M. Sadegh Riazi*, Ebrahim M. Songhori, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar

# Toward Practical Secure Stable Matching

**Abstract:** The Stable Matching (SM) algorithm has been deployed in many real-world scenarios including the National Residency Matching Program (NRMP) and financial applications such as matching of suppliers and consumers in capital markets. Since these applications typically involve highly sensitive information such as the underlying preference lists, their current implementations rely on trusted third parties. This paper introduces the first provably secure and scalable implementation of SM based on Yao's garbled circuit protocol and Oblivious RAM (ORAM). Our scheme can securely compute a stable match for 8k pairs four orders of magnitude faster than the previously best known method. We achieve this by introducing a compact and efficient sub-linear size circuit. We even further decrease the computation cost by three orders of magnitude by proposing a novel technique to avoid unnecessary iterations in the SM algorithm. We evaluate our implementation for several problem sizes and plan to publish it as open-source.

**Keywords:** privacy-preserving, stable matching, garbled circuit, secure function evaluation

DOI 10.1515/popets-2017-0005
Received 2016-05-31; revised 2016-09-01; accepted 2016-09-02.

## 1 Introduction

In Stable Matching (SM), there are two groups of individuals, e.g., men and women. Each individual ranks the members of the other group in a list sorted based on her preference. The goal is to assign the members of these two groups to each other while satisfying the following post-match condition: there shall be no pairs from the two groups such that they prefer each other more than their already assigned partners. SM has substantial real-world applications: The National Residency Matching Program (NRMP) matches around $32k$ graduating medical students to residency programs in the US every year [23, 33]. The New York City Department of Education (NYCDOE) matches over $90k$ entering students to public high schools [1]. Also there are many financial applications that require SM, such as vertical networks and their application in supply chains [25].

SM use cases typically involve sensitive preference lists which have to be kept private. The current practice to ensure data privacy in SM is by exposing the personal preferences to a third party server and relying on its trustworthiness to perform a secure matching. However, relying on the trusted third party might be unacceptable because there is still a high risk of information leakage and data abuse by the third party. Even if a third party server is indeed trustworthy, it can accidentally expose the user's private data in the event of a compromise. In addition to information leakage, multiple studies [9, 13] show that if certain individuals in a SM problem know the input of others, they could leverage this information to manipulate the results [29, 37]. [36] studied strategic issues in the SM model. They derived an optimal cheating strategy and showed that there is a possibility for a woman to misrepresent her input in order to achieve a more favorable assignment.

We can make the process of stable matching secure by utilizing Secure Function Evaluation (SFE) protocols. SFE allows to evaluate a function on private inputs from multiple parties where each party wants to keep her own inputs private. Today, the common drawback of SFE protocols is their low efficiency, mainly due to the communication, which prohibits the wide usage for real-world applications. Therefore, for SFE protocols to be widely adopted, it is crucial to devise methodologies that increase the efficiency.

The inefficiency problem becomes even more pronounced when one considers algorithms with intensive memory access and high complexity such as Dijkstra's algorithm, subtree matching, or substring search. This is due to the high cost of hiding memory content and access pattern required by SFE. A well-known example

***Corresponding Author: M. Sadegh Riazi:** University of California, San Diego, E-mail: mriazi@ucsd.edu
**Ebrahim M. Songhori:** Rice University, E-mail: ebrahim@rice.edu
**Ahmad-Reza Sadeghi:** TU Darmstadt, E-mail: ahmad.sadeghi@trust.cased.de
**Thomas Schneider:** TU Darmstadt, E-mail: thomas.schneider@crisp-da.de
**Farinaz Koushanfar:** University of California, San Diego, E-mail: farinaz@ucsd.edu

that involves processing sensitive data is the SM problem which has quadratic memory access complexity.

Golle proposed a privacy-preserving SM system based on Homomorphic Encryption [11]. Franklin et al. then improved this system and made it more efficient using an efficient multi-party indirect indexing [6, 7]. However, most of the previously proposed protocols for secure SM use a large number of expensive public-key operations and have not been implemented yet. The first solution based on symmetric key encryption is proposed by Keller et al. [15] (the previously best known method in terms of computation and communication complexity). They reported that their approach can solve secure SM for 8k pairs (1/4 of the size of the NRMP) in $1.5 \cdot 10^{12}$ seconds, i.e., almost 47 000 years! The work of [41] achieves a better runtime of 33 hours for set size 512 but it has higher computation and communication complexity which limits its scalability. (See Table 2 in Section 5 for a detailed comparison.)

This paper introduces the first scalable secure SM system. Our approach leverages a well-known SFE protocol, called Yao's Garbled Circuit (GC) [39], which is mainly based on efficient symmetric cryptographic operations. The input to this protocol is a Boolean circuit description of the function that needs to be evaluated securely. The total cost of the GC protocol is the total number of gates in the Boolean circuit. The conventional approaches in the GC protocol mainly relied on a combinational (directed acyclic) circuit description. In contrast, we utilize a recently proposed approach in [32] to describe the SM functionality as a compact sequential circuit. The size of the sequential circuit is less than the combinational circuit but it has to be evaluated for multiple iterations. Therefore, the total cost of the GC protocol is the product of the number of iterations and the size of the sequential circuit.

We achieve an unprecedented level of efficiency for solving secure SM by two sets of innovations. (i) We propose the first compact and efficient circuit with sub-linear size with respect to the number of pairs in SM ($n$). The size of our circuit is $\mathcal{O}(\log^3 n)$ which significantly improves over $\mathcal{O}(n^2 \log n)$ for the naïve sequential circuit. (ii) We present a novel technique, called *early termination*, to significantly decrease the computation time by reducing the total number of iterations that the SM sequential circuit needs to be evaluated. For example, for the set size $8k$, early termination results in three orders of magnitude improvement in computation time and communication (reducing the execution time from 5.62 years to 1.25 days). Although our focus throughout the paper is on the SM algorithm, our methodology can be applied to other memory intensive algorithms by adapting our novel early termination technique and creating a sub-linear sequential circuit.

### 1.1 Contributions

Here, we briefly list our contributions:

- We introduce the first feasible, scalable, and efficient secure SM for real-world set sizes.
- We design the first sequential circuit for the SM algorithm for Yao's GC protocol. The compactness achieved by the sequential description (as opposed to prior combinational one) reduces the circuit size from $\mathcal{O}(n^4 \log n)$ to $\mathcal{O}(n^2 \log n)$.
- We design the first sub-linear size circuit (w.r.t. the SM set size). This circuit achieves unprecedented computation and communication efficiency compared to the prior art by integrating sub-linear ORAM and various memory access strategies.
- We introduce mathematical and statistical methodologies for early protocol termination which allows us to trade-off between security and efficiency.
- We demonstrate a proof-of-concept implementation of our approach for different secure SM settings with various set sizes. We benchmark the state-of-the-art sub-linear ORAM schemes and report the best solution for various ranges of set sizes.

### 1.2 Paper Organization

In Section 2, we give a summary of our approach. In Section 3, we provide the formal description of the SM problem and summarize Yao's GC protocol which we use for our system. Details of our proposed circuits used in the GC protocol and their variations are discussed in Section 4. In Section 5, we present a survey of the related literature and describe the differences/new aspects of our approach compared to the earlier works. Comprehensive results and timing performance are given in Section 6. Section 7 discusses the trade-off between privacy and performance. Finally, we conclude in Section 8.

## 2 High Level Architecture

We implement a secure stable matching system based on Yao's GC protocol [39] which is an efficient method for secure function evaluation between two parties. Secure SM is inherently a multiparty SFE problem where

multiple parties provide their inputs. However, we use a known technique based on XOR-secret-sharing that translates this problem into two-party SFE.

In the GC protocol, the underlying function has to described as a Boolean circuit. The input of each party is an input to this circuit. Then two parties, called Garbler and Evaluator, run the GC protocol and find the results. Implementing the entire SM as a acyclic Boolean circuit (combinational) is neither practicable nor scalable and hence no one has tried to implement it so far. In our work, we design a compact sequential circuit which enables an efficient implementation of the SM algorithm.

Here, we explain how two proxies (the Garbler and Evaluator) are able to securely perform the SM for multiple parties using the GC protocol (see [22] for details). For each input bit $I$, each party does the following:

1. Generate a random mask $I_A$.
2. Compute $I_B = I_A \oplus I$.
3. Send $I_A$ to Garbler and $I_B$ to Evaluator.

The SM circuit is extended by one layer of XOR gates, one gate for each input bit. Each XOR gate receives $I_A$ and $I_B$ as inputs, provided by Garbler and Evaluator respectively, and computes $I = I_A \oplus I_B$ as output. The outputs of these gates are the inputs to the main SM circuit.

