Privacy-Preserving Electronic Bartering

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Stefan Wüller, M.Sc. RWTH
aus Aachen

Berichter: Univ.-Prof. Dr.-Ing. Ulrike Meyer
Univ.-Prof. Dr.-Ing. Susanne Wetzel

Tag der mündlichen Prüfung: 21. Februar 2018

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

To all those who did not dedicate their theses to themselves.

Abstract

E-commerce applications like online shopping, e-marketplaces, and e-banking are becoming more and more prevalent in our daily lives. While providing a lot of convenience, these applications generally require the disclosing of sensitive personal data. Typically, it is not transparent for their users what the personal data are used for. A majority of users may be willing to share sensitive personal data on the Internet to some extent when there is a balance between the involved benefits and drawbacks. However, this is certainly not the case if those data (e.g., one's room for negotiation) have the potential to adversely affect e-commerce transactions which, in particular, may occur particularly in the context of online bartering marketplaces.

The research goal of this thesis is to advance the privacy-protection of online bartering marketplaces such that users do not have to disclose private data (including their offers/demands as well as the quantities thereof) to anyone in order to find suitable trade partners. Our approach is to design privacy-preserving protocols that can be used as a key component of a bartering system allowing its users to barter their commodities in a privacy-preserving fashion. More precisely, we devise a novel privacy-preserving bartering protocol for the two-party case providing security against active adversaries as well as two novel privacy-preserving bartering protocols for the multi-party case which provide security against passive and active adversaries, respectively. The focus of this thesis is on the much more complicated multi-party case which, compared to the two-party case, requires fundamentally different design approaches as well as the development of novel privacy-preserving building blocks for comparison and selection operations that are of general interest beyond the context of bartering. Using our privacy-preserving multi-party bartering protocols (which are shown to be practical for a limited number of parties) as a key component, we model a bartering system that allows an arbitrary number of parties (arriving at the system over time) to barter their commodities in a privacy-preserving fashion. The implementation and the simulation of our novel privacy-preserving bartering model as well as the comparison to the most prominent conventional bartering models show that the modeled privacy-preserving bartering system is practical.

Zusammenfassung

In unserem täglichen Leben gewinnen Formen des elektronischen Handels wie Online Shopping, Online Banking und elektronische Marktplätze zunehmend an Bedeutung. Sie bieten ihren Nutzern viele Annehmlichkeiten, erfordern jedoch häufig die Preisgabe sensibler persönlicher Daten. Wofür diese Daten jedoch letztendlich verwendet werden, bleibt in vielen Fällen undurchsichtig, was viele Nutzer als Nachteil empfinden. Wenn eine Balance zwischen Vor- und Nachteilen besteht, ist eine Vielzahl von Nutzern bereit, sensible persönliche Daten im Internet preiszugeben. Diese Balance besteht jedoch nicht mehr, wenn sich die Preisgabe dieser Daten (zum Beispiel Verhandlungsspielräume) negativ auf ihre elektronischen Handelstransaktionen auswirken kann. Dieses Problem tritt insbesondere im Kontext von elektronischen Tauschmärkten auf.

Das Forschungsziel dieser Dissertation ist die Entwicklung kryptographischer Protokolle, die dem Schutz der Privatsphäre von Nutzern elektronischer Tauschmärkte dienen. Dazu werden Privatsphäre wahrende Protokolle entwickelt, die als Schlüsselkomponenten in einem Tauschsystem verwendet werden können, welches es seinen Nutzern ermöglicht, ihre Waren ohne die Preisgabe sensibler persönlicher Daten (einschließlich Angebot und Nachfrage) zu tauschen.

In der vorliegenden Arbeit wird ein neues Privatsphäre wahrendes Zweiparteientauschprotokoll (welches Sicherheit gegen aktive Angreifer bietet) sowie zwei neue Privatsphäre wahrende Mehrparteientauschprotokolle (sicher gegen passive bzw. aktive Angreifer) entwickelt. Der Schwerpunkt der Arbeit liegt auf dem wesentlich komplizierteren Mehrparteienfall, welcher im Vergleich zum Zweiparteienfall völlig neue Designansätze sowie die Entwicklung neuer kryptographischer Bausteine verlangt. Die neuen Mehrparteientauschprotokolle (welche aus Komplexitätsgründen nur zwischen einer begrenzten Anzahl von Parteien ausgeführt werden können) werden als Schlüsselkomponenten für die Modellierung eines Tauschsystems verwendet, das beliebig vielen Parteien (welche im Laufe der Zeit dem System beitreten) einen Privatsphäre wahrenden Tausch ihrer Waren ermöglicht. Die Implementierung und die Simulation des neuen Modells sowie der Vergleich mit den wichtigsten Modellen konventioneller Tauschsysteme offenbaren die Praxistauglichkeit des modellierten Privatsphäre wahrenden Tauschsystems.

Contents

List of Figures

List of Tables

List of Gates

List of Protocols

CHAPTER 1

Introduction

The rise of *e-commerce*—the process of exchanging goods or services via the Internet—turned out to be one of the unexpected impacts of the explosive emergence of the Internet [40]. Widely used e-commerce applications include online shopping, e-marketplaces, and e-banking. While providing a lot of convenience, these applications generally require the disclosing of sensitive personal data like credit card information, address details, or preferences. For an individual user it is typically not transparent what the personal data are used for and if these data are handled as claimed [110]. A majority of users may be willing to share sensitive personal data on the Internet to some extent when there is a balance between the involved benefits and drawbacks. However, this is certainly not the case if those data (e.g., one's room for negotiation) have the potential to adversely affect e-commerce transactions.

This problem can, for example, arise in the context of *e-bartering*. Bartering is defined as "the direct exchange of goods or services—without an intervening medium of exchange or money—either according to established rates of exchanges or by bargaining" [44] and has been practiced since the early days of humanity. Nowadays, traded goods and services include books, rental cars, apartments, office equipment, production surpluses, or idle times of employees. The attractiveness of bartering is a consequence of its simplicity and the absence of disadvantages resulting from the use of currencies like varying valuation of goods and services due to subjectivity, inflation, foreign exchange problems, liquidity problems of banks, as well as the concentration of economic power. Furthermore, compared to other e-commerce applications that involve money, bartering transactions allow for a richer structure of exchanges [82]: A trade takes place if the involved parties are satisfied with the exchange of their offered and desired commodities (i.e., goods or services). If the commodities first have to be converted into money, the corresponding prices have to be individually determined. Consequently, a party desir-

ing a commodity which is more expensive than its offer cannot barter, although a trade could have taken place if the commodities were traded directly [82].

The traditional form of bartering suffered from the inevitable need for coordination, i.e., a group of parties with coinciding offers and demands must be in the same place at the same time such that these parties can exchange their commodities in a cyclic fashion [89]. Today, a large fraction of bartering transactions is carried out via e-marketplaces represented by online platforms that facilitate the cumbersome search of trade cycles by providing various functionalities for finding suitable trade partners. However, a typical requirement of these platforms is that a user has to disclose sensitive personal data about her offered and desired commodities and the corresponding quantities to the platform operator and oftentimes to all other users. This does not only take away the freedom of a user to choose the level of transparency w.r.t. her bartering transactions but also can result in unfairness, manipulation, and corruption. For instance, by revealing the quantities of her offered and desired commodities to other users, a user may unintentionally limit her room for negotiation. Furthermore, a platform operator may control (to some extent) which parties eventually trade their commodities and thus can prefer certain users or grant advantages to himself when participating as a user. Also a complete or partial view of offers and demands in a market can be undesirable for some specific bartering contexts where reactive market dynamics should not be possible. For example, in the kidney donor exchange market (e.g., see [1]) tailoring a patient's medical characteristics to an offered kidney in order to increase the probability for being matched should not be possible.

In this thesis, we address these privacy-related issues that today's bartering platforms are facing by introducing privacy-preserving bartering protocols that allow for the distributed identification of trade partners without requiring any party to disclose what and how much it is willing to barter. The newly proposed protocols do not make use of any (trusted) third party (e.g., a platform operator), hence, it is not possible for any party to collect quotes, create user profiles, or record accomplished bartering transactions that—when leaked—deliberately or accidentally harm the privacy of the users.

More precisely, we consider a bartering setting in which each party provides a private quote which specifies its offered and desired commodity and the corresponding quantities. Our privacy-preserving protocols allow parties to jointly determine an actual trade (i.e., a trade to be executed) that arises from an actual trade partner constellation (indicating who trades with whom) and the commodities and quantities to be traded. During a protocol execution, the quotes of all parties remain private. The only new knowledge a party gains from participating in a protocol execution is its *local view* of the determined actual trade. The local view comprises a party's direct trade partners as well

as the commodities and the corresponding quantities to be sent and received. The determining of an actual trade includes a mechanism for the privacy-preserving negotiation of the quantities of the commodities to be traded. This mechanism motivates the parties to specify their true negotiation ranges.

In case that an actual trade is to be determined between more than two parties and there are multiple possibilities to select the actual trade partner constellation, we support the incorporation of any selection strategy, provided that the strategy is a function of the trade partner constellations and not of the private quotes. In this thesis, we present two concrete selection strategies where the first one allows the maximization of the number of parties that can trade and the second one additionally minimizes the lengths of trade cycles as a secondary priority criterion.

We demonstrate that our privacy-preserving bartering protocols (which are shown to be practical for a limited number of parties) can be used as a key component in a practical bartering system that allows an arbitrary number of parties (arriving at the system over time) to barter their commodities in a privacy-preserving fashion.

Our approach for designing privacy-preserving bartering protocols is based on *secure multi-party computation* (SMPC), a theoretically well-founded subfield of cryptography that allows multiple parties to jointly compute a functionality in a distributed fashion over their respective inputs while keeping their inputs private. More precisely, from the participation in the execution of an SMPC protocol, a party only learns its prescribed output and what can be deduced from it in combination with its private input. Two important characteristics that determine the SMPC setting are the *party case* and the *adversary model*. For the party case, one distinguishes between the *two-party case* and the *multi-party case*. The two most prominent adversary models are the *semi-honest adversary model* (considering passive adversaries) and the *malicious adversary model* (considering active adversaries). In this thesis, we present privacy-preserving bartering protocols for each of the resulting four SMPC settings.

1.1 Contributions

In the following, we detail the main contributions of this thesis.

Privacy-Preserving Bartering Protocols: Building on a privacy-preserving two-party bartering protocol secure in the semi-honest model [48], we present the first privacy-preserving bartering protocol for the two-party case that provides security in the malicious model. The focus of this thesis lies, however, on the much more complicated multi-party case which requires fundamentally different design approaches. This is due

to the fact that bartering protocols designed for the two-party case cannot be generalized to determine trade cycles between more than two parties. The corresponding difficulties and the relevance of privacy-preserving bartering protocols for the multi-party case have already been recognized in the literature (see, e.g., [51] and [54]). In this thesis, we present the first privacy-preserving bartering protocols for the multi-party case that provide security in the semi-honest and the malicious model, respectively. All novel privacy-preserving bartering protocols are formally proven to be secure (i.e., correct and privacy-preserving) in the respective adversary model. In part, this work has been published in [120–125].

