Secure Maximum Weight Matching Approximation on General Graphs

Andreas Brüggemann
Technical University of Darmstadt
Darmstadt, Germany
brueggemann@encrypto.cs.tu-darmstadt.de

Malte Breuer
RWTH Aachen University
Aachen, Germany
breuer@itsec.rwth-aachen.de

Andreas Klinger
RWTH Aachen University
Aachen, Germany
klinger@itsec.rwth-aachen.de

Thomas Schneider
Technical University of Darmstadt
Darmstadt, Germany
schneider@encrypto.cs.tu-darmstadt.de

Ulrike Meyer
RWTH Aachen University
Aachen, Germany
meyer@itsec.rwth-aachen.de

ABSTRACT

Privacy-preserving protocols for matchings on general graphs can be used for applications such as online dating, bartering, or kidney donor exchange. In addition, they can act as a building block for more complex protocols. While privacy-preserving protocols for matchings on bipartite graphs are a well-researched topic, the case of general graphs has experienced significantly less attention so far. We address this gap by providing the first privacy-preserving protocol for maximum weight matching on general graphs. To maximize the scalability of our approach, we compute an $1/2-$approximation instead of an exact solution. For N nodes, our protocol requires $O(N \log N)$ rounds, $O(N^3)$ communication, and runs in only 12.5 minutes for $N = 400$.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; • **Mathematics of computing** → Matchings and factors; *Approximation algorithms.*

KEYWORDS

Matching, General Graphs, Secure Multi-Party Computation, Approximation

ACM Reference Format:
Andreas Brüggemann, Malte Breuer, Andreas Klinger, Thomas Schneider, and Ulrike Meyer. 2022. Secure Maximum Weight Matching Approximation on General Graphs. In *Proceedings of the 2022 Workshop on Privacy in the Electronic Society (WPES '22), November 7, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3559613.3563209

1 INTRODUCTION

Secure Multi-Party Computation (MPC) allows multiple parties to evaluate some function on their private inputs using a distributed protocol that keeps everything that a party cannot derive from its input and output private [29]. It has been used to devise privacy-preserving protocols for a variety of matching problems on graphs, e.g., fingerprint identification [5], kidney donor exchange [6], and matching medical students to medical residency programs or students to schools or universities [10, 16, 23]. Additionally, matching protocols are used as subroutines for tackling entirely different problems as in [28]. While almost all of these approaches are specifically designed for bipartite graphs only, use cases such as online dating, bartering, or kidney exchange can profit from or even require to use general graphs. To the best of our knowledge, the only general graph approach is presented by Breuer et al. [6] and offers a solution to the maximum matching problem that is exact but has $O(N^5)$ communication (i.e., traffic) for N nodes resulting in a run time of more than a day for $N = 40$ already.

Our Contributions. We present the first secure and privacy-preserving approximation protocol for maximum weight matching (MWM) on weighted general graphs. Here, the objective is to find a matching that maximizes the sum of weights of its edges. Our protocol thus approximatively solves a more general problem than the maximum matching problem (i.e., the MWM problem where all weights are 1) for which Breuer et al. [6] present an exact solution. We achieve a high scalability while computing an approximation of the exact solution only. Yet, the computed matching is guaranteed to have at least half of the maximally possible weight, which dependent on the use case may be well-acceptable. The protocol offers a theoretical complexity of $O(N \log N)$ rounds and $O(N^3)$ communication. Assuming a semi-honest adversary and an honest majority, this yields 12.5 minutes run time and 16.4 GB communication for $N = 400$ in its implementation in the MPC framework MP-SPDZ [19]. Note that detailed information about protocol and results is provided in the full version of this paper [7].

Applications. Besides privacy-preserving matching protocols being of interest as building blocks for more complex protocols (i.e., [28]), our approach is also well suited for use cases such as multi-party bartering and online dating. In privacy-preserving multi-party bartering [26–28], parties want to barter their commodities while keeping their offers and demands private. If we only consider the case where in each trade two parties exchange their goods to reduce logistic effort, our MWM protocol can not only be applied but also allows to model the involved parties' satisfaction with chosen trades using weights. While dating is used as a prominent illustration of bipartite matching problems (e.g., in Hall's *marriage theorem* [17]), general graphs are required to abolish the restriction to heterosexuality. Thus, our protocol can be used for online dating regarding monogamous relationships free of further assumptions regarding sexuality while keeping private the users' data like sexuality and sexual preferences that may be subject to social stigma. We argue that for both use cases, the gain of attaining high scalability and hence being able to handle large pools of participants outweighs the losses from not receiving an exact solution.