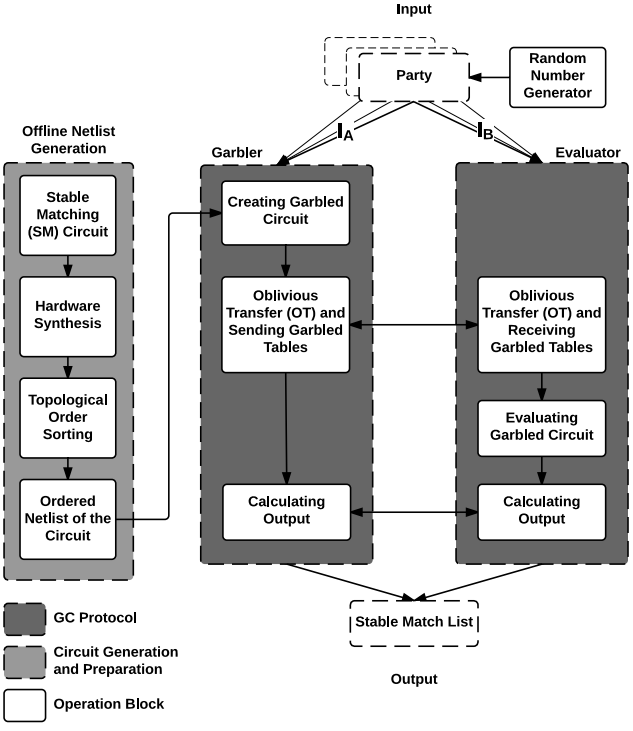


**Fig. 1.** Global flow of our secure SM system.

Figure 1 shows the global flow of our system which consists of two parts. First, the offline pre-processing and then the online execution which preforms the GC protocol. In the first part we synthesize the circuit, generate the netlist and sort the gates topologically. This ordered netlist is given to the Garbler to generate the garbled circuit for the online phase. In the online phase, the constant-round GC protocol is executed: the Evaluator gets all information that she needs to evaluate the garbled circuit. At the end, both Garbler and Evaluator share the results to find the outputs and deliver them to all parties.

The SM algorithm is considered to be secure if it outputs a stable match without revealing any additional information about preference list except what can be inferred from the output. Our scheme is secure against honest-but-curious adversaries. Our system realizes secure SM as it takes the encrypted inputs from each party and securely computes a stable match. Our methodology is secure as it relies on the security of the GC protocol which has been proven to be secure against honest-but-curious adversary in [18]; getting inputs from each party with the XOR-sharing technique is secure because Garbler and Evaluator see only random numbers from which they cannot infer any information [22]. The output is a correct stable match since our circuit for SM implements the Gale and Shapley algorithm that was proven to correctly compute a stable match [8].

It is worth mentioning that in our method, users do not have to install any cryptographic libraries or do complicated tasks. Each individual only needs to generate a random bit string and XOR it with her input. She then only needs to send the random string to one server and the XORed version to the other.

# 3 Background

## 3.1 Problem Statement and Notations

In this section, we illustrate the detailed description of the SM problem. A number of researchers have focused on addressing the SM problem, but Gale and Shapley [8] were the first to formalize the SM algorithm. They centered their work on the special case of the marriage problem. In this case, there is a set of men and a set of women. They introduced an algorithm which resulted in stable marriage. Gale and Shapley also proved that the stable match always exists. However, they showed that there can be more than one stable assignment and so the

stable matching is not a unique assignment. Roth [29] demonstrated that there is always a stable match preferred by men (male-optimal) and there is always a stable match preferred by women (female-optimal). The algorithm which Gale and Shapley proposed consists of a number of rounds in which the men propose and the women review these proposals. This algorithm always produces a match which is preferred by men and thus is male-optimal. (For more detail, see [13].)

In a general SM problem we have two groups of individuals that can represent various types of entities. Thus, for simplicity, we call these two groups women and men. We denote the size of the group of women by $|W|$ and that of men by $|M|$. In general the cardinality of these two sets can be different. We assign an ID from 0 to $|W|-1$ to each woman and 0 to $|M|-1$ to each man; ID is unique among each group. Each woman and man ranks the members of the other group as she/he prefers which we call vector of preferences. Each woman can rank up to $K_w$ men and each man can rank up to $K_m$ women. Generally, $K_w$ and $K_m$ can be different from $|M|$ and $|W|$ respectively. For simplicity we will show the complexity of our protocol in terms of $n$ throughout the paper (assuming $|M| = |W| = n$). The preference vectors altogether, will form a preference matrix. Thus we have one preference matrix for the women's group and one for men's in which each row represents the preference vector of each member of that group, see Figure 2 for an example: If woman #1 ranks the men as $[1, 2, 0]$, this means that she prefers man #1 over man #2 and so on.

| Woman ID | Priority Higher → Lower | | |
|---|---|---|---|
| 0 | 1 | 0 | 2 |
| 1 | 1 | 2 | 0 |
| 2 | 0 | 1 | 2 |

**(a)** Women

| Man ID | Priority Higher → Lower | | |
|---|---|---|---|
| 0 | 2 | 0 | 1 |
| 1 | 2 | 1 | 0 |
| 2 | 0 | 1 | 2 |

**(b)** Men

**Fig. 2.** Example of preference matrices.

Therefore the preference matrix is of size $|W| \times K_w$ for women and $|M| \times K_m$ for men. These two matrices are the only input to the SM algorithm. After the algorithm is finished, the result is a stable match, meaning that there are no two individuals that they both prefer to be matched to each other but are not already assigned.

Figure 3 shows one stable match and one unstable match. The match in Figure 3a is stable because it satisfies the above definition and the match in Figure 3b is unstable because man #0 prefers woman #2 over woman #1 and also woman #2 prefers man #0 over man #1 and this violates the definition of SM.

| Woman ID | Man ID |
|---|---|
| 0 | 2 |
| 1 | 1 |
| 2 | 0 |

**(a)** Stable

| Woman ID | Man ID |
|---|---|
| 0 | 2 |
| 1 | 0 |
| 2 | 1 |

**(b)** Unstable

**Fig. 3.** Example of stable and unstable match.

A stable match is called optimal for men or women if every member of that group is matched to the best person he/she prefers that could have been matched in any other stable match. Roth [29] showed that this specific version of SM can always be found for one of the sets but not for both.

## 3.2 Stable Matching Algorithms

Here, we explain the Gale-Shapley SM algorithm [8]; the authors also prove the stability of its outcome in their paper.

### 3.2.1 General Stable Matching

The first algorithm relates to the case when each individual ranks all of the members from the other group. More specifically, $K_m = |W|$ and $K_w = |M|$. There is a list of size $|W|$ which holds the up-to-now assigned partner to each woman. This temporary assignment will become finalized when the algorithm terminates. We have the notion of free and engaged persons during the execution of the algorithm. Engaged persons are those who have a partner up to that round and free persons are those who do not. Each man could become engaged and after that free and again engaged and so on. But as soon as a woman gets engaged, she remains engaged until the end of the matching process. The algorithm consists of $R$ proposals. In each proposal, a free man proposes to the woman he prefers the most and he has not proposed to yet. Then if that woman who is proposed to is free, they become engaged. If she is engaged, she looks at her preference list and sees whether she prefers the proposing man over her already assigned partner or not; if yes, she becomes engaged to the new man and the previous man becomes free and if she does not prefer,

she rejects the proposal and the man remains free. The algorithm runs until all men are engaged. Algorithm 1 shows the pseudo-code of the limited SM algorithm where $w \longleftrightarrow m$ denotes assigning man $m$ to woman $w$ and $pc$ is the proposal counter for men.

**Algorithm 1:** General Stable Matching

```
while |free men| ≠ 0 do
  m ← choose from free men
  w = most preferred woman that m has not yet proposed to
  if w is free then
    w ⟷ m
  else
    m' ← man who is currently engaged to w
    if w prefers m over m' then
      w ⟷ m
      m' gets free
    end if
  end if
end while
```

Gale and Shapley showed that this algorithm takes at most $R_{max} = n^2 - n + 1$ proposals ($R \leq R_{max}$), where $n$ is the size of each participating group. As proven by Gale and Shapley, a stable match always exists and the algorithm terminates with the male-optimal stable match.

### 3.2.2 Limited Stable Matching

In the following, we describe a variant of the SM algorithm where each person ranks only a subset of the other group. More specifically, $K_m < |W|$ and $K_w < |M|$. This version is called limited SM. In this case, each person ranks up to a certain number, meaning that he/she prefers to be unmatched rather than being assigned to a person who is not in the list. Algorithm 2 is the pseudo-code for limited SM where the only difference to Algorithm 1 is that there is a limit on the total number of listed preferences.

