Privacy-Preserving Maximum Matching on General Graphs and its Application to Enable Privacy-Preserving Kidney Exchange

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

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

Susanne Wetzel
swetzel@stevens.edu
Stevens Institute of Technology
Hoboken, NJ, USA

ABSTRACT

To this day, there are still some countries where the exchange of kidneys between multiple incompatible patient-donor pairs is restricted by law. Typically, legal regulations in this context are put in place to prohibit coercion and manipulation in order to prevent a market for organ trade. Yet, in countries where kidney exchange is practiced, existing platforms to facilitate such exchanges generally lack sufficient privacy mechanisms. In this paper, we propose a privacy-preserving protocol for kidney exchange that not only addresses the privacy problem of existing platforms but also is geared to lead the way in overcoming legal issues in those countries where kidney exchange is still not practiced. In our approach, we use the concept of secret sharing to distribute the medical data of patients and donors among a set of computing peers in a privacy-preserving fashion. These computing peers then execute our new Secure Multi-Party Computation (SMPC) protocol among each other to determine an optimal set of kidney exchanges. As part of our new protocol, we devise a privacy-preserving solution to the maximum matching problem on general graphs. We have implemented the protocol in the SMPC benchmarking framework MP-SPDZ and provide a comprehensive performance evaluation. Furthermore, we analyze the practicality of our protocol when used in a dynamic setting where patients and donors arrive and depart over time) based on a data set from the United Network for Organ Sharing.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Security and privacy** → **Privacy-preserving protocols**; **Privacy protections**; • **Social and professional topics** → **Patient privacy**.

KEYWORDS

Kidney Exchange, Privacy, Secure Multi-Party Computation, Matching Algorithms

ACM Reference Format:
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 Twelveth ACM Conference on Data and Application Security and Privacy (CODASPY '22), April 24–27, 2022, Baltimore, MD, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3508398.3511509

CODASPY '22, April 24–27, 2022, Baltimore, MD, USA

ACM ISBN 978-1-4503-9220-4/22/04…$15.00
https://doi.org/10.1145/3508398.3511509

1 INTRODUCTION

According to the World Health Organization kidney disease is the 10th most common cause of death worldwide [35]. While the preferable treatment for patients with kidney disease is transplantation, waiting lists for post-mortem kidney donation are very long, e.g., more than 90000 patients are currently on the waiting list for a kidney transplant in the US [37]. An alternative to post-mortem donation is to find a friend or relative who is willing to donate one of their kidneys. While many patients find such a living donor, often this donor is not compatible with the patient's medical characteristics.

A recent development to solve this problem is kidney exchange. The idea is to consider multiple patients with incompatible living donors (also referred to as incompatible patient-donor pairs) and find exchanges among them (e.g., crossover exchanges where the donor of one patient-donor pair donates to the patient of another pair and vice versa). While there are many countries around the world where kidney exchange is already practiced, to date there are still some countries (e.g., Germany[1]) where kidney exchange faces legal obstacles. This is mainly due to fear of manipulation, corruption, and coercion. In those countries where kidney exchange is already practiced, it is usually organized by large centralized platforms which are responsible for the computation of the exchanges. Thus, the patient-donor pairs have to fully trust these platforms not only with their medical data but also with the correct and fair computation of the exchanges. Furthermore, such a centralized approach makes the platforms a desirable target for attackers. Compromising a single entity allows them to access the medical data of all pairs or to manipulate the computation of the exchanges which could have life-threatening consequences for the involved patients.

To mitigate these shortcomings, we devise a decentralized approach for kidney exchange where medical data and exchange computation are distributed among multiple parties. This ensures privacy of the data as well as protection against manipulation, corruption, and coercion for existing platforms. Also, this may lead to an adoption of kidney exchange in countries where it is still off limits today. Specifically, this work provides four main contributions:

First, we devise the first privacy-preserving protocol for crossover kidney exchange based on secret sharing which at the same time is the first privacy-preserving protocol for kidney exchange with polynomial communication complexity. At the heart of our protocol is a privacy-preserving solution to the maximum matching problem on general graphs. To the best of our knowledge, we are the first to solve the maximum matching problem on general graphs in a

[1] §8 Transplantationsgesetz (German Transplantation Law)

privacy-preserving fashion which we consider to be of independent interest beyond the use case of kidney exchange.

Second, we implement our protocol in the state-of-the-art SMPC benchmarking framework MP-SPDZ [31] and carry out a comprehensive performance analysis. The source code is published in [18].

Third, we compare the implementation of our protocol to the only other privacy-preserving protocol for kidney exchange [19] known to date. To this end, we implement the protocol from [19] in MP-SPDZ based on secret sharing and thereby significantly improve its performance. While the protocol from [19] solves a more general problem than our newly developed protocol, we show that our protocol considerably outperforms the protocol from [19] for the special case of crossover exchanges.

Fourth, we establish the practicality of our protocol when used as part of a dynamic kidney exchange platform where patient-donor pairs arrive and leave over time. To this end, we run simulations based on real-world data from the United Network for Organ Sharing (UNOS) which is a major kidney exchange platform in the US. We compare the performance of a kidney exchange platform using our protocol to a conventional (non-privacy-preserving) approach and measure the number of transplants for both scenarios. As kidney exchange platforms differ substantially around the world, we run simulations for a wide range of parameters reflecting many different characteristics of various kidney exchange platforms. Our simulations show that the performance difference between our privacy-preserving approach and the conventional approach is negligible for those parameters that are most likely to occur in practice.

2 INTUITION AND APPROACH

Traditional Platforms. Typically, at the core of today's traditional kidney exchange platforms (cf. Figure 1) is the pool of those patient-donor pairs that are currently registered and seek for an exchange partner. Usually, a patient-donor pair is associated with a transplant center or hospital which registers it with a central platform by providing them the medical data of both patient and donor. Operators of these central platforms then carry out so-called *match runs* at specific points in time (denoted as t and $t+1$ in Figure 1).

A match run corresponds to solving the *Kidney Exchange Problem* (KEP) among all patient-donor pairs within the pool. The KEP is defined as finding a set of exchanges that maximizes the number of patients that can receive a transplant [1]. Usually, it is modeled as a graph problem where each patient-donor pair corresponds to one node and an edge is added between two nodes if the donor of the first pair can donate to the patient of the second pair. Exchanges are then computed such that the donor of a patient-donor pair only donates her kidney if the corresponding patient also receives a compatible kidney transplant in return.

The simplest form of such an exchange is a *crossover exchange* where the donor of an incompatible patient-donor pair A donates to the patient of another incompatible patient-donor pair B and vice versa. Larger exchange cycles where the donor of a pair always donates to the patient of the succeeding pair are also possible. However, large cyclic exchanges require a lot of medical resources as all involved transplants have to be carried out simultaneously to prevent a donor from backing out after the corresponding patient already received a transplant. Therefore, most countries only consider

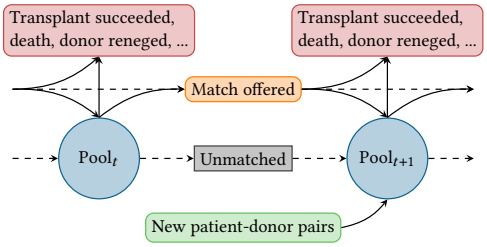

Figure 1: Model of a dynamic kidney exchange platform, adapted from [23].

exchange cycles of maximum size three (e.g., Netherlands, Spain, UK, and the major exchange platforms in the US [7, 11]) or even only crossover exchanges (e.g., all countries of Scandinavia [5, 11]). Besides such cyclic exchanges, also chains initiated by an altruistic donor without a corresponding patient are possible. However, they are still not allowed in many countries [11].

After the execution of a match run, those pairs that were matched are removed from the pool and informed of the computed exchange partner. If the match results in a transplant, the pairs leave the platform. If the match fails, they reenter the pool. Pairs that were not matched, simply remain in the pool until the next match run is executed. In between two match runs new patient-donor pairs may arrive at the pool and pairs that are already in the pool may leave for various reasons (e.g., death or illness of patient or donor).

Privacy Concerns. The traditional centralized setting exhibits two major shortcomings. First, the patient-donor pairs (or their representative hospitals) have to completely trust the single operator of the centralized platform in computing the exchanges correctly and treating each patient-donor pair equally. Second, an attacker only has to compromise a single entity in order to gain complete control over the exchange computation as well as the sensitive medical data of all patients and donors registered with the platform.

Intuition for the Privacy-Preserving Approach. Pursuing a decentralized approach based on *Secure Multi-Party Computation* (SMPC) allows for the distribution of the exchange computation among multiple parties, making it privacy-preserving. In our approach we substitute the central kidney exchange platform by multiple so-called *computing peers* who execute an SMPC protocol among each other to compute the exchanges in a distributed fashion. The patient-donor pairs are then also referred to as *input peers* as they (or their representative hospitals) just send their input (medical data) to the computing peers. They use secret sharing to guarantee that a single computing peer does not gain any knowledge on the actual medical data of any patient or donor. After the protocol execution, the computing peers then send the shares of the computed exchanges to the corresponding input peers. During the protocol execution, the computing peers do not learn anything about the medical data of the patient-donor pairs or the computed exchanges.

