Privacy-Preserving Trade Chain Detection

Stefan Wüller[1,2](✉), Malte Breuer[1], Ulrike Meyer[1], and Susanne Wetzel[2]

[1] RWTH Aachen University, Aachen, Germany
{wueller,meyer}@itsec.rwth-aachen.de
[2] Stevens Institute of Technology, Hoboken, NJ, USA
swetzel@stevens.edu

Abstract. In this paper, we present a novel multi-party protocol to facilitate the privacy-preserving detection of trade chains in the context of bartering. Our approach is to transform the parties' private quotes into a flow network such that a minimum-cost flow in this network encodes a set of simultaneously executable trade chains for which the number of parties that can trade is maximized. At the core of our novel protocol is a newly developed privacy-preserving implementation of the cycle canceling algorithm that can be used to solve the minimum cost flow problem on encrypted flow networks.

1 Introduction

Bartering refers to the direct exchange of goods or services for other goods or services [9]. Nowadays, traded goods and services include books, rental cars, apartments, production surpluses, or idle times of employees. The attractiveness of bartering stems from the fact that it does not suffer from shortcomings of currencies such as foreign exchange problems, inflation, liquidity problems of banks, or concentration of economic power.

Today, a large fraction of bartering transactions is carried out via centralized (online) platforms which support their users in finding suitable trade partners. Since bartering involves sensitive personal data (e.g., negotiation ranges), a main objective of prior work [14–16] is to replace these central platforms by decentralized privacy-preserving protocols which allow a fixed number of parties to privately barter their commodities thus eliminating the risk that a platform operator may not only learn sensitive personal data but (to some extent) can also control and manipulate which parties eventually trade their commodities.

Specifically, in the considered bartering setting, the privacy-preserving multi-party protocols of [14–16] allow each party to specify a quote that includes an offered and a desired commodity along with the corresponding quantity ranges at which a party is willing to trade. The protocols then obliviously detect a trade which consists of disjoint *trade cycles* (of lengths greater than or equal to two). These trade cycles encode how the parties can exchange their commodities (in a cyclic fashion) such that each one of the trade partners is satisfied with the trade. From participating in such a protocol, a party only learns its *local view* of a trade, i.e., its direct trade partners and what to exchange with them. Yet a party's quote remains private at all times.

Besides trade cycles, a *trade chain* is another exchange structure which is widely studied in the literature (see, e.g., [4,5]). Specifically, trade chains are of importance when so-called *donor parties* are considered which are altruistic parties that give their offered commodity away for free (i.e., without receiving another commodity in return). Analogously to a trade cycle, a trade chain indicates how the parties can exchange their commodities, with the difference that the first party in a chain is a donor party and the last party in the trade chain does not have to give away its offered commodity. The analysis of the impact of considering trade chains (instead of considering only trade cycles) in the context of conventional (i.e., non privacy-preserving) bartering is an active field of research (see, e.g., [4,5]). Recent results show that considering trade chains can lead to a significant increase of the overall number of parties that can trade. However, to the best of our knowledge, to date there is no privacy-preserving bartering protocol yet that was explicitly designed for the detection of trade chains (or a combination of trade chains and trade cycles).

In this paper, we present a first step to close this gap by introducing an efficient bartering protocol that enables the distributed detection of trade chains in a privacy-preserving fashion. Our protocol detects an optimal set of simultaneously executable trade chains so that the number of parties that can trade is maximized while the parties' quotes are kept private at all times. Furthermore, we formally prove that from participating in our novel protocol, a party only learns its direct trade partners and what to exchange with them. At the core of the protocol is a novel privacy-preserving protocol implementing the *cycle canceling algorithm* that allows multiple parties to solve the minimum cost flow problem on encrypted flow networks.

2 Preliminaries

Let $e \leftarrow_{\$} S$ indicate that e is drawn uniformly at random from S and let $\mathbb{N}_b := \{1, \ldots, b\}$. For a logical statement B (e.g., $0 \wedge 1$ or $5 < 6$), the *Iverson Bracket* $[B]$ evaluates to 1 if B is true and to 0 otherwise. The index set of parties $P_1, \ldots, P_\iota$ ($\iota \in \mathbb{N}$) that participate in a multi-party protocol is defined as $\mathscr{P} := \{1 \ldots, \iota\}$.

A *directed graph* is a graph $G = (V, E)$ where each edge $(v, w) \in E$ with $v, w \in V$ is directed from v to w. For a directed graph $G = (V, E)$, a tuple $(v_1, v_2, \ldots, v_l)$ with $v_i \in V$ and $(v_i, v_{i+1}) \in E$ ($\forall i \in \mathbb{N}_{l-1}$) is referred to as *path* (of length l). If additionally $(v_l, v_1) \in E$, tuple $(v_1, v_2, \ldots, v_l)$ is referred to as *cycle* (of length l). A (directed) graph $G = (V, E)$ is often represented by means of an *adjacency matrix* $A := (a_{i,j})_{|V| \times |V|}$ where for all $i, j \in V$ $a_{i,j} = 1$ if $(i, j) \in E$ and $a_{i,j} = 0$ otherwise.

Let $G = (V, E)$ be a directed graph and let $h : E \to S$ be a function that maps each edge $(v, w) \in E$ to a value in S. For convenience, we sometimes encode h as a matrix $H := (h_{i,j})_{|V| \times |V|}$ where for all $i, j \in V$ $h_{i,j} = h(i, j)$ if $(i, j) \in E$ and $h_{i,j} = 0$ otherwise.

The following definitions are based on [6] (extended by a cost function) and are essential for the formalizing of our approach for the detection of trade chains.

Definition 1 (Flow Network). *A* flow network *is a directed graph $G = (V, E)$ with a capacity function $u : V \times V \to \mathbb{R}^{\geq 0}$ and a cost function $c : V \times V \to \mathbb{R}$ such that $u(v, w) = c(v, w) := 0$ in case that $(v, w) \notin E$. Furthermore, if $(v, w) \in E$ then $(w, v) \notin E$. A flow network has one so-called* source *node and one so-called* sink *node where the source $s \in V$ has no incoming edges and the sink $t \in V$ has no outgoing edges.*

Definition 2 (Flow). *A* flow *f in a flow network $G = (V, E)$ with capacity function $u(v, w)$ and cost function $c(v, w)$ is a function $f : V \times V \to \mathbb{R}$ such that $0 \leq f(v, w) \leq u(v, w)$ and for all $w \in V \backslash \{s, t\}$ $\sum_{v \in V} f(v, w) = \sum_{v \in V} f(w, v)$. The* value *of a flow f is defined as $|f| := \sum_{v \in V} f(s, v)$. A* maximum flow *$f$ is a flow in G where $|f|$ is maximized. The cost of a flow f is given by $\sum_{(v,w) \in E} c(v, w) \cdot f(v, w)$. A minimum cost flow f is a flow with minimized cost.*

Definition 3 (Residual Network). *Given a flow network $G = (V, E)$ and a flow f, the* residual network *$G_f = (V, E_f)$ with* residual capacity *u_f and* residual cost *c_f is defined as $E_f := \{(v, w) \in V \times V : u_f(v, w) > 0\}$ and*

$$u_f(v, w) := \begin{cases} u(v, w) - f(v, w) & \mathit{if}\ (v, w) \in E \\ f(w, v) & \mathit{if}\ (w, v) \in E \\ 0 & \mathit{otherwise} \end{cases}, \quad c_f(v, w) := \begin{cases} c(v, w) & \mathit{if}\ (v, w) \in E \\ -c(v, w) & \mathit{if}\ (w, v) \in E \\ 0 & \mathit{otherwise} \end{cases}$$

2.1 Paillier Threshold Cryptosystem

Our privacy-preserving bartering protocol for trade chain detection relies on the additively homomorphic Paillier cryptosystem which has been proven to be semantically secure [12]. More precisely, we make use of the (τ, ι) threshold variant of the Paillier cryptosystem from [10] where the private key is distributed among ι parties such that at least τ parties have to cooperate in order to decrypt a ciphertext. Figure 1 gives a brief overview of the corresponding key generation procedure, the encryption function, and some homomorphic properties. In the remainder of this paper, we omit the public and private key from our notation, define $[\![m]\!] := E(m)$, and represent negative integers by the upper half $[\lceil n/2 \rceil, n - 1]$ of the plaintext space $\mathbb{P}$ (cf. Fig. 1). With $\mathbb{C}$ we denote the corresponding ciphertext space (see Fig. 1). For convenience, we write the encryption of a matrix $A = (a_{i,j})_{m \times n}$ as $[\![A]\!] := ([\![a_{i,j}]\!])_{m \times n}$ and define $[\![A[i, j]]\!] = [\![a_{i,j}]\!]$.