Privacy-Preserving Building Blocks: We introduce several novel multi-party comparison and selection functionalities that operate on encrypted values. For each of these functionalities, we devise efficient SMPC protocols. While used as secure building blocks in the context of designing our privacy-preserving bartering protocols, these protocols are of general interest beyond the context of bartering. Among the most important contributions are multi-party protocols (secure in the semi-honest, respectively, in the malicious adversary model) that implement the primitive of conditional random selection [119], a multi-party protocol (secure in the malicious model) that allows the sampling from a private (i.e., unknown) interval (specified by its encrypted bounds) according to an arbitrary discrete distribution, and a multi-party protocol (secure in the malicious model) that allows the oblivious shrinking of a private interval in case that the interval width is beyond a given publicly known threshold. We provide formal security proofs for all novel SMPC protocols. In part, this work has been published in [48,118–125].

Privacy-Preserving Bartering System: We introduce a novel bartering model for a privacy-preserving bartering system (independent of a specific adversary model) and provide a comprehensive evaluation of its practicality. To this end, we implemented our privacy-preserving multi-party bartering protocol (secure in the semi-honest model) in a newly devised SMPC library in order to determine an upper bound for the number of parties the protocol can handle efficiently and design a generic framework for the simulation of bartering models. This framework is implemented on top of an existing framework for discrete event simulation and is used to implement our newly proposed privacy-preserving bartering model as well as two prominent conventional (i.e., non-privacy-preserving) bartering models known from literature. We conduct a comprehensive series of simulations in order to compare the three considered bartering models w.r.t. the number of parties (resp., the average waiting time of parties) that are identified as trade partners. The evaluation of the simulation results shows that bartering in

the modeled privacy-preserving bartering system is practical and achieves comparable good results w.r.t. the considered measures as it is the case for modeled conventional bartering systems. The results of this part of the thesis are currently under submission.

1.2 Collaborations

The work which is presented in this thesis was conducted within a joint research project between Professor Meyer's group at RWTH Aachen University and Professor Wetzel's group at Stevens Institute of Technology. Each group worked on different aspects of privacy-preserving bartering. While the group at RWTH Aachen University focused on designing multi-party bartering protocols for the semi-honest and the malicious model as well as on bringing these protocols to practice, the group at Stevens Institute of Technology designed two-party bartering protocols (secure in the semi-honest model) with a focus on supporting multiple quotes for each party.

This joint research project was partially supported through DFG grant ME 3704/4-1 and NSF grant 1646999.

1.3 Outline

The remainder of this thesis is organized as follows: In Chapter 2, we present the general context of this thesis including a brief overview of privacy-preserving techniques for the processing of private data and of essential functionalities provided by conventional bartering platforms. Furthermore, we sketch our envisioned privacy-preserving bartering system and provide an overview of related work on privacy-preserving bartering. Chapters 3 and 4 establish the mathematical and cryptographical background for this thesis. Subsequently, we present existing as well as novel SMPC protocols which are used as building blocks for our novel privacy-preserving bartering protocols (Chapter 5). This chapter also captures the related work for the presented building blocks. In Chapters 6 and 7, we present the privacy-preserving bartering protocols for the two-party and the multi-party case, respectively. An implementation and a performance evaluation of our privacy-preserving multi-party bartering protocol providing security in the semi-honest model is described in Chapter 8. This protocol serves as a basis for modeling a first privacy-preserving bartering system which is presented and compared to conventional bartering models in Chapter 9. We conclude this thesis with a summary of the results and a discussion of future work (Chapter 10).

CHAPTER 2

General Context

In this chapter, we set the general context of this thesis. First, we provide a brief overview of some preselected state-of-the-art *privacy-enhancing technologies*. While this is a broad term and there is no commonly accepted definition [109], in this thesis, we focus on privacy-enhancing technologies for data processing rather than for identity management or anonymous communication. Here, the goal is to identify techniques that are suitable for our purpose to design a privacy-preserving bartering system rather than to give a comprehensive and complete overview. Second, we review essential functionalities provided by conventional e-bartering platforms (not making any use of privacy-enhancing technologies). Subsequently, we sketch the bartering setting (including the privacy requirements) for our envisioned privacy-preserving bartering system. We discuss which privacy-enhancing technologies are suitable and which functionalities can be implemented in such a system. Finally, we present the related work on privacy-preserving bartering.

Outline: In Sections 2.1 and 2.2, we present relevant privacy-enhancing technologies and review functionalities that are provided by conventional e-bartering platforms, respectively. Subsequently, we sketch the bartering setting of our envisioned privacy-preserving bartering system and review related work on privacy-preserving bartering (Section 2.3). This chapter closes with a summary.

2.1 Privacy-Enhancing Technologies

Privacy-enhancing technologies (PETs) for data processing can broadly be divided into statistical methods and cryptographic methods. The two approaches differ significantly w.r.t. the application scenarios they are suitable for. The focal point of statistical methods

is to protect identifying information (of a person or an object) in released data of a private database which, e.g., results from a response to a (possibly interactive) data query. Anonymization techniques used in this context are typically adding random noise to the database before responding to a query and removing critical identifiers like birthdays or addresses. On one hand, statistical methods are generally amenable for an efficient implementation (compared to cryptographic methods) but, on the other hand, they entail a loss in accuracy and/or meaningfulness w.r.t. the released data. Furthermore, most statistical methods only provide security against specific (known) adversarial strategies. It is important to note that how to privately compute an answer to a data query is not a matter of concern for statistical methods. Instead, the general assumption is that the database is held by some sort of trusted/legitimized party which receives data queries and computes responses.

In contrast, the focus of the cryptographic methods is on *how* to privately compute the response to a data query or more generally on *how* to privately compute a specific functionality on private data. While cryptographic methods come with a significant computational overhead (as they operate on encrypted data), they generally operate on the original unchanged data and thus provide accurate results. Using cryptographic techniques, it is also possible to remove the necessity of involving a trusted third party and to achieve provable security, i.e., allowing the design of protocols which provide security independent of an adversary's strategy.

In the following, we first outline statistical methods comprising k-anonymity and differential privacy (Section 2.1.1). In Section 2.1.2, we briefly review some of the most promising cryptographic methods for designing privacy-preserving bartering protocols like attribute-based encryption, homomorphic encryption, and secure multi-party computation.

2.1.1 Statistical Techniques

k-Anonymity. In [112], Sweeney introduced the *k-anonymity* privacy protection model which strives to prevent re-identification attacks of individuals contained in a data set (consisting of data records each prescribed to a unique individual) by the means of linking/matching released data to other already available data. The k-anonymity model is based on so called *quasi-identifiers* which are defined as attributes (e.g., name, address, and gender) of a data set that can be used for linking with externally released data. A data set providing k-anonymity assures that an individual cannot be distinguished from at least $k-1$ other individuals appearing in the same data set. More precisely, in the released data set, a data record which is projected to the quasi-identifiers appears at least k

times. This concept has several shortcomings: (1) only protection against known attacks is provided; (2) k-anonymity can be only guaranteed in case that the released data set is linked against foreseeable controlled data sources; (3) depending on the application field, it might be hard to determine the quasi-identifiers.

Differential Privacy. The overall privacy goal in the context of statistical databases, postulated by Dalenius [33], is to achieve semantic security, i.e., having access to a statistical database only reveals information that could have been learned without access. In [41], it has been formally proven that this goal is unreachable since it is not possible to control the auxiliary information available to an adversary. Therefore, in [41], a relaxed approach, referred to as *differential privacy*, has been formulated. Differential privacy captures that the loss of an individual's privacy should not substantially increase as a result of being represented by a data record in a statistical database. In other words, differential privacy assures that the answer to a database query will be independent of whether or not a specific individual appears in the database. This can be achieved by utilizing a mechanism proposed in [42] which adds appropriate random noise to the answer of a database query. The noise is chosen according to a function of the maximum influence an individual might have w.r.t. a query answer. According to [41], differential privacy can be maintained for successive and adaptive database queries.

Joint Differential Privacy. The concept of *joint differential privacy*, introduced in [75], is a variant of differential privacy suitable for an adapted application scenario where a number of parties individually provide private input to a kind of trusted third party which—after computing a functionality on that input—provides an individual private output to each party. The goal in this scenario is to protect the privacy of a party's input from the other parties w.r.t. what they can deduced from their own input and output. Informally speaking, joint differential privacy guarantees that the privacy of a specific party's input is protected w.r.t. the notion of differential privacy even if up to all of the other parties are colluding [70].

Marginal Differential Privacy. A relaxation of joint differential privacy is referred to as *marginal differential privacy* [72]. This concept fits to the same adapted application scenario as described for joint differential privacy. However, marginal differential privacy requires that no parties are colluding in order to guarantee privacy of a party's input.

2.1.2 Cryptographic Techniques

Identity-Based and Attribute-Based Encryption. In [108], Shamir asked for an *identiy-based encryption* (IBE) public key encryption scheme. Instead of randomly generating a public/private key pair followed by publishing a public key certificate, in an IBE scheme the public key of a user is set to a unique identifier (w.r.t. the application context) composed of attributes like email address, network address, office number, and/or phone number. The considered attributes should be undeniable as well as available to the public. The corresponding private key is generated by a trusted *private key generator* (PKG) using a master secret which is only known by the PKG. For the request of a private key, a user has to authenticate against the PKG, i.e., he has to prove that the identity belongs to him. The initial motivation behind IBE was to set up a secure email system which makes the cryptographic aspects transparent and which can be used by laymen [108]. In such a system, a user can send an encrypted email to a second user, where, e.g., the concatenation of the name and the email address of that second user constitutes his public key. Since the master secret of the PKG has to be used to issue a private key, the system guarantees that only the designated receiver and the PKG can decrypt the email. For the sender, this approach eliminates the burden to request and check certificates for intended recipients and allows the sending of an encrypted email to a user who did not even request his private key yet.

Attribute based encryption (ABE) can be considered as a generalization of IBE. According to [63], the concept of ABE is due to Sahai et al. In [103] they proposed a *fuzzy identity based encryption* (FIBE) scheme which allows a user—provided with a private key for identity A—to decrypt a ciphertext encrypted for identity A', provided that A and A' are close to each other w.r.t. some policy. In contrast to IBE, where identities A and A' have to be identical to perform this decryption operation, in FIBE, the identities become error-tolerant which allows the implementation of an IBE scheme based on biometric data. As a further application of FIBE, the authors of [63] conceptualize ABE which, generally speaking, allows the labeling of a private key and a ciphertext with sets of attributes. Ciphertexts can only be decrypted by a private key if the associated attributes match according to some specific policy. In [103], the policy requires that the number of matching attributes is above a certain threshold. One noteworthy property of the scheme presented by Sahai et al. is that it is resistant against collusion attacks where multiple users try to combine their attributes by utilizing their private keys in order to decrypt a ciphertext which none of them can decrypt on its own.

Secure Multi-Party Computation. *Secure multi-party computation* (SMPC), first introduced by Yao [126,127], allows one to perform distributed computations in a secure manner without involving a trusted third party: Some parties, each holding private data, want to jointly compute a functionality on their aggregated data but without leaking any information on their private input except what is absolutely necessary in order to compute the target functionality. Providing their private inputs to an SMPC protocol implementing the target functionality allows the parties to securely compute their prescribed output. More precisely, the secure computation of a functionality makes sure that each party does not learn anything beyond its correct output and what can be deduced from it in combination with its input. This must even hold if the protocol execution is under attack by an adversary controlling a specific subset of parties which participate in the protocol execution and which follow the adversary's instructions.