Choosing the optimal number of computing peers here means a trade-off between privacy and performance. On one hand, a larger number of computing peers, increases the number of peers that need to be corrupted in order to compromise the privacy of the

patient-donor pairs. On the other hand, a smaller number of computing peers decreases the communication overhead induced by the protocol and thus also decreases the protocol runtime. For our runtime measurements (cf. Section 5.3), we use three computing peers which is usually considered a good trade-off (e.g., [15, 16]). The computing peers could be governmental institutions, transplant centers, or institutions that cover the patients' interests.

For the actual SMPC protocol that is executed between the computing peers we then focus on pure crossover exchange. This allows us to reduce the KEP to finding a maximum matching in a general graph where an undirected edge is added between two nodes if the donor of one pair is compatible with the patient of the other and vice versa, i.e., if a crossover exchange between the two pairs is possible. Thus, computing a maximum matching is equivalent to maximizing the number of crossover exchanges. This is a huge advantage as the KEP is NP-complete if the cycle size is restricted to three or larger [1]. Solving the maximum matching problem on general graphs instead is possible in polynomial time (e.g., [25]). Using an efficient algorithm as basis for a privacy-preserving protocol is important as such protocols in general introduce additional overhead to the original algorithm. Another advantage of crossover exchanges is that some countries (e.g., Germany) only allow a transplantation if patient and donor know each other well which is much easier to guarantee for crossover exchanges than for larger cycles. [2]

3 PRELIMINARIES

In this section, we establish the relevant background for our privacy-preserving protocol for kidney exchange including basic terminology from graph theory (Section 3.1) and a description of the considered setting for secure multi-party computation (Section 3.2).

3.1 Graph Theory

An *undirected graph* is denoted as $G = (V, E)$ where V is a finite set of nodes and $(u, v) \in E$ indicates an undirected edge between nodes $u, v \in V$ with $u \neq v$. Two nodes u and v are called *adjacent* if there is an edge $(u, v) \in E$ and an edge $(u, v) \in E$ is said to be *incident* to node u and v. A *matching* is a set of edges M such that each node $v \in V$ is incident to at most one edge in M. An edge $(u, v) \in M$ is called a *matched* edge and an edge $(u, v) \notin M$ is called an *unmatched* edge. A node $v \in V$ that is incident to an edge in M is referred to as a *matched* node and the nodes u, v of an edge $(u, v) \in M$ are called *mates*. All nodes $v \in V$ that are not incident to an edge in M are called *exposed* nodes. A *path* is a sequence of pairwise different nodes $p = v_1, ..., v_k$ such that for all v_i with $i \in \{1, ..., k-1\}$ it holds that $(v_i, v_{i+1}) \in E$. A path P is called *alternating* w.r.t. a matching M if the edges in P are alternately matched and unmatched edges. We refer to an alternating path P beginning at an exposed node v_1 and ending at an exposed node v_2 (with $v_1 \neq v_2$) as an *augmenting* path. A *cycle* is a path with $(v_k, v_1) \in E$. We call a cycle *odd* if it contains an odd number of edges and *even*, otherwise. A fundamental property for matching algorithms is Berge's Theorem.

Definition 3.1. (Berge's Theorem [9]) *A matching M in a graph G is called a maximum matching if and only if G has no augmenting path with respect to M.*

[2] §8 Transplantationsgesetz (German Transplantation Law)

Intuitively, Berge's Theorem states that if there is an augmenting path $P = (v_1, ..., v_k)$ with respect to M in the graph G, then it is possible to increase the matching by adding all edges $(v_i, v_{i+1}) \notin M$ with $i \in \{1, ..., k-1\}$ to M and removing all edges $(v_i, v_{i+1}) \in M$ with $i \in \{1, ..., k-1\}$ from M. Thereby, we obtain a new matching M' with $|M'| = |M| + 1$. Thus, M cannot be a maximum matching.

3.2 Secure Multi-Party Computation

Generally speaking, SMPC allows a set of n parties to jointly compute a functionality in a distributed fashion such that no party learns anything beyond its private input and output and what can be deduced from both. We adopt the well-established approach (e.g., [15, 16]) of distinguishing between two sets of parties, so-called input peers and computing peers. While the former provide input to the functionality and receive their corresponding output, the latter execute the actual protocol among each other. In our protocol (cf. Section 5.2), the input peers correspond to the patient-donor pairs and the computing peers can be governmental institutions, transplant centers, or institutions that cover the patients' interests.

To realize SMPC, we use Shamir's (t, n) threshold secret sharing scheme [39] which allows the input peers to share a secret value x among n computing peers such that possession of a subset of at most t shares does not reveal any information on x itself. Restoring the secret x is possible iff at least $t + 1$ computing peers collaborate.

We require that all computations are carried out over a finite field $\mathbb{Z}_p$ for a prime $p > n$ and we represent a negative value $-x$ as $p - x \in \mathbb{Z}_p$. This allows for the correct addition and multiplication of negative values. We denote that a value x is secret shared among the n computing peers by $[x]$. For a vector $[V]$ of shared values, we write $[V](i)$ to denote the i-th entry of the vector. Analogously, we write $[M](i, j)$ for the entry in the i-th row and the j-th column of a matrix $[M]$ of shared values.

Building Blocks. Using Shamir's secret sharing scheme enables the computing peers to compute any linear combination of secret shared values locally whereas for multiplication the peers have to interact with each other by means of a multiplication protocol which can be constructed such that it runs in a constant number of rounds [8]. For the sake of readability we use the infix notation $[x] \leftarrow [y] \cdot [z]$ to denote the execution of a multiplication protocol. Note that we measure the communication complexity of an SMPC protocol in terms of the number of calls to the multiplication protocol. The round complexity refers to the number of messages transmitted during the SMPC protocol.

In addition to multiplication, we require several other primitives as building blocks for our main protocol (cf. Section 5.2). Secure comparison of two shared values $[x]$, $[y]$ is denoted by $[x] \overset{?}{<} [y]$ and can be implemented such that it has linear communication and constant round complexity [20]. Similarly, protocols for equality and inequality test can be constructed. We also require a protocol for conditional selection, i.e., given a secret shared bit $[b]$, choose $[x]$ if b is equal to 1 and $[y]$, otherwise. Such a protocol can be realized by computing $[b] \cdot ([x] - [y]) + [y]$ and we denote it by $[b]\ ?\ [x]\ :\ [y]$. Finally, we sometimes require to access or update an entry $[V]([i])$ of a vector where the index $[i]$ is a secret value. Such a secret index is first translated into a vector $[I]$ containing

the value 1 at position i and zeros at all other positions. This can be done in linear communication and constant round complexity [33]. Based on the index vector $[I]$, entry $[V]([i])$ can then be accessed by computing the inner product of $[V]$ and $[I]$ and updated by computing $[V](j) \leftarrow [I](j)\ ?\ [x]\ :\ [V](j)$ for $j \in \{1, |V|\}$. Both operations require $|[V]|$ multiplications which can be executed in parallel. Thus, accessing or updating an entry $[V]([i])$ overall requires linear communication and constant round complexity.[3]

Security. For our protocol (cf. Section 5.2), we consider security in the *semi-honest model* where the corrupted computing peers strictly follow the protocol specification but try to learn as much as possible on the honest computing peers' input. Security in this sense can be considered sufficient for kidney exchange as it prevents an adversary from learning the medical data of the patient-donor pairs and thus from influencing the computed exchanges in any meaningful way. Note that for the input peers it is guaranteed that they do not learn anything beyond their private input and output as they do not participate in the actual protocol execution. Furthermore, we assume an honest majority of computing peers as well as encrypted and authenticated channels between the computing peers and between computing peers and input peers.

To prove the security of our main protocol (cf. Section 5.2), we use the standard simulation-based security paradigm which intuitively states that the parties do not learn anything beyond their private input and output. For a detailed definition of the security setting and the adversary models, we refer the reader to [29].

4 RELATED WORK

In this section, we review pertinent related work: state-of-the-art algorithms for maximum matching on general graphs (Section 4.1); existing privacy-preserving protocols for matching on bipartite graphs (Section 4.2); and the only other privacy-preserving protocol known to date in the context of kidney exchange (Section 4.3).

4.1 Matching Algorithms

The first efficient algorithm to solve the maximum matching problem on general graphs was devised by Edmonds [25] and is based on Berge's Theorem (Definition 3.1). It starts with an initial matching M which can also be the empty matching. Then, it tries to find an augmenting path P with respect to M and revert the path such that a new matching M' of cardinality $|M|+1$ is obtained. This is repeated until no further augmenting path can be found. Due to Berge's Theorem the thus obtained matching is a maximum matching.