2.2 Secure Multi-Party Computation

Secure multi-party computation (SMPC) allows a set of ι parties to compute an ι-input functionality $\mathcal{F}$ such that each party only learns its prescribed output and what can be deduced from it in combination with its private input—even in the presence of an adversary. In this paper, we consider a semi-honest adversary that

Key Generation:

- Generate two primes $p = 2p' + 1, q = 2q' + 1$ of bit length $k/2$ s.t. p', q' are also primes
- Set $n := pq$, $n' := p'q'$ and select $\beta \leftarrow_\$ \mathbb{Z}_n^*$, $(a, b) \leftarrow_\$ \mathbb{Z}_n^* \times \mathbb{Z}_n^*$
- Set $g := (1 + n)^a \cdot b^n \ mod\ n^2$ and $\Theta := an'\beta \ mod\ n$
- Public key: (g, n, Θ), Private key: (τ, ι) sharing of $\beta n'$
- Plaintext space: $\mathbb{P} := \mathbb{Z}_n^*$, Ciphertext space: $\mathbb{C} := \mathbb{Z}_{n^2}^*$

Encryption:

- $m \in \mathbb{P}$, $r \leftarrow_\$ \mathbb{Z}_n^*$, $E(m) := g^m r^n \ mod\ n^2$

Homomorphic Properties:

- $E(m_1) +_h E(m_2) := E(m_1) \cdot E(m_2) = E(m_1 + m_2)$ (homomorphic addition)

  shorthand notation: $\sum_{i=1}^{l}{}_h [\![m_i]\!] := [\![m_1]\!] +_h \ldots +_h [\![m_l]\!]$
- $E(m) \times_h a := (E(m))^a = E(a \cdot m)$ and $E(m) \times_h 0 := E(0)$ with $a \in \mathbb{Z} \setminus \{0\}$ (hom. scalar mult.)
- $E(m_1) -_h E(m_2) := E(m_1) +_h (E(m_2))^{-1} = E(m_1 - m_2)$ (homomorphic subtraction)

Fig. 1. Overview of the threshold Paillier variant from [10].

corrupts and controls a fixed set of parties following the protocol specifications but trying to learn as much as possible about the inputs of the honest parties.

Let $\overline{x} := (x_1, \ldots, x_\iota)$ and let $\mathcal{F} : (\{0,1\}^*)^\iota \rightarrow (\{0,1\}^*)^\iota, \overline{x} \mapsto (\mathcal{F}_1(\overline{x}), \ldots, \mathcal{F}_\iota(\overline{x}))$ be a multi-party functionality where P_ℓ ($\ell \in \mathscr{P}$) provides input x_ℓ and obtains output $\mathcal{F}_\ell(\overline{x})$. Furthermore, let π be an ι-party protocol allowing to compute $\mathcal{F}$. With $I := \{i_1, \ldots, i_\kappa\} \subset \mathscr{P}$ we denote the index set of $1 \leq \kappa < \iota$ corrupted parties controlled by the semi-honest adversary.

Informally, party P_ℓ's *view* on the execution of a protocol π on input $\overline{x}$ consists of the messages received during the protocol execution as well as the party's internal random coin tosses. Let $\overline{x}_I$ and $\mathcal{F}_I(\overline{x})$ denote the κ-tuples $(x_{i_1}, \ldots, x_{i_\kappa})$ and $(\mathcal{F}_{i_1}(\overline{x}), \ldots, \mathcal{F}_{i_\kappa}(\overline{x}))$, respectively. A protocol π is said to *securely* (i.e., correctly and privately) compute functionality $\mathcal{F}$ if there exists a probabilistic polynomial time simulator S which on input I, $\overline{x}_I$, and $\mathcal{F}_I(\overline{x})$ simulates a protocol transcript that is computationally indistinguishable from the view of the corrupted parties resulting from an actual protocol execution. For the sake of clarity, we enclose simulated values with angle brackets $\langle \cdot \rangle$.

A gate ρ (resp., a gate functionality $\mathcal{G}$) is a special type of protocol (resp., functionality) which obtains encrypted input and/or returns encrypted output. In general, these ciphertexts come from a higher-level protocol (resp., functionality) and the corresponding plaintext values are not known to any party. We write $(o) \leftarrow \mathcal{G}(x)$ (resp., $(o) \leftarrow \rho(x)$) to indicate that all parties provide the same input and obtain the same output.

3 Overview

3.1 Bartering Related Terminology

We consider multiple parties $P_1, \ldots, P_\iota$ where each party specifies one offered and one desired commodity. In particular, given a finite set $\mathscr{C}$ of commodities $c_1, \ldots, c_{|\mathscr{C}|}$, each party P_ℓ ($\ell \in \mathscr{P}$) specifies one quote $\mathbf{q}^{(\ell)} := (\mathbf{o}^{(\ell)}, \mathbf{d}^{(\ell)})$ where

$\mathscr{P}$	offer	demand
P_1	$(\mathsf{A}, 8)$	$(\lambda, 0)$
P_2	$(\mathsf{B}, 10)$	$(\lambda, 0)$
P_3	$(\mathsf{C}, 5)$	$(\mathsf{A}, 7)$
P_4	$(\mathsf{A}, 6)$	$(\mathsf{C}, 4)$
P_5	$(\mathsf{D}, 6)$	$(\mathsf{A}, 5)$
P_6	$(\mathsf{D}, 5)$	$(\mathsf{B}, 8)$

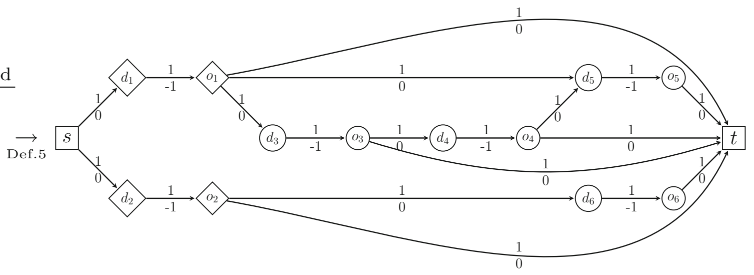

Fig. 2. Example for transforming a set of quotes $\mathbf{Q}$ into an exchange network. The source and the sink are represented by square nodes, the offer and the demand of donor parties are represented by diamond nodes, and the offer and the demand of ordinary parties are represented by round nodes. Each edge is annotated with its capacity (above) and its cost (below).