The number of proposals in this case is upper bounded to $R = K_m \cdot |M|$. Although this algorithm limits the number of choices, it takes a number of proposals which is far smaller than that of the general version: $R_{\text{general}} \in O(n^2)$ whereas $R_{\text{limited}} <= K_m \cdot |M| \in O(n)$ for constant $K_m$. Since the total number of proposals are fixed and are less than $R_{general}$, it might be the case that some individuals remain unmatched after the algorithm is finished.

**Algorithm 2:** Limited Stable Matching

```
initialize pc[i] = K for i ∈ {0, |M| − 1}
while ∃man : pc[man] > 0 ∧ man is free do
  m ← choose that free man
  w = most preferred woman that m has not yet proposed to
  pc[m] ← pc[m] − 1
  if w is free then
    w ⟷ m
  else
    m' ← man who is currently engaged to w
    if w prefers m over m' then
      w ⟷ m
      m' gets free
    end if
  end if
end while
```

## 3.3 Yao's GC Protocol

Here, we explain the underlying SFE protocol of our system which is Yao's GC protocol [39] and is one of the most promising solutions for two-party SFE. In this protocol, two parties (Garbler and Evaluator) jointly evaluate a function on their inputs while keeping their own data private. The function is represented as a Boolean circuit. The GC protocol consists of three algorithms: circuit garbling which is done only by the Garbler, data exchange which involves both parties, and evaluation which is done only by the Evaluator. First, Garbler assigns two random keys to each Boolean value in the circuit and then encrypts the truth table of each gate using the keys. Garbler sends the encrypted tables to the Evaluator together with the keys corresponding to her inputs. Evaluator receives the keys corresponding to his inputs from the Garbler through 1-out-of-2 Oblivious Transfer (OT) protocol, without letting her to know his inputs. Then, Evaluator decrypts the tables, one by one, using the received keys until he reaches the output keys. Finally, Garbler reveals the mapping of the keys to the semantic values to Evaluator in order to achieve the final result in plain-text.

### 3.3.1 Optimizations

We benefit from the state-of-the-art optimizations for the GC protocol. We utilize the Free-XOR technique [16] which makes the cost of garbling an XOR gate virtually zero. Therefore, the cost of the GC protocol can be measured only in terms of the number of AND gates in the circuit. We also use garbling with a fixed-key block-cipher [3] together with the half gates technique [40] for efficient evaluation of AND gates. For OT required in the initial data exchange of the GC protocol, we use the OT Extension method [2, 14]. We use TinyGarble [32], an automated framework for generating optimized Boolean circuits for the GC which is based on logic synthesis tools. It optimizes the generation of a Boolean circuit for the GC protocol by customizing the flow of the logic synthesis tool. It uses a customized technology library consisting of logical descriptions of basic gates. The library also includes the corresponding parameters like timing and area. In the case for the GC, the timing parameter is not needed as the GC depends only on the size of the circuit (area, in synthesis tools' terminology). The area parameter of an XOR gate is set to 0 and all other two input gates to 1. The circuit is then synthesized using Synopsys Design Compiler (DC) 2010.03-SP4 [35]. Synopsys DC minimizes the number of non-XOR gates because the area for these gates is set to one whereas for XOR gates it is zero.

TinyGarble also proposes using sequential circuits that are more compact than the conventional combinational circuits. In sequential circuits, the states of the computation are stored in some memory elements (registers). Unlike combinational circuits, the output of the sequential circuits depends both on the input to the circuit and the value of the registers. For a same functionality, a sequential circuit has less number of gates compared to a combinational one. However, the sequential circuit has to be evaluated for multiple iterations (clock cycles). For garbling a sequential circuit, TinyGarble stores the labels associated to registers and uses them in the next iteration. Sequential description reduces the memory footprint of the GC protocol because less number of labels has to be stored in the memory for garbling and evaluation.

### 3.3.2 Oblivious RAM

Goldreich and Ostrovsky [10] proposed a two-party mechanism that lets a client store her data on a remote server while hiding her data and access pattern from the server. They also showed that the lower bound for the cost of accessing a single entry in the memory is sub-linear with respect to the size of the memory. A naïve implementation of ORAM linearly scans the entire memory for each access such that the client can choose the desired entry using multiplexer (MUX) which is called Linear ORAM.

Gordon et al. [12] proposed to use ORAM mechanism inside two-party SFE (e.g., GC) in order to reduce the amortized cost of accessing a memory entry from linear to sub-linear. There are several improvements on the original idea of ORAM including [28, 31, 34] which reduced the amortized per-access complexity to $\mathcal{O}(\log^3 n)$.

An ORAM scheme uses an oblivious data structure in order to hide the access pattern and it must implement two protocols: *initialization* protocol and *access* protocol. Initialization protocol is used to create and initialize the oblivious data structure from the given array of data. Access protocol is used to implement the actual access to the data structure. It translates the logical address that is created in the SFE protocol to the sequence of physical addresses. In RAM-based secure computation (RAM-SC), the memory accesses are handled by ORAM. Once the secret logical address is generated inside the SFE protocol, it is translated into multiple physical addresses by the access protocol (client-side) that are revealed to both parties. Both parities then provide the requested memory entities back to the SFE protocol. At the end, the SFE protocol changes all the memory entities' data to hide which element was accessed and how it was changed and then it sends them to two parties to store them. There are different data structures for ORAM. Goldreich and Ostrovsky [10] introduced two hierarchical layered structure ORAMs: Square-Root ORAM and Hierarchical ORAM. Shi et al. [31] initiated a tree-based ORAM scheme.

So far, the best asymptotic complexity for ORAM inside SFE is Circuit ORAM proposed by Wang et al. [38] which is a tree-based ORAM. A very recent paper [41] revisits Square-Root ORAM, an alternative ORAM structure that has lower initialization cost than Circuit ORAM. Despite the fact that it has higher asymptotic complexity, it outperforms Circuit ORAM for medium-sized memory.

We have used the implementations and results of Circuit ORAM [38] and Square-root ORAM [41]. We have benchmarked them and we show the results in Section 6. Integrating ORAM with our circuit requires to put a Boolean circuit description of client-side functionality in our circuit to run the "access" algorithm described in [38, 41]. Also, two parties, Garbler and Eval-

uator, have to be the server-side and interact with the client-side Boolean circuit to deliver the memory entities.

### 3.3.3 Adversary Model

The GC protocol, on which we base our implementation, is secure against honest-but-curious (also known as semi-honest or passive) adversaries which assumes that all parties follow the protocol but they may be curious to extract additional information from the data which they are receiving. Although this security model is not the strongest attack model, it is a first step towards being secure and it is commonly used in the literature. Also there are several reasons for having this security model:

- It is acceptable in many scenarios, e.g., when parties are reasonably trusted like hospitals, companies or government agencies but they need to obscure their private information for legal reasons or to prevent future break-ins. Moreover, this model prevents against passive observers that want to steal information.
- Many useful privacy-preserving applications inherently have the properties that let them fit well into this security model. For instance, when all parties have an incentive to generate flawless outcomes like financial fraud detection when banks pull together all of the data to detect corrupt accounts or personalized medicine when a patient and a drug company collaborate to find the best medicine, they are basically interested to reach the correct result and hence they will adhere to the protocol steps.
- Since we design Boolean circuits that are evaluated with Yao's GC protocol, the very same circuits can be evaluated with (less efficient) protocols that provide security against stronger active/malicious adversaries, e.g., [17, 19, 24, 30].

# 4 Circuits for Stable Matching

As explained in Section 3.3, the GC protocol for the two-party SFE takes a Boolean circuit as a function description. In this section, we describe how we generate such circuits for SM. We describe a combinational circuit for SM in Section 4.1. Afterward, we introduce the first sequential circuit ever implemented for SM in Section 4.2. In Section 4.3, we analyze the circuit runtime. Section 4.4 describes a novel sub-linear size circuit for SM that we have designed. Finally, in Section 4.5 we analyze the memory usage and ORAMs' access cost complexities.