The major challenge when designing a matching algorithm in this way is the handling of so-called *blossoms*. A blossom is a cycle of odd length in the graph that contains alternating matched and unmatched edges. The problem with blossoms is that it is not trivial how to choose the matching edges of the blossom such that a possibly existing augmenting path is indeed found. Edmonds proposes to shrink a blossom into a single supernode and then continue the algorithm on the shrunken graph. After finding an augmenting path in this adapted graph, the supernode is expanded again and the matching edges of the blossom are chosen accordingly. While this concept is quite simple, the shrinking and expanding of blossoms are computationally rather complex operations leading to a complexity of $O(|V|^4)$ for Edmonds' algorithm.

The approach by Edmonds has been refined many times by introducing different techniques for handling blossoms, e.g., labeling techniques that allow to avoid the explicit shrinking and expanding of blossoms (e.g., [27, 36]). The most efficient solutions for the maximum matching problem run in $O(\sqrt{|V|} \cdot |E|)$ time [14, 28, 34].

However, for an algorithm to be suitable as the basis for a privacy-preserving protocol it is also important that it has an easy structure and does not rely on complex and dynamically growing datastructures as we have to make sure that the datastructures do not leak any information on the underlying data. Therefore, we use the matching algorithm designed by Pape and Conradt [36] which uses a very simple labeling technique to avoid the shrinking of blossoms as basis for our privacy-preserving protocol. Their approach is based on Edmonds' algorithm and achieves a complexity of $O(|V|^3)$.

4.2 Privacy-Preserving Bipartite Matching

While to the best of our knowledge we are the first to propose a privacy-preserving protocol for maximum matching on general graphs, there are privacy-preserving solutions for maximum matching on bipartite graphs (e.g., [4, 12, 41]) with complexities ranging from $O(|V|^3)$ [12] to $O(|V|^6)$ [41]. Furthermore, the maximum matching problem on bipartite graphs can be translated into the maximum flow problem. Aly et al. [3] present two protocols for the maximum flow problem based on the Edmonds-Karp algorithm and the Push-Relabel algorithm with complexities $O(|V|^4)$ and $O(|V|^5)$, respectively. Blanton et al. [13] compute a maximum flow based on the Ford-Fulkerson algorithm in $O(|V|^3|E|log(|V|))$.

Another related problem is private stable matching where two groups of individuals state their preferences over each other and are matched such that there are no pairs from the two groups who would prefer each other over their computed match. In private stable matching the individuals' preferences and the outcome of the computation are kept private. The first private stable matching algorithm goes back to Golle [30] and has complexity $O(|V|^5)$. The currently best solutions achieve complexity $O(|V|^2log^3(|V|))$ [24, 38].

However, it is not possible to adapt these solutions to crossover kidney exchange as a bipartite graph requires the division of the patient-donor pairs into two sets. The only meaningful way to do so, is to consider donors and patients as two separate sets which makes it impossible to guarantee that a donor only donates her kidney if the corresponding patient also receives a kidney transplant.

4.3 Privacy-Preserving Kidney Exchange

To the best of our knowledge, the only existing privacy-preserving approach for kidney exchange is the SMPC protocol for solving the kidney exchange problem by Breuer et al. [19]. The authors rely on SMPC based on additive homomorphic encryption and use a brute force approach to compute a set of exchange cycles that maximizes the number of patients that can receive a kidney transplant.

While their approach solves the KEP for an arbitrary maximum cycle size, the runtime of their protocol increases very fast for increasing numbers of patient-donor pairs as the KEP is NP-complete for a cycle size larger than 2. Therefore, we restrict our approach

[3] This can also be realized with sub-linear complexity using ORAM techniques (e.g., [32]). However, in practice ORAM only provides for a performance improvement for large vector sizes ($\geq \sim$1000) [32] and the vectors in our protocols are much smaller.

to crossover exchange enabling the computation of exchanges between a much larger number of patient-donor pairs.

Another drawback of their approach is that it requires each patient-donor pair to participate in the whole protocol execution which does not reflect a realistic scenario for the use case of kidney exchange since the patient-donor pairs lack the required expertise and the medical data lies with the transplant centers or hospitals anyway. Our approach instead only requires the pairs (or hospitals) to send the medical data to three computing peers using secret sharing and then wait for their output.

5 PROTOCOL FOR CROSSOVER KIDNEY EXCHANGE

Before providing the detailed specification of our privacy-preserving protocol for crossover kidney exchange (Section 5.2), we introduce the matching algorithm by Pape and Conradt [36] on which our protocol is based (Section 5.1). In Section 5.3, we present a performance analysis of the implementation of our protocol in MP-SPDZ [31].

5.1 Algorithm by Pape and Conradt

The main idea of the matching algorithm by Pape and Conradt [36] is to develop a blossom (an odd cycle) in two alternating paths instead of shrinking it which is a computationally expensive operation. Thereby, a node that is part of a blossom will be once at even distance along the path from the root and once at odd distance along a second path from the root. Thus, if nodes of the blossom are part of an augmenting path, this path is guaranteed to be found.

To develop a blossom in two alternating paths, an *alternating tree* is grown from each exposed node until either an augmenting path has been found or the tree can be grown no more. The root of an alternating tree is the exposed node r which forms level 0 of the tree. All nodes adjacent to r are added on level 1 and all matched edges incident to the nodes on level 1 (i.e., their mates) form the nodes on level 2. Level 3 then again contains all nodes adjacent to the nodes on level 2 (which are not already part of the tree) and so on. More formally, an alternating tree with respect to a matching M is rooted at an exposed node and all paths emanating from the root are alternating paths. The nodes at even levels of the tree are called *inner nodes* and those at odd levels are called *outer nodes*.

In the algorithm, three datastructures are used to represent an alternating tree. The queue Q contains all outer nodes that are currently in the tree but from which the tree has not been explored further. The binary array *nonTree* of size |V| keeps track of the nodes that are in the tree at even levels (i.e., the root node and all inner nodes). The entry $nonTree(v)$ is set to 0 if v is the root node or an inner node and to 1 if it is an outer node. The array *grandfather* of size |V| is used to trace back the path up to the root if an augmenting path has been found. For an outer node w, $grandfather(w) = u$ indicates that there is a path (u, v, w) in the alternating tree from u to w such that (v, w) is a matched edge.

Algorithm 1 contains pseudocode for the matching algorithm by Pape and Conradt [36]. The algorithm starts with an initial matching which can be any matching M in G including the empty matching. The matching is given by the array *mate* of size |V| which encodes the mate of each node $v \in$ V. The number of exposed nodes is given by *expo*. The algorithm iterates over all potential root nodes

Algorithm 1 Matching Algorithm by Pape and Conradt [40]

```
1:  Start with an initial matching
2:  for r ∈ V do
3:    if mate(r) = 0 ∧ expo ≥ 2 then
4:      for v ∈ V do
5:        nonTree(v) ← 1
6:      nonTree(r) ← 0
7:      Q.insert(r)
8:      found ← 0
9:      while Q ≠ ∅ ∧ found = 0 do
10:       x ← Q.removeHead()
11:       for y adjacent to x do
12:         if nonTree(y) = 1 then
13:           if mate(y) = 0 then
14:             Update matching
15:             expo ← expo − 2
16:             found ← 1
17:           else if mate(y) ≠ x then
18:             if y is not an ancestor of x then
19:               z ← mate(y)
20:               nonTree(y) ← 0
21:               grandfather(z) ← x
22:               Q.insert(z)
```

$r \in V$ and starts growing an alternating tree from r if r is an exposed node. The tree is grown based on a breadth-first search. Note that if the number of exposed nodes is less than 2, the matching is already a maximum matching and the algorithm can be aborted.

If M is not yet a maximum matching, the array *nonTree* is initialized such that only the entry for the root node r equals 0 indicating that the tree currently only contains r. Also, r is added to the queue Q of outer nodes to be explored from and the boolean *found* is set to 0 stating that an augmenting path has not yet been found.

Then, the first node x is removed from Q (which is the root node r in the first iteration) and a tree is grown from x. The algorithm parses over each node y that is adjacent to x. If $nonTree(y) = 0$, the edge (x, y) is ignored as y is an inner node, i.e., it is already part of the tree. If $nonTree(y) = 1$, the node is either matched or exposed. If y is exposed (i.e., $mate(y) = 0$), an augmenting path from root node r to y has been found. In that case, the matching M is updated, the number of exposed nodes is decreased by 2, the current tree is abandoned, and an alternating tree is grown from the next exposed node. If y is not exposed and the mate of y is not x itself, it has to be checked whether y is an ancestor of x in the tree. If y is an ancestor of x, they are part of a blossom (an odd cycle) which could lead to a false augmenting path. In that case the algorithm just continues with the next value for y. If y is not an ancestor of x, the inner node y and the outer node $z = mate(y)$ are added to the tree, i.e., z is added to Q, $grandfather(z)$ is set to x, and $nonTree(y)$ is set to 0.

A maximum matching has been found if there are less than two exposed nodes or if every exposed node has been tried as the root of an alternating tree. The complete algorithm has complexity $O(|V|^3)$. For a detailed complexity analysis, we refer the reader to [40].

