# Privacy-Preserving Graph Algorithms in the Semi-honest Model

Justin Brickell and Vitaly Shmatikov

The University of Texas at Austin, Austin TX 78712, USA

**Abstract.** We consider scenarios in which two parties, each in possession of a graph, wish to compute some algorithm on their *joint graph* in a privacy-preserving manner, that is, without leaking any information about their inputs except that revealed by the algorithm's output.

Working in the standard secure multi-party computation paradigm, we present new algorithms for privacy-preserving computation of APSD (all pairs shortest distance) and SSSD (single source shortest distance), as well as two new algorithms for privacy-preserving set union. Our algorithms are significantly more efficient than generic constructions. As in previous work on privacy-preserving data mining, we prove that our algorithms are secure provided the participants are "honest, but curious."

**Keywords:** Secure Multiparty Computation, Graph Algorithms, Privacy.

## 1 Introduction

In this paper, we investigate scenarios with two mutually distrustful parties, each in possession of a graph (representing, *e.g.*, a network topology, a distribution channel map, or a social network). The parties wish to compute some algorithm on their *combined* graph, but do not wish to reveal anything about their private graphs beyond that which will be necessarily revealed by the output of the algorithm in question.

For example, consider two Internet providers who are contemplating a merger and wish to see how efficient the resulting joint network would be without revealing the details of their existing networks; or two transportation companies trying to determine who has the greatest capacity to ship goods between a given pair of cities without revealing what that capacity is or which distribution channels contribute to it; or two social networking websites wishing to calculate aggregate statistics such as degrees of separation and average number of acquaintances without compromising privacy of their users, and so on.

In this paper, we construct privacy-preserving versions of classic graph algorithms for APSD (all pairs shortest distance) and SSSD (single source shortest distance). Our algorithm for APSD is new, while the SSSD algorithm is a privacy-preserving transformation of the standard Dijkstra's algorithm. We also show that minimum spanning trees can be easily computed in a privacy-preserving manner. As one of our tools, we develop protocols for privacy-preserving set union, which are results of independent interest.

B. Roy (Ed.): ASIACRYPT 2005, LNCS 3788, pp. 236–252, 2005.

We demonstrate that our constructions are significantly more efficient than those based on generic constructions for secure multi-party computation such as Yao's garbled circuits [39]. Some of the efficiency gain is due to our use of canonical orderings on graph edges. We believe that this technique may find applicability beyond the problems considered in this paper.

We prove that our constructions are secure in the *semi-honest* model. Assuming that a party correctly follows the protocol, there is no efficient adversary that can extract more information from the transcript of the protocol execution than is revealed by that party's private input and the result of the graph algorithm. Our choice of the semi-honest model follows previous work on privacy-preserving data mining such as Lindell and Pinkas' construction for a privacy-preserving version of the ID3 decision tree learning algorithm [28], and constructions by Yang *et al.* for privacy-preserving classification [38].

In general, the semi-honest model seems to be the right fit for our setting, where there is no realistic way to verify that the parties are submitting their true graphs as private inputs. The best we could hope for in the case of actively malicious participants is a protocol in which the parties first commit to their graphs, and then prove at every step of the protocol that their inputs match their commitments. This would greatly complicate the protocols without providing any protection against parties who maliciously choose their graphs in such a way that the result of the computation on the joint graph completely reveals the other party's input. We leave investigation of privacy-preserving graph algorithms in the model with malicious participants to future work.

This paper is organized as follows. We survey related work in section 2, then present our definition of privacy in section 3 and our cryptographic toolkit, including a construction for private set union, in section 4. Section 5 contains the main results of the paper: privacy-preserving APSD and SSSD algorithms. Their complexity is analyzed in section 6. Conclusions are in section 7.

## 2 Related Work

This paper follows a long tradition of research on privacy-preserving algorithms in the so called *secure multiparty computation* (SMC) paradigm. Informally, security of a protocol in the SMC paradigm is defined as computational indistinguishability from some *ideal functionality*, in which a trusted third party accepts the parties' inputs and carries out the computation. The ideal functionality is thus secure by definition. The actual protocol is secure if the adversary's view in any protocol execution can be simulated by an efficient simulator who has access only to the ideal functionality, *i.e.*, the actual protocol does not leak any information beyond what is given out by the ideal functionality. Formal definitions for various settings can be found, for example, in [6,7,22].

Any polynomial-time multi-party computation can be done in a privacy-preserving manner using generic techniques of Yao [39] and Goldreich, Micali, and Wigderson [23]. Generic constructions, however, are sometimes impractical due to their complexity. Recent research has focused on finding more efficient

privacy-preserving algorithms for specific problems such as computation of approximations [18], auctions [33], set matching and intersection [20], surveys [19], computation of the k-th ranked element [1] and especially data mining problems such as privacy-preserving computation of decision trees [28], classification of customer data [38], and mining of vertically partitioned data [16,37].

The techniques we use in this paper are closely related to those previously used in the cryptographic version of privacy-preserving data mining, *e.g.*, by Lindell and Pinkas in their privacy-preserving transformation of the ID3 algorithm [28]. We, too, use generic Yao's protocol [39,29] as a building block. Yao's protocol can be implemented using efficient constructions for oblivious transfer [31,32] and secure function evaluation [30].

In this paper, we aim to follow the SMC tradition and provide provable cryptographic guarantees of security for our constructions. Another line of research has focused on *statistical privacy* in databases, typically achieved by randomly perturbing individual data entries while preserving some global properties [4,2,5,3,26,12,17]. A survey can be found in [36]. The proofs of security in this framework are statistical rather than cryptographic in nature, and typically permit some leakage of information, while supporting more efficient constructions. In this paradigm, Clifton *et al.* have also investigated various data mining problems [10,24,35,25], while Du *et al.* researched special-purpose constructions for problems such as privacy-preserving collaborative scientific analysis [14,13,34,15]. Recent work by Chawla *et al.* [8] aims to bridge the gap between the two frameworks and provide rigorous cryptographic definitions of statistical privacy in the SMC paradigm.