$\mathbf{o}^{(\ell)}$ refers to P_ℓ's offer and $\mathbf{d}^{(\ell)}$ refers to P_ℓ's demand. The offer $\mathbf{o}^{(\ell)} := (c_o^{(\ell)}, \overline{q}_o^{(\ell)})$ indicates that P_ℓ offers a quantity of at most $\overline{q}_o^{(\ell)} \in \mathbb{N}$ of commodity $c_o^{(\ell)} \in \mathscr{C}$. Similarly, the demand $\mathbf{d}^{(\ell)} := (c_d^{(\ell)}, \underline{q}_d^{(\ell)})$ of P_ℓ specifies the minimum quantity $\underline{q}_d^{(\ell)}$ at which it desires commodity $c_d^{(\ell)} \in \mathscr{C}$. A quote $\mathbf{q}^{(\ell)}$ indicates that party P_ℓ is willing to give at most $\overline{q}_o^{(\ell)}$ units of commodity $c_o^{(\ell)}$ iff it receives a least $\underline{q}_d^{(\ell)}$ units of commodity $c_d^{(\ell)}$. We distinguish a special type of a party P_ℓ, called *donor party*, with quote $\mathbf{q}^{(\ell)} := ((c_o^{(\ell)}, \overline{q}_o^{(\ell)}), (\lambda, 0))$ that is willing to give away at most $\overline{q}_o^{(\ell)}$ units of commodity $c_o^{(\ell)}$ where symbol λ indicates the absence of P_ℓ's demand. A party P_ℓ is referred to as *endowed party* in case that P_ℓ receives a specific quantity of its desired commodity without having to give away anything to another party in return. The quotes $\mathbf{q}^{(\ell)} = (\cdot, \mathbf{d}^{(\ell)})$ and $\mathbf{q}^{(\ell')} = (\mathbf{o}^{(\ell)}, \cdot)$ of two parties P_ℓ and $P_{\ell'}$ ($\ell \neq \ell'$) are *partially compatible* iff for $\mathbf{d}^{(\ell)} = (c_d^{(\ell)}, \underline{q}_d^{(\ell)})$ and $\mathbf{o}^{(\ell')} = (c_o^{(\ell')}, \overline{q}_o^{(\ell')})$ it holds that $[(c_d^{(\ell)} = c_o^{(\ell')}) \wedge (\underline{q}_d^{(\ell)} \leq \overline{q}_o^{(\ell')})] = 1$. The set of quotes of all parties $P_1, \ldots, P_\iota$ is denoted as $\mathbf{Q} := \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$.

Definition 4 (Trade Chain). *For parties $P_1, \ldots, P_\iota$ and their corresponding set of quotes $\mathbf{Q}$, a* trade chain *of length m is a tuple $(P_{\ell_1}, P_{\ell_2}, \ldots, P_{\ell_m})$ (with $\ell_i \neq \ell_j$ for $i \neq j$) such that $\mathbf{q}^{(\ell_1)} = (\mathbf{o}^{(\ell_1)}, (\lambda, 0))$ and $\mathbf{q}^{(\ell_2)} = (\cdot, \mathbf{d}^{(\ell_2)})$ as well as $\mathbf{q}^{(\ell_i)} = (\mathbf{o}^{(\ell_i)}, \cdot)$ and $\mathbf{q}^{(\ell_{i+1})} = (\cdot, \mathbf{d}^{(\ell_{i+1})})$ are partially compatible ($i = 2, \ldots, m-1$). Two trade chains are called* disjoint *and are* simultaneously executable *in case they have no parties in common.*

3.2 Approach

The goal of this work is to design an efficient privacy-preserving bartering protocol to determine an optimal set of disjoint trade chains (that when executed simultaneously maximize the number of parties that can trade). Our approach is to first transform the parties' private quotes into a special type of flow network, referred to as *exchange network* (see Fig. 2).

Definition 5 (Exchange Network). *For a given set of ι parties $\mathscr{P}$ and the corresponding set of quotes* $\mathbf{Q}$*, an* exchange network *is a flow network $G^{EN} = (V, E)$ with two nodes $d_\ell, o_\ell \in V$ for each party P_ℓ ($\ell \in \mathscr{P}$) representing its demand and offer. Furthermore, $(d_\ell, o_\ell) \in E$ with $u(d_\ell, o_\ell) := 1$ and $c(d_\ell, o_\ell) := -1$ as well as $(o_\ell, d_{\ell'}) \in E$ with $u(o_\ell, d_{\ell'}) := 1$ and $c(o_\ell, d_{\ell'}) := 0$ in case that $\mathbf{q}^{(\ell)} = (\mathbf{o}^{(\ell)}, \cdot)$ and $\mathbf{q}^{(\ell')} = (\cdot, \mathbf{d}^{(\ell')})$ are partially compatible ($\ell, \ell' \in \mathscr{P}$). In addition, G^{EN} has a source and a sink node $s, t \in V$ where $(s, d_\ell) \in E$ iff $\mathbf{q}^{(\ell)} = (\cdot, (\lambda, 0))$ with $u(s, d_\ell) := 1$ and $c(s, d_\ell) := 0$ as well as (o_ℓ, t) with $u(o_\ell, t) := 1$ and $c(o_\ell, t) := 0$ ($\forall o_\ell \in V$). $G^{EN} \sim \mathbf{Q}$ indicates that G^{EN} is deduced from $\mathbf{Q}$.*

A maximum flow of minimum cost f in $G^{\text{EN}} = (V, E)$ encodes an optimal set of disjoint trade chains for $P_1, \ldots, P_\ell$ where an edge $(o_\ell, d_{\ell'}) \in E$ with $f(o_\ell, d_{\ell'}) = 1$ indicates that a party P_ℓ has to give away some specific amount (to be negotiated after all parties learned their trade partners) of its offered commodity to party $P_{\ell'}$ ($\ell, \ell' \in \mathscr{P}$). A sequence S of edges $(s, d_{\ell_1}), (d_{\ell_1}, o_{\ell_2}), \ldots, (o_{\ell_m}, t)$ in G^{EN} with $f(v, w) = 1$ ($\forall (v, w) \in S$) encodes a single trade chain corresponding to $(P_{\ell_1}, P_{\ell_2}, \ldots, P_{\ell_m})$ where $\ell_1, \ell_2, \ldots, \ell_m \in \mathscr{P}$. This correlation is due to the construction of an exchange network: Each party P_ℓ is represented by two nodes in G^{EN} (with an edge of capacity 1 in between) where one node is associated with P_ℓ's demand and the second node is associated with P_ℓ's offer.[1] This ensures that there is at most a flow of 1 "through" each party enforcing that each party is involved in at most one trade chain. There are directed edges of capacity 1 between the source node and the demand nodes of all donor parties to ensure that each trade chain is initiated by a donor party. Furthermore, there are directed edges of capacity 1 between the sink node and the offer nodes of all parties so that in principle each party can become the end of a trade chain. Our cost encoding is motivated by the fact that for each additional party that is added to a trade chain, the cost of the flow is decreased by one such that determining a maximum flow of minimum cost f in G^{EN} is analogous to determining a set of disjoint trade chains maximizing the number of parties that can trade.

The problem of computing a (maximum) flow of minimum cost is known as the *minimum cost flow problem* (for a maximum flow).

Definition 6 (Minimum Cost Flow Problem). *Given a flow network $G = (V, E)$, a capacity function u, a cost function c, and a maximum flow f in G, the* minimum cost flow problem *is to find a flow f' of minimum cost with $|f| = |f'|$.*

One direct and efficient approach for solving the minimum cost flow problem (for a maximum flow) is to use the *cycle canceling algorithm* [11] (cf. Algorithm 1). This algorithm takes a flow network $G = (V, E)$, a capacity function u, as well as a cost function c as input and first computes a (maximum) flow f in G. Then, it iteratively eliminates directed cycles with negative cost (i.e., cycles for which the sum of the costs associated with its edges is negative) in the residual network arising from G (together with u and c) and f. To this end, the flow along the

[1] Note that there is also a demand node for each donor party in order to ensure that no information about them (e.g., the number of all donor parties) is leaked in our privacy-preserving bartering protocol.