5.2 SMPC Protocol Specification

Our privacy-preserving protocol for kidney exchange is based on the matching algorithm by Pape and Conradt [36] (in the following

referred to as conventional PC algorithm) introduced in Section 5.1. Specifically, we first have to construct the graph on which we compute the matching such that each node corresponds to one patient-donor pair and an undirected edge is added between two nodes if the corresponding pairs are medically compatible, i.e., the donor of one pair can donate to the patient of the other and vice versa. Since each edge then corresponds to one crossover exchange, computing a maximum matching on the constructed graph then is equivalent to maximizing the number of crossover exchanges.

The detailed specification of protocol Crossover-KE for privacy-preserving crossover kidney exchange is given in Protocol 1. Prior to the protocol execution, all input peers (patient-donor pairs) send their input (medical data) to the computing peers using secret sharing. Thus, each computing peer holds a share of the medical data of each patient-donor pair P_v with $v \in \mathrm{V}$ where $|\mathrm{V}|$ corresponds to the number of patient-donor pairs for which the protocol is run.

Graph Initialization Phase. At the beginning of the protocol, a shared adjacency matrix $[\mathrm{A}]$ is computed such that an entry $[\mathrm{A}](u, v)$ encodes whether patient-donor pairs P_u and P_v $(u, v \in \mathrm{V})$ are compatible. To this end, we use the building block Comp-Check introduced by Breuer et al. [19]. The input to the compatibility check comprises secret binary indicator vectors for the donor bloodtype $[B^d]$, the patient bloodtype $[B^p]$, the donors' antigens $[A^d]$, and the patients' antibodies $[A^p]$ for both patient-donor pairs. The donor of pair P_u is then considered compatible with the patient of pair P_v if $B_u^d(i) = B_v^p(i) = 1$ for at least one $i \in \{1, ..., |B^d|\}$ and if for all $i \in \{1, ..., |A^p|\}$ with $A_u^d(i) = 1$ it holds that $A_v^p(i) \neq 1$, i.e., if the patient has no antibody against the donor's antigens.[4] As compatibility has to be computed between the donor of pair P_u and the patient of pair P_v and vice versa, this requires $2 \cdot |B^d| \cdot |A^d|$ comparisons which can be executed in parallel. Note that even if the protocol indicates that two pairs are compatible, the final choice of whether a transplant is carried out still lies with medical experts. Before starting the matching computation, the adjacency matrix is shuffled at random to ensure that two pairs with the same input have the same probability of being matched independent of their index. It is sufficient that $\frac{n}{2} + 1$ of the n computing peers each input a secret permutation matrix. All rows and columns of the matrix are then shuffled once with each of these permutation matrices.[5]

Path Finding Phase. After the computation of the adjacency matrix, we initialize an empty matching by setting the mate of each patient-donor pair P_u $(\forall u \in \mathrm{V})$ to $[0]$. In the privacy-preserving setting the initial matching does not influence the protocol performance as the worst case number of iterations have to be executed for all loops in order to hide the structure of the alternating trees.

The computation of a maximum matching then starts with the for-loop over all potential root nodes r. In contrast to the conventional PC algorithm, we cannot abort the tree construction if the node r is not exposed as we have to keep all information about the structure of the underlying graph private. Thus, we set $[root]$ to $[0]$ if r is not exposed, i.e., all relevant conditions in the remainder

Protocol 1 Crossover-KE

for $u, v \in \mathrm{V}$ **do**
  $[A](u,v) \leftarrow$ Comp-Check($[B_u^d], [A_u^d], [B_u^p], [A_u^p], [B_v^d], [A_v^d], [B_v^p], [A_v^p]$)
$[\mathrm{A}] \leftarrow$ Shuffle($[\mathrm{A}]$)
for $u \in \mathrm{V}$ **do**
  $[mate](u) \leftarrow [0]$
for $r \in \mathrm{V}$ **do**
  $[root] \leftarrow ([mate](r) \stackrel{?}{=} [0])\ ?\ [r]\ :\ [0])$
  for $v \in \mathrm{V}$ **do**
    $[nonTree](v) \leftarrow [1]$
  $[nonTree](r) \leftarrow [root] \stackrel{?}{=} [0]$
  $[Q] \leftarrow [Q].insert([root])$
  $[found] \leftarrow [0]$
  for $|\mathrm{V}|$ times **do**
    $[x] \leftarrow [Q].deleteHead()$
    $[x] \leftarrow [found]\ ?\ [0]\ :\ [x]$
    for $y \in \mathrm{V}$ **do**
      $[augPath] \leftarrow [nonTree](y) \cdot [A]([x], y) \cdot ([mate](y) \stackrel{?}{=} [0]) \cdot ([mate](r) \stackrel{?}{=} [0])$
      $[found] \leftarrow [augPath] + [found] - [augPath] \cdot [found]$
      $[mate] \leftarrow$ Update-M($[augPath], [mate], [grandf.], [x], y$)
      $[y\_anc\_x] \leftarrow$ Anc-Check($[grandfather], [x], y$)
      $[c_2] \leftarrow [nonTree](y) \cdot ([x] \stackrel{?}{\neq} [mate](y)) \cdot ([found] \stackrel{?}{=} [0]) \cdot [A]([x], y) \cdot ([y\_anc\_x] \stackrel{?}{=} [0])$
      $[z] \leftarrow [c_2]\ ?\ [mate](y)\ :\ [0])$
      $[nonTree](y) \leftarrow ([z] \stackrel{?}{\neq} [0])\ ?\ [0]\ :\ [nonTree](y)$
      $[grandfather]([z]) \leftarrow [x]$
      $[Q] \leftarrow [Q].insert([z])$
$[mate] \leftarrow$ Reverse-Shuffle($[mate]$)
for $v \in \mathrm{V}$ **do**
  Send share of $[mate](v)$ to patient-donor pair P_v

of the loop will also be equal to $[0]$ and thus the matching will not be altered for $[root] = [0]$. Note that the entire protocol is designed such that if a node v has value 0, no further modifications to the current state of the tree are made (i.e., we use 0 as a *dummy node*).

Then, we initialize the array $[nonTree]$, the queue $[Q]$, and the boolean $[found]$ as in the conventional PC algorithm. If $[root] = [0]$, $[nonTree](r)$ is set to $[1]$ and $[0]$ is inserted into the queue $[Q]$.

In the conventional PC algorithm, one would now iterate over the entries in the queue $[Q]$ until it is empty or an augmenting path is found. However, to avoid any information leakage w.r.t. the structure of the alternating tree, we have to execute the loop for a fixed number of iterations. In particular, we have to execute the loop for the worst case number of iterations which is $|\mathrm{V}|$ as each node can be added at most once to the queue $[Q]$.

The loop then starts with the first node $[x]$ from the queue $[Q]$. In the privacy-preserving setting, the queue has to be implemented such that it returns $[0]$ if it is empty. Thus, if the queue is empty, the protocol continues computing on the dummy node 0.

In the next step, we check whether we have already found an augmenting path for this tree (i.e., $[found] = [1]$). If that is the case, we set $[x]$ to $[0]$. Otherwise, $[x]$ remains unchanged.

In the conventional PC algorithm, one would now only iterate over those nodes that are adjacent to $[x]$. However, as we have to keep the nodes that are indeed adjacent to $[x]$ hidden, we have

[4] Additional criteria for compatibility can be checked in the same way if required.

[5] The graph initialization phase is the only part of protocol Crossover-KE specific to kidney exchange. To consider a different use case, one just has to change the computation of the adjacency matrix. Thus, our protocol is of independent interest for any use case requiring the privacy-preserving computation of a matching on general graphs.

Protocol 2 UPDATE-M

1: $[mate](y) \leftarrow [augPath]\ ?\ [x]\ :\ [mate](y)$
2: **for** |V| times **do**
3: $\quad [next] \leftarrow [augPath] \cdot [mate]([x])$
4: $\quad [mate]([x]) \leftarrow [augPath]\ ?\ [y]\ :\ [mate]([x])$
5: $\quad [x] \leftarrow [augPath] \cdot [grandfather]([x])$
6: $\quad [mate]([next]) \leftarrow [augPath] \cdot [x]$
7: $\quad [y] \leftarrow [augPath] \cdot [next]$

to iterate over all nodes $y \in V$. First, we check if there is an augmenting path rooted at r and ending in y. This is the case if y is still not part of the tree (i.e., $[nonTree] = [1]$), y is adjacent to $[x]$ (i.e., $[A]([x], y) = [1]$), and y and r are both exposed nodes (i.e., $[mate](y) = [mate](r) = [0]$). If all these conditions hold, their product equals $[1]$ and thus $[augPath]$ is set to $[1]$. Afterwards, we can update the secret variable $[found]$ which is used in the next round to check whether we have already found an augmenting path for the tree rooted at r. Furthermore, we call the protocol UPDATE-M to increase the matching along the found augmenting path.