Another line of cryptographic research on privacy focuses on private information retrieval (PIR) [9,21], but the problems and techniques in PIR are substantially different from this paper.

## 3 Definition of Privacy

We use a simplified form of the standard definition of security in the static semi-honest model due to Goldreich [22] (this is the same definition as used, for example, by Lindell and Pinkas [28]).

**Definition 1.** *(computational indistinguishability): Let* $S \subseteq \{0,1\}^*$. *Two ensembles (indexed by* $S$*),* $X \stackrel{\text{def}}{=} \{X_w\}_{w \in S}$ *and* $Y \stackrel{\text{def}}{=} \{Y_w\}_{w \in S}$ *are* computationally indistinguishable (by circuits) *if for every family of polynomial-size circuits,* $\{D_n\}_{n \in \mathbb{N}}$*, there exists a negligible (*i.e.*, dominated by the inverse of any polynomial) function* $\mu : \mathbb{N} \mapsto [0,1]$ *so that*

$$|\Pr[D_n(w, X_w) = 1] - \Pr[D_n(w, Y_w) = 1]| < \mu(|w|)$$

*In such a case we write* $X \stackrel{c}{\equiv} Y$.

Suppose $f$ is a polynomial-time functionality (deterministic in all cases considered in this paper), and $\pi$ is the protocol. Let $x$ and $y$ be the parties' respective

private inputs to the protocol. For each party, define its *view* of the protocol as $(x, r^1, m_1^1, \ldots, m_k^1)$ (respectively, $(y, r^2, m_1^2, \ldots, m_l^2)$), where $r^{1,2}$ are the parties' internal coin tosses, and $m_j^i$ is the $j^{th}$ message received by party $i$ during the execution of the protocol. We will denote the $i^{th}$ party's view as $\mathsf{view}_i^\pi(x, y)$, and its output in the protocol as $\mathsf{output}_i^\pi(x, y)$.

**Definition 2.** *Protocol $\pi$ securely computes deterministic functionality $f$ in the presence of static semi-honest adversaries if there exist probabilistic polynomial-time simulators $S_1$ and $S_2$ such that*

$$\{S_1(x, f(x, y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{\equiv} \{\mathsf{view}_1^\pi(x, y)\}_{x,y\in\{0,1\}^*}$$
$$\{S_2(y, f(x, y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{\equiv} \{\mathsf{view}_2^\pi(x, y)\}_{x,y\in\{0,1\}^*}$$

*where $|x| = |y|$.*

Informally, this definition says that each party's view of the protocol can be efficiently simulated given only its private input and the output of the algorithm that is being computed (and, therefore, the protocol leaks no information to a semi-honest adversary beyond that revealed by the output of the algorithm).

## 4 Tools

As building blocks for our algorithms, we use protocols for privacy-preserving computation of a minimum $\min(x, y)$ and set union $S_1 \cup S_2$.

In the minimum problem, the parties have as their respective private inputs integers $x_1$ and $x_2$ which are representable in $n$ bits. They wish to privately compute $m = \min(x_1, x_2)$. Because this problem is efficiently solved by a simple circuit containing $O(n)$ gates, it is a good candidate for Yao's generic method [39]. An implementation of this functionality with Yao's garbled circuit requires 2 communication rounds with $O(n)$ total communication complexity and $O(n)$ computational complexity.

### 4.1 Privacy-Preserving Set Union

In the set union problem, parties $P_1$ and $P_2$ have as their respective private inputs sets $S_1$ and $S_2$ drawn from some finite universe $U$. They wish to compute the set $S = S_1 \cup S_2$ in a privacy-preserving manner, *i.e.*, without leaking which elements of $S$ are in the intersection $S_1 \cap S_2$. We will define $|S_1| = s_1$, $|S_2| = s_2$, $|S| = s$, and $|U| = u$.

In this section, we give two solutions for privacy-preserving set union: the iterative method, and the tree-pruning method. Both require communication and computational complexity that is logarithmic in $u$, provided $s$ is small (note that even if we are not concerned about privacy, computing the set union requires at least $O(s \lg u)$ bandwidth, although it can be done in 1 round). Appendix B surveys several previously proposed techniques that can be used to compute the set union, but these techniques are all either linear in $u$ (or worse), or do not fully preserve privacy.

*Iterative method.* The basic idea of the iterative method is to build up $S$ one element at a time, from "smallest" to "largest." Before the protocol begins, both parties agree upon a canonical total ordering for the entire universe $U$. As a result, each element in $U$ is given an integer label with $\lg u$ bits. In addition, we need a label representing $\infty$, for which can simply use the integer $u+1$. The protocol proceeds as follows:

*Step 1.* Set $S = \emptyset$.
*Step 2.* $P_1$ selects $m_1$ as the canonically smallest element in $S_1$, or sets $m_1 = \infty$ if $S_1 = \emptyset$. $P_2$ likewise selects $m_2$ as the canonically smallest element in $S_2$, or sets $m_1 = \infty$ if $S_1 = \emptyset$.
*Step 3.* Using a protocol for private minimum, $P_1$ and $P_2$ privately compute $m = \min(m_1, m_2)$.
*Step 4.* If $m = \infty$, stop and return $S$. Otherwise, $S = S \cup \{m\}$ and the parties remove $m$ from their input sets (it may be present in one or both). Then return to step 2.

The protocol preserves privacy because, given the output set $S$, a simulator can determine the value of $m$ at each iteration. The protocol used for computing the minimum is private, so there exists an efficient algorithm that can simulate its execution to the party $P_1$ given its input and the output $m$ (likewise for $P_2$). The simulator for the iterative method protocol uses the simulator for the minimum protocol as a subroutine, following the standard hybrid argument.

The iterative method protocol requires $s+1$ iterations, and in each iteration the minimum of two $(\lg u)$-bit integers is privately computed. Using Yao's method, this requires a circuit with $2 \lg u$ inputs and $O(\lg u)$ gates. The $2 \lg u$ oblivious transfers can all take place in parallel, and since Yao's method requires a constant number of rounds the whole protocol takes $O(s)$ communication rounds. The total communication and computational complexity for the iterative method is $O(s \lg u)$.

