. All other pairs will write 0, to a dummy location. Similar to the read operation, this also requires $O(t\text{polylog}(n))$ communication.

In step 6 and 9, the t MAs need to jointly compute a circuit on their inputs.

All the circuits computed throughout the protocol have size $O(t\text{polylog}(n))$ gates. We can use the techniques of [2] (a generalization of Yao's garbled circuit to the multiparty setting) to implement these circuits securely. This requires $O(t^2\text{polylog}(n))$ communication for each circuit implemented, and leads to a total communication complexity of $O(t^2n^2\text{polylog}(n))$ among the MAs for the whole protocol.

We do not study the adversarial structure for this communication network, but would like mention that this generalization has the following advantages over the 2-MA case. (1) The computation is now distributed among the t MAs instead of only 2 MAs and (2) even if a pair of MAs collude, they only learn a portion of private data as opposed to all of it.

We claim that there may be many practical situations where a participant trusts at least 1 MA. Consider the application of matching medical residents and hospitals. Many universities have non-disclosure policies for their students' private records. Medical students might, then, be able to absolutely trust their campus matching authority with secrets, but have no reason to trust the matching authority representing the hospitals, the other medical schools, or some outside party. Thus, a model where a participant is able to consistently pairwise-share her inputs with an MA she trusts is very useful, especially when it means work and communication savings.

[1] The reason for adjusting the index is that each pair only holds portions of the data structure. Note that it is publicly known which portion each pair is holding.