Matching Update Phase. The specification of protocol UPDATE-M is given in Protocol 2. The general idea of the protocol is to trace back the path from the exposed node y to the root node r and change all matched edges to unmatched edges and vice versa. Thereby, the cardinality of the matching is increased by exactly 1. Of course, this should only be done if we indeed found an augmenting path, i.e., $[augPath] = [1]$. To this end, at the beginning of protocol UPDATE-M $[mate](y)$ is set to $[x]$ if $[augPath] = [1]$. Afterwards, we iterate over the edges of the path and switch matched with unmatched edges. We use an auxiliary variable *next* which stores the next node on the path. If $[augPath] = [1]$, we set $[next]$ to the mate of $[x]$ and thus to the next node on the path from y to r. Furthermore, we set the new mate of $[x]$ to $[y]$ and $[x]$ itself to the grandfather of $[x]$. Then, we set the new mate of $[next]$ to the next value for $[x]$, i.e., to the grandfather of $[x]$. Furthermore, we update $[y]$ such that it now contains the next node to consider on the path (i.e., $[next]$). Thereby, $[y]$ and $[x]$ become the two next nodes on the path to the root. We have to execute the whole loop |V| times as this is the maximum number of nodes of the augmenting path. Note that UPDATE-M does not alter the matching if $[x] = [0]$ or $[augPath] = [0]$.

Tree Extension Phase. In the last part of the tree growing in Protocol 1, we check if the nodes y and $[z]$ ($[z]$ is the mate of y) have to be added to the tree. First, we check if y is an ancestor of $[x]$ in the tree rooted at r using the protocol ANC-CHECK. If this is the case, the two nodes are part of an odd cycle and should not be added to the tree as they could lead to a false augmenting path.

The specification of ANC-CHECK is given in Protocol 3. In each iteration, we check whether the current node $[x]$ is equal to y. If this is the case, we know that $[x]$ is an ancestor of y. Otherwise, we update the value of $[x]$ to its grandfather. If we reach the root, $[x]$ is set to $[0]$ as $[grandfather](r) = [0]$. Thus, at the end of the protocol $[y\_anc\_x] = [1]$ if and only if y is an ancestor of $[x]$.

Returning to Protocol 1, we add $[z]$ and $[y]$ to to the tree if all of the following conditions hold: y is not part of the tree already (i.e., $[nonTree](y) = 1$), $[mate](y)$ is not equal to $[x]$, no path has been found in the current tree so far (i.e., $[found] = [0]$), y is

Protocol 3 ANC-CHECK

1: $[y\_anc\_x] \leftarrow [0]$
2: **for** $\lceil |V|/2 \rceil$ times **do**
3: $\quad [y\_anc\_x] \leftarrow ([x] \stackrel{?}{=} [y])\ ?\ [1]\ :\ [y\_anc\_x]$
4: $\quad [x] \leftarrow [grandfather]([x])$

adjacent to $[x]$ (i.e., $[A]([x], y) = [1]$), and y is not an ancestor of $[x]$. Otherwise, we set $[z]$ to $[0]$. Afterwards, we update $[nonTree]$, $[grandfather]$, and $[Q]$ accordingly, i.e., if $[z] = [mate](y)$, we set $[nonTree](y)$ to $[0]$, $[grandfather](z)$ to $[x]$, and add $[z]$ to $[Q]$.

Output Phase. At the end of the protocol, the entries in $[mate]$ are unshuffled using the inverses of the permutation matrices that were used to shuffle the adjacency matrix during the graph initialization phase. Afterwards, each patient-donor pair P_v with $v \in V$ obtains the shares of $[mate](v)$ indicating its exchange partner.

Correctness. The correctness of Protocol 1 follows from the correctness of the conventional PC algorithm and the fact that computations on the dummy node 0 do not alter the computed matching.

Security. Since throughout Protocol 1 all computations are executed on shared values and all building blocks are called on shared inputs, the simulator can just call the simulator of the corresponding building block on random input for each building block that is executed during protocol CROSSOVER-KE. Security then follows from the properties of Shamir's secret sharing scheme, the security of the primitives introduced in Section 3.2, and the data-obliviousness of Protocol 1. The latter refers to the property that no information on the structure of the trees is leaked as always all possible branches of a tree are considered (using dummy iterations if an augmenting path has already been found or the queue is empty).

Complexity. The computation of the adjacency matrix can be done in $O(|V|^3)$ communication and $O(|V^2|)$ round complexity as protocol COMP-CHECK exhibits linear communication and constant round complexity. The most time consuming part of protocol CROSSOVER-KE, however, is the construction of the alternating trees for all potential root nodes. The corresponding loop is executed |V| times as are the inner loops in lines 12 and 15 of Protocol 1. Note that it is not possible to abort these loops early as this would leak information on the underlying graph. Protocols UPDATE-M and ANC-CHECK both have communication complexity $O(|V^2|)$ and round complexity $O(|V|)$ and are called once in the innermost loop. Thus, the construction of the alternating trees has communication complexity $O(|V^5|)$ and round complexity $O(|V|^4)$ which are also the complexities for the complete protocol CROSSOVER-KE.

5.3 Performance Analysis

We have implemented our novel protocol CROSSOVER-KE in the SMPC benchmarking framework MP-SPDZ [31] and evaluated its performance. The source code is published in [18].

In our evaluation, we use LXC containers on a single server with an AMD EPYC 7702P 64-core processor for the computing peers and the input peers. Each container runs Ubuntu 20.04 LTS and is equipped with one core and 4GB RAM. Each computing peer corresponds to a single container and a fourth container manages

Table 1: Runtime and network traffic of protocol Crossover-KE for different numbers of input peers and different latencies L for a fixed bandwidth $B = 1Gbps$.

input peers	runtime			traffic
	L = 1ms	L = 5ms	L = 10ms	
5	21s	97s	192s	51MB
10	6m	27m	53m	759MB
15	30m	2h	5h	4GB
20	96m	7h	15h	13GB
25	4h	18h	35h	33GB
30	8h	37h	73h	70GB
35	15h	68h	134h	148GB
40	25h	116h	-	260GB
45	40h	-	-	423GB
50	61h	-	-	663GB
55	89h	-	-	990GB
60	127h	-	-	1440GB

sending input and receiving output for all input peers (patient-donor pairs). For all measurements where the runtime of a single protocol execution is less than one hour, we execute ten repetitions and average the results. For all others, we only execute the protocol once as deviations between the different runs are comparably low for such long runtimes. As performance measures we consider the runtime and the induced network traffic which corresponds to the accumulated network traffic of all three computing peers. We start a measurement when the computing peers have received all inputs and end it when they have sent the output back to the input peers.

Since all containers are hosted on the same server, the *latency* L in our setup is very low. In order to mimic real-world scenarios, we include the capability to artificially set the latency and the bandwidth to realistic values. As these may differ considerably depending on the setup of the three computing peers in practice, we have to determine values that resemble pertinent use cases. With respect to the bandwidth, it seems reasonable to assume that the computing peers have a high-end Internet connection as they are hosted by large (possibly governmental) institutions. Thus, we assume a bandwidth $B = 1Gbps$ for all our measurements. [6] Concerning the latency, the values may differ considerably depending on the location of the different institutions that host the computing peers. Therefore, we consider three different values for the latency L. For a best-case scenario, where all institutions are hosted in the same network, we consider latency $L = 1ms$. As an average case latency, we use $L = 5ms$, and as a worst-case setup, we use $L = 10ms$.

Table 1 shows runtime and network traffic for increasing numbers of input peers. We stopped a measurement if it took longer than 7 days.[7] Let us first consider the runtimes for $L = 1ms$. We observe that the runtime increases considerably for increasing numbers of input peers. For up to 5 input peers, the protocol still finishes within one minute for $L = 1ms$, whereas the runtime for 20 input peers amounts to more than 1 hour and for 40 input peers the protocol requires more than 24 hours to finish. Finally, for 60 input peers the protocol runs for 5 days and 7 hours. The network traffic scales similarly: for up to 5 input peers it is still below $50MB$, for 20 input peers it increases to more than $10GB$, for 40 input peers $260GB$, and for 60 input peers $1.4TB$ are sent in one protocol run.

[6] Results for smaller bandwidths can be found in the extended version of this paper [17].

[7] We decided to only allow protocol runtimes for up to 7 days as for longer runtimes the state of the patient-donor pairs inside the pool of a kidney exchange platform may already have changed substantially before the actual protocol execution is finished.

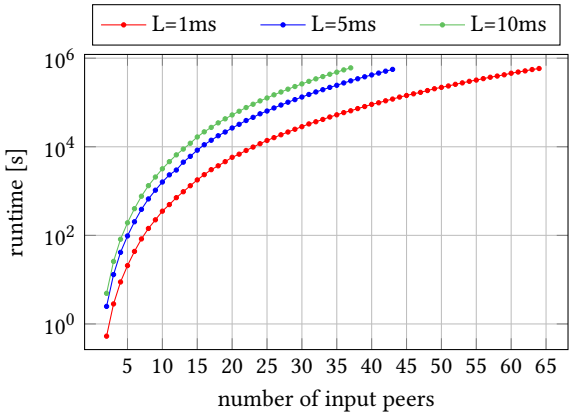

Figure 2: Runtime of protocol Crossover-KE for different latencies L and fixed bandwidth $B = 1Gbps$.

We also observe that the runtime increases for increasing latencies. On average the runtime for $L = 1ms$ is about 4.66 times larger than for $L = 5ms$ and about 9.22 times larger than for $L = 10ms$.