*Tree-pruning method.* Before the tree-pruning protocol begins, the participants agree on a $(\lg u)$-bit binary label for each element in the universe (note that a canonical total ordering would automatically provide such a label). The basic idea of the protocol is that the participants will consider label prefixes of increasing length, and use a privacy-preserving Bit-Or protocol (see appendix C) to determine if either participant has an element with that prefix in his set.

Initially, the single-bit prefixes "0" and "1" are set "live." The protocol proceeds through $\lg u$ rounds, starting with round 1. In the $i$th round, the participants consider the set P of $i$-bit "live" prefixes. For each prefix $p \in P$, each participant sets his respective 1-bit input to 1 if he has an element in his set with prefix $p$, and to 0 if he does not have any such elements. The participants then execute a privacy-preserving Bit-Or protocol on their respective 1-bit inputs. If the result of the Bit-Or protocol is 1, then $p0$ and $p1$ are set as live $(i+1)$-bit prefixes. Otherwise, $p0$ and $p1$ are dead prefixes.

By a simple inductive argument, the number of live prefixes in each round does not exceed $2 \cdot |S|$, because an $i$-bit prefix $p_i = b_1 \ldots b_i$ can be live if and

only if at least one of the participants has an element whose label starts with $b_1 \ldots b_{i-1}$, and the number of such elements cannot exceed the total number of elements in the union, *i.e.*, $|S|$.

In the last round ($i = \lg u$), the length of the prefix is the same as the length of the binary labels, and the entire set $P$ of live prefixes is declared to be the output $S$ of the privacy-preserving set union protocol.

The tree-pruning protocol preserves privacy because, given the output set $S$, a simulator can determine the output of each of the Bit-Or protocols. As in the case of the iterative method protocol, we can construct a simulator for the tree-pruning protocol that uses a simulator for the Bit-Or protocol as a subroutine, and prove its correctness using a hybrid argument. The construction is simple and is omitted for brevity.

The tree-pruning protocol requires $\lg u$ iterations, and in each iteration the pairwise Bit-Or of at most $2s$ bits is computed. These computations can all take place in parallel, so the protocol requires $O(\lg u)$ communication rounds. Each iteration requires $O(s)$ communication and computational complexity, so the entire protocol has complexity $O(s \lg u)$. Both the iterative method and tree pruning protocols have the same complexity, but different numbers of rounds. The iterative method requires fewer rounds when $s = o(\lg u)$.

## 5 Privacy-Preserving Algorithms on Joint Graphs

We now present our constructions that enable two parties to compute algorithms on their joint graph in a privacy-preserving manner. Let $G_1$ and $G_2$ be the two parties' respective weighted graphs. Assume that $G_1 = (V_1, E_1, w_1)$ and $G_2 = (V_2, E_2, w_2)$ are complete graphs on the same set of vertices, that is, $V_1 = V_2$ and $E_1 = E_2$. Let $w_1(e)$ and $w_2(e)$ represent the *weight* of edge $e$ in $G_1$ and $G_2$, respectively. To allow incomplete graphs, the excluded edges may be assigned weight $\infty$. We are interested in computing algorithms on the parties' joint *minimum* graph $\text{gmin}(G_1, G_2) = (V, E, w_{min})$ where $w_{min}(e) = \min(w_1(e), w_2(e))$, since minimum joint graphs seem natural for application scenarios such as those considered in section 1.

### 5.1 Private All Pairs Shortest Distance (APSD)

The All Pairs Shortest Distance (APSD) problem is the classic graph theory problem of finding shortest path distances between all pairs of vertices in a graph (see, *e.g.*, [11]). We will think of APSD($G$) as returning a complete graph $G' = (V, E', w')$ in which $w'(e_{ij}) = d_G(i, j)$ and $V$ is the original edge set of $G$. Here $d_G(i, j)$ represents the shortest path distance from $i$ to $j$ in $G$. This problem is particularly well suited to privacy-preserving computation because the solution "leaks" useful information that can be used by the simulator.

To motivate the problem, consider two shipping companies who are hoping to improve operations by merging so that they can both take advantage of fast shipping routes offered by the other company. They want to see how quickly the merged company would be able to ship goods between pairs of cities, but

they don't want to reveal all of their shipping times (and, in particular, their inefficiencies) in case the merger doesn't happen. In other words, they wish to compute APSD($G$) where $G = \text{gmin}(G_1, G_2)$.

The basic idea behind our construction is to build up the solution graph by adding edges in order from shortest to longest. The following algorithm takes as input the parties' complete graphs $G_1$ and $G_2$. The graphs may be directed or undirected, but they must have strictly positive weight functions.

1. For notational convenience we introduce a variable $k$, initially set to 1, that represents the iteration count of the algorithm. Color each edge in $E$ "blue" by letting $B^{(k)}$ denote the set of blue edges in the edge set $E$ at iteration $k$, and setting $B^{(0)} = E$. Let $R^{(k)}$ denote the set of "red" edges, $R^{(k)} \stackrel{def}{=} E - B^{(k)}$. The lengths of red edges have reached their final values and will not change as the algorithm proceeds, while the lengths of blue edges may still decrease.
2. A public graph $G_0^{(0)} = (V, E, w_0^{(0)})$ is created. Its edges are all initially weighted as $w_0^{(0)}(e) = \infty$. When the algorithm terminates after $n$ iterations, we will have $w_0^{(n)}(e_{ij}) = d_G(i, j)$ and $B^{(n)} = \emptyset$.
3. The parties compute the following public value

$$m_0^{(k)} = \min_{e \in B^{(k-1)}} w_0^{(k-1)}(e) \tag{1}$$

and the respective private values

$$m_1^{(k)} = \min_{e \in B^{(k-1)}} w_1(e), \text{ and} \tag{2}$$

$$m_2^{(k)} = \min_{e \in B^{(k-1)}} w_2(e) \tag{3}$$

4. Now the parties *privately* compute the length of the smallest blue edge among all three graphs, $m^{(k)} = \min(\min(m_1^{(k)}, m_0^{(k)}), \min(m_2^{(k)}, m_0^{(k)}))$, using a generic protocol for private minimum (section 4). This protocol does not reveal the larger value.
5. The parties form the following public set