Homomorphic Encryption. Informally, *homomorphic encryption* is a special type of encryption that allows the computing of functions on encrypted data (without the need of decrypting it first). One distinguishes between *partially homomorphic encryption* and *fully homomorphic encryption*. While partially homomorphic cryptosystems (e.g., [62,94]) allow the computation of a restricted set of functions only (depending on the underlying homomorphism), fully homomorphic cryptosystems (e.g., [26,56,116]) allow one to compute arbitrary functions over encrypted data. Homomorphic encryption can be used, besides other techniques, to perform secure multi-party computations.

Oblivious RAM. The concept of ORAM (Oblivious Random-Access Machine) has been introduced by Goldreich and Ostrovsky [61]. The original motivation was the protection of (expensive) software from illegitimate software duplication. First approaches include the combination of a physically shielded CPU and encrypted software: The CPU, connected to the considered system, is equipped with a decryption key allowing the decryption of the associated software. In order to run the encrypted software, the CPU has to fetch an encrypted instruction from memory followed by decrypting and executing it. Goldreich and Ostrovsky argue that this approach does not provide the intended security which is due to the fact that an adversary can, e.g., distinguish between different types of memory accesses or observe memory access patterns. These observations might reveal program structures which in turn might facilitate the reconstruction of the encrypted software. It is in this context that Goldreich and Ostrovsky propose the ORAM construction which ensures that an adversary observing an encrypted software running on a shielded CPU learns nothing beyond the running time and the memory consumption of that software. The concept of ORAM allows the performing of secure multi-party

computations by letting the parties jointly implement the CPU and by letting the cells of the memory represent the parties' private inputs. This approach can be beneficial in case that the functionality which is to be computed on the parties' private inputs is more suitable to be represented by a RAM than by a circuit [83].

2.2 Conventional E-Bartering

In the following, we give an overview of essential functionalities provided by current online platforms for conventional e-bartering aiming at increasing the possibilities and the convenience to find trade partners.

While bartering has been practiced since the early days of humanity, nowadays, bartering transactions are primarily processed via online bartering platforms. There are a variety of bartering platforms which offer a diverse range of functionalities to their users (e.g., BarterQuest, SwapRight, U-Exchange, readitswapit, BarterOnly, BizXchange [10,11,16, 100,111,115]). Some of these platforms are focused on a limited spectrum of commodities (i.e., goods and services) and on specific user groups like individuals or businesses (e.g., readitswapit, BizXchange) while other platforms are rather generic and allow one to barter almost everything (e.g., BarterQuest, BarterOnly). These platforms can be classified into two groups which we refer to as *bartering dashboards* and *bartering systems*. Bartering dashboards (e.g., U-Exchange) are bulletin boards allowing their users to publish classified ads listing their offered and desired commodities as well as their contact information. In contrast, bartering systems (e.g., BarterQuest) are characterized by the fact that they provide a variety of functionalities from supporting a user when specifying her offered and desired commodities up to completely automating the bartering process. These functionalities facilitate the finding of suitable trade partners and also enable the determining of trade cycles involving several parties which generally cannot be determined by bartering dashboards.

Irrespective of a specific bartering platform, a bartering process can be divided into three rudimentary steps: (1) search for potential trades, (2) filter the potential trades based on criteria like state of commodities or user reputation, and (3) determine an actual trade from the results of the second step which is designated to be executed. Depending on the bartering platform, the third step involves some form of bargaining where the details of a trade can be negotiated, e.g., the quantities of the commodities to be exchanged or how the exchange will be carried out. Furthermore, whether or not some of these steps can be automated depends on the respective platform.

In the following, we focus on bartering systems and detail their essential functionalities based on [18,19].

Functionality 1 (User Data Management)**.** A general approach to manage user data is to maintain user accounts. A user account allows the linking of different kind of data (e.g., contact information, history of bartering transactions) to one single user. Keeping a record of user specific data substantially improves the usability of a bartering system. Besides the fact that user specific data (e.g., contact information) have to be entered only once, a user account allows the reusing of search queries, to remember preference w.r.t. offered and desired commodities, as well as to maintain a history of all prior transactions. This in turn allows a bartering system to make suggestions on commodities offered or desired by other users and to establish a reputation system allowing to blame unreliable users as well as to increase the system's trustworthiness. Furthermore, the management of user data allows a bartering system to provide automation, user notification, and the presentation of pending trades.

Functionality 2 (Specification of Offered and Desired Commodities)**.** In order to actively use a bartering system, a user has to specify some offered commodities and/or some desired commodities. Note that offered commodities must not necessarily be linked to desired commodities (and vice versa). A user just specifying an offered commodity can wait for a another user suggesting a *barter offer*. Due to the variety of functionalities provided by bartering systems the specification of commodities requires users to complete a predefined form usually comprising a separate specification of offered and desired commodities, a specification of some commodity specific attributes, and the classification of commodities w.r.t. a given set of categories. The procedure of specifying a commodity can be assisted by the system, e.g., by providing access to a database of already specified commodities (possibly entered by other users) or by suggesting attributes and categories based on previously entered information. In the following, we refer to the set of offered (resp., desired) commodities of one party as *offer list* (resp., *demand list*). Some forms of automation (e.g., automated detection of trade cycles) provided by bartering systems require the specification of *barter listings* which detail the conditions (e.g., quantities of commodities) under which a user is willing to barter some of its offered commodities for some of its desired commodities.

Functionality 3 (Manual Searching and Filtering)**.** One of the most fundamental functionalities provided by bartering systems are searching and filtering offered and demanded commodities as well as barter listings. Usually, a user is presented a single search field or search mask where the latter consists of several search fields a user has to complete. The supported search criteria are system-depended and are aligned with how commodities have to be specified. For example, a user may be able to search for a specific user, for users offering (resp., demanding) a specific commodity, or for com-

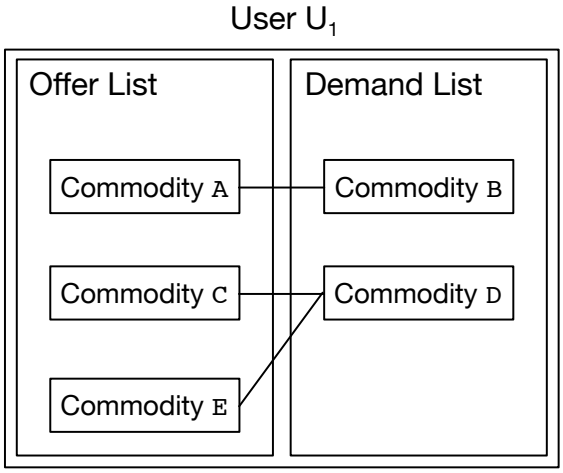


Figure 2.1: Offer and demand list of user U_1.

modities of a certain category. An important feature of search masks is the possibility to filter the search results, e.g., by categories or attributes of a commodity or by restricting the search results to users resident in a specific location. The possibility of browsing a category can be of interest if a user is interested in any commodity of a category rather than in a specific one. Furthermore, based on the specification of commodities it may be difficult to decide whether or not two items match. By keeping the search criteria more general, a user can be more successful even if the follow-up work on the search results can be much more time-consuming for the user. It is important to note that in general the manual searching and filtering functionalities are intended to facilitate the arrangement of trades by the users themselves rather than to automate this process.

Functionality 4 (Automatic Detection of Trade Partner Pairs)**.** The automatic detection of trade partner pairs can be considered as an advanced functionality which is provided by only a few bartering systems. One example is BarterQuest which is based, i.a., on [19] which provides a detailed description of automatic trade partner detection. One approach to implement this functionality is to allow users to create links between commodities in their offer and demand lists. An example is given in Figure 2.1 where an edge between commodity `A` (in the offer list) and commodity `B` (in the demand list) indicates that user U_1 is willing to give away commodity `A` in exchange for commodity `B`.[1] Based on the linked lists consisting of one or more barter listings (cf. Figure 2.1 which illustrates three barter listings) the bartering system has to identify pairs of users

[1]Similarly, a link between commodities `C`, `E` (in the offer list) and commodity `D` (in the demand list), respectively, indicates that user U_1 is willing to give away commodity `C` or `E` in exchange for commodity `D`.

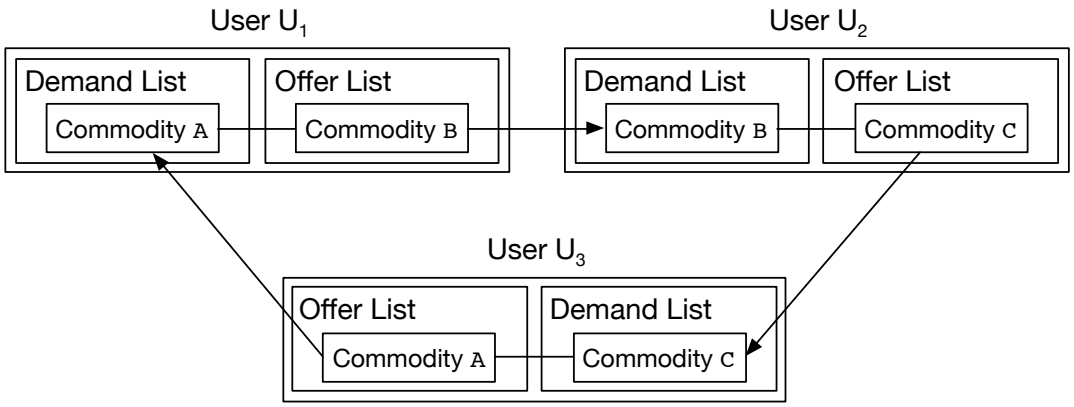


Figure 2.2: Trade cycle of length three.

with matching barter listings. Standard approaches for determining whether or not two commodities are equal or similar comprise natural language processing algorithms, similarity matching algorithms, and the usage of fine-grained categories. After a potential trade has been identified by the system, the involved users are informed and further interaction (possibly including a negotiation of the final exchange rates) is required in order to conclude the trade.

Functionality 5 (Automatic Detection of Trade Cycles)**.** While it is often not possible or at least time-consuming to find a pair of users such that some of their offered and desired commodities mutually match, determining an alternative trade involving more than two users may be possible. In these sort of trades, users trade in a cyclic fashion (as long as nobody gives anything away for free). Consider the example given in Figure 2.2. While there is no potential trade between any pair of users there is a trade cycle of length three where directed edges in Figure 2.2 indicate the exchange direction of the offered commodities. In practice, it is often preferable to limit the considered size of trade cycles. This is not only because of performance reasons—the underlying decision problem is NP-complete—but also to reduce the impact in case a user involved in the trade cycle intentionally or unintentionally drops out. The latter aspects are discussed in more detail in Chapter 7.