2 BACKGROUND

2.1 Graph Theory

We consider a weighted, undirected, and finite graph $G = (V, E, \omega)$ with vertices $V = \{0, ..., N-1\}$ where N is the number of nodes. Furthermore, we restrict the weight function ω to map edges in E to positive integer weights only. A matching $\mathfrak{M}$ on G is a subset of its edges such that each node is covered by at most one edge of $\mathfrak{M}$. A maximum weight matching (MWM) is a matching where the sum of its edges' weights is maximal. Furthermore, a c−approximation algorithm for MWM is an algorithm that computes a matching with at least a fraction c of the maximally possible weight.

2.1.1 Approximation Algorithms for Maximum Weight Matching. While for MWM on general graphs there exist Edmond's blossom algorithm [14] and multiple improvements, a lack of scalability has led to the development of several approximation algorithms [11–13, 18, 21, 22]. We observe that higher approximation guarantees generally come at the price of higher run time complexity or more sophisticated algorithms. Such sophisticated algorithms contain nested loops, recursive calls and many conditional blocks that depend on the input which must not be leaked by the control flow. Thus, a translation to an MPC protocol is prone to increase the complexity.

2.2 Secure Multi-Party Computation

2.2.1 Arithmetic Black-Box. To make our high-level protocol independent of low-level primitives, we base it on a generic *arithmetic black-box* (ABB) [9] that allows to secret-share and reconstruct values in the arithmetic or in the binary domain, and that allows to compute linear combinations and products of such secret-shared values. By $[x]$, we denote an arithmetic secret-sharing of value x where the domain is some field or ring.

2.2.2 Building Blocks. We proceed by giving a brief overview over some fundamental building blocks used by our protocol. As we require multiple non-linear functions, e.g., for comparisons we use edaBits [15] and daBits [24] to switch between arithmetic and binary domains which significantly improves practical performance. We use comparison gates from [8, 15]. Furthermore, we require a gate DEMUX^n that given input $[x]$, where n is the maximum value of x, i.e., $0 \leq x \leq n$, returns $[y_i]_{0 \leq i < n}$ s.t. for all $0 \leq i < n$, $[y_i] = [1]$ if $i = x$ and $[y_i] = [0]$ otherwise. This can be constructed from a bit decomposition and the DEMUX protocol in [20].

3 RELATED WORK

To the best of our knowledge, to date there is no privacy-preserving protocol for MWM on general graphs. The protocol of Breuer et al. [6] computes an unweighted maximum matching on a general graph based on an algorithm that is specifically designed for the unweighted case. Furthermore, its results' exactness comes at the cost of low run time performance. There also exist multiple works regarding different matching problems on bipartite graphs [1, 5, 10, 16, 23, 28] which are not applicable in our case as they heavily rely on the graph's bipartiteness. Finally, Araki et al. [3] propose a highly scalable protocol for graph analysis. While they show how to apply it to a selection of graph problems, they depend on existing message-passing algorithms for these problems. As it is unclear how to turn any of the existing matching algorithms into a message-passing algorithm, their protocol appears not to be applicable.

4 PRIVACY-PRESERVING MAXIMUM WEIGHT MATCHING APPROXIMATION

To achieve high scalability, we base our protocol on a greedy 1/2−approximation for MWM (cf. Alg. 1) mentioned by Avis [4] and Preis [22] due to its low concrete run time and a particularly easy structure that is promising for an efficient MPC protocol.

Algorithm 1: Greedy MWM Approximation [22]

```
Input  : Weighted graph $G = (V, E, \omega)$
Output : Matching $\mathfrak{M}$ on $G$
$\mathfrak{M} \leftarrow \emptyset$;
while $E \neq \emptyset$ do
    Let $e \in E$ be an edge of maximal weight;
    $\mathfrak{M} \leftarrow \mathfrak{M} \cup \{e\}$ ;                // add edge $e$ to $\mathfrak{M}$
    $E \leftarrow E \setminus \{f \in E \mid e \cap f \neq \emptyset\}$ ; // del. blocked edges
end
```

Note that the algorithm specifies that in each iteration an edge of maximal weight is chosen but does not specify which edge to choose if there are multiple options. Deriving an unambiguous and deterministic version is easy by defining an arbitrary but fixed ordering $\prec_d$ on all potential edges, i.e., pairs of nodes of the graph. Then, if multiple candidate edges of maximal weight exist in an iteration, we select the candidate that is minimal w.r.t. $\prec_d$.