Algorithm 1. Cycle Canceling Algorithm for Minimum Cost Flow.

```
Input : Flow network $G = (V, E)$ with capacity function $u$ and cost function $c$.
Output: A maximal flow $f$ in $G$ with minimum cost.

Initialization Phase
1  $f \leftarrow$ MaximumFlow$(G, u)$;
2  if $|f| = 0$ then
3      return $\perp$
4  $(G_f, u_f, c_f) \leftarrow$ ResidualNetwork$(G, u, c, f)$;
Main Phase
5  $N \leftarrow$ NegativeCostCycle$(G_f, u_f, c_f)$;
6  while $N$ exists do
7      $u^* \leftarrow \min\{u_f(e) : e$ is an edge of $N\}$;
8      $f \leftarrow$ AugmentFlow$(G, f, N, u^*)$;
9      $(G_f, u_f, c_f) \leftarrow$ ResidualNetwork$(G, u, c, f)$;
10     $N \leftarrow$ NegativeCostCycle$(G_f, u_f, c_f)$;
11 return $f$;
```

negative cost cycle is augmented by the minimum value of the residual capacities of the edges belonging to the cycle. This operation does not change $|f|$. The algorithm terminates once all negative cost cycles are eliminated. According to the *negative cycle optimality condition* [2], this approach allows the computation of a maximum flow with minimum cost.

The maximum flow f in G (Step 1, Algorithm 1) can be computed by using the *push-relabel algorithm* (see, e.g., [6]). A negative cost cycle (Steps 5 and 10, Algorithm 1) can be computed by an extension of the *Bellman-Ford algorithm* (see, e.g., [6]) that not only determines whether a negative cost cycle exists but also computes the edges belonging to such a cycle. After computing the negative cost cycle N, the minimum residual capacity u^* in the cycle is determined and the flow is augmented accordingly (Step 8, Algorithm 1).

At the core of our novel privacy-preserving bartering protocol for the detection of an optimal set of disjoint trade chains (Sect. 5) is a newly developed privacy-preserving implementation of the cycle canceling algorithm.

4 Gates

In the following, we review gates secure against semi-honest adversaries which are used as building blocks for our novel privacy-preserving bartering protocol.

4.1 Secure Basic Operations

Definition 7 ($\mathcal{G}_{\text{Mult}}$: Secure Multiplication). *Let $P_1, \ldots, P_\iota$ hold ciphertexts $[\![x]\!]$ and $[\![y]\!]$. Then, gate functionality $\mathcal{G}_{\text{Mult}}$ is given by $([\![x \cdot y]\!]) \leftarrow \mathcal{G}_{\text{Mult}}(([\![x]\!], [\![y]\!]))$.*

A gate ρ_{Mult} implementing $\mathcal{G}_{\mathrm{Mult}}$ for the semi-honest model can be derived from the multiplication gate presented in [7,8] which has communication and round complexities in $\mathcal{O}(\iota k)$ and $\mathcal{O}(1)$, respectively, where k refers to the security parameter of the Paillier cryptosystem.

Definition 8 ($\mathcal{G}_{\mathrm{LT}}$: Secure Less Than Comparison). *Let $P_1, \ldots, P_\iota$ hold ciphertexts $[\![x]\!]$ and $[\![y]\!]$. Then, gate functionality $\mathcal{G}_{\mathrm{LT}}$ is given by $([\![b]\!]) \leftarrow \mathcal{G}_{\mathrm{LT}}(([\![x]\!], [\![y]\!]))$ with $b := [x < y]$.*

A gate ρ_{LT} implementing $\mathcal{G}_{\mathrm{LT}}$ for the semi-honest model has been presented in [13]. This gate has communication and round complexities in $\mathcal{O}(\iota k)$ and $\mathcal{O}(\iota)$, respectively. Based on gate ρ_{LT} it is straight-forward to derive the corresponding greater than (GT), less than or equal (LTE), greater than or equal (GTE), and equality test (ET) variants as sketched, e.g., in [13].

4.2 Secure Negative Cost Cycle Computation

We use an adaptation of the *Bellman-Ford algorithm* (see, e.g., [6]) for the computation of negative cost cycles in exchange networks.

In general, the Bellman-Ford algorithm can be used to solve the *single-source shortest-paths* problem on a weighted directed graph $G = (V, E)$ for a given source node $s \in G$ where the weights of the edges are defined by a cost function $c : E \rightarrow \mathbb{R}$ (cf. [6]). The single-source shortest-paths problem is to find a shortest path (i.e., the path with the lowest cost) from the source node to all other nodes in G. Since the Bellman-Ford algorithm supports negative edge costs, there can be negative cost cycles in G implying that no shortest path can be found. In this case, the Bellman-Ford algorithm indicates that no solution exists. Otherwise, the algorithm provides a solution to the single-source shortest-paths problem. In particular, the Bellman-Ford algorithm iterates $|V|$ times over all edges $(v, w) \in E$ and for each node maintains the current lowest cost from the source node as well as the associated predecessor node. In case that the current solution can still be improved in the $|V|$-th iteration step, then G contains at least one negative cost cycle. Since we are not only interested in learning whether a negative cost cycle exists in G but also have to determine the edges of a negative cost cycle, the Bellman-Ford algorithm has to be slightly adapted such that, e.g., the node for which the last cost update is obtained is used in combination with the currently stored predecessors for all nodes to find the nodes of the negative cost cycle that induced the last cost update.

Definition 9 ($\mathcal{G}_{\mathrm{NCC}}$: Secure Negative Cost Cycle Computation). *Let $P_1, \ldots, P_\iota$ hold the encrypted adjacency matrix $[\![\mathcal{A}]\!] \in \mathbb{C}^{|V| \times |V|}$ of a directed graph $G = (V, E)$ and the encrypted cost matrix $[\![\mathcal{C}]\!] \in \mathbb{C}^{|V| \times |V|}$ encoding the cost function of G. The index of the source node s of G is publicly known. Then, gate functionality $\mathcal{G}_{\mathrm{NCC}}$ is given by $[\![\mathcal{N}]\!] \leftarrow \mathcal{G}_{\mathrm{NCC}}(([\![\mathcal{A}]\!], [\![\mathcal{C}]\!], s))$ where $[\![\mathcal{N}]\!] \in \mathbb{C}^{|V| \times |V|}$ is an encrypted adjacency matrix encoding a negative cost cycle in G.*

In [3], a secure protocol implementing the Bellman-Ford algorithm is proposed. It is straight-forward to modify this protocol to additionally extract the encrypted adjacency matrix $[\![\mathcal{N}]\!]$ encoding a negative cost cycle in G. The communication and round complexities of the adapted protocol from [3] are in $\mathcal{O}(\iota^4 k)$ and $\mathcal{O}(\iota^4)$ where k refers to the security parameter of the Paillier cryptosystem.

5 Protocol

In this section, we now present our novel privacy-preserving bartering protocol for the detection of trade chains.