## 4.1 Combinational Circuit

To the best of our knowledge, there exists no implementation of the combinational circuit that gets the men's and women's preference matrices as an input and outputs the stable match. The reason might be that it is relatively complicated to design such a large circuit without proper tool support. In this section, we calculate a lower bound for the size of this circuit. Considering $n$ to be the number of the pairs in SM, we have preference matrices of size $\mathcal{O}(n^2 \log n)$ bits, because each of the $n$ individuals ranks the $n$ members of the other group and each entry needs to be represented by $\log n$ bits. Accessing an entry, without using a sub-linear ORAM, requires a multiplexer which needs $\mathcal{O}(n^2 \log n)$ AND gates. Comparing two entities of length $log\, n$ requires $log\, n$ AND gates. We have to compare $n$ entities for each man, yielding total $\mathcal{O}(n^2 \log n)$ AND gates. This is just for accessing and comparing two preferences. To implement the algorithm, it is extraordinarily hard to design a generic circuit. One solution might be to design a circuit that has $R$ (the number of proposals) layers and each layer processes one proposal made by a man until we reach the stable match. Knowing that in the worst case, we need $R_{max} = n^2 - n + 1$ proposals, this accounts for the number of layers of the circuit. Another solution might be for each man to find his match instantly. This requires that for each of the $n$ women on his list, we find whether she accepts the proposal by that man or not and this also depends on the preference list of other men. So in this way, we need $n$ circuits of size $\mathcal{O}(n)$ (for each woman) times $n$ (for each man) to compare the preferences. This results in $\mathcal{O}(n^3)$ comparisons on $\mathcal{O}(n \log n)$ values. In both cases, the lower bound of the circuit size is $\mathcal{O}(n^4 \log n)$ which is a huge circuit size even for relatively small values of $n$. Therefore, the actual implementation is not feasible for real-world applications, since there is no automated tool to generate and synthesize a circuit of such size.

## 4.2 Sequential Circuit

Describing the SM circuit in a sequential way enables us to implement the circuit in a highly compact format.

This compact format allows us to synthesize the circuit for larger group size and meet the size of real-world applications. This compactness also provides us with the ability to store the circuit on the platforms with lower computational and memory capabilities.

In the rest of this section, we describe our sequential circuit for SM. Figure 4 shows the block diagram representation of the circuit. The inputs to the circuit are the preference matrices of men and women. The outputs of the circuit comprise the final stable match list and the finish signal. Jumping ahead, when using the early termination technique (ETT) (discussed in Section 4.3), we output the finish signal to both parties at each iteration whereas the final list is output only if the finish signal is 1, but kept secret otherwise. The circuit consists of the following modules: Algorithm Computation Circuit (ACC), vectorized Preference Compare Circuit (PCC), and Man Selection Circuit (MSC), and Memory (registers for storing the state values). We consider linear-complexity ORAM techniques to store the data for this circuit.

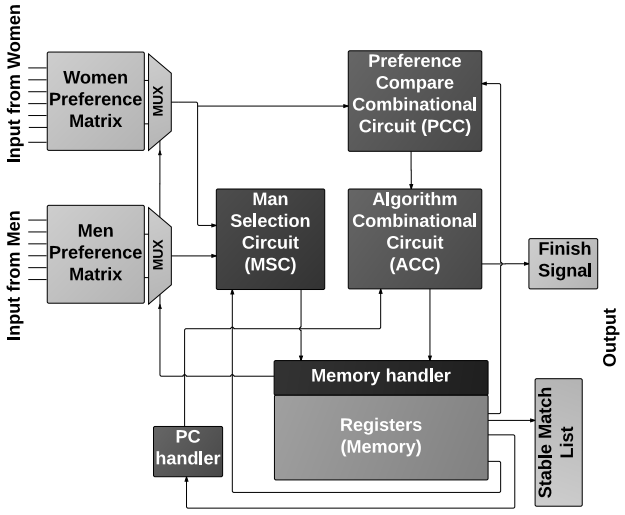


**Fig. 4.** Block diagram architecture of the sequential circuit for implementing secure SM.

Here, we explain how the entire circuit works. We design the sequential circuit such that it processes one proposal in one iteration of the sequential circuit. There is a very close similarity between the Boolean circuit and Algorithm 1, e.g., MSC implements "*choose from free men*" (it outputs the ID of the next free man that can propose), PCC implements "$w$ *prefers* $m$ *over* $m'$", and ACC implements the rest of the main body of the algorithm.

MSC finds the next free man. The internal structure of MSC is similar to a priority encoder. Therefore, this module looks in the men's list and selects the first free man for the next iteration of the sequential circuit. In Section 4.3, we will explain how MSC and the finish signal work together. Once a free man is selected, he should propose to the most preferred woman that he has not already proposed to (see Algorithm 1). Therefore, a counter storing the number of proposals is required for each man; we call it Proposal Counter (PC). In order to access the desired women ID, a MUX on the man's preference list is required. This MUX needs $\mathcal{O}(n^2 \log n)$ AND gates.

ACC implements the main part of Algorithm 1 as a Boolean circuit. ACC processes the proposal: if a woman is free, then the man and she will be engaged and the status of the assignment will be updated in the memory. If she is not free, ACC needs to check whether or not she prefers the man over her current match. Therefore, we need to access the preference vector of the woman using MUX and deliver it to PCC together with the values for the man ID and the women's current match ID. PCC linearly scans the preference vector and checks which one of the two men is more preferable (which one comes first in the preference list). It is important that PCC does this task in one iteration, otherwise the entire circuit will be unnecessarily evaluated and the efficiency will be reduced. As a result, one proposal can be done in *one* iteration of the sequential circuit. In Section 4.3, we will explain how many iterations the circuit should be evaluated.

Figure 5 illustrates the inside of the PCC. This circuit is designed to minimize the number of AND gates. In fact, PCC looks at the woman preference list and sees which man ID came first. The 1-bit output becomes 1 if the new man is more preferable. The first two layers' wires are $\log n$ bits. Then there is one linear layer of XOR gates and three linear layers of AND gates which at the end are OR-ed and produce the output. The final logical OR gate is implemented by $n-1$ two-input AND gates and $3n-3$ XOR gates. Thus, PCC uses $2n \log n + 3(n-1)$ AND gates. In summary, the total number of AND gates in the circuit is $\mathcal{O}(n^2 \log n)$ dominated by MUXs for accessing the preference list.

## 4.3 Number of Proposals and Circuit Runtime

Gale and Shapley have proved that after at most $R_{max} = n^2 - n + 1$ proposals in Algorithm 1, the algo-

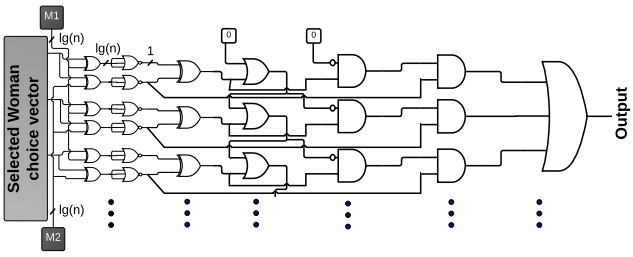


**Fig. 5.** Preference Compare Circuit (PCC).

rithm will output a stable match. However in the GC, the intermediate status of each man's engagement is encrypted and cannot be accessed in plain-text. Therefore, the process of finding the next free man becomes challenging. One naïve implementation can be to iterate over all men and let them to propose regardless of the engagement status. But if the man was not free, the proposal should be invalidated. This approach is dramatically inefficient. The reason is as follows: since all the information in the circuit is encrypted, there is no way to distinguish a valid proposal from an invalid one. Thus, we have to evaluate the circuit regardless of the validity of the proposal. As a result, we need to run the circuit even for more iterations than the mathematical worst case of the number of proposals ($R_{max}$). This problem arises both for general SM and limited SM. To overcome this problem, we have designed Man Selection Circuit (MSC). This circuit wisely choses the next free man ahead of time to avoid invalid proposals (unnecessary execution of the circuit).

MSC makes the number of iterations that the circuit should be evaluated *exactly equal* to the number of required proposals. In the worst case, this number is equal to $R_{max}$. However, based on our extensive statistical analysis, we have observed that the average number of required proposals is linearithmic in terms of the set size $n$. Our experiment is performed for 10 000 different randomly generated preference matrices for each set size. It can be seen from Figure 6 that the statistical maximum of the number of proposals is also linearithmic with respect to the set size $n$. The error bars show the minimum and maximum number of proposals for each set size. In [20], authors provide a probabilistic analysis and they show that the average number of proposals is indeed $\Theta(n \log n)$.