$$S_0^{(k)} = \{e | w_0^{(k-1)}(e) = m^{(k)}\} \tag{4}$$

and the respective private sets

$$S_1^{(k)} = \{e | w_1(e) = m^{(k)}\}, \text{ and} \tag{5}$$

$$S_2^{(k)} = \{e | w_2(e) = m^{(k)}\} \tag{6}$$

By construction, $S_0^{(k)}$, $S_1^{(k)}$, and $S_2^{(k)}$ contain only blue edges.
6. First, the parties *privately* compute the set union $S^{(k)} = S_0^{(k)} \cup S_1^{(k)} \cup S_2^{(k)}$. This is done using the privacy-preserving set union algorithm from section 4.

Next, the color of each edge $e \in S^{(k)}$ is changed from blue to red by setting $B^{(k)} = B^{(k-1)} - S^{(k)}$. Define a weight function $w_0'^{(k)}$ by

$$w_0'^{(k)}(e) = \begin{cases} m^{(k)} & \text{if } e \in S^{(k)} \\ w_0^{(k-1)}(e) & \text{otherwise} \end{cases} \tag{7}$$

7. Examine triangles with an edge $e_{ij} \in S^{(k)}$, an edge $e_{jk} \in R^{(k)}$, and an edge $e_{ik} \in B^{(k)}$. Define the weight function $w_0^{(k)}$ by fixing these triangles if they violate the triangle inequality under $w_0'^{(k)}$. More precisely, if $w_0'^{(k)}(e_{ij}) + w_0'^{(k)}(e_{jk}) < w_0'^{(k)}(e_{ik})$, then define $w_0^{(k)}(e_{ik}) = w_0'^{(k)}(e_{ij}) + w_0'^{(k)}(e_{jk})$. Do the same for triangles with an edge $e_{ij} \in R^{(k)}$, an edge $e_{jk} \in S^{(k)}$, and an edge $e_{ik} \in B^{(k)}$.
8. If there are still blue edges, go to step 3. Otherwise stop; the graph $G_0^{(k)}$ holds the solution to APSD($G$).

The algorithm is proved correct in appendix A. The proof of privacy follows.

*Proof (Privacy).* We describe a simulator for $P_1$; the simulator is given $P_1$'s input to the protocol, $x$, and the output of the protocol, $f(x, y) = G'$. The simulators are identical for $P_1$ and $P_2$ except for the asymmetry in the simulation of the set union and minimum subprotocols. We assume that simulators for the subprotocols exist because they are private protocols. For instance, if Yao's protocol is used then we can use the simulator in [29].

We will assume that there are $n$ protocol rounds. The view of $P_1$ is

$$\{RT^m(x_1, y_1), RT^u(x_2, y_2), RT^m(x_3, y_3), \ldots, RT^u(x_{2n}, y_{2n})\} \tag{8}$$

where $RT^m$ denotes the *real transcript* of the private minimum protocol, and $RT^u$ denotes the real transcript of the private set union protocol.

We will show in later theorems that the output of each of these protocol executions can be computed by the simulator as a polynomial function of $G'$, which we will denote as $h_i^m(G')$ and $h_i^u(G')$. We will also show that $P_1$'s input to each of these protocol executions can be computed as a polynomial function of $x$ and $G'$ which we will denote as $g_i^m(x, G')$ and $g_i^u(x, G')$. The simulator can therefore use the subprotocol simulators as subroutines, producing the simulated transcript

$$\{ST^m(g_1^m(x, G'), h_1^m(G')), \ldots, ST^u(g_{2n}^u(x, G'), h_{2n}^u(G'))\} \tag{9}$$

where $ST^m$ and $ST^u$ denote the simulated transcripts of the minimum and union protocols, respectively.

We prove a hybrid argument over the simulated views for the minimum and set union protocols. First, define the hybrid distribution $H_i$ in which the first $i$ minimum/union protocols are simulated and the last $2n - i$ are real. Formally, let $H_i(x, y)$ denote the distribution:

$$\{ST^m(g_1^m(x, G'), h_1^m(G')), \ldots, ST^u(g_i^u(x, G'), h_i^u(G')),$$
$$RT^m(x_{i+1}, y_{i+1}), RT^u(x_{i+2}, y_{i+2}), \ldots, RT^u(x_{2n}, y_{2n})\}$$

We now prove that $H_0(x,y) \stackrel{c}{\equiv} H_{2n}(x,y)$ by showing that for all $i$, $H_i(x,y) \stackrel{c}{\equiv} H_{i+1}(x,y)$. For the sake of contradiction, assume the opposite, and choose $i$ so that $H_i(x,y) \stackrel{c}{\not\equiv} H_{i+1}(x,y)$. These two distributions differ in only one term, so there must be a polynomial-time distinguisher for either

$$ST^u(g_i^u(x,G'),h_i^u(G')) \text{ and } RT^u(x_i,y_i) \text{ or}$$
$$ST^m(g_i^m(x,G'),h_i^m(G')) \text{ and } RT^m(x_i,y_i)$$

However, this contradicts the privacy of the subprotocols, which implies that no such polynomial-time distinguishers exist.

We now show that for each execution of the set union and minimum subprotocols, $P_1$'s subprotocol input and the subprotocol output are computable as functions of $P_1$'s input and the output of the entire APSD protocol.

**Theorem 1.** *$m^{(k)}$ is efficiently computable as a function of $G'$.*

*Proof.* The edge weights found in $G'$ are $m^{(1)} < m^{(2)} < \ldots < m^{(n)}$. Therefore $m^{(k)}$ is the $k$th smallest edge weight in $G'$.

**Theorem 2.** *$S^{(k)}$ is efficiently computable as a function of $G'$.*

*Proof.* $S^{(k)}$ is the set of edges in $G'$ with weight $m^{(k)}$.