Definition 10 ($\mathcal{F}_{\text{OTCD}}$: Optimal Trade Chain Detection). *Let P_ℓ hold private input $\mathbf{q}^{(\ell)}$ ($\forall \ell \in \mathscr{P}$). Then, protocol functionality $\mathcal{F}_{\text{OTCD}}$ is given by $(T^{(1)}, \ldots, T^{(\iota)}) \leftarrow \mathcal{F}_{\text{OTCD}}(\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)})$ where $T^{(\ell)} := (T^{(\ell)}_{\mathsf{send}}, T^{(\ell)}_{\mathsf{rec}})$ refers to the indices of P_ℓ's direct trade partners w.r.t. the detected trade chains derived from a maximum flow f^* with minimum cost in $G^{EN} \sim \mathbf{Q} := \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$ where f^* is chosen uniformly at random from all maximum flows of minimum cost in G^{EN}.*

5.1 Intuition

Intuitively, our novel protocol π_{OTCD} (securely implementing functionality $\mathcal{F}_{\text{OTCD}}$ in the semi-honest model) can be divided into four phases (see Algorithm 2). In the first phase, the parties compute the encrypted capacity matrix $[\![\mathcal{U}]\!]$ and the cost matrix $[\![\mathcal{C}]\!]$, encoding the capacity function and the cost function of the private exchange network $G^{\text{EN}} = (V, E) \sim \mathbf{Q}$. These matrices are computed in an oblivious fashion such that no party learns any information about the quote of another party. In the second phase, a maximum flow f (not necessarily having minimum cost) is computed in an oblivious fashion where the result is encoded in an encrypted matrix $[\![\mathcal{F}]\!]$. Based on $[\![\mathcal{F}]\!]$, the encrypted capacity matrix of the residual network of G^{EN} is computed in an oblivious fashion. The third phase uses gate ρ_{NCC} (see Sect. 4) to iteratively find a negative cost cycle (where in ρ_{NCC} the order of the edges to be processed is chosen uniformly at random) in the current residual network of G^{EN} in a privacy-preserving fashion. In order to eliminate the negative cost cycles, the flow as well as the residual capacities are updated by performing homomorphic operations on $[\![\mathcal{F}]\!]$ and $[\![\mathcal{U}_f]\!]$. At the end of the third phase, a maximum flow f^* with minimum cost is encoded by means of $[\![\mathcal{F}]\!]$ (which in turn represents an optimal set of trade chains). In the fourth phase, the parties jointly extract the identifiers $T^{(\ell)}_{\mathsf{send}}$, $T^{(\ell)}_{\mathsf{rec}}$ of the trade partners of each party P_ℓ from $[\![\mathcal{F}]\!]$ such that a party only learns its own trade partners as prescribed by Definition 10. The identifiers indicate that party P_ℓ has to give away some quantity of its offered commodity to party $P_{T^{(\ell)}_{\mathsf{send}}}$ (receiver) and is to receive some quantity of its desired commodity from party $P_{T^{(\ell)}_{\mathsf{rec}}}$ (sender). An identifier of value 0 is used to indicate that a sender (resp., receiver) does not exist. For example, $T^{(\ell)}_{\mathsf{rec}} = 0$ for a donor party P_ℓ and $T^{(\ell')}_{\mathsf{send}} = T^{(\ell')}_{\mathsf{rec}} = 0$ for a party $P_{\ell'}$ that is not part of a trade chain.

The main challenge of designing a protocol that securely implements functionality $\mathcal{F}_{\text{OTCD}}$ is to keep the parties' quotes $\mathbf{Q}$ (and with that the structure of the resulting exchange network $G^{\text{EN}} \sim \mathbf{Q}$) private. Consequently, it is necessary to design a data oblivious protocol (i.e., the protocol flow is independent of the parties private input) that provides individualized output (i.e., each party only learns its local view of the detected trade chains). Algorithm 1 (see Sect. 3.2) is not data oblivious because the while loop in the main phase terminates once there are no further negative cost cycles. Furthermore, it cannot be used to provide individualized output as it is designed to operate on a public flow network. By fixing the number of iterations for finding negative cost cycles to $|\iota|$, we ensure that all negative cost cycles are found while the protocol flow becomes data oblivious. In case that there are no further negative cost cycles, our protocol obliviously operates on encrypted dummy cycles that do not influence the already computed encrypted maximum flow of minimum cost. Finally, we adopt a technique from [16] to extract the local view of each party from the computed optimal set of trade chains represented by the encrypted maximum flow of minimum cost.

5.2 Protocol Description

In the following, we present the details of protocol π_{OTCD}. For convenience, we associate the 2ℓ-th and the $(2\ell + 1)$-th row (resp., column) of the encrypted adjacency matrices used for the protocol specification of π_{OTCD} with node d_ℓ and node o_ℓ of G^{EN} ($\forall \ell \in \mathscr{P}$), respectively. The first and the last row (resp., column) are associated with the source node s and the sink node t, respectively.

1. Exchange Network Construction Phase: The purpose of the first phase is to compute the encrypted matrices $[\![\mathcal{U}]\!], [\![\mathcal{C}]\!] \in \mathbb{C}^{|V|\times|V|}$ with $|V| = 2\iota + 2$ in an oblivious fashion. These matrices encode the capacity and the cost function of the exchange network $G^{\text{EN}} = (V, E)$ resulting from the parties' private input quotes $\mathbf{Q}$ (see Definition 5) and represent $G^{\text{EN}} \sim \mathbf{Q}$.

First, P_1 initializes the entries at position $(2\ell, 2\ell + 1)$ of the matrices $[\![\mathcal{U}]\!]$ and $[\![\mathcal{C}]\!]$ which encode the directed edge from nodes d_ℓ to o_ℓ representing the demand and offer of party P_ℓ ($\forall \ell \in \mathscr{P}$). In particular, P_1 sets these entries in $[\![\mathcal{U}]\!]$ to $[\![1]\!]$ and in $[\![\mathcal{C}]\!]$ to $[\![-1]\!]$. Additionally, the entries $[\![\mathcal{C}[2\ell + 1, 2\ell]]\!]$ representing the corresponding reverse edges from o_ℓ to d_ℓ (which are not in G^{EN} but may exist in the residual network) are set to $[\![1]\!]$ for later use. Furthermore, party P_1 sets the capacity of the edges from each node o_ℓ ($\forall \ell \in \mathscr{P}$) to the sink node t appropriately by $[\![\mathcal{U}[2\ell+1, t]]\!] := [\![1]\!]$. All other entries of $[\![\mathcal{U}]\!]$ and $[\![\mathcal{C}]\!]$ are set to $[\![0]\!]$ before they are broadcasted by P_1 (see Steps 1–6, Algorithm 2). Subsequently, all parties obliviously determine the donor parties and update the capacities of the edges between the source node and the demand nodes of the donor parties in $[\![\mathcal{U}]\!]$ to $[\![1]\!]$. Finally, for all pairs of parties $(P_\ell, P_{\ell'})$ with $\ell, \ell' \in \mathscr{P}$ it is obliviously checked whether $\mathbf{q}^{(\ell)} = (\mathbf{o}^{(\ell)}, \cdot)$ and $\mathbf{q}^{(\ell')} = (\cdot, \mathbf{d}^{(\ell')})$ are partially compatible. If this is the case, the encrypted capacity matrix is obliviously updated by setting $[\![\mathcal{U}[2\ell + 1, 2\ell']]\!] := [\![1]\!]$ (see Steps 7–10, Algorithm 2).