We can take advantage of this fact by two different approaches. (i) The first approach is to run the circuit for a constant number of times. This number can be statistically guaranteed instead of mathematically guaranteed and it means that we can run the circuit for some fixed statistically driven number like $m + \alpha \times \sigma$, where $m$ is the average number of proposals, $\sigma$ is the standard deviation, and $\alpha$ is a constant to be determined by the probability of proper termination. $\alpha$ is fixed prior to running the protocol and is known by both Garbler and Evaluator. Higher $\alpha$ means it is more likely that by the time the protocol is finished, the output is a final stable match but higher $\alpha$ also requires more iterations to be run. Figure 6 plots the statistical worst case for $\alpha = 8$ (this is chosen so that $m+\alpha\times\sigma$ is higher than the maximum error bar for all set sizes). (ii) The second approach is to produce the "finish signal" by ACC and stop the algorithm as soon as the stable match is reached. We call this approach Early Termination Technique (ETT). ACC produces the finish signal by checking if all the men (or women) are engaged. It tracks the status of all men and if all of them are matched, then the finish signal is true.

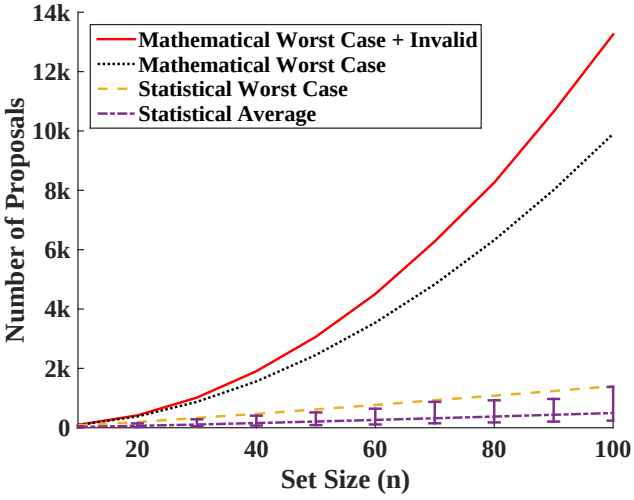


**Fig. 6.** Number of proposals needed for different scenarios in the general SM. Statistical worst case is computed for $\alpha = 8$). The error bar shows the minimum and maximum value in the simulation. Experiment is performed 10 000 times for each set size with randomly generated preference list.

In order to verify the termination condition in the middle of the GC protocol, the parties have to reveal the shared secret associated only to the finish signal. Revealing this secret in an iteration will not leak any information about the rest of the secure computation except the fact whether or not the computation is finished up to this iteration. Thus in our implementation, the computation and communication can be scaled down almost by a factor of $\mathcal{O}(n)$. This is a privacy/efficiency trade-

off though (see Section 7 for more details on information leakage and Section 6 for performance results).

## 4.4 Sub-linear Sequential Circuit

In Section 4.2, we proposed a sequential circuit for general and limited SM. The size of the circuit is $\mathcal{O}(n^2 \log n)$. We also discussed in Section 4.3 that the circuit should run for $\mathcal{O}(n^2)$ iterations for the worst case scenario and can be reduced using early termination (ETT) to $\mathcal{O}(n)$. Therefore, the overall computation and communication complexity of the secure SM is $\mathcal{O}(n^3 \log n)$ using ETT. This overall cost is the product of the number of iterations and the size of the circuit. For the set size of the NRMP ($n = 32k$), the cost is in the order of $2^{50}$ AND gates which incurs $32PB$ communication! The number of iterations (proposals) is already minimized by MSC and ETT and cannot be reduced further. Thus, the only way is to reduce the circuit size. To do so, we propose four tweaks for the sequential circuit.

First, we replace the MUXs accessing the memory (e.g., preference list of men and women) with sub-linear ORAM. It allows us to access the memory of size $m$ with the cost of less than $\mathcal{O}(m)$. We will discuss the choice of ORAMs and their trade-offs in Section 4.5.

Second, we need to change the size of PCC from linear to sub-linear. PCC outputs a binary indicating if a woman prefers the new proposing man over her already assigned man. PCC linearly scans the woman's preference list (a vector of man's ID) in order to find which man came first in the list. (Note that the list is sorted from most desirable to the least.) This incurs $\mathcal{O}(n)$ AND gates. To make PCC sub-linear, we store the inverse of the preference list of each woman. For example if man #5 is the $3^{rd}$ one in the preference list, in the inverse list, the value of the $5^{th}$ position is 3. Now instead of a linear search, we need two accesses to the inverse list and compare their values to determine the more desirable man. However, access cost using MUX is still $\mathcal{O}(n)$. But the MUX can be replaced with sub-linear ORAM to make the total cost sub-linear with respect to $n$.

Third, ETT needs to access the status of all men in order to determine the finish signal ($\mathcal{O}(n)$ AND gates). Accessing the men's status list using sub-linear ORAM does not solve the problem since we still need to access men's status list $n$ times at each iteration. This makes the overall cost even worse than Linear ORAM. Therefore, we store the total number of engaged men in a counter. We initialize the counter with zero and each time a free man becomes matched with a woman, we increase the counter by one. When a new assignment requires old assignment to break, we simply do not change the counter. Also, when a proposal gets rejected, we do not change the counter. Using this approach, we only need to store $\log n$ bits, add a addition block, and put a multiplexer before the registers. The total overhead for this approach is $\mathcal{O}(\log n)$.

Fourth, MSC requires scanning over the status of all men at each iteration to choose the next free man. Similar to ETT, this incurs $\mathcal{O}(n)$ AND gates. Unlike the solution for ETT, we cannot skip this computation for any iteration. We propose to store the ID of all free men in a *queue*. We dequeue a free man and process his proposal. If his proposal gets rejected, we enqueue him to the back of the queue. Or if in the process of the proposal, a previously assigned man becomes free, we enqueue this man to the queue. The queue should be initialized by all men at the beginning of the SM. Therefore, the capacity of the queue should be exactly $n$. We implement this queue using an array of size $n$ and circular indexing. This is the most efficient and simplest solution and incurs a constant number of accesses per each enqueue or dequeue operation. By storing this array in a sub-linear ORAM, the size of MSC will become sub-linear.

## 4.5 Memory Analysis and Choosing ORAM Scheme

There are several ORAM schemes in the literature [38, 41]. Each of them has different initialization cost and access complexity. Depending on the intensity of memory accesses and size of the memory, each ORAM may outperform the others. Thus, it is important to find at which memory size, one ORAM scheme starts to outperform the others. This is called the *breakeven* point. To the best of our knowledge, Circuit ORAM [38] has the best asymptotic complexity $\mathcal{O}(log^3 n)$. However, for small memory sizes, Linear ORAM (MUX) outperforms all sub-linear ORAMs including Circuit ORAM. The reason is twofold: (i) The cost of initialization for Linear ORAM is negligible compared to sub-linear ORAMs. (ii) The constant coefficient for its access cost complexity is lower than the one for sub-linear ORAMs.

For medium set size, another sub-linear ORAM, Square-Root ORAM [41], outperforms both Linear ORAM and Circuit ORAM. The reason is that it has lower initialization cost compared to Circuit ORAM but has better access cost complexity of $\mathcal{O}(\sqrt{n \log n})$ com-

pared to Linear ORAM. As the set size increases, Circuit ORAM eventually outperforms both Square-Root ORAM and Linear ORAM.

Based on the characteristics and performance of different ORAM schemes, we integrated our sequential circuit with the best ORAM scheme. Integrating ORAM means putting a client-side Boolean circuit (for handling access requests) inside the main garbled circuit and connecting necessary interfaces to our sub-linear sequential circuit. Also two parties should act as servers who then store the memories and communicates with the client circuit to deliver the requested memory entity to the client (the servers hold the encrypted memory).

The choice of ORAM also depends on which variant of secure SM we want to use (General SM or Limited SM) and whether or not we are interested to use the early termination technique.

Table 1 shows the overall asymptotic complexity which includes both initialization and per-access cost. Note that in order to find the most efficient one, we need to perform experiments. In Section 6.2, we will show what is the best choice of ORAM depending on the set size and SM variant based on our experiments.

# 5 Related Work

Golle [11] was the first to develop privacy-preserving SM. He persuasively illustrates that implementing such an algorithm has a great practical impact. In his framework, he devises a variant of the classic Gale-Shapley algorithm with some techniques for concealment. Each party should send the encrypted preference list to some honest-but-curious matching authorities. During the algorithm men and women are divided into two disjoint groups, those who are engaged and those who are free. Then, $m$ fake men are added and are engaged to women at the beginning of the algorithm and this enables some appealing concealment properties, such as that the number of engaged men and free men are always constant and this could prevent any information leaking of intermediate changes of free and engaged men sets. Golle then defines a bid as an encrypted representation of the preference of one man for a woman in addition to some "book-keeping" information. There are free and engaged bids, a bid paired up with a woman. The Algorithm follows 4 steps for $R$ iterations beginning with initial bids to reach a stable match by resolving a conflict between bids at each time. Those four steps are (i) randomly choosing a free bid and opening it mutually by matching authorities, (ii) finding a conflict bid since there is always exactly one conflicting bid, (iii) resolving the conflict, and (iv) mixing bids internally and externally. Golle uses an additively homomorphic semantically secure threshold public key encryption scheme like a threshold version [4, 5] of Paillier [26] and re-encryption mix networks to implement this. Golle argues that the algorithm terminates after $n$ iterations and reaches a stable match between $n$ men and $n$ women. The complexity then is dominated by $\mathcal{O}(n^3)$ modular exponentiations and the corresponding communication complexity is also $\mathcal{O}(n^2\text{polylog}(n))$.