Functionality 6 (Negotiation)**.** After determining one or multiple potential trades, a subsequent negotiation phase might be necessary to come up with an actual trade. For example, this phase may include the negotiation of the quantities of the commodities to be traded or the shipping conditions. Usually, bartering systems do not go any further than providing the means to privately communicate, e.g., by means of a private

chat room but leave the actual negotiation to their users. However, depending on the possibilities to specify a commodity, negotiable facets can be part of the commodity description which makes the negotiation phase obsolete. The latter is of particular interest for systems supporting trade cycle detection in order to reduce the risk that a trade cycle is identified but discarded due to disagreements w.r.t. the exchange rates.

2.3 Privacy-Preserving E-Bartering

In Section 2.3.1, we briefly sketch the bartering setting (including the privacy requirements) for our envisioned privacy-preserving bartering system. Based on Section 2.1 and Section 2.2, we now discuss which of the reviewed PETs are best suited as well as which functionalities of conventional bartering systems can be realized for the implementation of the privacy-preserving bartering system. Subsequently, we review related work on privacy-preserving bartering as well as other related topics like privacy-preserving matching and fair exchange (see Section 2.3.2).

2.3.1 Envisioned Privacy-Preserving Bartering System

Our bartering setting is as follows: Each user can specify a quote comprising an offered and a desired commodity as well as the corresponding quantity ranges at which the user is willing to trade these commodities. As a kind of simplification, we stipulate that a user's demand is satisfied by at most one other user in an actual trade. The envisioned privacy requirements are aligned to our goal to provide a maximum level of privacy, i.e., to reveal only those information on the quotes and the determined actual trade that is absolutely necessary to maintain the functionality of bartering divisible commodities. In particular, we postulate:

Privacy Requirement 1. A user only learns what she receives (from which party) and what she has to send (to which party) as well as what can be deduced from this information.

Privacy Requirement 2. A user can determine her trade partners without involving a trusted third party.

Additionally, we require unbiasedness:

Privacy Requirement 3. All users can provably be treated equally, i.e., no one is preferred over another by the bartering system.

Privacy Requirements 1 and 2 entail further, more fine-grained requirements:

Privacy Requirement 1.A. A user only learns the offered (resp., desired) commodity of a direct trade partner as well as the quantity of that commodity to receive (resp., send) and the fact that this quantity lies in the trade partner's negotiation range.

Privacy Requirement 1.B. A user must not learn anything about the quote and the transactions of any user that is not her direct trade partner.

Privacy Requirement 1.C. A user which is involved in a trade cycle of length greater than two must not be able to observe the transactions of its direct trade partners except those in which she is involved.

Privacy Requirement 1.D. In case of considering divisible commodities, the negotiation of the final exchange rates has to be a one-shot process. Otherwise, by making multiple offers and counteroffers, the involved users learn more information than permitted by Privacy Requirement 1.

Privacy Requirement 2.A. Since there is no trusted third party, the task to find trade partners is essentially left to the users (as participants in a distributed multi-party protocol) themselves.

Since we postulate the absence of a trusted third party, statistical PETs (taken in isolation) are not suitable for our purposes. Furthermore, our privacy requirements necessitate a secure computation for determining suitable trade partners for a user.

Considering the cryptographic PETs as reviewed in Section 2.1, there are multiple techniques that seem to be suitable to realize our envisioned privacy-preserving bartering system. The approach that we follow in this thesis is to use homomorphic encryption: Besides providing confidentiality of the parties' quotes, a homomorphic cryptosystem allows the computation of functionalities on ciphertexts. This feature can be used to compute complex trade conditions on a set of quotes allowing for a distributed computation of whether or not there exist bartering opportunities between the parties providing their encrypted quotes. More precisely, a barter market consisting of multiple parties can be cleared at once. Instead of using fully homomorphic encryption which to date lacks practicality (see, e.g., [43,57,67]), we use partially homomorphic encryption as an underlying technique for SMPC in order to enable the computation of functionalities on encrypted data. Other alternative approaches are briefly discussed in Section 2.4.

Now that we identified PETs suitable for our envisioned privacy-preserving bartering system, we can assess which functionalities provided by conventional bartering platforms (i.e., Functionalities 1-6 introduced in Section 2.2) can be realized without violating our privacy requirements. Since there is no trusted third party which can either observe or coordinate bartering transactions, essential benefits (e.g., establishing a reputation

system or suggesting attributes or categories for a commodity based on commodities specified by other users) of managing user accounts (Functionality 1) and maintaining a database of commodities specified by other users (Functionality 2) are lost. Furthermore, the postulated privacy requirements thwart the possibility to browse and filter the offered and demanded commodities as well as barter listings specified by other users (Functionality 3).

In contrast, the detection of trade partner pairs and trade cycles (Functionality 4 and Functionality 5) can be expressed as a functionality computed on the encrypted private quotes of the users, although it remains a challenge to handle multiple potential trades. The automation in this context which may be a functionality provided by a conventional bartering system becomes a distributed task between the users in a privacy-preserving bartering system. It is also possible to negotiate the actual quantities at which the actual commodities are to be exchanged (Functionality 6) in a privacy-preserving fashion—provided that the negotiation process is intended to be an integral part of the bartering system and shall not be left to the users—by defining an appropriate functionality on the users encrypted quotes honoring their preferences.

2.3.2 Related Work on Privacy-Preserving E-Bartering

Due to the fact that bartering transactions allow for a richer structure of exchanges compared to e-commerce applications that involve money (see Chapter 1), privacy-preserving protocols for buyer/seller transactions (e.g., [3,4]) or auctions (e.g., [22,92]) cannot directly be applied in the context of privacy-preserving bartering and are not considered in the following.

2.3.2.1 Privacy-Preserving Two-Party E-Bartering

To the best of our knowledge, for the two-party case there exists only one privacy-preserving bartering approach that has been proposed by Frikken et al. [54]. Based on a public set of indivisible commodities and a private utility function each party has to specify individually, the protocol allows the privacy-preserving computation of a so-called *win-win trade*. The utility function of a party maps each commodity in the union of both commodity sets to a monetary value. A win-win trade describes an exchange of a subset of commodities from which both parties profit. More precisely, the utility of the commodities a party receives is strictly greater than the utility of the commodities that a party gives away. The main object of [54] is to enable two parties to barter without revealing their individual utility functions, because, if it is published by one party, a second party can adapt its own utility function in order to get a better deal. The approach of

Frikken et al. is based on secure multi-party computation and provides security against passive adversaries. Their work differs from our approach in that we additionally allow the parties to keep their offered and desired commodities private. Furthermore, we do not restrict our solutions to indivisible commodities and we do not require that each party has to specify a valuation for each commodity. Instead, we allow the parties to privately specify quantity ranges of their commodities that indicate how much of a commodity they are at most willing to give away if they receive at least some lower bound of another commodity. On the basis of their quantity ranges the actual quantities of the commodities to be traded are automatically negotiated. While the solution of [54] is restricted to the two-party case and provides security against passive adversaries only, we additionally cover the multi-party case and provide security against active adversaries. Compared to our approach, the approach from [54] does not provide Functionalities 5, 6 and does not satisfy Privacy Requirements 1, 1.A, 1.C, 1.D which can be attributed to the fact that the offered and desired commodities are not kept private, only indivisible commodities are considered, and only bartering between two parties is supported.

2.3.2.2 Privacy-Preserving Multi-Party E-Bartering

Similarly to the two-party case, to the best of our knowledge, there is only one approach to privacy-preserving multi-party bartering that has been proposed in [72]: Kannan et al. introduce a protocol where each party holds exactly one indivisible commodity from a publicly known finite set of commodities as well as a totally ordered preference list over all commodities in the set. Their goal is then to determine an actual trade between multiple parties such that the computed commodity allocation is pareto optimal while the input of each party (commodity and preference list) is kept private. Specifically, the protocol protects the parties' input under the notion of marginal differential privacy (see Section 2.1.1) which corresponds to the assumption that there are no colluding parties participating in the protocol which try to subvert the privacy of another party. The substantial difference between the approach from [72] and our approach is that the former one focalizes on the privacy of the parties' input *after* the functionality is computed while the major goal of our approach is to provide privacy *during* and *after* the computation of the actual trade. This difference can be attributed to the different notions of privacy underlying the approaches (cf. Section 2.1.1 and Section 2.1.2). Further differences to our work are that the protocol from [72] requires a trusted third party in order to determine an actual trade and that they use a weaker privacy notion that assumes non-colluding parties. Furthermore, only indivisible commodities are supported. In our approach, an actual trade is computed without the help of a trusted third party and we allow that all

but one party may be controlled by an adversary. In addition, our approach supports divisible commodities. Compared to the direction we follow in this thesis, the work from [72] does not satisfy Privacy Requirements 2, 3, 1.D, 2.A which can be attributed to the fact that a trusted third party is required and only indivisible commodities are supported.

2.3.2.3 Privacy-Preserving Matching

A barter market consisting of a set of parties along with their (offered and desired) commodities can be encoded as a bipartite weighted graph (see, e.g., [1]): Each offered and desired commodity is associated with a node; the set of the offered and the desired commodities form the two disjoint sets of nodes underlying the bipartite graph. An edge of weight $x > 0$ is added between the offered and desired commodity of the same party and an edge of weight y with $y > x$ connects a party's desired commodity with the offered commodity of another party in case both commodities coincide. A protocol for finding a *maximum-weight perfect matching* (see Definition 3.6, Section 3) can be used to compute a maximum cycle cover on that graph which constitutes a trade between the given set of parties such that (1) each party giving away its offered commodity is to receive its desired commodity and (2) the number of parties involved in the trade is maximized. Although a maximum-weight perfect matching can be computed in time polynomial in the number of parties, this approach has some substantial practical disadvantages. On the one hand, it is not possible to specify trade cycle lengths constraints (i.e., to restrict the maximum allowed cycle length) which is a serious limitation since it is a requirement that arises naturally from practice (see Section 7.4 and, e.g., [1,85]). On the other hand, this approach does not allow prioritizing potential trades such that, e.g., a potential trade consisting of two trade cycles of length two cannot be preferred over a trade cycle of length four although in the latter case the impact of a party dropping out is higher. The functionalities and privacy requirements that can be provided are the same as for our approach, except that the maximum-weight perfect matching approach is restricted w.r.t. Functionality 5 in that it does not support to restrict the trade cycle length.

It is important to note that protocols for finding a *maximum matching* (see Section 7.6) on a given bipartite graph cannot be used since they do not allow the consideration of the parties' preferences. Consequently, a trade where each party keeps its offered commodity can be selected as an actual trade even if there exist other trades where parties do exchange their commodities.

To the best of our knowledge, there exists only one privacy-preserving protocol for

computing a maximum-weight perfect matching on bipartite graphs which is targeted towards multimedia analysis and retrieval [25]. The protocol from [25] is restricted to the two-party case and may leak crucial information when used for bartering as the protocol flow depends on the protocol input to some extent. However, in the context of multimedia analysis and retrieval this kind of leakage is not a serious issue.

In Section 7.6, we sketch a novel protocol for privacy-preserving maximum-weight perfect matching for multiple parties where the protocol flow (as opposed to [25]) does not depend on the private input. Furthermore, it is elucidated in what respect this protocol is suitable for our envisioned bartering setting.