We proceed by explaining how to implement the deterministic greedy algorithm in §4.1, and then discuss problems resulting from using $\prec_d$ and how to overcome them using randomization in §4.2.

4.1 Deterministic Matching

The input of our protocol for a fixed number of nodes N is a secret-shared adjacency matrix $[\text{weight}_{u,v}]_{0 \leq u < v < N}$. As no information about the edges besides the output may be leaked, our protocol works on all unordered pairs of nodes $u \neq v$. By the definition of the adjacency matrix, potential edges $\{u, v\}$ that are no real edges can be identified by $[\text{weight}_{u,v}]$ being $[0]$. We proceed by discussing how to privately implement each iteration of the greedy algorithm (Alg. 1 using $\prec_d$) which

(1) selects the minimal edge w.r.t. $\prec_d$ among all edges of maximal weight (§4.1.1),
(2) adds the selected edge to the matching (§4.1.2),
(3) and deletes all edges that are incident to the selected one (§4.1.3).

In our approach, we execute a fixed number of iterations. Otherwise, the number of iterations would leak the size of the resulting matching. Thus, our protocol runs for $\lfloor N/2 \rfloor$ iterations which is the maximal possible size of a matching between N nodes. If no real edges remain, further iterations do not extend the matching.

4.1.1 Selecting the Edge of Maximal Weight. The naïve approach to select an edge of maximal weight is to run a linear search over all $O(N^2)$ potential edges which requires $O(N^2)$ comparisons run in sequential order. Optimized techniques that are employed instead in non MPC settings [4] do not allow to be translated to a protocol without an increase of complexity. Instead, we still search over all edges in each iteration, but do so using a parallel reduction tree as in [8]. Over multiple iterations, pairs of edges are internally compared in parallel and only the edges of higher weight are kept for the next iteration until only one edge remains. The number of required iterations, and hence, the resulting round complexity is $O(\log N)$. The communication complexity remains at $O(N^2)$.

Recall that we need to incorporate ordering $\prec_d$ to decide which edge to select if multiple edges share the same maximal weight. This is easily realized by sorting all potential edges by public $\prec_d$ without interaction and then using the reduction operator $\geq$ regarding the edge weights. If no edge remains, the search returns $([N],[N])$.

4.1.2 Adding an Edge to the Matching. We represent each edge $\{u,v\}$ in the matching with a vector $[\text{partner}_i]_{0\leq i<N}$ by setting $[\text{partner}_u]$ to $[v]$ and $[\text{partner}_v]$ to $[u]$. By $[\text{partner}_u] = [N]$ we encode that node u is matched to no other node. Thus, all secret-shared values $[\text{partner}_u]$ are initialized to $[N]$.

If a real edge $\{a,b\}$ is chosen in some iteration, it is not possible to directly access $[\text{partner}_a]$ and $[\text{partner}_b]$ as this would leak the chosen edge. Instead, we use gate DEMUX^N to generate two indicator vectors $[\text{indicator\_a}_i]_{0\leq i<N}$ and $[\text{indicator\_b}_i]_{0\leq i<N}$ that contain value $[1]$ at index a respectively b and value $[0]$ at other indices. Then, we add $([b]-N)\cdot[\text{indicator\_a}_i]_{0\leq i<N}$ and $([a]-N)\cdot[\text{indicator\_a}_i]_{0\leq i<N}$ to $[\text{partner}_i]_{0\leq i<N}$. If no edge was found, then $[a]=[b]=[N]$ so that no entry of the output vector is changed. The total complexity is $O(N)$ communication as each vector entry is accessed to hide which one is changed, and $O(\log\log N)$ rounds due to the required call to DEMUX^N.

4.1.3 Deleting Incident Edges. Deleting all edges incident to a real edge $\{a,b\}$ again requires to access all edges in order to not leak any information about $\{a,b\}$. Recall that to delete an edge, it is sufficient to set the corresponding secret-shared weight to zero. Now, we simply add the indicator vectors from the previous step without interaction to obtain a binary vector $[\text{indicator}_i]_{0\leq i<N}$ that contains $[1]$ exactly at indices a and b. Thus, it now suffices to multiply each weight $[\text{weight}_{u,v}]$ by $(1-[\text{indicator}_u])\cdot(1-[\text{indicator}_v])$. This step has a complexity of $O(N^2)$ communication in $O(1)$ rounds.