However, Franklin et al. [6] show that Golle's proposal is not promising and in the worst case, his algorithm will be executed $n^2$ iterations. This results in a computation complexity of $\mathcal{O}(n^5)$ modular exponentiations and $\mathcal{O}(\nu\, n^5)$ communication complexity, where $\nu$ is the number of matching authorities. They also develop another solution for this problem based on Golle's framework but also add $n$ fake women. There are some modifications to the original framework. They added Private Information Retrieval (PIR) for reading and writing privately in a shared database. PIR is implemented using the protocol of Naor and Nissim [21] which is based on the Oblivious Transfer and requires $\mathcal{O}(\text{polylog}\, n)$ communication and is used a constant number of times in each iteration. Franklin et al. use the GC protocol to compare the preference of women and also to increment the number of times that each man has proposed. The communication complexity is dominated by re-encryption mixnets just like Golle's which results in $\mathcal{O}(\nu\, n^3\text{polylog}\, n)$ communication complexity. Yet, computation complexity is dominated by accessing the database. In each iteration, the database access takes $\mathcal{O}(n^2\sqrt{\log n})$ work and the total computation complexity is $\mathcal{O}(n^4\sqrt{\log n})$ public key operations. In that work they also optimize the protocol when there are exactly 2 matching authorities and they achieve a better performance as listed in Table 2. However, this protocol is complicated and not yet implemented.

Naor et al. [22] suggest an architecture for privacy-preserving protocols for mechanism design and they mention SM as one of the related applications. They argue that the complexity of implementing their architecture for SM problem depends on a combinational circuit which implements the SM algorithm. They did not implement such a system nor designed such a circuit. Naor et al. suggest that the classical Gale-Shapley algorithm necessitates the usage of indirect addressing of a RAM and that translation into a circuit is inefficient.

**Table 1.** Total complexity for implementing each variant of SM using different ORAM schemes, where $n$ is the set size and $k$ is the limit of number of proposals in limited SM.

| ORAM Scheme | General SM | General SM + ETT | Limited SM |
|---|---|---|---|
| Linear ORAM | $\mathcal{O}(n^4 \log n)$ | $\mathcal{O}(n^3 \log n)$ | $\mathcal{O}(n^2 k^2 \log n)$ |
| Square-Root ORAM | $\mathcal{O}(n^3 \log^2 n)$ | $\mathcal{O}(n^2 \log^2 n)$ | $\mathcal{O}((n k)^{1.5} \log^{1.5}(n k) \log^{0.5} n)$ |
| Circuit ORAM | $\mathcal{O}(n^2 \log^3 n)$ | $\mathcal{O}(n^2 \log^3 n)$ | $\mathcal{O}(n k \log^2(n k) \log n)$ |

**Table 2.** Different protocols for performing secure SM and corresponding complexities, where $n$ is the size of each group and $\nu$ is the number of matching authorities. Our protocol is faster by orders of magnitudes as it uses efficient Symmetric-Key operations instead of costly Public-Key operations.

| Protocol | Computation Complexity | | Communication Complexity | Round Complexity |
|---|---|---|---|---|
| Golle [11] | $\mathcal{O}(n^5)$ | Public-Key Operations | $\mathcal{O}(\nu n^5)$ | $\mathcal{O}(n^3 \text{polylog} n)$ |
| Franklin et al. [6] | $\mathcal{O}(n^4\sqrt{\log n})$ | Public-Key Operations | $\mathcal{O}(\nu n^3)$ | $\mathcal{O}(n^2 \text{polylog} n)$ |
| Franklin et al. [7] | $\mathcal{O}(n^4 \text{polylog} n)$ | Public-Key Operations | $\mathcal{O}(\nu n^2)$ | $\mathcal{O}(n^2 \text{polylog} n)$ |
| Ours without using ORAM | $\mathcal{O}(n^4 \log n)$ | Symmetric-Key Operations | $\mathcal{O}(n^4 \log n)$ | $\mathcal{O}(1)$ |
| Revisiting SQRT ORAM [41] | $\mathcal{O}(n^3 \log^2 n)$ | Symmetric-Key Operations | $\mathcal{O}(n^3 \log^2 n)$ | $\mathcal{O}(n^2 \text{polylog} n)$ |
| Ours using ORAM | $\mathcal{O}(n^2 \log^3 n)$ | Symmetric-Key Operations | $\mathcal{O}(n^2 \log^3 n)$ | $\mathcal{O}(n^2 \text{polylog} n)$ |
| Keller et al. [15] | $\mathcal{O}(n^2 \log^3 n)$ | Symmetric-Key Operations | $\mathcal{O}(n^2 \log^3 n)$ | $\mathcal{O}(n^2 \text{polylog} n)$ |
| Ours using ORAM and ETT | $\mathcal{O}(n^2 \log^3 n)$ | Symmetric-Key Operations | $\mathcal{O}(n^2 \log^3 n)$ | $\mathcal{O}(n^2 \text{polylog} n)$ |

Franklin et al. [7] develop an efficient multiparty Look-Up Table (mLUT) protocol and suggest that one can design a secure SM system by implementing the algorithm of [6, Section 5] into a circuit with access to a RAM. Especially in the multiparty setting, mLUT for array/matrix access reduces the complexity of such a setting to the situation when we have 2 matching authorities as shown in Table 2. However, this system has only been suggested but has not been implemented yet.

Keller et al. [15] were the first that reported implementing secure SM using GC and oblivious memory. Without presenting implementation details, they reported that in the worst case, the general SM with $8k$ pairs can be done in $1.5 \cdot 10^{12}$ seconds. The complexity of related work is reported in Table 2. Zahur et al. [41] reported a runtime of more than 33 hours for set size of 512 pairs. Although they have the previously best known runtime for this set size, their computation and communication complexity is $\mathcal{O}(n^3 \log^2 n)$ which limits the scalability of their approach significantly. While our computation and communication complexity is $\mathcal{O}(n^2 \log^3 n)$ without using ETT, we also achieve a better runtime of 8 hours without using ETT and 5 minutes by using ETT for the same set size.

# 6 Evaluation and Comparison

In this section we summarize and compare the complexity of different algorithms for general SM and its limited variant.

## 6.1 Evaluation Setup

We use Synopsys Design Compiler 2010.03-SP4 to generate our sequential circuits. The timing analysis are done on two similar machines with Intel Core i7-2600 CPU @ 3.4GHz with 12GB RAM on an Ubuntu 15 operating system connected using 1 Gbps Ethernet. In all of the experiments, the GC security parameter is 128-bit and the security failure probability for ORAM is set to at most $2^{-80}$. The experiments that take more than $10^5$ seconds are estimated based on the computational and communication complexity. For sub-linear ORAMs (Circuit ORAM and Square-Root ORAM) we used the circuit size of ORAM based on the results in [41].

## 6.2 ORAM Analysis

Figure 7 shows the memory cost for the general SM running for worst case scenario. There are 3 different zones. For very small set sizes ($< 2^5$), the linear ORAM

(MUX) is the most efficient solution. In the next region up to set sizes ($\sim 2^{12}$), the Square-Root ORAM outperforms linear ORAM. For set sizes larger than ($\sim 2^{12}$), the Circuit ORAM outperforms all other solutions. For the NRMP, the set size is roughly $2^{15}$ and therefore the Circuit ORAM is the most efficient method. However, it takes $10^8$ seconds ($\sim 3$ years) and seems still not practical.