2.3.2.4 The Fair Exchange Problem

An orthogonal line of work to privacy-preserving bartering is *fair exchange*. In fact, the final state of a bartering protocol can be the starting point for fair exchange [51]. For a set of parties (each having specified an offered and a desired commodity) a bartering protocol computes an actual trade that determines at which rate a party has to send (resp., to receive) its offered (resp., desired) commodity to (resp., from) which other party. Now, the goal of a fair exchange protocol is to assure that a party only *physically* receives its desired commodity iff all other parties involved in the same trade cycle *physically* receive their desired commodities too. Basically, fair exchange is concerned with the transfer of digital commodities in the context of digital contract signing, digital payment, certified emails, non-repudiation, and exchanging secrets [102]. Most fair exchange protocols are based on the assumption that the considered digital commodities are idempotent, i.e., possessing one commodity is equivalent to possessing a multiple of that commodity [102]. Furthermore, most protocols are not concerned with safeguarding the value, quality, or correctness of digital documents. Instead, it is often assumed that there is the possibility to verify these properties on the received data and to handle appropriately in case that the verification fails.

According to [102], there are three types of fair exchange protocol constructions where the classification is based on the degree of the involvement of trusted third parties. The first group of protocols—essentially feasibility results—do not make use of a trusted third party (e.g., [13,98]). These protocols approximate a fair exchange by stipulating a gradual release of information such that a party deciding to cheat only gets a minimal advantage over the honest parties. The second group is characterized by the fact that the protocols require an online trusted third party which has to be involved in each exchange (e.g., [51,52]). Protocols belonging to the third group, also referred to as *optimistic* protocols, only make use of a (offline) trusted third party in case of detecting a protocol

failure or malicious behavior (e.g., [87,95]). Further classifications of fair exchange protocols can be made based on, e.g., the number of supported parties, different levels of fairness, and different trust assumptions on an invoked third party.

2.4 Summary

In this chapter, we set the general context of this thesis. We provided an overview of specific privacy-enhancing technologies and reviewed essential functionalities provided by conventional bartering systems. Furthermore, we sketched the bartering setting for our envisioned privacy-preserving bartering system and discussed which of the considered privacy-enhancing technologies and the functionalities provided by conventional bartering systems are suitable for, respectively, can be implemented with regard to our purposes.

In order to design our envisioned privacy-preserving bartering system, the approach that we follow in this thesis is to use SMPC from homomophic encryption techniques. Nevertheless, there are other interesting approaches that could be pursued. For example, one could consider the use of SMPC from ORAM techniques as well as to make use of attribute based encryption. With regard to the latter approach, one can think of a privacy-preserving bartering protocol that works as follows: User U_1 publishes an encryption of his contact information on a bartering dashboard where the corresponding ciphertext is labeled in such a way that only a user U_2 whose quote is compatible with the quote of user U_1 can decrypt it. However, this approach seems to be less promising than the approach that we follow in this thesis since it is unclear how to determine trade cycles of length greater than two.

CHAPTER 3

Preliminaries

In this chapter, we introduce the mathematical and cryptographical foundations and principles which are used throughout this thesis. An overview of the most important notations is given at the end of this thesis.

Outline: In Section 3.1, we present basic notations, definitions, and concepts. Section 3.2 deals with partially homomorphic encryption with a focus on the additively homomorphic Paillier encryption scheme and one of its threshold variants. Eventually, we provide a brief overview of zero knowledge proofs of knowledge and Sigma-protocols in Section 3.3.

3.1 Basic Definitions and Notations

3.1.1 Basic Notations

We define $\mathbb{N}^0 := \mathbb{N} \cup \{0\}$, $\mathbb{R}^{>0}$ as the set of positive real numbers, and $\mathbb{R}^{\geq 0} := \mathbb{R}^{>0} \cup \{0\}$. $\mathbb{N}_u := \{1, \ldots, u\}$ refers to the set of natural numbers less than or equal to $u \in \mathbb{N}$.

For a given integer $x \in \mathbb{N}$ and $m \in \mathbb{N}$ such that $0 \leq x < 2^m$ we write $x_{bin} := (x_0, x_1, \ldots, x_{m-1})$ with $x_0, \ldots, x_{m-1} \in \{0, 1\}$ to refer to the bit representation of x. With $|x|$ we denote the bit length of $x \in \mathbb{N}^0$.

With the notation $V[i]$, we refer to the i-th entry v_i of a vector $V = (v_1, \ldots, v_n)$ ($i \in \mathbb{N}_n$).

For an arbitrary element a which is listed n times, we define

$$a, \overset{n}{\ldots}, a := \underbrace{a, \ldots, a}_{n \text{ times}}.$$

For a given logical statement B, the *Iverson Bracket* $[\cdot]$ is defined as

$$[B] := \begin{cases} 1 & \text{if } B \text{ is true} \\ 0 & \text{otherwise} \end{cases}.$$

3.1.2 Sets and Intervals

For a set or interval A, we write $a \leftarrow_{\$} A$ to denote that a is drawn uniformly at random from A. Similarly, we write $a \leftarrow_{\mathcal{D}} A$ to denote that a is drawn from A according to probability distribution $\mathcal{D}$.

Definition 3.1 (Non-Negative Closed Integer Interval)**.** *Let $(\mathbb{N}^0, \leq)$ be a totally ordered set and let $\lambda, \mu \in \mathbb{N}^0$. A* non-negative closed integer interval *$[\lambda, \mu]$ is defined as $\{x \in \mathbb{N}^0 \mid \lambda \leq x \leq \mu\}$.*

Whenever we speak of an interval, we refer to a non-negative closed integer interval.

Definition 3.2 (Interval Width)**.** *The* interval width *of an interval $[\lambda, \mu]$ is defined as $|[\lambda, \mu]| := \mu - \lambda$.*

3.1.3 Number Theory

Definition 3.3 (Strong Prime Number)**.** *A* strong prime number *is a prime number p of the form $p = 2p' + 1$ such that p' is a prime number too.*

Definition 3.4 (Lagrange Interpolation, based on [32])**.** *Let $\mathbb{F}$ be a finite field. For a given set of n data points $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{F} \times \mathbb{F}$ there exists a unique polynomial L of degree $n - 1$ referred to as* Lagrange interpolation polynomial *with*

$$L(x_j) = y_j, \ \forall j \in \mathbb{N}_n$$

which is given by

$$L(x) := \sum_{j=1}^{n} y_j l_{j,n}(x)$$

where the Lagrange basis polynomials *$l_{j,n}(x)$ are given by*

$$l_{j,n}(x) := \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}.$$

3.1.4 Graph Theory

Most of the graph theory related notations and definitions that are introduced in the following are adapted from [39].

An *undirected graph* G is a tuple (V, E) where V is a finite set of *nodes* and E—a binary relation on V—refers to a set of edges. Each edge corresponds to a set $\{v, w\}$ of two nodes $v, w \in V$ with $v \neq w$. A *directed graph* $G = (V, E)$ is a graph where each edge has a direction, i.e., an edge corresponds to a tuple (v, w) of two nodes $v, w \in V$ with $v \neq w$. A *weighted graph* $G = (V, E, \omega_\epsilon)$ is a graph where each edge has an associated weight defined by the weight function $\omega_\epsilon : E \to \mathbb{R}^{\geq 0}$. Furthermore, a *bipartite graph* $G = (U, W, E)$ is an undirected graph with two disjoint sets of nodes U and W such that for each edge $\{u, w\} \in E$ it holds that $u \in U$ and $w \in W$.

Two nodes $v, w \in V$ in an undirected (resp., directed) graph $G = (V, E)$ are *adjacent* if there exists an edge $\{v, w\} \in E$ (resp., $(v, w) \in E$). Node v is called the *neighbor* of w (and vice versa) and edge $\{v, w\}$ (resp., edge (v, w)) is said to be *incident* to node v and w. For an undirected graph (resp., directed) graph, a node $w \in V$ is said to be *reachable* from node $v \in V$ iff there is a *path* $e_1, e_2, ..., e_m = \{v, u_1\}, \{u_1, u_2\}, \ldots, \{u_m, w\}$ (resp., $e_1, e_2, ..., e_m = (v, u_1), (u_1, u_2), \ldots, (u_m, w)$) such that $e_1, \ldots, e_m \in E$ and $v, w, u_1, \ldots, u_m \in V$ are pairwise different. If $p = \{u_1, u_2\}, \{u_2, u_3\}, \ldots, \{u_{m-1}, u_m\}$ is an undirected path (resp., $p = (u_1, u_2), (u_2, u_3), \ldots, (u_{m-1}, u_m)$ is a directed path), then $p, \{u_1, u_m\}$ (resp., $p, (u_m, u_1)$) is an m-cycle for $m \geq 3$ (resp., $m \geq 2$). The *degree* of a node v in V corresponds to the number of incident edges, written as $deg(v)$. For a directed graph, we distinguish between the incoming edges $deg^+(v)$ and the outgoing edges $deg^-(v)$ of a node v such that $deg(v) = deg^+(v) + deg^-(v)$.

A graph $G' = (V', E')$ is referred to as *subgraph* of a graph $G = (V, E)$, written as $G' \sqsubseteq G$, iff $V' \subseteq V$ and $E' \subseteq E$. Furthermore, a subgraph $G' = (V', E')$ of a graph $G = (V, E)$ is referred to as *component* of G if any two nodes $v', w' \in V'$ are reachable from each other and no node $v \in V \backslash V'$ is reachable from any $v' \in V'$. The union of two graphs $G' = (V', E')$ and $G'' = (V'', E'')$, written as $G' \sqcup G''$, yields the graph $G = (V' \cup V'', E' \cup E'')$.

Definition 3.5 ((Maximum, Perfect) Matching)**.** *Given a bipartite graph $G = (U, W, E)$, a matching M in G is a subset of edges $M \subseteq E \subseteq U \times W$ such that for all $v \in V := U \cup W$ it holds that v occurs as endpoint of an edge in M at most once, i.e., $|\{\{v, v'\} \in M : v' \in V\}| \leq 1$. A maximum matching M_{max} is a matching such that for any other matching M in G it holds that $|M_{max}| \geq |M|$. A perfect matching M_{perf} (which does not necessarily exist in G) is a matching such that $|M_{perf}| = |U| = |W|$.*

Definition 3.6 (Maximum Weight (Perfect) Matching)**.** *Let $G = (U, W, E, \omega_\epsilon)$ be a weighted bipartite graph with weight function ω_ϵ and let $\mathbb{M}$ (resp., $\mathbb{M}_{perf}$) be the set of all matchings (resp., perfect matchings) in G. Let $\omega_\epsilon(M) := \sum_{e \in M} \omega_\epsilon(e)$ for a matching M in G. A maximum weight matching M_{ω_ϵ} (resp., a maximum weight perfect matching $M_{\omega_\epsilon\text{-}perf}$) is a matching $M \in \mathbb{M}$ (resp., perfect matching $M_{perf} \in \mathbb{M}_{perf}$) such that for any $M' \in \mathbb{M}$ (resp., $M'_{perf} \in \mathbb{M}_{perf}$) $\omega_\epsilon(M) \geq \omega_\epsilon(M')$ (resp., $\omega_\epsilon(M_{perf}) \geq \omega_\epsilon(M'_{perf})$).*