4.1.4 Complexity. Recall that we run $\lfloor N/2 \rfloor$ iterations. Thus, the total communication complexity is $O(N^3)$ in $O(N\log N)$ rounds.

4.2 Randomization

Recall that we introduced the ordering $\prec_d$ to fix which edge to select if multiple candidates share the same weight. Depending on the use case, this can introduce significant problems as the ordering induces advantages for some edges in being selected which are not justified by the edge weights. Now, for instance in multi-party bartering or online dating, nodes represent people who compete in getting matched. Ideally, no person should gain an unfair advantage solely resulting from how people are mapped to nodes together with the arbitrary usage of ordering $\prec_d$.

4.2.1 Idea. One way of circumventing such bias is to randomize the mapping between people and nodes. With our input being an already generated adjacency matrix of a graph G, we realize an equivalent functionality by shuffling all nodes, i.e.,

(1) select a permutation π on N nodes uniformly at random,
(2) apply π to G as a graph isomorphism to obtain G',
(3) run the greedy algorithm using $\prec_d$ on G', and
(4) apply π^{-1} to the resulting matching.

The resulting probabilistic functionality $\mathcal{F}$ is *invariant under node permutation*, i.e., for each graph G with N nodes and each graph isomorphism τ on N nodes, distributions $\mathcal{F}(G)$ and $\tau^{-1}(\mathcal{F}(\tau(G)))$ are equal. In other words, the value that represents a node does not influence the outcome and hence, the mapping from persons to nodes does not influence the probability that a particular person is matched to another person.

4.2.2 Implementation. To implement said functionality $\mathcal{F}$, we first select the graph isomorphism π, i.e., a random permutation over N elements. We refer to the full version of our paper [7] for a discussion on why this permutation must be kept private. It is easy to see that applying π to G represented by an adjacency matrix is possible by first permuting its rows and then permuting its columns by π. Then, we run the deterministic greedy protocol from §4.1 and finally apply π^{-1} to the resulting matching. Regarding the last step, recall that we encode the matching by setting $[\text{partner}_u]$ to $[v]$ and vice versa when u and v are matched. As we run the greedy algorithm on $\pi(G)$, we now have that $[\text{partner}_{\pi(u)}] = [\pi(v)]$ instead. First, we permute $[\text{partner}_u]_{0\leq u<N}$ by π^{-1} so that we have $[\text{partner}_u] = [\pi(v)]$. Then, we compute the dot product of $\text{DEMUX}^N([\pi(v)])$ and $[\pi^{-1}(i)]_{0\leq i<N}$ where the second vector is obtained by applying π to $[i]_{0\leq i<N}$. It is easy to verify that the dot product evaluates to $[\pi^{-1}(\pi(v))] = [v]$. This can easily be extended to also correctly work for unmatched nodes [7].

It remains to show how to obliviously permute matrices and vectors by a private random permutation π and its inverse. MP-SPDZ [19] already provides such functionality by utilizing an oblivious Waksman network [25] as demonstrated by Zahur et al. [30]. The most expensive permutation operation then is to permute the adjacency matrix which requires $O(pN^2\log N)$ communication in $O(p \log N)$ rounds where p is the number of MPC parties.[1] We note that, e.g., for $p = 3$ parties there exist faster appropriate shuffling approaches as in [3] with $O(N^2)$ communication in constant rounds.

4.2.3 Alternative Randomization. In the full version [7] we discuss a second approach that yields somewhat higher unbiasedness guarantees by enforcing that in each iteration of the greedy algorithm, the selection is done uniformly at random among the edges of maximal weight. Interestingly, this is possible using a single oblivious random permutation on all edges. As the resulting protocol's run time is $\approx 2.9\times$ of the randomized protocol introduced in §4.2.1, we do not discuss it any further in this paper.