Algorithm 2. π_{OTCD} for optimal trade chain detection.

```
   Input  : Quote $\mathbf{q}^{(\ell)}$ of party $P_\ell$ ($\forall \ell \in \mathscr{P}$).
   Output: Tuple $T^{(\ell)} \in \mathscr{P} \cup \{0\} \times \mathscr{P} \cup \{0\}$ for party $P_\ell$ ($\forall \ell \in \mathscr{P}$).
   Exchange Network Construction Phase
1  Party $P_1$:
2      Initialize $[\![\mathcal{U}]\!], [\![\mathcal{C}]\!]$ of size $|V|\times|V|$, $|V|{:=}2\iota{+}2$ by $[\![\mathcal{U}[v,w]]\!]{=}[\![\mathcal{C}[v,w]]\!]{:=}[\![0]\!]$ $\forall v, w{\in}V$;
3      foreach $\ell \in \mathscr{P}$ do
4          Set $[\![\mathcal{U}[2\ell, 2\ell+1]]\!] := [\![1]\!]$ and $[\![\mathcal{U}[2\ell+1, t]]\!] := [\![1]\!]$;
5          Set $[\![\mathcal{C}[2\ell, 2\ell+1]]\!] := [\![-1]\!]$ and $[\![\mathcal{C}[2\ell+1, 2\ell]]\!] := [\![1]\!]$;
6      Broadcast $[\![\mathcal{U}]\!], [\![\mathcal{C}]\!]$;
7      foreach $\ell \in \mathscr{P}$ all parties do Jointly compute $([\![\mathcal{U}[s, 2\ell]]\!]) \leftarrow \rho_{\mathrm{ET}}(([\![c_d^{(\ell)}]\!], [\![\lambda]\!]))$;
8      foreach $\ell, \ell' \in \mathscr{P}$ $(\ell \neq \ell')$ all parties do
9          Jointly comp. $([\![cond_1]\!]){\leftarrow}\rho_{\mathrm{ET}}(([\![c_o^{(\ell)}]\!], [\![c_d^{(\ell')}]\!]))$,
             $([\![cond_1']\!]){\leftarrow}\rho_{\mathrm{GTE}}(([\![\overline{q}_o^{(\ell)}]\!], [\![\underline{q}_d^{(\ell')}]\!]))$;
10         Jointly compute $([\![\mathcal{U}[2\ell+1, 2\ell']]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![cond_1]\!], [\![cond_1']\!]))$;
   Flow Initialization Phase
11 Party $P_1$:
12     Initialize $[\![\mathcal{F}]\!], [\![\mathcal{U}_f]\!]$ of size $|V| \times |V|$ by $[\![\mathcal{F}[v,w]]\!] = [\![\mathcal{U}_f[v,w]]\!] := [\![0]\!]$ $\forall v, w \in V$;
13     foreach $\ell \in \mathscr{P}$ do Set $[\![\mathcal{F}[s, 2\ell]]\!] = [\![\mathcal{F}[2\ell, 2\ell+1]]\!] = [\![\mathcal{F}[2\ell+1, t]]\!] := [\![\mathcal{U}[s, 2\ell]]\!]$;
14     Broadcast $[\![\mathcal{F}]\!], [\![\mathcal{U}_f]\!]$;
15 foreach $v, w \in V$ $(v \neq w)$ all parties do
16     Jointly compute $([\![cond_2]\!]) \leftarrow \rho_{\mathrm{ET}}(([\![\mathcal{F}[v,w]]\!], [\![1]\!]))$;
17     Jointly compute $([\![\mathcal{U}_f[v,w]]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![cond_2]\!], [\![\mathcal{U}[v,w]]\!] -_h [\![\mathcal{F}[v,w]]\!]))$
         $+_h\ \rho_{\mathrm{Mult}}(([\![1]\!] -_h [\![cond_2]\!], [\![\mathcal{U}_f[v,w]]\!]))$;
18     Jointly compute $([\![\mathcal{U}_f[w,v]]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![1]\!] -_h [\![cond_2]\!], [\![\mathcal{U}_f[w,v]]\!])) +_h [\![cond_2]\!]$;
   Cycle Canceling Phase
19 Repeat $\iota$ many times
20     All parties jointly compute $[\![\mathcal{N}]\!] \leftarrow \rho_{\mathrm{NCC}}(([\![\mathcal{U}_f]\!], [\![\mathcal{C}]\!], t))$;
21     foreach $v, w \in V$ $(v \neq w)$ all parties do
22         Jointly compute $([\![cond_3]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![\mathcal{N}[v,w]]\!], [\![\mathcal{U}[v,w]]\!]))$;
23         Jointly compute $([\![cond_3']\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![\mathcal{N}[v,w]]\!], [\![\mathcal{U}[w,v]]\!]))$;
24         Jointly compute $([\![\mathcal{F}[v,w]]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![cond_3]\!], [\![\mathcal{F}[v,w]]\!] +_h [\![1]\!]))$
             $+_h\ \rho_{\mathrm{Mult}}(([\![1]\!] -_h [\![cond_3]\!], [\![\mathcal{F}[v,w]]\!]))$;
25         Jointly compute $([\![\mathcal{F}[w,v]]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![cond_3']\!], [\![\mathcal{F}[w,v]]\!] -_h [\![1]\!]))$
             $+_h\ \rho_{\mathrm{Mult}}(([\![1]\!] -_h [\![cond_3']\!], [\![\mathcal{F}[w,v]]\!]))$;
26     foreach $v, w \in V$ $(v \neq w)$ all parties do
27         Jointly compute $([\![cond_4]\!]) \leftarrow \rho_{\mathrm{ET}}(([\![\mathcal{F}[v,w]]\!], [\![1]\!]))$;
28         Jointly compute $([\![\mathcal{U}_f[v,w]]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![cond_4]\!], [\![\mathcal{U}[v,w]]\!] -_h [\![\mathcal{F}[v,w]]\!]))$
             $+_h\ \rho_{\mathrm{Mult}}(([\![1]\!] -_h [\![cond_4]\!], [\![\mathcal{U}_f[v,w]]\!]))$;
29         Jointly compute $([\![\mathcal{U}_f[v,u]]\!]) \leftarrow \rho_{\mathrm{Mult}}(([\![1]\!] -_h [\![cond_4]\!], [\![\mathcal{U}_f[v,u]]\!])) +_h [\![cond_4]\!]$;
   Output Extraction Phase
30 foreach $\ell \in \mathscr{P}$ do
31     Party $P_\ell$:
32         Compute $[\![T_{\mathsf{send}}^{(\ell)}]\!] := \sum_{i=1}^{\iota}{}_h (i \times_h [\![\mathcal{F}[2\ell{+}1, 2i]]\!])$, $[\![T_{\mathsf{rec}}^{(\ell)}]\!] := \sum_{i=1}^{\iota}{}_h (i \times_h [\![\mathcal{F}[2i{+}1, 2\ell]]\!])$;
33         Broadcast $[\![T_{\mathsf{send}}^{(\ell)}]\!]$, $[\![T_{\mathsf{rec}}^{(\ell)}]\!]$;
34     All parties jointly decrypt $[\![T_{\mathsf{send}}^{(\ell)}]\!]$ and $[\![T_{\mathsf{rec}}^{(\ell)}]\!]$ s.t. only $P_\ell$ learns the result;
35     Party $P_\ell$ sets $T^{(\ell)} := (T_{\mathsf{send}}^{(\ell)}, T_{\mathsf{rec}}^{(\ell)})$;
36     Party $P_\ell$ outputs $T^{(\ell)}$;
```

2. Flow Initialization Phase: The purpose of the second phase is to obliviously compute a maximum flow f in G^{EN} which is encoded by the encrypted matrix $[\![\mathcal{F}]\!] \in \mathbb{C}^{|V|\times|V|}$. Based on $[\![\mathcal{F}]\!]$, the residual capacities of G^{EN} are initialized and encoded by the encrypted matrix $[\![\mathcal{U}_f]\!] \in \mathbb{C}^{|V|\times|V|}$. Instead of using a (privacy-preserving) variant of the *push-relabel algorithm* for computing a maximum flow in G^{EN}(see Sect. 3.2), we follow a more efficient approach that exploits G^{EN}'s particular structure: A flow of value 1 is sent from source node s to each demand node associated with a donor party. Then, the flow continues on to the corresponding offer node of the donor party and from there on directly to the sink node t. Note that such a flow is maximal since the maximum flow in G^{EN} is upper bounded by the number of donor parties. Furthermore, from the construction of an exchange network (see Definition 5) it follows that there always is such a flow in G^{EN}. In protocol π_{OTCD}, party P_1 obliviously determines this flow locally by setting $[\![\mathcal{F}[s, 2\ell]]\!] = [\![\mathcal{F}[2\ell, 2\ell+1]]\!] = [\![\mathcal{F}[2\ell+1, t]]\!] := [\![\mathcal{U}[s, 2\ell]]\!]$ $(\forall \ell \in \mathscr{P})$. In Steps 15–18 of Algorithm 2, the parties jointly compute the entries of $[\![\mathcal{U}_f]\!]$ based on $[\![\mathcal{U}]\!]$ and $[\![\mathcal{F}]\!]$. More precisely, in Step 16 it is obliviously checked whether or not there is a flow between two nodes v, w $(\forall v, w \in V,\ v \neq w)$. Based on the result, $[\![\mathcal{U}_f]\!]$ is obliviously updated according to Definition 3 (see Steps 17–18).