Figure 2 shows plots of the runtime for the three considered latencies. Note that the scale of the y-axis is logarithmic. We observe that the overall runtime is less than exponential. Furthermore, the plots show that the factor between the runtimes for the different latencies is nearly constant. This is as expected since the latency is a constant offset by which each sent message is delayed.

While the runtimes of Protocol 1 may seem large, this is not necessarily an issue for our use case as most kidney exchange platforms typically compute a matching only once every couple of days and some even only once every three months [7]. In Section 6, we analyze to what extent the runtimes of Protocol 1 allow for its application in large scale kidney exchange platforms.

5.3.1 Comparison to Related Work. We compare our novel protocol Crossover-KE to the privacy-preserving protocol for kidney exchange by Breuer et al. [19] (cf. Section 4.3). While their protocol allows to solve the KEP for any maximum cycle size, it is also possible to restrict it to pure crossover exchange. To enable a fair comparison between the two approaches, we implemented the protocol from [19] in MP-SPDZ considering the same setting as for our protocol Crossover-KE (secret sharing is used instead of homomorphic encryption and the protocol is run by three computing peers whereas the patient-donor pairs correspond to the input peers). We refer to the implementation from [19] based on homomorphic encryption as protocol KEP-Rnd-HE and to our implementation of the protocol from [19] based on secret sharing as protocol KEP-Rnd-SS. We publish the source code for protocol KEP-Rnd-SS in [18].

Table 2 shows the results for our protocol Crossover-KE and the two protocols KEP-Rnd-SS and KEP-Rnd-HE (restricted to crossover exchange).[8] We observe that the runtime of protocol KEP-Rnd-HE is worse than for both other protocols for all numbers of input peers.

[8] We only evaluated the protocol KEP-Rnd-HE for up to 10 parties as this already showed that the other two protocols drastically outperform protocol KEP-Rnd-HE.

Table 2: Runtime and network traffic for our protocol Crossover-KE (for latency $L = 1ms$), our implementation KEP-Rnd-SS of the protocol from [19] (for latency $L = 1ms$), and the implementation KEP-Rnd-HE from [19] where all parties are connected by a LAN with latency $L < 0.5ms$.

input	Crossover-KE		KEP-Rnd-SS		KEP-Rnd-HE	
peers	runtime	traffic	runtime	traffic	runtime	traffic
2	0.5s	4MB	0.2s	0.4MB	14s	1MB
3	3s	10MB	0.7s	0.8MB	23s	4MB
4	9s	24MB	2s	2MB	36s	11MB
5	21s	51MB	3s	3MB	64s	32MB
6	43s	101MB	4s	6MB	136s	93MB
7	83s	186MB	5s	12MB	6m	307MB
8	2m	318MB	8s	39MB	19m	1GB
9	4m	492MB	13s	147MB	66m	5GB
10	6m	759MB	30s	585MB	4h	23GB
11	8m	1GB	2m	2GB	-	-
12	12m	1.5GB	7m	10GB	-	-
13	16m	2GB	26m	40GB	-	-
14	22m	3GB	115m	166GB	-	-
15	30m	4GB	8.5h	684GB	-	-

In particular, by using secret sharing instead of homomorphic encryption we improved the performance of the protocol from [19] significantly. When comparing our novel protocol Crossover-KE to the protocol KEP-Rnd-SS, we observe that for up to 12 input peers, protocol KEP-Rnd-SS outperforms our protocol Crossover-KE. However, for larger numbers of input peers, our protocol Crossover-KE is much more efficient. We prescribe this to the brute-force nature of the approach from [19], i.e., an exhaustive search is run over the set of all possible exchange constellations that can exist between the input peers. Since this set increases very fast for increasing numbers of input peers, their approach is only feasible for small numbers of input peers (e.g., for 15 input peers protocol KEP-Rnd-SS already takes 17 times longer than our protocol Crossover-KE).

Our results show that for countries where only crossover exchange is allowed our specialized protocol for crossover exchange is better suited than the existing approach from [19]. This was to be expected as the general KEP is an NP-complete problem [1].

6 DYNAMIC KIDNEY EXCHANGE

Our protocol Crossover-KE (Protocol 1) allows for the computation of a maximum matching between a fixed set of patient-donor pairs. However, kidney exchange is dynamic by nature, i.e., patient-donor pairs arrive at and leave from the platform over time. In this section, we analyze the performance of a dynamic kidney exchange platform (cf. Section 2) using our protocol compared to a non-privacy-preserving approach (as explained in Section 2). To this end, we developed a simulation framework based on DESMO-J [22] which is a discrete event simulation framework written in Java.

Recall that at the core of a kidney exchange platform is the pool of patient-donor pairs who want to find an exchange partner. We simulate the computed transplants over time using our protocol Crossover-KE (Protocol 1) for computing the matching inside the pool at a certain point in time. Depending on the number of pairs in the pool at the time of a match run, the runtime of our protocol may be infeasible. This means that the protocol execution does not finish before the next match run is scheduled, e.g., if we compute a matching once a week, the protocol runtime should be below 7 days. In such a case, we split the pool into multiple sub-pools and distribute the pairs uniformly at random among these. Of course, this may lead to fewer pairs being matched in the privacy-preserving case than in the conventional case (where a matching can always be computed efficiently among all pairs in the pool). Our policy is to split the pool into the minimum number of equally sized sub-pools whose execution time is below the match run interval.

The goal of our simulations is then to evaluate the impact of this pool splitting on the number of patient-donor pairs that can receive a transplant. Thereby, we can evaluate the impact of the larger runtimes of our privacy-preserving protocol compared to the non-privacy-preserving solution and thus determine the practicality of our protocol when used in a real-world kidney exchange platform.

6.1 Data Set

We use a data set from the Organ Procurement and Transplantation Network (OPTN) provided by the United Network for Organ Sharing (UNOS) containing all patients and donors that have participated in their living donor exchange program between 27th October 2010 and 29th December 2020. [9] The data set contains 2738 patients that were registered with UNOS. There are 150 patients that have registered with more than one donor over time. We consider these as different unique pairs leading to an overall number of 2913 unique patient-donor pairs from which we sample in our simulations. We use the straight-forward approach of choosing the entering patient-donor pairs as described in [7], i.e., we sample randomly from all pairs when inserting a new pair into the pool.

6.2 Simulation Setup

For each simulation run, we consider a time horizon of five years and evaluate the number of patients that receive a kidney transplant as well as the average waiting time (i.e., the average time a patient has to wait until receiving a transplant). We compare the approach using our protocol Crossover-KE (Protocol 1) to a conventional approach which is not resistant against manipulation and does not protect the privacy of the patient-donor pairs' medical data. For the conventional approach, we assume that computing the matching is done instantly using a state-of-the-art matching algorithm. In the privacy-preserving case, we simulate the protocol execution using a non-privacy-preserving implementation of our protocol and then schedule the completion of the protocol execution by adding the runtime measured in Section 5.3 to the simulation time. Thereby, we reduce the time and resources required by the simulations which allows us to simulate larger parameter ranges.

A platform as described in Section 2 suggests the following parameters that can vary among different platforms (and countries) and for which we run simulations to determine those platforms (and countries) for which our privacy-preserving protocol scales best:

The *arrival rate* states the frequency at which new patient-donor pairs register with the platform (e.g., arrival rate 2 indicates that a new pair arrives every 2 days). In the UNOS data set, a new

[9] The data reported here have been supplied by the United Network for Organ Sharing as the contractor for the Organ Procurement and Transplantation Network. The interpretation and reporting of these data are the responsibility of the author(s) and in no way should be seen as an official policy of or interpretation by the OPTN or the U.S. Government.

pair registers every 1.4 days on average. However, the arrival rate differs among existing platforms depending mainly on the size of the population that participates in the exchange. Therefore, we follow Ashlagi et al. [6] who suggest arrival rates between 1 and 14 days.

The *match run interval* is the interval at which a matching among the pairs in the pool is computed (e.g., a match run interval of size 1 indicates that a matching is computed on a daily basis). While platforms in the US such as APKD (Alliance for Paired Kidney Donation) and NKR (National Kidney Registry) have switched to daily match runs or at least multiple match runs per week (UNOS), other countries such as Australia, UK, Netherlands, and Canada compute a matching only once every 2-4 months [7, 10, 21, 26]. Therefore, we consider values spanning from one day to four months.

The *departure rate* is the probability that a patient-donor pair leaves the pool due to illness/death of donor/patient or due to a donor backing out. Based on data from APKD, Ashlagi et al. [6] estimate that a pair on average stays for about 420 days in the pool without being matched. They simulate this by assuming that a pair leaves each day with probability $\frac{1}{420}$. Besides, they also consider pairs staying for 800 and 1000 days on average. We follow their approach and consider pairs staying for 400, 800, and 1000 days on average.

The *match refusal probability* is the probability that a match is refused by the patient-donor pair or the hospital for any other reason than a positive crossmatch.[10] For large US exchanges, this probability is between 20% and 35% [2, 6] whereas in reports on kidney exchange platforms in Europe there is no mention of such high probabilities [10, 26]. As we aim to cover all existing platforms, we consider match refusal probabilities 0%, 10%, 20%, 30%, and 40%.