5 EVALUATION

We evaluated run time and network traffic for our MWM approximation protocol in its randomized variant (cf. §4.2) using the MPC framework MP-SPDZ [19] version 0.3.2[2]. As ABB implementation, we used the three-party protocol by Araki et al. [2] with domains $\mathbb{F}_q$ (arithmetic) and $\mathbb{F}_2$ (binary). The protocol provides security against a semi-honest adversary given an honest majority. We ran the protocol on an AMD EPYC 7702P with a base clock of 2.0 GHz, one core assigned to each party, and a simulated LAN with a bandwidth of 1 GBit/s and a round-trip time of 1 ms. For different numbers of nodes N up to $N = 400$, we executed the protocol 10 times each and took the arithmetic means as final results.[3]

Recall that randomization was specifically motivated by nodes representing certain entities that want to get matched. To give a pointer on the cost of matrix generation, we benchmarked our protocol together with an exemplary computation of the adjacency matrix. We let each node have an associated input vector of dimension 50. Then, we consider more similar vectors for two nodes as a better match. We compute their squared Euclidean distance, connect them by an edge if the distance is below a certain threshold and then set the weight to a fixed offset minus the squared distance.

5.1 Results

In Fig. 1 we give the average run times of our randomized protocol (cf. §4.2) including the previously introduces matrix generation. Further benchmark results are given in §A. The average run times are $\approx$14.0 s for $N = 100$ nodes, $\approx$5.1 min for $N = 300$, and $\approx$12.5 min for $N = 400$. The total traffic generated by all three parties together amounts to $\approx$274.3 MB for $N = 100$, $\approx$7.0 GB for $N = 300$, and $\approx$16.4 GB for $N = 400$. Run time and traffic scale polynomially bounded which matches our previous theoretical complexity analysis. We deduce that our protocol scales well where, depending on the use case, matching thousands of nodes may still be feasible.

We also measured the share that each phase of the protocol has in the overall run time. The protocol phases that we distinguish between are the previously described example adjacency matrix generation, applying π to the adjacency matrix (cf. §4.2.1), computing a matching (cf. §4.1), and applying π^{-1} to the result (cf. §4.2.1).

[1]Note that a factor of N regarding communication comes from each oblivious swap being between two columns or rows containing N values each.
[2]https://github.com/data61/MP-SPDZ/releases/tag/v0.3.2
[3]Loops in MP-SPDZ are entirely unrolled at compile-time if they are fully parallelized leading to immense RAM utilization. Thus, a restriction to 200 GiB of RAM prevented us from benchmarking our protocol for $N > 400$.

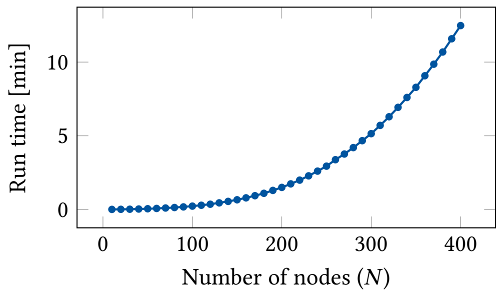

Figure 1: Run time of the randomized protocol (cf. §4.2).

Generating the adjacency matrix takes $\approx$38.2% of the total run time for $N = 10$, drops below 2% for $N = 160$ and is $\approx$0.7% for $N = 400$. In absolute values, its run time increases to $\approx$5.0 s for $N = 400$. Shuffling the adjacency matrix costs $\approx$4.5% of the total run time for $N = 10$ decreasing down to $\approx$0.6% for $N = 400$ corresponding to $\approx$4.8 s. Finally, reversing the shuffling never takes more than 0.2 s.

The matching phase clearly dominates the total run time for increasing N taking $\approx$54.8% of it for $N = 10$ but already more than 90% for $N \geq 60$ and more than 95% for $N \geq 100$. Its share increases up to $\approx$98.7% for $N = 400$. This is due to it having the highest asymptotic complexity w.r.t. N. Note that the phase is the deterministic base protocol from §4.1. This shows that achieving invariance under node permutation has only negligible overhead.

6 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we have presented a secure protocol for maximum weight matching 1/2−approximation. We have demonstrated our protocol's scalability in theory and in practice and furthermore identified how to overcome a bias problem using randomization.

We see three main directions for further research. First, our protocol's performance may be evaluated using an ABB for a different setting, e.g., for malicious security. Second, new protocols can be developed using either other approximations that offer better quality guarantees or even exact algorithms as their basis. Here it is of special interest how much performance one must pay in order to get better results, i.e., investigating the trade-off between quality of results and run time. Third, while our approximation guarantees results to be at least half as good as exact ones, their concrete quality can be significantly higher depending on the properties of the input graphs. Thus, an empirical study of the resulting matchings' quality could be of interest when using our protocol for a specific use case.