3. Cycle Canceling Phase: In the conventional cycle canceling algorithm (see Algorithm 1), the while loop in the main phase is executed until all negative cost cycles are eliminated. In order to leak no information on the structure of the private exchange network G^{EN}, protocol π_{OTCD} has to be data oblivious and thus the number of searches for negative cost cycles has to correspond to the upper bound of necessary searches (to eliminate all negative cost cycles) which is equal to $\iota := |\mathscr{P}|$ (see Theorem 1).

At the beginning of each iteration of the cycle canceling phase (see Step 20, Algorithm 2) gate ρ_{NCC} is used to obliviously compute a negative cost cycle in the residual network of G^{EN}. In gate ρ_{NCC}, the order of the edges to be processed is chosen uniformly at random. First, assume that such a cycle exists. This cycle is encoded by the encrypted matrix $[\![\mathcal{N}]\!] \in \mathbb{C}^{|V|\times|V|}$ which constitutes the output of gate ρ_{NCC}. In Steps 21–25, for each edge $(v, w) \in E$ that is part of the determined negative cost cycle, the flow is obliviously updated in the following way: In case that the edge under consideration is part of the exchange network, the corresponding entry in $[\![\mathcal{F}[v, w]]\!]$ is obliviously incremented by one (see Step 24, Algorithm 2). Otherwise, the edge results from a residual flow over (w, v) and thus entry $[\![\mathcal{F}[w, v]]\!]$ is obliviously decreased by one (see Step 25, Algorithm 2). Based on the updated flow $[\![\mathcal{F}]\!]$, in Steps 26–29, the residual capacities $[\![\mathcal{U}_f]\!]$ are updated analogously to the flow initialization phase.

In case that there is no (further) negative cost cycle before the end of the ι-th iteration, then all entries of $[\![\mathcal{N}]\!]$ correspond to a fresh encryption of 0 and thus the privacy-preserving computations performed on $[\![\mathcal{F}]\!]$ and $[\![\mathcal{U}_f]\!]$ are just re-randomizations of the existing encrypted entries. Consequently, in each iteration of the cycle canceling phase it is kept private whether a (further) negative cost cycle exists.

4. Output Extraction Phase: At the end of the cycle canceling phase, the encrypted matrix $[\![\mathcal{F}]\!]$ encodes a maximum flow of minimum cost in G^{EN} that in turn represents an optimal set of trade chains for the participating parties. The purpose of the output extraction phase is to extract the parties' local views w.r.t. the computed trade chains from $[\![\mathcal{F}]\!]$. In particular, party P_ℓ ($\forall \ell \in \mathscr{P}$) locally computes the encryption of $T^{(\ell)}_{\mathsf{send}}$ (resp., $T^{(\ell)}_{\mathsf{rec}}$) as the homomorphic sum of the ℓ-th row (resp., ℓ-th column) where each encrypted entry is multiplied with the corresponding column (resp., row) index by using the homomorphic scalar multiplication operation of the underlying cryptosystem (see Step 32, Algorithm 2). Party P_ℓ broadcasts the resulting encrypted values $[\![T^{(\ell)}_{\mathsf{send}}]\!]$ and $[\![T^{(\ell)}_{\mathsf{rec}}]\!]$ which are jointly decrypted by all parties in such a way that only P_ℓ learns the indices of its direct trade partners.

Complexity. The complexity of the exchange network construction phase is dominated by the $\mathcal{O}(\iota^2)$ calls of gates ρ_{ET}, ρ_{GTE}, and ρ_{Mult} (see Steps 8–10, Algorithm 2). The flow initialization phase has the same communication and round complexity which results from the iteration over all pairs of nodes in G^{EN} in order to compute the residual capacities (see Steps 15–18). The ι executions of gate ρ_{NCC} dominate the cycle canceling phase since the communication complexity (resp., round complexity) of protocol ρ_{NCC} is in $\mathcal{O}(\iota^4 k)$ (resp., $\mathcal{O}(\iota^4)$). Finally, the output extraction phase has a communication complexity (resp., round complexity) in $\mathcal{O}(\iota k)$ (resp., in $\mathcal{O}(1)$). The overall complexity of protocol π_{OTCD} is dominated by the cycle canceling phase, i.e., the communication complexity (resp., round complexity) of π_{OTCD} is in $\mathcal{O}(\iota^5 k)$ (resp., in $\mathcal{O}(\iota^5)$).

Theorem 1. *Let party P_ℓ hold private input $\mathbf{q}^{(\ell)}$ ($\forall \ell \in \mathscr{P}$). Then, protocol π_{OTCD} securely computes functionality $\mathcal{F}_{\mathrm{OTCD}}$ in the semi-honest model.*

Proof. Correctness (sketch): In the following, we show that on input $\mathbf{Q} = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\ell)}\}$, protocol π_{OTCD} computes functionality $\mathcal{F}_{\mathrm{OTCD}}$ (see Definition 10).

In the exchange network construction phase, $\mathbf{Q}$ is obliviously transformed into an exchange network $G^{\mathrm{EN}} \sim \mathbf{Q}$ represented by the encrypted matrices $[\![\mathcal{U}]\!]$ and $[\![\mathcal{C}]\!]$. These matrices are constructed according to Definition 5 based on local as well as distributed computations on the parties' private input quotes.

In the flow initialization phase, an initial maximum flow (not necessarily with minimum cost) from the source node through the donors' demand and offer nodes to the sink node is computed locally by party P_1. This flow (with a flow value equal to the number of donor parties) always exists due to the construction of an exchange network (see Definition 5). This flow is also maximal since each edge (with capacity 1) leaving the source node is incident to a donor's demand node. The capacity of the residual network of G^{EN} w.r.t. to the initial flow is computed according to Definition 3 (the cost function of the residual network was already set during the exchange network construction phase).

The correctness of the cycle canceling phase can be reduced to the correctness of Algorithm 1 and the correctness of gate ρ_{NCC}. This phase essentially implements Steps 5–10 of Algorithm 1 in a privacy-preserving fashion. After determining a negative cost cycle in the current residual network of G^{EN} by using gate ρ_{NCC}, the flow along the negative cost cycle is augmented by 1 and the residual network is updated accordingly. Unlike Algorithm 1, the number of iterations of the while loop (see Step 6, Algorithm 1) in Algorithm 2 is fixed to ι in order to achieve data obliviousness. First, it is important to note that ι iterations are sufficient to eliminate all negative cost cycles because there are ι edges with negative costs in G^{EN} and in each iteration the flow along at least one edge with negative cost is increased by 1. Furthermore, in case that there are no negative cost cycles before the last iteration of the loop has terminated, the encrypted adjacency matrix $[\![\mathcal{N}]\!]$ (see Step 20, Algorithm 2) merely consists of fresh encryptions of 0 and the maximum flow of minimum cost in G^{EN} that is already computed and encoded by $[\![\mathcal{F}]\!]$ is not modified by the operations performed in Steps 21–29 of Algorithm 2.