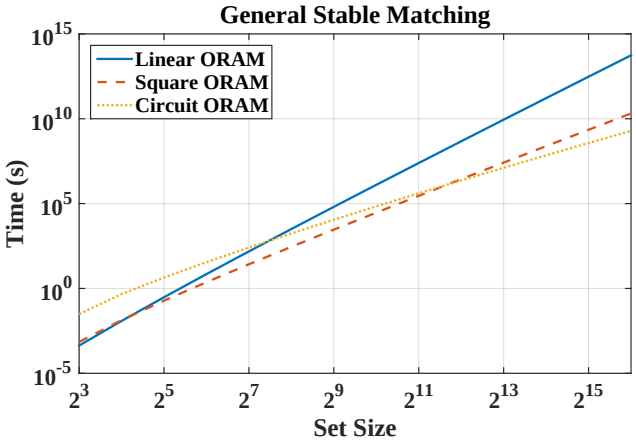


**Fig. 7.** Total running time of the GC protocol using different ORAM schemes for general SM without using ETT.

Figure 8 shows the memory cost for the general SM using ETT. In this case the number of iterations is reduced by a factor of $\mathcal{O}(n)$. ETT makes the algorithm less memory intensive thus initialization cost of ORAM becomes dominant. Since the initialization cost of Square-Root ORAM is less than Circuit ORAM, it outperforms Circuit ORAM for any set size. Here we have only one breakeven point in which Square-Root starts to outperform Linear ORAM at set size $2^5$. For the NRMP ($2^{15}$ pairs), the Square-Root ORAM is the most efficient method and it takes $10^6$ seconds ($\sim 24$ days). This is the first time that the NRMP stable matching can be securely evaluated in less than a month. The solution of [15] reports 47 000 years of computation for SM on a set size 1/4 of NRMP.

Figure 9 shows the memory cost for limited SM where the number of preferences for each person is limited to $k = 20$. The breakeven point between Linear ORAM and Square-Root ORAM is at set size $2^5$. In this scenario, the cost of initialization is as important as the cost of accessing the memory. Hence, Square-Root ORAM outperforms Circuit ORAM for a wide range of set sizes.

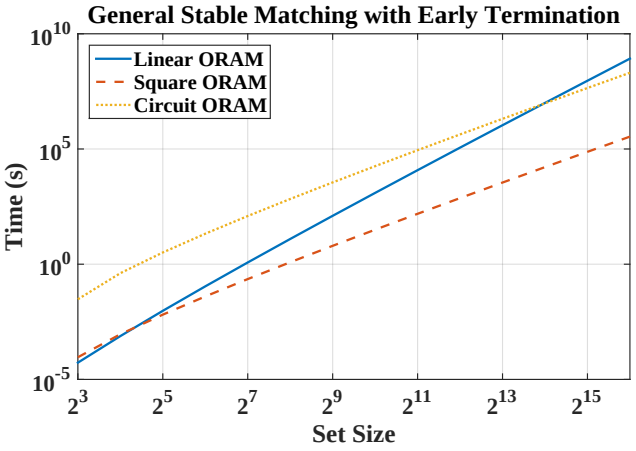


**Fig. 8.** Total running time of the GC protocol using different ORAM schemes for general SM using early termination technique.

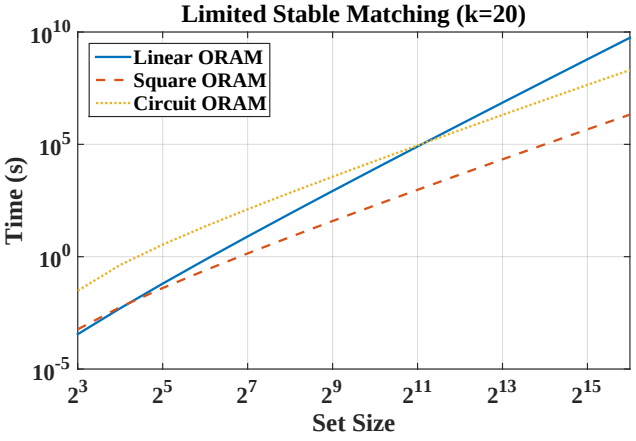


**Fig. 9.** Total running time of the GC protocol using different ORAM schemes for limited SM with $k = 20$.

## 6.3 End-to-End Secure Stable Matching

Table 3 shows the total end-to-end execution time and communication cost of general SM for various set sizes when using three different ORAM schemes. For each set size, the best result is shown in bold format. Our scheme can securely compute a stable match for $8k$ pairs four orders of magnitude faster than the method of [15] (the previously best method in terms of complexity).

Table 4 shows the total end-to-end execution time and communication cost of general SM with ETT for various set sizes when using three different ORAM schemes. For each set size, the best result is shown in bold format. The computation and communication cost are further decreased, for example for set size $8k$ by three orders of magnitude compared to general SM without ETT.

**Table 3.** Timing and communication results without using early termination technique.

| Set Size | Linear ORAM | | Square-Root ORAM | | Circuit ORAM | |
|---|---|---|---|---|---|---|
| | Time | Comm. | Time | Comm. | Time | Comm. |
| 8 | **9.1 ms** | **1.22 MB** | 25.7 ms | 3.44 MB | 107.1 ms | 14.37 MB |
| 128 | 23.21 min | 186.92 GB | **5.43 min** | **43.78 GB** | 31.68 min | 255.18 GB |
| 512 | 5.3 d | 61.52 TB | **7.9 h** | **3.81 TB** | 1.05 d | 12.17 TB |
| 2k | 4.54 y | 19.25 PB | **26.59 d** | **308.4 TB** | 36.95 d | 428.54 TB |

**Table 4.** Timing and communication results using early termination technique.

| Set Size | Linear ORAM | | Square-Root ORAM | | Circuit ORAM | |
|---|---|---|---|---|---|---|
| | Time | Comm. | Time | Comm. | Time | Comm. |
| 8 | **5.69 ms** | **764 KB** | 15 ms | 2.01 MB | 100.5 ms | 13.48 MB |
| 128 | 54.4 s | 7.3 GB | **12.75 s** | **1.71 GB** | 2.12 min | 17.11 GB |
| 512 | 1.24 h | 600.81 GB | **4.63 min** | **37.32 GB** | 41.96 min | 337.93 GB |
| 2k | 4.05 d | 46.99 TB | **1.56 h** | **753.83 GB** | 13.79 h | 6.66 TB |
| 8k | 306.53 d | 3.55 PB | **1.25 d** | **14.54 TB** | 11.96 d | 138.7 TB |
| 32k | - | - | **23.47 d** | **272.2 TB** | 251.65 d | 2.91 PB |

Table 5 shows the total end-to-end execution time and communication cost of limited SM with $k = 20$ for various set sizes when using three different ORAM schemes. For each set size, the best result is shown in bold format.

Table 6 compares our timing results with the previously best known runtime for mid-size set sizes presented in [41] (they did not report the communication results). We achieve approximately 4× and 450× improvement without and with ETT respectively compared to the results reported in [41]. The maximum set size that is compared is 512 because they have not reported set sizes higher than 512.

# 7 Discussion

**Security.** Security of our protocol follows from the proof of security of Yao's GC protocol in [18] and that of ORAM constructions for secure computations [38, 41]. The security of XOR sharing is based on the security of One-Time Pad (OTP). Our optimizations are applied to the Boolean circuit (used inside one step of the GC protocol) leaving the flow of the GC protocol intact. The only exception is the optional Early Termination Technique for which we discuss the privacy-efficiency trade-off next.

**What would be leaked by early termination?** As we discussed earlier in Section 4.3, we can terminate the GC protocol as soon as it has reached the stable results in order to enhance the performance greatly. We do this by producing a 1-bit "finish signal" and revealing its real value at each iteration. Revealing this one bit means sharing the actual wire label by Evaluator and the type of the wire by Garbler where the real value of finish signal wire would be the XOR of these two.

Please note that both parties share the information about this 1-bit wire *only*. Since the type of each wire is generated randomly, revealing the type of one wire does not convey any information about other bits and hence this technique does not affect the rest of the protocol and no more information would be revealed.