ACKNOWLEDGMENTS

This project received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). In addition, it was funded by the Deutsche Forschungsgemeinschaft (DFG) within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230, and projects 419340256 and 2236/1. It received funding from the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

REFERENCES

[1] Balamurugan Anandan and Chris Clifton. 2017. Secure Minimum Weighted Bipartite Matching. In *2017 IEEE Conference on Dependable and Secure Computing*. 60–67. https://doi.org/10.1109/DESEC.2017.8073798

[2] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. 805–817. https://doi.org/10.1145/2976749.2978331

[3] Toshinori Araki, Jun Furukawa, Kazuma Ohara, Benny Pinkas, Hanan Rosemarin, and Hikaru Tsuchida. 2021. Secure Graph Analysis at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21)*. 610–629. https://doi.org/10.1145/3460120.3484560

[4] David Avis. 1983. A Survey of Heuristics for the Weighted Matching Problem. *Networks* 13, 4 (1983), 475–493. https://doi.org/10.1002/net.3230130404

[5] Marina Blanton and Siddharth Saraph. 2015. Oblivious Maximum Bipartite Matching Size Algorithm with Applications to Secure Fingerprint Identification. In *Computer Security – ESORICS 2015 (Lecture Notes in Computer Science)*. 384–406. https://doi.org/10.1007/978-3-319-24174-6_20

[6] Malte Breuer, Ulrike Meyer, and Susanne Wetzel. 2022. Privacy-Preserving Maximum Matching on General Graphs and its Application to Enable Privacy-Preserving Kidney Exchange. In *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy (CODASPY '22)*. 53–64. https://doi.org/10.1145/3508398.3511509

[7] Andreas Brüggemann, Malte Breuer, Andreas Klinger, Thomas Schneider, and Ulrike Meyer. 2022. Secure Maximum Weight Matching Approximation on General Graphs. Cryptology ePrint Archive, Paper 2022/1173. https://eprint.iacr.org/2022/1173.

[8] Octavian Catrina and Sebastiaan de Hoogh. 2010. Improved Primitives for Secure Multiparty Integer Computation. In *Security and Cryptography for Networks (Lecture Notes in Computer Science)*. 182–199. https://doi.org/10.1007/978-3-642-15317-4_13

[9] Ivan Damgård and Jesper Buus Nielsen. 2003. Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In *Advances in Cryptology - CRYPTO 2003 (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg, 247–264. https://doi.org/10.1007/978-3-540-45146-4_15

[10] Jack Doerner, David Evans, and abhi shelat. 2016. Secure Stable Matching at Scale. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. 1602–1613. https://doi.org/10.1145/2976749.2978373

[11] Doratha E Drake and Stefan Hougardy. 2003. A Simple Approximation Algorithm for the Weighted Matching Problem. *Inform. Process. Lett.* 85, 4 (Feb. 2003), 211–213. https://doi.org/10.1016/S0020-0190(02)00393-9

[12] Doratha E. Drake Vinkemeier and Stefan Hougardy. 2005. A Linear-Time Approximation Algorithm for Weighted Matchings in Graphs. *ACM Transactions on Algorithms* 1, 1 (July 2005), 107–122. https://doi.org/10.1145/1077464.1077472

[13] Ran Duan and Seth Pettie. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. ACM* 61, 1 (Jan. 2014), 1:1–1:23. https://doi.org/10.1145/2529989

[14] Jack Edmonds. 1965. Maximum Matching and a Polyhedron With 0,1-Vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics* 69B (Jan. 1965), 125. https://doi.org/10.6028/jres.069B.013

[15] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. 2020. Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits. In *Advances in Cryptology – CRYPTO 2020 (Lecture Notes in Computer Science)*. 823–852. https://doi.org/10.1007/978-3-030-56880-1_29

[16] Philippe Golle. 2006. A Private Stable Matching Algorithm. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science)*. 65–80. https://doi.org/10.1007/11889663_5

[17] P. Hall. 1935. On Representatives of Subsets. *Journal of the London Mathematical Society* s1-10, 1 (1935), 26–30. https://doi.org/10.1112/jlms/s1-10.37.26

[18] Sven Hanke and Stefan Hougardy. 2010. New Approximation Algorithms for the Weighted Matching Problem. http://www.or.uni-bonn.de/%7Ehougardy/paper.html Report No. 101010, Research Institute for Discrete Mathematics, University Of Bonn.