3.1.5 Complexity Theory

Definition 3.7 (Asymptotic Notation [74])**.** *Let $f, g : \mathbb{N} \to \mathbb{R}^{>0}$ be functions from non-negative integers to non-negative reals. We write $f(n) = \mathcal{O}(g(n))$ or $f(n) \in \mathcal{O}(g(n))$ to indicate that there exist $c, n' \in \mathbb{N}$ such that for all $n > n'$ it holds that $f(n) \leq c \cdot g(n)$.*

Definition 3.8 (Negligible Function [74])**.** *A function $f : \mathbb{N} \to \mathbb{R}^{>0}$ is called* negligible *if for every polynomial p, there exists an $n' \in \mathbb{N}$ such that for all integers $n > n'$ it holds that $f(n) < 1/p(n)$.*

Definition 3.9 (PPT Algorithm)**.** *A* probabilistic polynomial time *(PPT) algorithm is a probabilistic algorithm which runs in time polynomial in k where $k \in \mathbb{N}$ refers to a security parameter.*

Definition 3.10 (Probability Ensemble, based on [68])**.** *A* probability ensemble *$X = \{X(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ is an infinite sequence of random variables $X(x,k)$ indexed by $x \in \{0,1\}^*$ and $k \in \mathbb{N}$.*

Definition 3.11 (Computational Indistinguishablity, based on [27])**.** *Two probability ensembles $X = \{X(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ and $Y = \{Y(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ are* computationally indistinguishable*, denoted $X \stackrel{c}{\equiv} Y$, if for every PPT distinguisher D there exists a negligible function $negl$ and a $k' \in \mathbb{N}$ such that for all $k > k'$, all $x \in \{0,1\}^*$, and $a \in \{0,1\}^*$:*

$$\Big| Pr[\mathsf{D}(1^k, x, a, X(x,k)) = 1] - Pr[\mathsf{D}(1^k, x, a, Y(x,k)) = 1] \Big| \leq negl(k).$$

Definition 3.12 (Statistical Indistinguishability, based on [27])**.** *Two probability ensembles $X = \{X(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ and $Y = \{Y(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ are* statistically indistinguishable*, denoted $X \stackrel{s}{\equiv} Y$, if there exists a $k' \in \mathbb{N}$ such that for all $k > k'$ and all $x \in \{0,1\}^*$:*

$$1/2 \sum_{y \in \{0,1\}^*} |Pr[X(x,k) = y] - Pr[Y(x,k) = y]| < negl(k).$$

Definition 3.13 (Perfect Indistinguishability, based on [27])**.** *Two probability ensembles $X = \{X(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ and $Y = \{Y(x,k)\}_{x \in \{0,1\}^*, k \in \mathbb{N}}$ are* perfectly indistinguishable*, denoted $X \stackrel{p}{\equiv} Y$, if for all k and $x \in \{0,1\}^*$, $X(x,k)$ and $Y(x,k)$ are identically distributed.*

3.1.6 Secret Sharing

A (τ, ι)-threshold secret sharing scheme allows the distribution of a secret S amongst ι parties such that each subgroup of at least τ parties can recover S from their own shares. In the following, we review two such schemes (*XOR secret sharing* and *Shamir's secret sharing*) by following the presentation provided in [74]. Both schemes are information-theoretically secure, i.e., any $\tau - 1$ parties learn no information about S from their aggregated shares.

3.1.6.1 XOR Secret Sharing

The XOR secret sharing scheme is a simple and restricted scheme which requires that $\tau := \iota$ and $S \in \{0, 1\}$.

Share Generation: Select $S_1, \ldots, S_{\iota-1} \leftarrow_\$ \{0, 1\}$ and compute $S_\iota = S \oplus S_1 \oplus \ldots \oplus S_{\iota-1}$. Share S_i is given to party P_i ($\forall i \in \mathbb{N}_\iota$).

Share Reconstruction: In case that all ι shares are aggregated, the parties can reconstruct the secret by computing $S = S_1 \oplus \ldots \oplus S_\iota$.

Note that it is straight-forward to extend this scheme in order to support the XOR sharing of a secret $S \in \{0, 1\}^n$ ($n \in \mathbb{N}$).

3.1.6.2 Shamir's Secret Sharing

A more powerful scheme is Shamir's (τ, ι)-threshold secret sharing scheme [107]. Let $\mathbb{F}$ be a finite field, $S \in \mathbb{F}$, and $|\mathbb{F}| > \iota$. The sharing and reconstruction method of Shamir's scheme make use of the fact that for any τ tuples $(x_1, y_1), \ldots (x_\tau, y_\tau) \in \mathbb{F} \times \mathbb{F}$ with pairwise different values x_i ($i \in \mathbb{N}_\tau$), there exists a unique polynomial $p(X)$ of degree $\tau - 1$ over $\mathbb{F}$ such that $p(x_i) = y_i$.

Share Generation: Let $x_1, \ldots, x_\iota \in \mathbb{F}$ be pairwise different and publicly known. A secret $S \in \mathbb{F}$ is (τ, ι)-threshold shared by selecting coefficients $a_1, \ldots, a_{\tau-1} \leftarrow_\$ \mathbb{F}$ and constructing polynomial $p(X) := S + \sum_{i=1}^{\tau-1} a_i X^i$. Share $S_i := p(x_i)$ is given to party P_i ($\forall i \in \mathbb{N}_\iota$).

Share Reconstruction: W.l.o.g., let $P_1, \ldots, P_\tau$ aggregate their shares $S_1, \ldots, S_\tau$. By utilizing Lagrange interpolation (see Definition 3.4), the parties can construct the $(\tau - 1)$-degree polynomial $p'(X)$ with $p'(x_i) = S_i$ and $p'(0) = S$ ($\forall i \in \mathbb{N}_\tau$) as p' is unique, $p' = p$, and $p(0) = S$.

3.1.7 Public Key Cryptography

In the following, the terms *encryption scheme* and *cryptosystem* are used interchangeably.

Definition 3.14 (Public Key Encryption Scheme [74])**.** *A tuple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is called a* public-key encryption scheme *if the following three properties are satisfied:*

1. *On input* 1^s *of a security parameter* $s \in \mathbb{N}$*, the key generation algorithm* Gen *outputs a key pair* (pk, sk) *whereas we refer to the first entry as* public key *and the second entry as* private key.

2. *On input* pk *and a message* m *from the* plaintext space $\mathbb{P}$*, the* (*probabilistic*) *encryption algorithm* Enc *outputs a ciphertext* c*, written as* $c \leftarrow \mathsf{Enc}_{pk}(m)$.

3. *On input* sk *and a ciphertext* $c \leftarrow \mathsf{Enc}_{pk}(m)$ *from the* ciphertext space $\mathbb{C}$*, the deterministic decryption algorithm* Dec *outputs a plaintext* m*, written as* $m := \mathsf{Dec}_{sk}(c)$.

Definition 3.15 (Public Key (τ, ι)-Threshold Encryption Scheme[1])**.** *A tuple* $(\mathsf{Gen}^{\tau,\iota}, \mathsf{Enc}^{\tau,\iota}, \mathsf{Dec}^{\tau,\iota})$ *is called a* public-key (τ, ι)-threshold encryption scheme *if the following three properties are satisfied:*

1. *On input* τ*,* ι*, and* 1^s *with* $s, \tau, \iota \in \mathbb{N}$ *and* $\tau \leq \iota$*, the key generation algorithm* $\mathsf{Gen}^{\tau,\iota}$ *outputs a public key* pk *and private key shares* sk_i $(i \in \mathbb{N}_\iota)$.

2. *On input* pk *and a message* m *from the plaintext space* $\mathbb{P}$*, the (probabilistic) encryption algorithm* $\mathsf{Enc}^{\tau,\iota}$ *outputs a ciphertext* c*, written as* $c \leftarrow \mathsf{Enc}^{\tau,\iota}_{pk}(m)$.

3. *On input* $c \leftarrow \mathsf{Enc}^{\tau,\iota}_{pk}(m)$ *and at least* τ *shares of the secret key* sk*, the deterministic decryption algorithm* $\mathsf{Dec}^{\tau,\iota}$ *computes* m*, written as* $m := \mathsf{Dec}^{\tau,\iota}_{sk}(c)$.

If the used encryption scheme is clear from context, for convenience, we sometimes write $[\![m]\!]$ in order to refer to an encryption of plaintext m and omit the public key from notation.

Definition 3.16 (IND-CPA Security [74])**.** *A public-key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with security parameter* s *has indistinguishable* (*IND*) *encryptions under chosen plaintext attacks* (*CPA*) *and is said to provide* IND-CPA security *if there exists a negligible function* $negl$ *such that all PPT adversaries* A *win the following game with probability less than or equal to* $1/2 + negl(s)$*:*

1. *Challenger* C *uses* $\mathsf{Gen}(1^s)$ *to generate a key pair* (pk, sk) *and provides adversary* A *with* pk.

[1] Definition 3.15 is a generalized variant of the corresponding definition presented in [50].

2. A *selects a pair of (equal-length) plaintexts* $m_0, m_1 \in \mathbb{P}$ *and gives both to* C.

3. C *chooses bit* $b \leftarrow_\$ \{0,1\}$, *computes* $c \leftarrow \mathsf{Enc}_{pk}(m_b)$, *and gives* c *to* A.

4. A *outputs* b'. *If* $b = b'$ *the adversary wins the game.*

Note that a public key encryption scheme has indistinguishable encryptions under chosen plaintext attacks if and only if it is *semantically secure* under chosen plaintext attacks (see Definition 5.4.8 and Theorem 5.4.11 in [59]). Thus, in the following, we use the terms IND-CPA security and semantic security interchangeably.

Definition 3.17 (Threshold IND-CPA Security [50])**.** *Let* $\mathsf{P\text{-}D\text{-}Ora}_{(sk_1,\ldots,sk_\iota)}$ *be a partial decryption oracle that on input* $m \in \mathbb{P}$ *returns* ι *partial decryptions of* m. *Furthermore, let* $\mathsf{E\text{-}Ora}_{pk,b}$ *be an encryption oracle that on input* $m_0, m_1 \in \mathbb{P}$ *computes* $c \leftarrow \mathsf{Enc}_{pk}(m_b)$. *A public key* (τ, ι)*-threshold encryption scheme provides* threshold IND-CPA security *if there exists a negligible function* $negl$ *such that all PPT adversaries* A *win the following game with probability less than or equal to* $1/2 + negl(s)$*:*

1. *Challenger* C *uses* $\mathsf{Gen}^{\tau,\iota}(1^s)$ *to generate threshold key* $(pk, sk_1, \ldots, sk_\iota)$.

