Poster: Secure Gale-Shapley
Efficient Stable Matching for Multi-Party Computation

Jack Doerner
University of Virginia
jhd3pa@virginia.edu

David Evans
University of Virginia
evans@virginia.edu

abhi shelat
University of Virginia
abhi@virginia.edu

Abstract—**The execution of stable matching algorithms, such as for matching aspirants to medical residency programs and candidates to sororities, is often outsourced to trusted arbiters in order to preserve the privacy of the participants' preferences. Multi-party computation presents a privacy-preserving alternative that does not require any trusted third party, but executing a stable matching algorithm in a secure multi-party context has thus far been infeasible for nontrivial inputs. We adapt the classic Gale-Shapley algorithm for use in such a context, and show experimentally that our modifications yield a lower complexity and more than an order of magnitude in practical cost improvement over previous techniques.**

I. Introduction

The Gale-Shapley algorithm provides a method for finding a stable pairing between two sets of n members, each member having a ranking of the members of the other set [2]. This concept has proven important to understanding and optimizing market allocations, and versions of it are used in many interesting applications, including matching medical residents to hospitals.

In practice, stable matching processes are often outsourced to a trusted arbiter in order to hide the participants' reported preferences from their counterparties. We consider how to run instances of stable matching using secure computation, in order to obviate the need for a trusted third party while preserving the participants' privacy. As in previous private stable matching work, we assume all members of each of the pairing sets trust one representative, so the protocol can be run as a two-party secure computation.

Executing an algorithm as complex as Gale-Shapley in a secure computation has historically been too expensive to be practical. For example, the protocols of Golle [3] and Franklin et al. [1] required roughly O(n^5) public-key operations and were too complicated to implement. Among other properties, secure computation requires that all data-dependent memory accesses be hidden in order to maintain security.

Recent advances in ORAM design [9] have reduced costs significantly, but oblivious Gale-Shapley remains impractical for any interesting scale. We overcome this by using both state-of-the-art ORAM designs and efficient custom oblivious data structures to adapt the stable matching algorithm for secure computation.

II. Algorithm

We first consider the structure of the textbook Gale-Shapley algorithm, typically presented via a process in which suitors (i.e. members of one set) make proposals to reviewers (members of the other set). The algorithm proceeds through each suitor's preference list from most-preferred to least, swapping between suitors as they become matched or invalidated by other matches. The algorithm makes at most n^2 pairings, but, critically, we cannot determine in advance which suitor's preferences it will be evaluating on any particular iteration, nor how far along that suitor's preference list it will have advanced.

As any iteration could require access to any pairing, the naïve approach requires that the preferences be stored within an ORAM. Consequently, the algorithm must perform n^2 accesses to an ORAM of length n^2. All other ORAMs and queues required by the textbook algorithm are of length n; thus, by finding a way to avoid storing the preferences within an ORAM, we could significantly improve both complexity and real-world performance.

Oblivious Linked Multi-lists. We observe that in this algorithm, each suitor's *individual* preference list is accessed strictly in order, and each element is accessed only once. Furthermore, an oblivious implementation of Gale-Shapley does not require any accesses to be dependent on oblivious conditions (the algorithm must obliviously select *which* list is accessed, but exactly one preference list is always accessed). In short, we need

Pairs	Textbook (s)			Improved (s)	
	Linear	Circuit	Sqrt	Circuit	Sqrt
64	611	1,329	145	88	22
128	9,543	6,164	1,396	392	113
256	152,781	34,747	12,264	1,916	597
512	-	188,973	119,405	12,574	3,252

TABLE I: **Execution Time vs Pair Count.** Values are mean wall-clock times in seconds for full protocol execution including initialization, for implementations using Linear Scan, Circuit ORAM, and Square-Root ORAM.

a data structure that iterates over n elements, in order, while hiding the progress of that iteration.