**Theorem 3.** *$m_1^{(k)}$ is efficiently computable as a function of $G_1$ and $G'$.*

*Proof.* $m_1^{(k)}$ is the smallest edge weight in $G_1$ that is $> m^{(k-1)}$, allowing that $m^{(0)} = 0$. This is because all edges with weight $\leq m^{(k-1)}$ are in $R^{(k-1)}$.

**Theorem 4.** *$S_1^{(k)}$ is efficiently computable as a function of $G_1$ and $G'$.*

*Proof.* $S_1^{(k)}$ is the set of edges in $G_1$ with weight $m^{(k)}$.

### 5.2 Private All Pairs Shortest Path

While there is only a single all pairs shortest distance solution for a given graph, there may be many all pairs shortest path solutions, because between a pair of points there may be many paths that achieve the shortest distance. As a side effect of engaging in the protocol described in section 5.1, the two participants learn an APSP solution. When defining the weight function $w_0^{(k)}$ by fixing violating triangles in $w_0^{\prime(k)}$ during step 7, a shortest path solution may be associated with the fixed edge. Specifically, if $w_0^{\prime(k)}(e_{ij}) + w_0^{\prime(k)}(e_{jk}) < w_0^{\prime(k)}(e_{ik})$, then the shortest path from $i$ to $k$ is through $j$.

In step 6 of subsequent iterations, when adding an edge $e_{ij} \in S^{(k)}$ to the set of blue edges, we can conclude that the shortest path from $i$ to $j$ is the edge $e_{ij}$ itself if $e_{ij} \notin S_0^{(k)}$, or is the shortest path solution as computed above if $e_{ij} \in S_0^{(k)}$.

Note that learning this APSP solution does not imply any violation of privacy, as it is the APSP solution implied by the APSD solution.

### 5.3 Private Single Source Shortest Distance (SSSD)

The Single Source Shortest Distance (SSSD) problem is to find the shortest path distances from a source vertex $s$ to all other vertices [11]. An algorithm to solve APSD also provides the solution to SSSD, but leaks additional information beyond that of the SSSD solution and cannot be considered a private algorithm for SSSD. Therefore, this problem warrants its own investigation.

Similar to the protocol of section 5.1, the SSSD protocol on the minimum joint graph adds edges in order from smallest to largest. This protocol is very similar to Dijkstra's algorithm, but is modified to take two graphs as input.

1. Set $w_1^{(0)} = w_1$ and $w_2^{(0)} = w_2$. Color all edges incident on the source $s$ blue by putting all edges $e_{si}$ into the set $B^{(0)}$. Set the iteration count $k$ to 1.
2. Both parties privately compute the minimum length of blue edges in their graphs.
$$m_1^{(k)} = \min_{e_{si} \in B^{(k-1)}} w_1^{(k-1)}(e_{si}),$$
$$m_2^{(k)} = \min_{e_{si} \in B^{(k-1)}} w_2^{(k-1)}(e_{si})$$
3. Using the privacy-preserving minimum protocol, compute
$$m^{(k)} = \min(m_1^{(k)}, m_2^{(k)}).$$
4. Each party finds the set of blue edges in its graph with length $m^{(k)}$.
$$S_1^{(k)} = \{e_{si} | w_1^{(k-1)}(e_{si}) = m^{(k)}\}, \text{ and}$$
$$S_2^{(k)} = \{e_{si} | w_2^{(k-1)}(e_{si}) = m^{(k)}\}$$
5. Using the privacy-preserving set union protocol, compute
$$S^{(k)} = S_1^{(k)} \cup S_2^{(k)}.$$
6. Color the edges in $S^{(k)}$ red by setting $B^k = B^{(k-1)} - S^{(k)}$. Define a weight function $w_1'^{(k)}$ by
$$w_1'^{(k)}(e) = \begin{cases} m^{(k)} & \text{if } e \in S^{(k)} \\ w_1^{(k-1)}(e) & \text{otherwise} \end{cases} \tag{10}$$
and a weight function $w_2'^{(k)}$ by
$$w_2'^{(k)}(e) = \begin{cases} m^{(k)} & \text{if } e \in S^{(k)} \\ w_2^{(k-1)}(e) & \text{otherwise} \end{cases} \tag{11}$$
7. Similar to the APSD algorithm, form the weight function $w_1^{(k)}$ by fixing the triangles in $w_1'^{(k)}$ that violate the triangle inequality and contain edges in $S^{(k)}$. $w_{2^{(k)}}$ is likewise formed from $w_2'^{(k)}$.
If there are still blue edges remaining, go to step 2. Otherwise stop; both parties now have a graph with each edge incident on $s$ colored red, and with the weight of these edges equal to the shortest path distance from $s$ to each vertex.

### 5.4 Minimum Spanning Tree

Suppose that two frugal telephone companies wish to merge. Each company has a cost function for connecting any pair of houses, and they want to connect every house as cheaply as possible using the resources available to the merged company. In other words, they wish to compute $\mathrm{MST}(\mathrm{gmin}(G_1, G_2))$. If they can perform this computation privately, then both companies can see the final result without revealing their entire cost functions.

Both Kruskal's and Prim's algorithms for MST are easily turned into private protocols using our techniques, because the algorithms already consider edges in order from smallest to largest. At each iteration, Kruskal's algorithm adds the shortest edge such that its addition does not form a loop. It is a simple task for each party to compute the set of edges which would not form loops, and then to privately compute the length of the shortest edge in this set. One problem arises when there are multiple edges that share this length. In the shortest path algorithms, we addressed this issue by adding all edges of appropriate length at the same time using the private set union protocol, but this will not work for MST. Instead, we can assign a canonical ordering to the edges, and at each step find the shortest length edges that are canonically "first." This will allow a simulator to determine, given the final MST, in what order the edges arrived.