The *crossmatch failure probability* states the failure rate of offered matches due to a positive crossmatch. Platforms in the US (AKPD, UNOS, NKR) [7] as well as in other countries, e.g., Canada [21] report that the rate lies at about 35% for highly sensitized patients and 10% for others. We adopt these values for our simulations.

The *reentering delay* indicates the time it takes for a patient-donor pair to reenter the pool in case of a match offer and a positive crossmatch failure or a refusal of a matched pair. We follow Ashlagi et al. [6] who suggest a delay of 2 days for reentering to account for a match refusal and 7 days to account for a positive crossmatch.

Table 3 shows all parameters together with the values for which we run simulations. For each combination, we execute 50 simulation runs with different seeds for the random sampling of entering pairs.

6.3 Evaluation Results

The performance of the privacy-preserving approach depends highly on the runtime of the underlying SMPC protocol which in turn depends on the network latency L (cf. Section 5.3). Therefore, we evaluate the privacy-preserving approach for three different scenarios each using the runtime for a different latency (cf. Table 1).

As we cannot present the results for all parameter values in this paper, we focus on an average time before departure of 400 days and a match refusal probability of 20% as these are the values most commonly found in existing platforms. Besides, our simulations have shown that departure rate and match refusal probability only slightly influence the performance of the privacy-preserving approach compared to the conventional approach. Results for additional values are included in the extended version of this paper [17].

[10] A crossmatch is a test stating if a patient and a donor are medically compatible. In Protocol 1, we compute a virtual crossmatch which does not always match the result of a physical crossmatch. A physical crossmatch, however, can only be carried out after a match is computed as it requires mixing the blood of patient and donor.

Table 3: Simulation parameters and their respective values.

parameter	values
arrival rate	1, 2, 4, 7, 14 days
match run interval	1, 2, 4, 7, 14, 30, 60, 120 days
average time before departure	400, 800, 1000 days
match refusal probability	0, 10, 20, 30, 40 %

Table 4 shows the number of transplants for a departure rate of 1/400 and a match refusal probability of 20%. To compare the two approaches, we compute the percentage of transplants measured for the privacy-preserving approach compared to those measured for the conventional approach (e.g., a percentage of 95% indicates that the privacy-preserving approach only leads to 5% fewer transplants over the considered time horizon of five years). We highlight all entries where the percentage is larger than 95% in Table 4 indicating those parameters for which the negative impact of the privacy-preserving approach on the number of transplants is very small.

General Observations. The privacy-preserving approach is better suited for large arrival rates. This was to be expected since in the privacy-preserving approach the pool has to be split into multiple sub-pools if the number of pairs inside the pool becomes too large. This potentially leads to two pairs which are compatible ending up in different sub-pools. Thus, it is better for the privacy-preserving approach if the patient-donor pairs arrive less frequently as this also means that on average there are fewer patient-donor pairs in the pool which in turn decreases the necessity for pool splitting.

The privacy-preserving approach performs best if the match run interval is low. This is also as expected since a low match run interval also means that in general fewer patient-donor pairs are in the pool at each match run. Furthermore, if match runs are executed frequently, the probability that two potentially matching pairs are in the same sub-pool is higher since the pairs are distributed uniformly at random among all sub-pools for each match run.

The privacy-preserving approach sometimes even outperforms the conventional approach. We prescribe this to the fact that there is no known best policy for choosing the match run interval [6]. While, in general, executing match runs with a high frequency seems to be best, there are cases where waiting for further pairs to enter the pool can be better. Consider the following example: Pair A is compatible with pairs B and C whereas pair D is only compatible with pair B. If a match run is executed before pairs C and D arrive, only A and B can be matched. If one however waits until pairs C and D have arrived, then A can be matched with C and B with D.

Influence of the Latency. We observe that the smaller the latency in the network connecting the three computing peers, the better the performance of the privacy-preserving approach. This was to be expected as the runtime of the protocol determines the number of pairs for which the pool can still be executed without splitting it into sub-pools. For latency $L = 1ms$, we are able to find nearly the same number of transplants as in the conventional approach independent of the arrival rate when considering a small match run interval (e.g., over the complete time span of 5 years

Table 4: Comparison between the number of transplants measured for the conventional approach and the privacy-preserving approach (for different values for the latency L) with a departure rate of $1/400$ and a match refusal probability of 20%. The percentages state how close the privacy-preserving approach comes to the conventional approach.

		conv. model	privacy-preserving model					
arrival rate	match run interval	transplants	$L = 1ms$		$L = 5ms$		$L = 10ms$	
			transplants	percentage	transplants	percentage	transplants	percentage
1	1	690.52	682.24	98.80%	676.52	97.97%	676.80	98.01%
	2	689.80	672.88	97.55%	669.24	97.02%	661.08	95.84%
	4	683.72	655.60	95.89%	645.16	94.36%	640.96	93.75%
	7	678.84	642.36	94.63%	621.04	91.49%	615.64	90.69%
	14	658.92	604.04	91.67%	578.36	87.77%	560.68	85.09%
	30	612.24	529.36	86.46%	489.52	79.96%	480.20	78.43%
	60	555.12	440.20	79.30%	399.44	71.96%	383.88	69.15%
	120	460.48	329.84	71.63%	297.04	64.51%	282.84	61.42%
2	1	298.96	299.12	100.05%	295.52	98.85%	295.28	98.77%
	2	298.68	295.12	98.81%	294.60	98.63%	290.28	97.19%
	4	296.88	288.68	97.24%	283.44	95.47%	286.52	96.51%
	7	291.36	283.72	97.38%	279.44	95.91%	277.44	95.22%
	14	282.64	272.80	96.52%	264.88	93.72%	260.08	92.02%
	30	264.16	242.52	91.81%	229.44	86.86%	227.56	86.14%
	60	239.00	206.52	86.41%	188.56	78.90%	182.56	76.38%
	120	199.96	158.92	79.48%	146.52	73.27%	140.52	70.27%
4	1	123.48	122.44	99.16%	122.48	99.19%	124.88	101.13%
	2	123.20	122.40	99.35%	124.40	100.97%	123.20	100.00%
	4	123.84	122.28	98.74%	122.08	98.58%	119.00	96.09%
	7	122.20	119.64	97.91%	120.04	98.23%	117.64	96.27%
	14	117.44	115.00	97.92%	113.84	96.93%	113.08	96.29%
	30	109.80	104.12	94.83%	103.04	93.84%	101.20	92.17%
	60	99.56	93.20	93.61%	87.92	88.31%	85.64	86.02%
	120	80.72	70.76	87.66%	67.32	83.40%	66.56	82.46%
7	1	56.52	57.16	101.13%	57.08	100.99%	57.00	100.85%
	2	56.40	56.56	100.28%	57.28	101.56%	56.12	99.50%
	4	55.68	56.40	101.29%	57.44	103.16%	55.00	98.78%
	7	56.92	56.60	99.44%	55.48	97.47%	55.84	98.10%
	14	54.20	54.20	100.00%	53.92	99.48%	52.76	97.34%
	30	50.80	50.84	100.08%	49.00	96.46%	48.32	95.12%
	60	45.68	44.88	98.25%	42.72	93.52%	40.64	88.97%
	120	38.16	38.12	99.90%	33.28	87.21%	31.28	81.97%
14	1	20.00	20.76	103.80%	19.76	98.80%	19.60	98.00%
	2	19.76	19.48	98.58%	20.00	101.21%	20.12	101.82%
	4	19.96	21.04	105.41%	21.12	105.81%	21.72	108.82%
	7	19.96	20.00	100.20%	20.00	100.20%	19.88	99.60%
	14	20.44	20.32	99.41%	20.32	99.41%	20.24	99.02%
	30	17.48	17.48	100.00%	17.40	99.54%	17.44	99.77%
	60	17.08	17.08	100.00%	17.12	100.23%	17.04	99.77%
	120	13.72	13.72	100.00%	13.76	100.29%	13.24	96.50%

there are on average only 8.28 fewer transplants for arrival rate 1 and even 0.76 additional transplants for arrival rate 14). We observe similar findings for $L = 5ms$ while the percentages are in general a bit smaller than for $L = 1ms$. The same holds for $L = 10ms$. However, even for such a large latency, the number of transplants for small match run intervals is still very close to the conventional approach (e.g., independent of the arrival rate the privacy-preserving approach still achieves more than 98% of the number of transplants for the conventional approach if a match runs are executed daily).

Average Waiting Time. The average waiting time (i.e., the average time a patient waits until receiving a transplant) is nearly always larger for the privacy-preserving approach as the SMPC protocol execution may take up to 7 days whereas the runtime of the matching algorithm in the conventional approach is negligible. However, for those parameters where the number of transplants in the privacy-preserving approach is close to the conventional approach, the average waiting time only differs by a few days. This is acceptable as even in the conventional approach the patients on average wait multiple months before receiving a transplant. The corresponding simulation results can be found in the extended version [17].