2. *Adversary* A *chooses* τ *parties to corrupt and learns* pk *and the key shares of the corrupted parties.*

3. C *chooses bit* $b \leftarrow_\$ \{0,1\}$.

4. A *chooses* $m \in \mathbb{P}$ *and queries* $\mathsf{P\text{-}D\text{-}Ora}_{(sk_1,\ldots,sk_\iota)}(m)$. *This step is repeated an arbitrary number of times.*

5. A *chooses* $m_0, m_1 \in \mathbb{P}$ *and queries* $\mathsf{E\text{-}Ora}_{pk,b}(m_0, m_1)$. *This step is repeated an arbitrary number of times.*

6. A *outputs* b'. *If* $b = b'$, A *wins the game.*

Definition 3.18 (IND-MULT-CPA Security [74])**.** *Let* $\mathsf{E\text{-}Ora}_{pk,b}$ *be an encryption oracle that on input* $m_0, m_1 \in \mathbb{P}$ *computes* $c \leftarrow \mathsf{Enc}_{pk}(m_b)$. *A public-key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with security parameter* s *has indistinguishable* (IND) *encryptions for multiple* $(MULT)$ *messages under chosen plaintext attacks* (CPA) *and is said to provide* IND-MULT-CPA security *if there exists a negligible function* $negl$ *such that all PPT adversaries* A *win the following game with probability less than or equal to* $1/2 + negl(s)$*:*

1. *Challenger* C *uses* $\mathsf{Gen}(1^s)$ *to generate a key pair* (pk, sk) *and provides adversary* A *with* pk.

2. C *chooses bit* $b \leftarrow_{\$} \{0,1\}$.

3. A *selects* $m_0, m_1 \in \mathbb{P}$ *and queries* $\mathsf{E\text{-}Ora}_{pk,b}(m_0, m_1)$. *This step is repeated an arbitrary number of times.*

4. A *outputs* b'. *If* $b = b'$, A *wins the game.*

It is important to note that the plaintexts chosen by A for the oracle queries (Step 3) may depend on each other.

Theorem 3.1 ([74])**.** *If a public key encryption scheme is IND-CPA secure, then it is also IND-MULT-CPA secure.*

Definition 3.19 (Random Oracle Model, based on [74])**.** *The* random oracle model *postulates oracle access to a truly random hash function* $H : X \to Y$. *Receiving a query* $x \in X$, *the oracle returns* $y \leftarrow_{\$} Y$ *and stores* (x, y) *in case that* x *has not been queried before. Otherwise, the oracle accesses the stored queries, selects the unique entry* (x', y') *with* $x = x'$, *and returns* y'.

3.2 Partially Homomorphic Encryption

Characteristic for *partially homomorphic public key encryption schemes* is a homomorphism between the groups $(\mathbb{P}, \boxplus)$ and $(\mathbb{C}, \boxtimes)$ where $\boxplus$ and $\boxtimes$ are fundamental operations of the same adicity. Two relevant types of homomorphisms in this context are additive homomorphisms and multiplicative homomorphisms. An additively (resp., multiplicatively) homomorphic encryption scheme provides a 2-adic operation which performed on two ciphertext results in the encrypted sum (resp., product) of the corresponding plaintexts.

Definition 3.20 (Partially Homomorphic Public Key Encryption Scheme, based on [74])**.** *A public key encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is partially homomorphic if for all* s *and for all* (pk, sk) *generated by* $\mathsf{Gen}(1^s)$, *the corresponding plaintext space* $\mathbb{P}$ *and ciphertext space* $\mathbb{C}$ *form groups* $(\mathbb{P}, \boxplus)$ *and* $(\mathbb{C}, \boxtimes)$ *such that for any* $m_1, m_2 \in \mathbb{P}$ *and* $c_1 \leftarrow \mathsf{Enc}_{pk}(m_1)$, $c_2 \leftarrow \mathsf{Enc}_{pk}(m_2)$ *it holds that* $\mathsf{Dec}_{sk}(c_1 \boxtimes c_2) = m_1 \boxplus m_2$.[2]

In this thesis, we focus on a variant of the Paillier encryption scheme [94] (see Sections 3.2.1-3.2.3) which is an additively homomorphic and semantically secure encryption scheme. Utilizing the additively homomorphic property of Paillier, it is possible to derive other homomorphic operations we extensively make use of in this thesis (see Section 3.2.3). Furthermore, using Paillier in the context of designing SMPC protocols

[2]Note that a partially homomorphic public key encryption scheme is called additively (resp., multiplicatively) homomorphic if $(\mathbb{P}, \boxplus)$ is an additive (resp., multiplicative) group.

allows one to draw on a lot of previous work in this field (see Chapter 5). However, the contributions of this thesis are independent of the Paillier encryption scheme which can be replaced by any other suitable additively homomorphic encryption scheme (see Section 4.5.2).

3.2.1 Paillier Encryption Scheme

The Paillier encryption scheme [94], in the following simply referred to as Paillier, is a probabilistic additively homomorphic public-key encryption scheme which is IND-CPA secure. After a brief description of its components Gen, Enc, and Dec, we present a (τ, ι)-threshold variant of Paillier (Section 3.2.2) and the homomorphic properties of ((τ, ι)-threshold) Paillier (Section 3.2.3).

Key Generation: Let $\lambda(n) : \mathbb{N} \to \mathbb{Z}_n$ refer to the *carmichael function* which for any $n \in \mathbb{N}$ returns the minimal $m \in \mathbb{N}$ such that $a^m = 1 \; mod \; n$ for all $a \in \mathbb{Z}_n$ with $gcd(a, n) = 1$. On input 1^s, Gen randomly selects two strong primes p, q of bit size $s/2$, sets $N := p \cdot q$ and selects $\alpha \in \mathbb{N}_{\lambda(N)}$ and $g \in \mathbb{Z}^*_{N^2}$ such that g is of order $N\alpha$ in $\mathbb{Z}^*_{N^2}$. Gen outputs (pk, sk) with $pk := (N, g)$ and $sk := \lambda(N)$.

Encryption: On input (N, g) and $m \in \mathbb{Z}_N$, Enc selects $r \leftarrow_\$ \mathbb{Z}^*_N$ and outputs ciphertext

$$c := g^m \cdot r^N \; mod \; N^2.$$

Decryption: On input $\lambda(N)$ and $c \in \mathbb{Z}^*_{N^2}$, Dec outputs plaintext

$$m = \frac{L(c^{\lambda(N)} \; mod \; N^2)}{L(g^{\lambda(N)} \; mod \; N^2)} \; mod \; N,$$

where $\forall u \in \mathbb{Z}_{N^2}$ with $u = 1 \; mod \; N$

$$L(u) := (u - 1)/N.$$

The plaintext space $\mathbb{P} := \mathbb{Z}_N$ forms an additive group $(\mathbb{Z}_\mathbb{N}, +)$ and the ciphertext space $\mathbb{C} := \mathbb{Z}^*_{N^2}$ forms a multiplicative group $(\mathbb{Z}^*_{N^2}, \cdot)$. The security of Paillier is based on the *decisional composite residuosity assumption*:

Definition 3.21 (Decisional Composite Residuosity Assumption (DCRA), based on [74])**.** *Let N be as described above and let $r \leftarrow_\$ \mathbb{Z}_{N^2}$. For every PPT distinguisher* D *there exists a negligible function $negl$ such that*

$$\Big| Pr[\mathsf{D}(N, r^N \; mod \; N^2) = 1] - Pr[\mathsf{D}(N, r) = 1] \Big| \leq negl(s).$$

Theorem 3.2 ([94],[74]). *The Paillier encryption scheme is IND-CPA secure if and only if the computational composite residuosity assumption holds.*

3.2.2 Paillier Threshold Variant

Several (τ, ι)-threshold variants of Paillier have been proposed (e.g., [35,36,50]). Such a variant is also referred to as *threshold Paillier* and allows the sharing of a private Paillier key sk between ι parties. By the means of its key share sk_i a party P_i ($i \in \mathbb{N}_\iota$) can only compute a partial decryption of a given ciphertext. In order to recover the original plaintext of a given ciphertext, at least τ parties have to jointly decrypt the ciphertext by using their private key shares.

In the following, we present the threshold variant from [50] which we (arbitrarily) choose to use as an underlying technique for designing secure multi-party protocols. However, any other variant can be used in the context of this thesis as well.

Key Generation: On input τ, ι, and 1^s, Gen randomly selects two strong primes p, q of bit size $s/2$ and sets $N = p \cdot q$ and $N' = p'q'$. Next, Gen selects $\beta \leftarrow_\$ \mathbb{Z}_N^*$, selects $(a, b) \leftarrow_\$ \mathbb{Z}_N^* \times \mathbb{Z}_N^*$, and sets $g = (1+N)^a \cdot b^N \ mod\ N^2$. The secret key $sk = \beta N'$ is shared by using the Shamir threshold secret sharing scheme (see Section 3.1.6.2): choose $a_1, \ldots, a_{\tau-1} \leftarrow_\$ \mathbb{N}_{(N \cdot N'-1)}^0$, and set $p(X) := sk + \sum_{i'=1}^{\tau-1} a_{i'} X^{i'}$. Share $S_i = p(i) \ mod\ NN'$ is given only to P_i ($\forall i \in \iota$) while the public key $pk = (N, g, \theta)$, $\theta := aN'\beta \ mod\ N$ is published.

Encryption: On input (N, g) and $m \in \mathbb{Z}_N$, Enc selects $r \leftarrow_\$ \mathbb{Z}_N^*$ and outputs ciphertext

$$c := g^m \cdot r^N \ mod\ N^2.$$

Decryption: For a given ciphertext c, P_i computes the partial decryption $c_i = c^{2\iota! S_i}$. Let T be any set of τ partial decryptions. On input T, Dec outputs plaintext

$$m = L\left(\prod_{j \in T} c_j^{2\mu_{0,j}^T} \ mod\ N^2\right) \cdot \frac{1}{4\theta(\iota!)^2} \ mod\ N,$$

where $\mu_{0,j}^T = \iota! \cdot \prod_{j' \in T \setminus \{j\}} j'/(j'-j)$. Depending on how the partial decryptions are distributed, it can be controlled which parties learn plaintext m.

Theorem 3.3 ([50]). *In the random oracle model, the threshold variant of Paillier is threshold IND-CPA secure if the computational composite residue assumption holds.*

The proof of Theorem 3.3 from [50] shows that if there exists a PPT algorithm that breaks the threshold IND-CPA security of the threshold variant of Paillier from [50], it can be used to break the IND-CPA security of Paillier.

3.2.3 Homomorphic Properties of (Threshold) Paillier

The Paillier encryption scheme as well as the (τ, ι)-threshold variants provide the following homomorphic operations:

Homomorphic Addtion: Let $m_1, m_2 \in \mathbb{P}$. The *homomorphic addition* of two ciphertexts $\mathsf{Enc}(m_1) = g^{m_1} \cdot r_1^N \ mod\ N^2$ and $\mathsf{Enc}(m_2) = g^{m_2} \cdot r_2^N \ mod\ N^2$, written as $\mathsf{Enc}(m_1) +_h \mathsf{Enc}(m_2)$, can be computed as

$$\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = g^{m_1} r_1^N g^{m_2} r_2^N \ mod\ N^2 = g^{m_1+m_2}(r_1 r_2)^N \ mod\ N^2 = \mathsf{Enc}(m_1 + m_2).$$

Homomorphic Subtraction: Let $m_1, m_2 \in \mathbb{P}$. The *homomorphic subtraction* of two ciphertexts $\mathsf{Enc}(m_1) = g^{m_1} \cdot r_1^N \ mod\ N^2$ and $\mathsf{Enc}(m_2) = g^{m_2} \cdot r_2^N \ mod\ N^2$, written as $\mathsf{Enc}(m_1) -_h \mathsf{Enc}(m_2)$, can be computed as

$$\mathsf{Enc}(m_1)/\mathsf{Enc}(m_2) = g^{m_1} r_1^N (g^{m_2} r_2^N)^{-1} \ mod\ N^2 = g^{m_1-m_2} r_1^N r_2^{-N} \ mod\ N^2 = \mathsf{Enc}(m_1 - m_2).$$

Homomorphic Scalar Multiplication: Let $m \in \mathbb{P}$ and $a \in \mathbb{Z} \backslash \{0\}$. The *homomorphic scalar multiplication* of a ciphertext $\mathsf{Enc}(m)$ and scalar a, written as $a \times_h \mathsf{Enc}(m)$ or $\mathsf{Enc}(m) \times_h a$, can be computed as

$$(\mathsf{Enc}(m))^a = \underbrace{\mathsf{Enc}(m) +_h \ldots +_h \mathsf{Enc}(m)}_{a\text{-times}} = \mathsf{Enc}(a \cdot m).$$

In order to allow the computation of $0 \times_h \mathsf{Enc}(m)$, we define the result to be a fresh encryption of zero.