## 6 Complexity Analysis

For each algorithm considered in this paper, we calculate the number of rounds, the total communication complexity, and the computational complexity, and compare them with the generic method. Using Yao's method on a circuit with $m$ gates and $n$ inputs requires $O(1)$ rounds, $O(m)$ communication, and $O(m+n)$ computational overhead. Lindell and Pinkas note in [28] that the computational overhead of the $n$ oblivious transfers in each invocation of Yao's protocol typically dominates the computational overhead for the $m$ gates, but for correct asymptotic analysis we must still consider the gates.

*Complexity of privacy-preserving APSD.* For our analysis we will assume that the edge set $E$ has size $n$, and that the maximum edge length is $l$. The generic approach to this problem would be to apply Yao's Method to a circuit that takes as input the length of every edge in $G_1$ and $G_2$, and returns as output $G = \mathrm{APSD}(\mathrm{gmin}(G_1, G_2))$. Clearly, such a circuit will have $2n \log l$ input bits. To count the number of gates, note that a circuit to implement Floyd-Warshall requires $O(n^{3/2})$ minimums and $O(n^{3/2})$ additions. For integers represented with $\log l$ bits, both of these functionalities require $\log l$ gates, so we conclude that Floyd-Warhsall requires $O(n^{3/2} \log l)$ gates. To compute gmin requires $O(n \log l)$ gates, but this term is dominated by the gate requirement for Floyd-Warshall. We conclude that the generic approach requires $O(1)$ rounds, $O(n^{3/2} \log l)$ communication, and $O(n^{3/2} \log l)$ computational overhead.

The complexity of our approach depends on the number of protocol iterations $k$, which is equal to the number of different edge lengths that appear in the solution graph. In iteration $i$, we take the minimum of two $(\lg l)$-bit integers, and compute a set union of size $s_i$. Because each edge in the graph appears in exactly one of the set unions, we also know that $\sum_{i=1}^{k} s_i = n$.

First we will determine the contribution to the total complexity made by the integer minimum calculations. If we use Yao's protocol, then each integer minimum requires a constant number of communication rounds, $O(\lg l)$ inputs, and $O(\lg l)$ gates, so the $k$ calculations together contribute $O(k)$ rounds, $O(k \lg l)$ communication complexity, and $O(k \lg l)$ computational complexity.

Complexity contribution of the set union subprotocols depends on whether we use the iterative method or the tree pruning method as described in section 4. If the iterative method is used, then the $k$ invocations of set union require a total of $O(n)$ rounds, $O(k \lg n)$ communication complexity, and $O(k \lg n)$ computational complexity. If the tree-pruning method is used, then $O(k \lg n)$ rounds are required, but the communication and computational complexity remains the same. The asymptotically better performance of the iterative method hides the fact that each of the $k$ rounds requires $O(\lg n)$ oblivious transfers, which are considerably more expensive than the $O(|s_i|)$ private Bit-Or computations performed in each of the $\lg u$ rounds of the tree-pruning method.

Using the iterative method for set union, and noting that $k = O(n)$, we conclude that our APSD protocol requires $O(n)$ communication rounds, $O(n \log n + n \log l)$ communication complexity, and $O(n \log n + n \log l)$ computational complexity. As compared to the generic approach, we have traded more rounds for better overall complexity.

*Complexity of privacy-preserving SSSD.* Complexity of SSSD is similar to that of APSD, except that the number of rounds is $k = O(v)$ and the total number of set union operations is $v$, where $v$ is the number of vertices ($O(e^{1/2})$). We conclude that our protocol requires $O(v)$ rounds, $O(v(\log v + \log l))$ oblivious transfers, and $O(v(\log v + \log e))$ gates. A generic solution, on the other hand, would require $O(v^2 \log l)$ oblivious transfers.

## 7 Conclusions

In this paper, we presented privacy-preserving protocols that enable two honest but curious parties to compute APSD and SSSD on their *joint graph*. A related problem is how to construct privacy-preserving protocols for graph *comparison*. Many of these problems (*e.g.*, comparison of the graphs' respective maximum flow values) reduce to the problem of privacy-preserving comparison of two values, and thus have reasonably efficient generic solutions. For other problems, such as graph isomorphism, there are no known polynomial-time algorithms even if privacy is not a concern. Investigation of other interesting graph algorithms that can be computed in a privacy-preserving manner is a topic of future research.

## References


1. G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k-th ranked element. In *Proc. Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 40–55. Springer-Verlag, 2004.
2. D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 247–255. ACM, 2001.
3. R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. 2003 ACM SIGMOD International Conference on Management of Data*, pages 86–97. ACM, 2003.
4. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. 2000 ACM SIGMOD International Conference on Management of Data*, pages 439–450. ACM, 2000.
5. M. Bawa, R. Bayardo, and R. Agrawal. Privacy-preserving indexing of documents on the network. In *Proc. 29th International Conference on Very Large Databases (VLDB)*, pages 922–933. Morgan Kaufmann, 2003.
6. D. Beaver. Foundations of secure interactive computing. In *Proc. Advances in Cryptology - CRYPTO 1991*, volume 576 of *LNCS*, pages 377–391. Springer-Verlag, 1992.
7. R. Canetti. Security and composition of multiparty cryptograpic protocols. *J. Cryptology*, 13(1):143–202, 2000.
8. S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Towards privacy in public databases. In *Proc. 2nd Theory of Cryptography Conference (TCC)*, volume 3378 of *LNCS*, pages 363–385. Springer-Verlag, 2005.
9. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.
10. C. Clifton, M. Kantarcioglou, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2):28–34, 2002.
11. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
12. I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. 22nd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 202–210. ACM, 2003.
13. W. Du and M. Atallah. Privacy-preserving cooperative scientific computations. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 273–294. IEEE, 2001.
14. W. Du and M. Atallah. Privacy-preserving cooperative statistical analysis. In *Proc. 17th Annual Computer Security Applications Conference (ACSAC)*, pages 102–112. IEEE, 2001.
15. W. Du, Y. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: linear regression and classification. In *Proc. 4th SIAM International Conference on Data Mining (SDM)*, pages 222–233. SIAM, 2004.
16. C. Dwork and K. Nissim. Privacy-preserving data mining on vertically partitioned databases. In *Proc. Advances in Cryptology - CRYPTO 2004*, volume 3152 of *LNCS*, pages 528–544. Springer-Verlag, 2004.
17. A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