The last phase of protocol π_{OTCD} extracts each party's trade partners from $[\![\mathcal{F}]\!]$ which encodes an optimal set of trade chains. In Step 32 of Algorithm 2, $[\![T^{(\ell)}_{\mathsf{send}}]\!]$ (resp., $[\![T^{(\ell)}_{\mathsf{rec}}]\!]$) corresponds to the encryption of index $2i$ (resp., index $2i+1$) of the $(2\ell+1)$-th row (resp., 2ℓ-th column) where $[\![\mathcal{F}[2\ell+1, 2i]]\!] := [\![1]\!]$ (resp., $[\![\mathcal{F}[2i+1, 2\ell]]\!] := [\![1]\!]$). From the computation of $[\![\mathcal{F}]\!]$ it follows that $T^{(\ell)} = (T^{(\ell)}_{\mathsf{send}}, T^{(\ell)}_{\mathsf{rec}})$ provides party P_ℓ $(\forall \ell \in \mathscr{P})$ with the indices of its trade partners w.r.t. the computed optimal set of trade chains.

Privacy (sketch): In the following, we describe a simulator S which, given $\mathbf{q}^{(i_1)}, \ldots, \mathbf{q}^{(i_\kappa)}$ and $(T^{(i_1)}_{\mathsf{send}}, T^{(i_1)}_{\mathsf{rec}}), \ldots, (T^{(i_\kappa)}_{\mathsf{send}}, T^{(i_\kappa)}_{\mathsf{rec}})$, simulates the view of the corrupted parties $P_{i_1}, \ldots, P_{i_\kappa}$ $(I := \{i_1, \ldots, i_\kappa\} \subset \mathscr{P})$ that are controlled by a semi-honest adversary.

The initialization of $[\![\mathcal{U}]\!]$ and $[\![\mathcal{C}]\!]$ which is computed and broadcasted by P_1 in Steps 1–6 (Algorithm 2) can be computed in the same way by S. The following steps of the exchange network construction phase are simulated by using the subsimulators of ρ_{ET}, ρ_{GTE}, and ρ_{Mult} and by setting $\langle [\![\mathcal{C}[s, 2\ell]]\!] \rangle \leftarrow_{\$} \mathbb{C}$, $\langle [\![cond_1]\!] \rangle \leftarrow_{\$} \mathbb{C}$, $\langle [\![cond'_1]\!] \rangle \leftarrow_{\$} \mathbb{C}$, and $\langle [\![\mathcal{C}[2\ell+1, 2\ell']]\!] \rangle \leftarrow_{\$} \mathbb{C}$. The flow initialization phase can be simulated analogously to the first phase of protocol π_{OTCD}. In order to simulate the ι iterations of the cycle canceling phase, S uses the subsimulator of ρ_{NCC} and sets $\langle [\![\mathcal{N}]\!] \rangle \leftarrow_{\$} \mathbb{C}^{|V| \times |V|}$. The remaining steps can be simulated in the same way as described for the exchange network construction phase. The broadcasts sent in the output extraction phase (see Step 33, Algorithm 2) are simulated as $\langle [\![T^{(\ell)}_{\mathsf{send}}]\!] \rangle \leftarrow_{\$} \mathbb{C}$ and $\langle [\![T^{(\ell)}_{\mathsf{rec}}]\!] \rangle \leftarrow_{\$} \mathbb{C}$, respectively. The output of the individual decryption operations is simulated based on the corrupted parties' protocol output which is given to S as input.

6 Related Work and Discussion

To the best of our knowledge, in the literature there are only privacy-preserving multi-party protocols that allow the detection of trade cycles:

The authors of [15] propose a privacy-preserving bartering protocol (secure in the semi-honest model) by means of which multiple parties can compute a set of trade cycles based on their private input quotes. In this protocol, the parties' private quotes are transformed into logical formulae which are evaluated in an oblivious fashion. From participating in the protocol, a party only learns its direct trade partners. The protocol in [15] has two interesting features: First, it allows to put arbitrary restrictions on the lengths of the trade cycles to be detected. Depending on the bartering context an upper bound on the trade cycle lengths is essential in order to reduce the impact of a dropout and to facilitate simultaneous exchanges preventing that a party gives away its offered commodity but does not receive its desired commodity in return (cf. [1,4]). Second, the protocol supports the integration of arbitrary selection strategies for the detection of trade cycles (e.g., a strategy that maximizes the number of parties that can trade their commodities). However, the complexity of the protocol can grow exponentially in the number of participating parties which is inevitable as soon as a restriction on the trade cycle length (greater than 2) is supported because the underlying decision problem is NP-complete [1].

Another privacy-preserving bartering protocol for the detection of trade cycles has been presented in [16]. This protocol follows a completely different approach compared to the protocol from [15]. The parties' private quotes are transformed into a private weighted bipartite graph. At the core of the protocol is a privacy-preserving variant of the *Hungarian algorithm* which is used to obliviously compute a maximum weight matching in the weighted bipartite graph which encodes an optimal set of trade cycles maximizing the number of parties that can trade. The communication and the round complexities of this protocol are in $\mathcal{O}(\iota^6 k)$ and $\mathcal{O}(\iota^6)$, respectively. A restriction of the trade cycle lengths is not supported and thus the number of applications is limited.

In contrast to trade cycles, there is no need to restrict the length of trade chains in order to prevent that a party gives away its offered commodity without receiving its desired commodity: By conducting the trades in the order specified by a trade chain (starting with the donor party), in the worst case the trade chain is just aborted prematurely. Obviously, it is possible to reduce the problem of the privacy-preserving detection of trade chains to the privacy-preserving detection of trade cycles by setting the demand of a donor party to a dummy entry that matches with all offers. Then, it is straight-forward to use the protocol from [16] for the privacy-preserving detection of trade chains in time polynomial in the number of participating parties. However, using our novel *direct* approach (see Sect. 5) it is possible to reduce the communication complexity (resp., the round complexity) from $\mathcal{O}(\iota^6 k)$ (resp., $\mathcal{O}(\iota^6)$) to $\mathcal{O}(\iota^5 k)$ (resp., $\mathcal{O}(\iota^5)$). Based on these theoretical results, we expect that our novel protocol yields a significant performance improvement over the existing (more general) protocols for the privacy-preserving detection of trade chains.

Acknowledgments. In part, this work was supported by NSF grant #1646999 and DFG grant ME 3704/4-1. This work was carried out while one of the authors was at the National Science Foundation. 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.

References

1. Abraham, D.J., Blum, A., Sandholm, T.: Clearing algorithms for barter exchange markets: enabling nationwide kidney exchanges. In: Proceedings of the 8th ACM Conference on Electronic Commerce, pp. 295–304. ACM (2007)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc., Englewood Cliffs (1993)
3. Aly, A., Cuvelier, E., Mawet, S., Pereira, O., Van Vyve, M.: Securely solving simple combinatorial graph problems. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 239–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39884-1_21
4. Anderson, R., Ashlagi, I., Gamarnik, D., Kanoria, Y.: A dynamic model of barter exchange. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1925–1933 (2014)
5. Anderson, R., Ashlagi, I., Gamarnik, D., Kanoria, Y.: Efficient dynamic barter exchange. Oper. Res. **65**(6), 1446–1459 (2017)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, Cambridge (2009)
7. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. Technical report (2000)
8. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–300. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44987-6_18
9. Encyclopedia Britannica. www.britannica.com
10. Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45472-1_7
11. Klein, M.: A primal method for minimal cost flows with applications to the assignment and transportation problems. Manag. Sci. **14**(3), 205–220 (1967)
12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
13. Wüller, S.: Privacy-preserving electronic bartering. Ph.D. thesis, RWTH Aachen University (2018)
14. Wüller, S., Meyer, U., Wetzel, S.: Privacy-preserving multi-party bartering secure against active adversaries. In: Fifteenth Annual Conference on Privacy, Security and Trust. IEEE (2017)
15. Wüller, S., Meyer, U., Wetzel, S.: Towards privacy-preserving multi-party bartering. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 19–34. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_2
16. Wüller, S., Vu, M., Meyer, U., Wetzel, S.: Using secure graph algorithms for the privacy-preserving identification of optimal bartering opportunities. In: Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, pp. 123–132. ACM (2017)