Although using the finish signal may leak the number of proposals $R$, there are plenty of different cases when this setting could be employed. Such as when each matching authority agrees not to reveal the number of real proposals because no one has any interest to do so. In this case, there is no desire to reveal the number of proposals. If we add a random initiator to the MSC and the next-man-selection algorithm, we can achieve the randomness which will make $R$ nondeterministic. In this case, even for the same inputs, each time $R$ is different. However, leaking the number of total proposals *can* convey some information about the preference lists. For example, if the matched partners were their mutually first choices, then the algorithm will terminate immediately after the first iteration. This is very different from the case when each individual is matched with her last choice, in which case the algorithm will terminate after

**Table 5.** Timing and communication results for limited SM where $k = 20$.

| Set Size | Linear ORAM | | Square-Root ORAM | | Circuit ORAM | |
|---|---|---|---|---|---|---|
| | Time | Comm. | Time | Comm. | Time | Comm. |
| 8 | **22.76 ms** | **3.05 MB** | 51.8 ms | 6.95 MB | 200 ms | 26.84 MB |
| 128 | 3.62 min | 29.2 GB | **50.97 s** | **6.84 GB** | 5.73 min | 46.15 GB |
| 512 | 4.97 h | 2.4 TB | **18.52 min** | **149.21 GB** | 1.42 h | 688.2 GB |
| 2k | 16.21 d | 187.98 TB | **6.23 h** | **3.01 TB** | 20.2 h | 9.76 TB |
| 8k | 3.35 y | 14.21 PB | **5.01 d** | **58.11 TB** | 13.94 d | 161.74 TB |

**Table 6.** Timing results comparing with previously best method [41] (in terms of runtime).

| Set Size | Revisiting SQRT ORAM | Our best case | Our best case w/ Early Termination |
|---|---|---|---|
| 8 | 510 ms | 9.1 ms | 5.69 ms |
| 64 | 2.41 min | 33.19 s | 2.59 s |
| 512 | 1.38 d | 7.9 h | 4.63 min |

$\mathcal{O}(n^2)$ iterations. Therefore, knowing $R$ can give some information about the quality of the assignment. This problem can be mitigated by running the protocol for at least the statistical worst case (see Figure 6). In order to avoid unnecessary iterations and terminate the protocol, $R$ is the least information possible to leak.

It is worth mentioning that in real-world, as shown in Figure 6, the standard deviation of actual number of proposals is relatively small, meaning that the execution time for different inputs would be about the same and cannot convey significant information about the preference lists. In a nutshell, this variant of our protocol is a trade-off between privacy and efficiency where we have an unprecedented performance improvement but the total number of proposals is leaked. Note that the early termination technique is optional and can be avoided.

It is in fact a recent direction in secure computation to give up some privacy in order to achieve better runtime, see for example the private DBMS BlindSeer which leaks some (hard to quantify) information about the search tree [27].

# 8 Conclusion

We present the first efficient and scalable implementation of secure Stable Matching (SM). This is the first work which makes secure SM feasible for real-world applications such as the National Residency Matching Program (NRMP) with $32k$ pairs in a reasonable time. Although the execution time for very large inputs is still substantial, this is not unreasonable, as applications of SM usually do not have to be run online, but instead can be run offline, e.g., during a month. This is because in many applications, a large latency is not the main concern, but performing the stable match privately is. We designed different variations of SM (general and limited) and a novel technique to significantly decrease the computation time by reducing the total number of SM iterations. We also proposed a sequential circuit with sub-linear amortized size with respect to the number of pairs in SM. We benchmark three different Oblivious RAMs (ORAM) in our evaluation and determine the breakeven points for each variant of SM.

**Acknowledgements**

We would like to thank the anonymous reviewers for their helpful comments. This work is partially supported by National Science Foundation grant (CNS-1513063) and a Multidisciplinary University Research Initiative grant (FA9550-14-1-0351/ Rice 14-0538) to the ACES lab at Rice University. This work has been co-funded by the European Union's 7th Framework Program (FP7/2007-2013) under grant agreement n. 609611 (PRACTICE), by the German Federal Ministry of Education and Research (BMBF) within CRISP, by the DFG as part of project E3 within the CRC 1119 CROSSING.

# References


[1] Atila Abdulkadiroğlu, Parag A Pathak, and Alvin E Roth. The New York city high school match. *American Economic Review*, pages 364–367, 2005.

[2] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In *ACM CCS'13*, pages 535–548. ACM, 2013.

[3] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE S&P'13*, pages 478–492. IEEE, 2013.

[4] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC'01*, volume 1992 of *LNCS*, pages 119–136. Springer, 2001.

[5] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. In *FC'01*, volume 1962 of *LNCS*, pages 90–104. Springer, 2001.

[6] Matthew Franklin, Mark Gondree, and Payman Mohassel. Improved efficiency for private stable matching. In *CT-RSA'07*, volume 4377 of *LNCS*, pages 163–177. Springer, 2006.

[7] Matthew Franklin, Mark Gondree, and Payman Mohassel. Multi-party indirect indexing and applications. In *ASIACRYPT'07*, volume 4833 of *LNCS*, pages 283–297. Springer, 2007.

[8] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, pages 9–15, 1962.

[9] David Gale and Marilda Sotomayor. Ms. Machiavelli and the stable matching problem. *American Mathematical Monthly*, pages 261–268, 1985.

[10] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.

[11] Philippe Golle. A private stable matching algorithm. In *FC'06*, volume 4107 of *LNCS*, pages 65–80. Springer, 2006.

[12] S Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *ACM CSS'12*, pages 513–524. ACM, 2012.

[13] Dan Gusfield and Robert W Irving. *The stable marriage problem: Structure and algorithms*, volume 54. MIT press Cambridge, 1989.

[14] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO'03*, volume 2729 of *LNCS*, pages 145–161. Springer, 2003.

[15] Marcel Keller and Peter Scholl. Efficient, oblivious data structures for MPC. In *ASIACRYPT'14*, volume 8874 of *LNCS*, pages 506–525. Springer, 2014.

[16] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *ICALP'08*, volume 5126 of *LNCS*, pages 486–498. Springer, 2008.

[17] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT'07*, volume 4515 of *LNCS*, pages 52–78. Springer, 2007.

[18] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[19] Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of Cryptology*, 25(4):680–722, 2012.

[20] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[21] Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *STOC'01*, pages 590–599. ACM, 2001.

[22] Moni Naor, Benny Pinkas, and Reuben Sumner. Privacy preserving auctions and mechanism design. In *ACM conference on Electronic Commerce (EC'99)*, pages 129–139. ACM, 1999.

[23] National Residency Matching Program. http://www.nrmp.org.

[24] Jesper Buus Nielsen and Claudio Orlandi. LEGO for two-party secure computation. In *TCC'09*, volume 5444 of *LNCS*, pages 368–386. Springer, 2009.

[25] Michael Ostrovsky. Stability in supply chain networks. *American Economic Review*, 98(3):897–923, 2008.

[26] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, 1999.

[27] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos Keromytis, and Steve Bellovin. Blind seer: A scalable private dbms. In *2014 IEEE Symposium on Security and Privacy*, pages 359–374. IEEE, 2014.

[28] Benny Pinkas and Tzachy Reinman. Oblivious RAM revisited. In *CRYPTO'10*, volume 6223 of *LNCS*, pages 502–519. Springer, 2010.

[29] Alvin E Roth. The economics of matching: Stability and incentives. *Mathematics of operations research*, 7(4):617–628, 1982.

[30] Abhi Shelat and Chih-hao Shen. Two-output secure computation with malicious adversaries. In *EUROCRYPT'11*, volume 6632 of *LNCS*, pages 386–405. Springer, 2011.

[31] Elaine Shi, T-H Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $\mathcal{O}(\log^3 n)$ worst-case cost. In *ASIACRYPT'11*, volume 7073 of *LNCS*, pages 197–214. Springer, 2011.

[32] Ebrahim M Songhori, Siam U Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. TinyGarble: Highly compressed and scalable sequential garbled circuits. In *IEEE S&P'15*, pages 411–428. IEEE, 2015.

[33] Stable Matching Algorithms. http://www.dcs.gla.ac.uk/research/algorithms/stable/.

[34] Emil Stefanov, Elaine Shi, and Dawn Song. Towards practical oblivious RAM. In *NDSS'12*, 2012.

[35] Synopsys inc. Design compiler. http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler, 2000.

[36] Chung-Piaw Teo, Jay Sethuraman, and Wee-Peng Tan. Gale-Shapley stable marriage problem revisited: Strategic issues and applications. In *Integer Programming and Combinatorial Optimization (IPCO'99)*, volume 1610 of *LNCS*, pages 429–438. Springer, 1999.

[37] Chung-Piaw Teo, Jay Sethuraman, and Wee-Peng Tan. Gale-Shapley stable marriage problem revisited: Strategic

issues and applications. *Management Science*, 47(9):1252–1267, 2001.

[38] Xiao Wang, Hubert Chan, and Elaine Shi. Circuit ORAM: On tightness of the Goldreich-Ostrovsky lower bound. In *ACM CCS'15*, pages 850–861. ACM, 2015.

[39] Andrew Yao. How to generate and exchange secrets. In *FOCS'86*, pages 162–167. IEEE, 1986.

[40] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In *EUROCRYPT'15*, volume 9057 of *LNCS*, pages 220–250. Springer, 2015.

[41] Samee Zahur, Xiao Wang, Mariana Raykova, Adria Gascon, Jack Doerner, David Evans, and Jonathan Katz. Revisiting square root ORAM: Efficient random access in multi-party computation. 2016. In IEEE S&P'16. Online: http://ieee-security.org/TC/SP2016/papers/0824a218.pdf.