18. J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. Wright. Secure multiparty computation of approximations. In *Proc. 28th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *LNCS*, pages 927–938. Springer-Verlag, 2001.
19. J. Feigenbaum, B. Pinkas, R. Ryger, and F. Saint-Jean. Secure computation of surveys. In *Proc. EU Workshop on Secure Multiparty Protocols*, 2004.
20. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proc. Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, 2004.
21. Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. Computer and System Sciences*, 60(3):592–629, 2000.
22. O. Goldreich. *Foundations of Cryptography: Volume II (Basic Applications)*. Cambridge University Press, 2004.
23. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. Annual 19th ACM Symposium on Theory of Computing (STOC)*, pages 218–229. ACM, 1987.
24. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *Proc. ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*. ACM, July 2002.
25. M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 599–604. ACM, 2004.
26. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proc. 3rd IEEE International Conference on Data Mining (ICDM)*, pages 99–106. IEEE, 2003.
27. L. Kissner and D. Song. Privacy-preserving set operations. In *Proc. Advances in Cryptology - CRYPTO 2005 (to appear)*. Springer-Verlag, 2005.
28. Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.
29. Y. Lindell and B. Pinkas. A proof of Yao's protocol for secure two-party computation. `http://eprint.iacr.org/2004/175`, 2004.
30. M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 590–599. ACM, 2001.
31. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proc. 12th Annual Symposium on Discrete Algorithms (SODA)*, pages 448–457. ACM, 2001.
32. M. Naor and B. Pinkas. Computationally secure oblivious transfer. *J. Cryptology*, 18(1):1–35, 2005.
33. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. 1st ACM Conference on Electronic Commerce*, pages 129–139. ACM, 1999.
34. H. Polat and W. Du. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Proc. 3rd IEEE International Conference on Data Mining (ICDM)*, pages 625–628. IEEE, 2003.
35. J. Vaidya and C. Clifton. Privacy-preserving association rule mining in vertically partitioned data. In *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 639–644. ACM, 2002.

36. V. Verykios, E. Bertino, I. Fovino, L. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.
37. R. Wright and Z. Yang. Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In *Proc. 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 713–718. ACM, 2004.
38. Z. Yang, S. Zhong, and R. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proc. 5th SIAM International Conference on Data Mining (SDM)*. SIAM, 2005.
39. A. Yao. How to generate and exchange secrets. In *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE, 1986.

## A Proof of Private APSD Protocol Correctness

Before proving the algorithm correct, we prove some supporting lemmas.

**Lemma 1.** *If an edge* $e \in R^k$ *and* $w_0^{(k)}(e) = l$ *then* $\forall j > k, w_0^{(j)}(e) = l$.

*Proof.* Intuitively, this says that once the protocol establishes the length of a red edge, it never changes. This follows from the protocol lacking operations that alter the length of red edges.

**Lemma 2.** *For an edge* $e \in R^{(k)}$, $w_0^{(k)}(e) \leq m^{(k)}$.

*Proof.* In step 6 of iteration $k$, for edges $e \in S^{(k)}$ we set $w_0^{(k)}(e) = m^{(k)}$ and $e \in R^{(k)}$. Apply lemma 1 to complete the proof.

**Lemma 3.** *For an edge* $e \in B^{(k)}$, $w_0^{(k)}(e) > m^{(k)}$.

*Proof.* First, we show that for an edge $e \in B^{(k)}$, $w_0'^{(k)}(e) > m^{(k)}$. If $w_0'^{(k)}(e) = m^{(k)}$ then $e \in S^{(k)}$ (and $e \notin B^{(k)}$). If $w_0'^{(k)}(e) < m^{(k)}$ and $e \in B^{(k)}$, then $w_0^{(k-1)}(e) < m^{(k)}$ and we would have defined a smaller $m^{(k)}$.

Now, for those edges $e$ where we have $w_0^{(k)}(e) < w_0'^{(k)}(e)$ because of step 7, we still have $w_0^{(k)}(e) > m^{(k)}$ because the right-hand side of the assignment is strictly greater than $m^{(k)}$.

**Lemma 4.** *For all edges* $e$, $e \in R^{(k)} \leftrightarrow w_0^{(k)}(e) \leq m^{(k)}$ *and* $e \in B^{(k)} \leftrightarrow w_0^{(k)}(e) > m^{(k)}$.

*Proof.* This is an immediate consequence of lemmas 2 and 3.

**Lemma 5.** *For every red edge* $e_{ij} \in R^{(k)}$, $w_0^{(k)}(e_{ij}) = d_G(i, j)$.

*Proof.* The proof is by induction on $k$. For $k = 0$, the result is trivial. We will now assume that the result holds for values less than $k$ and prove it for $k$.

Because of lemma 1, it is sufficient to prove that for edges $e_{ij} \in S^{(k)}$, $d_G(i, j) = m^{(k)}$. We consider two cases.

1. The shortest path from $i$ to $j$ in $G$ is the edge $e_{ij}$.
   In this case, $d_G(i,j) = \min(w_1(e_{ij}), w_2(e_{ij}))$. To complete the proof, it's enough to show that $w_0^{(k-1)}(e_{ij}) \geq d_G(i,j)$. Suppose that in some iteration $h < k$ we set $w_0^{(h)}(e_{ij}) = w_0'^{(h)}(e_{ik}) + w_0'^{(h)}(e_{kj})$ in step 7. Then by inductive hypothesis, this implies a shorter path from $i$ to $j$ than the edge $e_{ij}$ which is a contradiction.
2. The shortest path from $i$ to $j$ in $G$ is through $k$.
   In this case, $d_G(i,j) = d_G(i,k) + d_G(k,j)$. WLOG, assume that $w_0^{(k)}(e_{ik}) \geq w_0^{(k)}(e_{kj})$. Then by lemmas 1 and 4, we have that for some $h < k$, $w_0^{(k)}(e_{ik}) = m^{(h)}$. This means that in step 7 of iteration $h$ the protocol set $w_0^{(h)}(e_{ij}) = w_0^{(h)}(e_{ik}) + w_0^{(h)}(e_{kj})$. By the inductive hypothesis, $w_0^{(h)}(e_{ik}) = d_G(i,k)$ and $w_0^{(h)}(e_{kj}) = d_G(k,j)$. We conclude that $w_0^{(h)}(e_{ij}) = d_G(i,k) + d_G(k,j)$ and therefore that $w_0^{(k)}(e_{ij}) \leq d_G(i,k) + d_G(k,j)$. By the same argument as in the first case, we also have $w_0^{(k)}(e_{ij}) \geq d_G(i,k) + d_G(k,j)$. Therefore, $m^{(k)} = d_G(i,k) + d_G(k,j) = d_G(i,j)$.