Instead of using a generic ORAM, we design a new *oblivious linked list* data structure, which satisfies these requirements more efficiently. Unlike an ORAM or an oblivious queue, our oblivious linked list can be accessed in $\Theta(1)$. We can extend this construction to iterate through multiple preference lists by permuting multiple lists together in a single array, and storing their metadata in another data structure. While our solution still requires a general ORAM of n elements in order to maintain the current matches for each of the reviewers, removing the n^2 element ORAM makes a significant improvement in practice and a slight improvement in theory. Furthermore, our new data structure can be initialized by sorting, which dramatically reduces the initialization time compared to previous techniques.

Complexity Analysis. The textbook Gale-Shapley algorithm performs $\Theta(n^2)$ operations upon an n^2 length memory, incurring a total complexity of $\Theta(n^4)$ for implementations based upon linear scan. Using Square-Root ORAM reduces the complexity to $\Theta(n^3\sqrt{\log(n)})$. Our formulation of the Gale-Shapley algorithm performs $\Theta(n^2)$ operations upon an n length memory; using a Square-Root ORAM for array accesses yields a complexity of $\Theta(n^{\frac{5}{2}}\sqrt{\log(n)})$. Using Circuit ORAM [7] reduces the asymptotic complexity to $\Theta(n^2 log^3(n))$, but is less efficient in practice for reasonable values of n because of the high concrete costs of Circuit ORAM.

III. Results

We implemented and benchmarked our algorithm, along with the textbook version, using Obliv-C [8] and the fastest available implementations of Square-Root and Circuit ORAM. We ran it on a pair of Amazon EC2 C4.2xlarge nodes located within the same datacenter and connected via a (measured) 1.03 Gbps link.

Table I presents our findings, which are consistent with our analytical results. At 512×512 members, we achieve more than an order of magnitude improvement over the previous best technique, for a total execution time of one hour, compared to thirty three hours as reported by Zahur et al. [9]. Previously, Terner et al. [6] found that a secure stable matching protocol on 100×100 participants required more than 13 hours, and Keller and Scholl [4] reported a method that can match 128×128 participants in roughly 2.5 hours, but it also requires 1000 processor-days of offline compute time (i.e., work independent of the input).

IV. Future Work

We are interested in producing an secure version of the algorithm used by the National Resident Matching Program to match medical residents to hospitals [5], which places about 20,000 physicians in jobs each year. Although this is more than an order of magnitude larger than the largest benchmarks we have run so far, the hospital-resident matching algorithm does not require that participants rank all of their potential partners — only that they rank a small subset of the most preferred. This significantly reduces the number of available pairings, enabling solutions that are much more efficient than when full preference rankings are used.

References

[1] Matthew Franklin, Mark Gondree, and Payman Mohassel. Improved Efficiency for Private Stable Matching. In *Topics in Cryptology – CT-RSA*, 2007.

[2] David Gale and Lloyd S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[3] Phillippe Golle. A Private Stable Matching Algorithm. In *Tenth International Conference on Financial Cryptography and Data Security*, 2006.

[4] Marcel Keller and Peter Scholl. Efficient, Oblivious Data Structures for MPC. In *Advances in Cryptology – ASIACRYPT*, 2014.

[5] Alvin E. Roth and Elliott Peranson. The Redesign of the Matching Market for American Physicians: Some Engineering Aspects of Economic Design. *American Economic Review*, 1999.

[6] Ben Terner and abhi shelat. Secure Stable Matching. Unpublished result, personal communication, 2014.

[7] Xiao Wang, Hubert Chan, and Elaine Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *ACM Conference on Computer and Communications Security*, 2015.

[8] Samee Zahur and David Evans. Obliv-C: A Lightweight Compiler for Data-Oblivious Computation. Cryptology ePrint Archive, Report 2015/1153, 2015. http://oblivc.org.

[9] Samee Zahur, Xiao Wang, Mariana Raykova, Adrià Gascón, Jack Doerner, David Evans, and Jonathan Katz. Revisiting Square Root ORAM: Efficient Random Access in Multi-Party Computation. In *IEEE Symposium on Security and Privacy*, 2016.