[19] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1575–1590. https://doi.org/10.1145/3372297.3417872

[20] John Launchbury, Iavor S. Diatchki, Thomas DuBuisson, and Andy Adams-Moran. 2012. Efficient Lookup-Table Protocol in Secure Multiparty Computation. *ACM SIGPLAN Notices* 47, 9 (Sept. 2012), 189–200. https://doi.org/10.1145/2398856.2364556

[21] Seth Pettie and Peter Sanders. 2004. A Simpler Linear Time $2/3-\epsilon$ Approximation for Maximum Weight Matching. *Inform. Process. Lett.* 91, 6 (Sept. 2004), 271–276. https://doi.org/10.1016/j.ipl.2004.05.007

[22] Robert Preis. 1999. Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs. In *STACS 99 (Lecture Notes in Computer Science)*. 259–269. https://doi.org/10.1007/3-540-49116-3_24

[23] M. Sadegh Riazi, Ebrahim M. Songhori, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. 2017. Toward Practical Secure Stable Matching. *Proceedings on Privacy Enhancing Technologies* 2017, 1 (Jan. 2017), 62–78. https://doi.org/10.1515/popets-2017-0005

[24] Dragos Rotaru and Tim Wood. 2019. MArBled Circuits: Mixing Arithmetic and Boolean Circuits with Active Security. In *Progress in Cryptology – INDOCRYPT 2019 (Lecture Notes in Computer Science)*. 227–249. https://doi.org/10.1007/978-3-030-35423-7_12

[25] Abraham Waksman. 1968. A Permutation Network. *J. ACM* 15, 1 (Jan. 1968), 159–163. https://doi.org/10.1145/321439.321449

[26] Stefan Wüller, Ulrike Meyer, and Susanne Wetzel. 2017. Privacy-Preserving Multi-Party Bartering Secure Against Active Adversaries. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. 205–20509. https://doi.org/10.1109/PST.2017.00032

[27] Stefan Wüller, Ulrike Meyer, and Susanne Wetzel. 2017. Towards Privacy-Preserving Multi-party Bartering. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science)*. 19–34. https://doi.org/10.1007/978-3-319-70278-0_2

[28] Stefan Wüller, Michael Vu, Ulrike Meyer, and Susanne Wetzel. 2017. Using Secure Graph Algorithms for the Privacy-Preserving Identification of Optimal Bartering Opportunities. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society (WPES '17)*. 123–132. https://doi.org/10.1145/3139550.3139557

[29] Andrew C. Yao. 1982. Protocols for Secure Computations. In *23rd Annual Symposium on Foundations of Computer Science*. 160–164. https://doi.org/10.1109/SFCS.1982.38

[30] Samee Zahur, Xiao Wang, Mariana Raykova, Adrià Gascón, Jack Doerner, David Evans, and Jonathan Katz. 2016. Revisiting Square-Root ORAM: Efficient Random Access in Multi-party Computation. In *2016 IEEE Symposium on Security and Privacy (S&P)*. 218–234. https://doi.org/10.1109/SP.2016.21

A DETAILED RUN TIME AND TRAFFIC

Table 1: Run time and global communication of the randomized protocol (cf. §4.2). Run time is divided into the generation of the adjacency matrix, shuffling of G by permutation π, the matching phase itself (cf. §4.1), and reversing the shuffling using π^{-1} on the output.

N	Run time					Comm.
	Gen.	π	Match	π^{-1}	**Total**	
50	0.3 s	0.1 s	3.0 s	0.0 s	3.3 s	41.1 MB
100	0.4 s	0.2 s	13.3 s	0.0 s	14.0 s	274.3 MB
150	0.8 s	0.7 s	38.2 s	0.1 s	39.8 s	909.4 MB
200	1.2 s	0.9 s	1.5 min	0.1 s	1.5 min	2.1 GB
250	2.1 s	1.1 s	2.9 min	0.1 s	2.9 min	4.0 GB
300	2.9 s	3.4 s	5.0 min	0.1 s	5.1 min	7.0 GB
350	4.0 s	4.1 s	8.1 min	0.2 s	8.3 min	11.1 GB
400	5.0 s	4.8 s	12.3 min	0.2 s	12.5 min	16.4 GB