It is now a simple task to prove algorithm correctness.

*Proof (Correctness).* Suppose the algorithm terminates after $n$ iterations. Then $R^{(n)} = E$. Apply lemma 5.

## B Survey of Privacy-Preserving Set Union Protocols

*Generic Yao's method.* It is easy to construct a circuit for computing the set union. Each party $P_p$ inputs one bit for every element $e$ in the universe $U$. The input bit $b_{pi}$ is set to 1 if party $P_p$ has element $e_i$ in his set, and 0 otherwise. The circuit consists of $|U|$ AND gates, each of which takes as inputs $b_{0i}$ and $b_{1i}$ and outputs $o_i = b_{0i} \wedge b_{1i}$. Then $o_i = 1$ iff element $e_i$ is in the set union. Since this circuit has $O(u)$ inputs and $O(u)$ gates, we conclude that the computational overhead and the communication complexity are both $O(u)$.

*Commutative encryption.* Clifton *et al.* [10] present a simple construction for privacy-preserving set union that uses commutative encryption. Each party encrypts the elements in its set, exchanges the encrypted sets with the other party, and then encrypts the other party's encrypted elements with its own key. The double-encrypted sets are then combined. Due to commutativity of encryption, all elements in the intersection appear as duplicates. They are removed, and the remaining elements are decrypted. Scrambling the order of elements may hide which elements are in the intersection, but the size of the intersection is still revealed, thus this method is not secure in the standard sense of definition 2. This protocol requires communication and computational complexity $O(|s_1| + |s_2|)$.

*Complement of set intersection.* When the universe $U$ is small, it is possible to use complementation and take advantage of the fact that $S_1 \cup S_2 = \overline{\bar{S}_1 \cap \bar{S}_2}$.

Freedman *et al.* [20] present a privacy-preserving protocol for set intersection that uses homomorphic encryption which requires $O(k)$ communication overhead and $O(k \ln \ln k)$ computation overhead, where $k$ is the size of the set intersection. For applications considered in this paper, sets $S_1$ and $S_2$ are very small, so their complements are of size $O(u)$. As a result, this method requires $O(u \ln \ln u)$ computation, which is unacceptable.

*Polynomial set representation.* Kissner and Song [27] present a method for representing sets as polynomials, and give several privacy-preserving protocols for set operations using these representations. They do not provide a protocol for the standard set union problem. Instead, they give a protocol for the "threshold set union" problem, in which the inputs are multi-sets and the output is the set of elements whose multiplicity of appearance in the union exceed some threshold; the intersection of the input sets is also revealed. When applied to regular sets (as opposed to multi-sets) this protocol does not preserve privacy as the intersection is the only information one can hope to keep private.

## C Privacy-Preserving Bit-Or

First, observe that the circuit for computing Or of 2 bits consists in a single gate. Therefore, even the generic construction using Yao's protocol [39] is efficient, requiring a single *1-out-of-2* oblivious transfer.

An alternative construction without oblivious transfers is provided by a semantically secure homomorphic encryption scheme such as ElGamal. Suppose Alice and Bob want to compute Or of their respective bits $b_A$ and $b_B$ in a privacy-preserving manner (Alice and Bob are honest, but curious). Alice picks some cyclic group $G$ of prime order $q$ with generator $g$ where the Decisional Diffie-Hellman problem is presumed hard, *e.g.*, the group of quadratic residues modulo some large prime $p = 2q + 1$, and chooses its secret key $k$ at random from $\{0, \ldots, q-1\}$. Alice sends to Bob its public key $q, g, g^k$ together with its ciphertext $c_A$, which is created as follows. If $b_A = 0$, then $c_A = (g^r, g^{kr})$, where $r$ is randomly selected from $\{0, \ldots, q-1\}$. If $b_A = 1$, then $c_A = (g^r, g \cdot g^{kr})$.

Upon receipt of $c_A = (\alpha, \beta)$ and Alice's public key, Bob computes $c_B$ as follows. First, it randomly picks $r' \in \{0, \ldots, q-1\}$. If $b_B = 0$, then $c_B = (\alpha^{r'}, \beta^{r'})$. If $b_B = 1$, then $c_B = (\alpha^{r'}, g^{r'} \cdot \beta^{r'})$. Bob returns $c_B$ to Alice.

Alice computes bit $b$ by decrypting $c_B = (\gamma, \delta)$ with its private key $k$, *i.e.*, $b = \frac{\delta}{\gamma^k}$. Clearly, if $b_A = b_B = 0$, then $b = 1$. In this case, Alice declares that $b_A \vee b_B = 0$. If $b \neq 1$, then Alice declares that $b_A \vee b_B = 1$.

To verify that this construction preserves privacy, observe that secrecy of $b_A$ follows from the semantic security of ElGamal. Now suppose $b_A = 1$. If $b_B = 0$, then the decrypted plaintext $b = g^{r'}$. If $b_B = 1$, then $b = g^{2r'}$. Since $B$ does not know $r'$, it cannot tell the difference. Thus, $A$ does not learn $b_B$ if $b_A = 1$.

(We are grateful to Stas Jarecki for a helpful discussion of constructions for privacy-preserving Bit-Or).