Summary. The negative impact of the privacy-preserving approach can be considered negligible for arrival rates of 4 to 14 days if the match run interval is between 1 and 7 days and for arrival rates of 1 or 2 days if the match run interval equals 1 or 2 days. Thus, our approach is even practical for large platforms such as UNOS in the US where patient-donor pairs arrive very frequently. Also, our approach performs best for those parameters for which the number of transplants is highest. This coincides with those values which one would prefer in real-world platforms. Thus, the worse performance of our approach for large match run intervals is not of relevance in practice. Finally, while 95% of the number of transplants compared to the conventional approach may not seem acceptable in countries where kidney exchange is already practiced, our approach would increase the number of possible transplants from 0% to 95% of the maximum possible number of transplants in countries such as Germany where kidney exchange is still not practiced.

7 CONCLUSION AND FUTURE WORK

We have presented a privacy-preserving protocol for crossover kidney exchange and evaluated its performance for different properties of the underlying network. Our protocol clearly outperforms the existing protocol for kidney exchange and offers reasonable runtimes for up to 64 patient-donor pairs depending on the network latency. We evaluated the practicality of our protocol when used in a dynamic kidney exchange platform and compared it to a conventional approach. Overall, the cost induced by adding privacy is low. Specifically, if match runs are executed frequently, the difference between the privacy-preserving and the conventional approach is negligible w.r.t. the number of transplants that can be found.

One direction for future work is the development of protocols that can handle larger exchange cycles or chains such that also platforms that allow for such more sophisticated exchange structures can be made privacy-preserving. Furthermore, we will strive to add the possibility of prioritization and robustness (in case a computed match fails) to our protocol further closing the gap between conventional and privacy-preserving kidney exchange.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Resarch Foundation) - project number (419340256) and NSF grant CCF-1646999. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Computations for our simulations were performed with computing resources granted by RWTH Aachen University under Project RWTH0438.

REFERENCES

[1] David J Abraham, Avrim Blum, and Tuomas Sandholm. 2007. Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchange. *ACM Conference on Electronic Commerce*.

[2] Nikhil Agarwal, Itai Ashlagi, Eduardo Azevedo, Clayton Featherstone, and Ömer Karaduman. 2018. What Matters for the Productivity of Kidney Exchange?. In *AEA Papers and Proceedings*, Vol. 108.

[3] Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Van Vyve. 2013. Securely solving simple combinatorial graph problems. *International Conference on Financial Cryptography and Data Security*.

[4] Balamurugan Anandan and Chris Clifton. 2017. Secure minimum weighted bipartite matching. *IEEE Conference on Dependable and Secure Computing*.

[5] Tommy Andersson and Jörgen Kratz. 2020. Pairwise kidney exchange over the blood group barrier. *The Review of Economic Studies* 87, 3.

[6] Itai Ashlagi, Adam Bingaman, Maximilien Burq, Vahideh Manshadi, David Gamarnik, Cathi Murphey, Alvin E Roth, Marc L Melcher, and Michael A Rees. 2018. Effect of match-run frequencies on the number of transplants and waiting times in kidney exchange. *American Journal of Transplantation* 18, 5 (2018).

[7] Itai Ashlagi and Alvin E Roth. 2021. *Kidney Exchange: An Operations Perspective*. Technical Report. National Bureau of Economic Research.

[8] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computations. In *ACM Symposium on Theory of Computing*. ACM.

[9] Claude Berge. 1957. Two Theorems in Graph Theory. *Proceedings of the National Academy of Sciences of the United States of America* 43, 9 (1957).

[10] Péter Biró, Bernadette Haase-Kromwijk, Tommy Andersson, Eyjólfur Ingi Ásgeirsson, Tatiana Baltesová, Ioannis Boletis, Catarina Bolotinha, Gregor Bond, Georg Böhmig, Lisa Burnapp, et al. 2019. Building Kidney Exchange Programmes in Europe – An Overview of Exchange Practice and Activities. *Transplantation* 103, 7 (2019).

[11] Péter Biró, Joris van de Klundert, David Manlove, William Pettersson, Tommy Andersson, Lisa Burnapp, Pavel Chromy, Pablo Delgado, Piotr Dworczak, Bernadette Haase, et al. 2019. Modelling and optimisation in european kidney exchange programmes. In *European Journal of Operational Research*. Elsevier.

[12] Marina Blanton and Siddharth Saraph. 2015. Oblivious maximum bipartite matching size algorithm with applications to secure fingerprint identification. *European Symposium on Research in Computer Security*.

[13] Marina Blanton, Aaron Steele, and Mehrdad Alisagari. 2013. Data-oblivious graph algorithms for secure computation and outsourcing. *ACM SIGSAC symposium on Information, computer and communications security*.

[14] Norbert Blum. 1990. A new approach to maximum matching in general graphs. In *International Colloquium on Automata, Languages, and Programming*. Springer.

[15] Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. 2015. How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation. In *International conference on financial cryptography and data security*. Springer.

[16] Dan Bogdanov, Sven Laur, and Jan Willemson. 2008. Sharemind: A framework for fast privacy-preserving computations. In *European Symposium on Research in Computer Security*. Springer.

[17] Malte Breuer, Ulrike Meyer, and Susanne Wetzel. 2022. Privacy-Preserving Maximum Matching on General Graphs and its Application to Kidney Exchange (Extended Version). *arXiv preprint arXiv:2201.06446*.

[18] Malte Breuer, Ulrike Meyer, and Susanne Wetzel. Source Code. https://gitlab.com/rwth-itsec/crossover-exchange.

[19] Malte Breuer, Ulrike Meyer, Susanne Wetzel, and Anja Mühlfeld. 2020. A Privacy-Preserving Protocol for the Kidney Exchange Problem. In *Workshop on Privacy in the Electronic Society*. ACM.

[20] Octavian Catrina and Sebastiaan De Hoogh. 2010. Improved primitives for secure multiparty integer computation. *International Conference on Security and Cryptography for Networks*.

[21] Edward H Cole, Peter Nickerson, Patricia Campbell, Kathy Yetzer, Nick Lahaie, Jeffery Zaltzman, and John S Gill. 2015. The Canadian kidney paired donation program: a national program to increase living donor transplantation. *Transplantation* 99, 5 (2015).

[22] DESMO-J. http://desmoj.sourceforge.net. Accessed 30-Sep-2021.

[23] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. 2019. Failure-Aware Kidney Exchange. *Management Science* 65, 4 (2019).

[24] Jack Doerner, David Evans, and Abhi Shelat. 2016. Secure stable matching at scale. *ACM SIGSAC Conference on Computer and Communications Security*.

[25] Jack Edmonds. 1965. Paths, Trees, and Flowers. *Canadian Journal of mathematics* 17 (1965).

[26] Paolo Ferrari, Willem Weimar, Rachel J Johnson, Wai H Lim, and Kathryn J Tinckam. 2015. Kidney paired donation: principles, protocols and programs. *Nephrology Dialysis Transplantation* 30, 8 (2015).

[27] Harold N Gabow. 1976. An Efficient Implementation of Edmonds' Algorithm for Maximum Matching on Graphs. *J. ACM* 23, 2 (1976).

[28] Harold N Gabow and Robert E Tarjan. 1991. Faster scaling algorithms for general graph matching problems. *J. ACM* 38, 4 (1991).

[29] Oded Goldreich. 2004. *Foundations of Cryptography: Volume 2 - Basic Applications*. Cambridge University Press.

[30] Philippe Golle. 2006. A private stable matching algorithm. *International Conference on Financial Cryptography and Data Security*.

[31] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *Computer and Communications Security*. ACM.

[32] Marcel Keller and Peter Scholl. 2014. Efficient, oblivious data structures for MPC. *International Conference on the Theory and Application of Cryptology and Information Security*.

[33] John Launchbury, Iavor S Diatchki, Thomas DuBuisson, and Andy Adams-Moran. 2012. Efficient lookup-table protocol in secure multiparty computation. *ACM SIGPLAN international conference on Functional programming*.

[34] Silvio Micali and Vijay V Vazirani. 1980. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *Symposium on Foundations of Computer Science*. IEEE.

[35] World Health Organization. Top 10 Causes of Death. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death. Accessed 30-Sep-2021.

[36] U Pape and D Conradt. 1980. Maximales Matching in Graphen. *Ausgewählte Operations Research Software in FORTRAN* (1980).

[37] Organ Procurement and Transplantation Network. https://optn.transplant.hrsa.gov/data/view-data-reports/national-data/. *Accessed 30-Sep-2021*.

[38] M Sadegh Riazi, Ebrahim M Songhori, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. 2017. Toward Practical Secure Stable Matching. *Proc. Priv. Enhancing Technol.* 2017, 1.

[39] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979).

[40] Maciej Syslo, Narsingh Deo, and Janusz S Kowalik. 1983. *Discrete Optimization Algorithms with Pascal Programs*. Prentice Hall.

[41] Stefan Wüller, Michael Vu, Ulrike Meyer, and Susanne Wetzel. 2017. Using Secure Graph Algorithms for the Privacy-Preserving Identification of Optimal Bartering Opportunities. *Workshop on Privacy in the Electronic Society*.