Homomorphic Scalar Division: Let $m \in \mathbb{P}$ and $b \in \mathbb{Z}_N^*$. The *homomorphic scalar division* of a ciphertext $\mathsf{Enc}(m)$ and scalar b, written as $\mathsf{Enc}(m)/_h\, b$, can be computed as

$$\mathsf{Enc}(m) \times_h (b^{-1} \ mod\ N) = \mathsf{Enc}(m \cdot b^{-1} \ mod\ N).$$

Homomorphic Scalar XOR: Let $b_1, b_2 \in \{0, 1\} \subseteq \mathbb{P}$. The *homomorphic XOR operation* of a ciphertext $\mathsf{Enc}(b_1)$ and a bit b_2, written as $\mathsf{Enc}(b_1) \oplus_h b_2$, can be computed as

$$\mathsf{Enc}(b_1) -_h (2b_2 \times_h \mathsf{Enc}(b_1)) +_h \mathsf{Enc}(b_2) = \mathsf{Enc}(b_1 \oplus b_2).$$

Homomorphic Concatenation: Let $m_1, m_2 \in \mathbb{P}$. Given an upper bound for the number d of decimal places of m_2, the *homomorphic concatenation* of two ciphertexts $\mathsf{Enc}(m_1)$ and $\mathsf{Enc}(m_2)$, written as $\mathsf{Enc}(m_1) \,||_h\, \mathsf{Enc}(m_2)$, can be computed as

$$\mathsf{Enc}(m_1) \times_h 10^d +_h \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1||m_2).$$

Ciphertext Blinding: Let $m \in \mathbb{P}$. A ciphertext $\mathsf{Enc}(m)$ can be *blinded* (i.e., *rerandomized*), written as $\mathsf{Blind}(\mathsf{Enc}(m))$ by homomorphically adding a fresh encryption of zero.

3.3 Zero Knowledge Proofs of Knowledge

Based on [68], we briefly introduce the notion of *zero knowledge proofs of knowledge* (ZKPOK) including the notion of Sigma-protocols (Section 3.3.1) which are interactive proofs of a simple structure that can be efficiently transformed into ZKPOK. Furthermore, we present some specific Sigma-protocols (Section 3.3.2) which can be used in the context of designing secure multi-party protocols to enforce honest behavior, i.e., each party has to prove to the other parties that it follows the protocol specification.

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. For a pair $(x, w) \in R$, x refers to a public instance of a computational problem while w refers to the corresponding solution and we write $x \sim w$. A ZKPOK is an interactive proof which allows a prover P to convince a verifier V that it knows a witness w for some instance x in such a way that the verifier learns nothing beyond the fact that the prover knows w such that $(x, w) \in R$.

Definition 3.22 (Zero Knowledge Proof of Knowledge)**.** *A protocol π is referred to as* zero knowledge proof of knowledge *(ZKPOK) for a given relation R if π satisfies the following properties:*

1. *Completeness: If P holds $(x, w) \in R$, V holds x, and both parties follow the protocol specification, then V always accepts.*

2. *Validity: For every prover P^* that is accepted by V with non-negligible probability there exists a PPT knowledge extractor that can extract w' from interacting with P^* such that $(x, w') \in R$.*

3. *Zero Knowledge: For every verifier V^* there exists a PPT simulator that on input x outputs a transcript which is perfectly indistinguishable from the transcripts resulting from the execution of π between P and V for instance x.*

P: x, w s.t. $(x, w) \in R$		V: x
1. Compute commitment a		
	2. $\xrightarrow{\quad a \quad}$	
		3. Generate random challenge $e \in \{0,1\}^t$
	4. $\xleftarrow{\quad e \quad}$	
5. Compute response z based on a, e, w		
	6. $\xrightarrow{\quad z \quad}$	
		7. Accept or reject (a, e, z)

Protocol 3.1: General three-move structure of Σ-protocols.

3.3.1 Sigma-Protocols

Sigma-protocols (hereinafter written as Σ-protocols) are interactive proofs with a three-move interaction: P sends a commitment a; V returns a random challenge e with $|e| = t$; P returns a value z computed based on a, e, and w (cf. Protocol 3.1). Depending on the resulting transcript, the verifier decides whether or not to accept the proof. The computations steps and the verification procedure depend on the specific Σ-protocol.

Definition 3.23 (Σ-Protocol, based on [68])**.** *A protocol π is referred to as* Σ-protocol *for a given relation R if π is a three-move protocol and it satisfies the following properties:*

1. *Completeness: If* P *holds $(x, w) \in R$,* V *holds x, and both parties follow the protocol specification, then* V *always accepts the resulting transcript (a, e, z).*

2. *Special Soundness: There exists a PPT algorithm that computes w on input x and two accepting transcripts (a, e, z), (a, e', z') with $e \neq e'$.*

3. *Special Honest Verifier Zero Knowledge: There exists a PPT simulator that on input x and e computes a transcript (a, e, z) which is perfectly indistinguishable from transcripts resulting from the execution of π between honest* P *and honest* V *for instance x.*

Σ-protocols can be efficiently transformed into ZKPOK (see, e.g., [68]). Thus, instead of designing a ZKPOK from scratch, one can concentrate on the simpler problem of constructing a Σ-protocol followed by applying the necessary transformations to achieve a ZKPOK [68].

Theorem 3.4. *Let π be a Σ-protocol for relation R. Every π can be efficiently transformed into a zero knowledge proof of knowledge with knowledge error 2^{-t}.*

The knowledge error indicates the probability that P can convince V without knowing w.

Apart from exhibiting an efficient transformation to ZKPOK, Σ-protocols have some further useful properties, for example, they are invariant under parallel repetition and they can be used to prove compound statements (combining statements by, e.g., AND and OR operations) [68]. Moreover, the *Fiat-Shamir heuristic* [47] can be used to transform Σ-protocols into extraordinary efficient *non-interactive* ZKPOKs. On the one hand this transformation removes the honest-verifier assumption (i.e., soundness can be achieved w.r.t. malicious verfiers) and on the other hand it removes the interaction between prover P and verifier V which allows P to convince multiple verifiers at the same time. Essentially, the transformation replaces the verifier's random challenge by instructing the prover to apply a hash function idealized by a random oracle on its first message. Note that this transformation guarantees soundness only in the random oracle model [12]. In the literature, there are two variants of the Fiat-Shamir heuristic differing w.r.t. the input of the idealized hash function: the *weak Fiat-Shamir heuristic* and the *strong Fiat-Shamir heuristic* [15]. The former (formalized in [12]) applies the hash function on the prover's first message while the latter (suggested, e.g., in [49]) demands to additionally include the problem instance in the input of the hash function. Since security issues have been identified for the weak Fiat-Shamir heuristic (e.g., [15]), we assume the use of the strong Fiat-Shamir heuristic in this thesis.

3.3.2 Specific Sigma-Protocols

In the following, we review a selection of Σ-protocols which are used explicitly in the context of this thesis. We write $\mathsf{Enc}(\cdot, r)$ in order to make the randomization of a probabilistic encryption function Enc explicit.

Proof of Plaintext Knowledge (POPK): Let $c \in \mathbb{C}$. POPK refers to a Σ-protocol for proving the relation $c \sim (m, r) \Leftrightarrow (m \in \mathbb{P}) \wedge (c = \mathsf{Enc}(m, r))$. A POPK Σ-protocol for Paillier ciphertexts has been proposed in [27].

Proof of Plaintext Bit (POPB): Let $c \in \mathbb{C}$. POPB refers to a Σ-protocol for proving the relation $c \sim (m, r) \Leftrightarrow (c = \mathsf{Enc}(m, r)) \wedge m \in \{0, 1\}$. For the Paillier cryptosystem, POPB can be implemented by using the *1-out-of-2 n-th power* Σ-protocol from [35].

Proof of Plaintext Equality (POPE): Let $c_1, c_2 \in \mathbb{C}$. POPE refers to a Σ-protocol for proving the relation $(c_1, c_2) \sim (m_1, m_2, r_1, r_2) \Leftrightarrow (c_1 = \mathsf{Enc}(m_1, r_1)) \wedge (c_2 = \mathsf{Enc}(m_2, r_2)) \wedge (m_1 = m_2)$. For the Paillier cryptosystem, POPE can be implemented by using the *n-th power* Σ-protocol from [35].

CHAPTER 4

Secure Multi-Party Computation

Building on the brief introduction to secure multi-party computation (SMPC) presented in Chapter 2 (Section 2.1.2), we now provide the theoretical background by focusing on the semi-honest and the malicious adversary model. The goal is to give a precise definition of the term *security* and to provide the necessary preliminaries required to formally prove the security of an SMPC protocol in one of the considered adversary models. While proving the security of a protocol in the semi-honest model is straightforward, we use the CDN-framework [27,28] in order to carry out tight security proofs in the more complicated malicious model.

Outline: First, we provide some informal background information on secure multi-party computation (Section 4.1) and give an overview of the general security model considered in this thesis (Section 4.2). Subsequently, we present the notions of security in the semi-honest model (Section 4.3) and in the malicious model (Section 4.4). Section 4.4 also contains a sketch of the CDN-framework (Section 4.4). In Section 4.5, we address some general aspects of SMPC like the distribution of key material and the used complexity measures.

4.1 Background

The overall goal of secure multi-party computation is to allow a set of ι parties $P_1, \ldots, P_\iota$ to securely (i.e., privately and correctly) compute an ι-input functionality $\mathcal{F}$ such that each participating party P_i ($i \in \mathscr{P} := \{1, \ldots, \iota\}$) learns nothing but its prescribed output and what can be deduced from it in combination with its private input—even in the presence of an adversary that can corrupt a subset of parties. In order to achieve this goal, an SMPC protocol π securely implementing functionality $\mathcal{F}$ has to be designed. The

principal goal of an adversary is to learn some information on the honest parties' private input or to influence the computation result. In this context, an important characteristic of SMPC protocols is provable security (given the security of the underlying cryptographic techniques), i.e., a protocol provides security independent of an adversary's strategy. Important fields of applications for SMPC are electronic voting (e.g., [30]), electronic auctions (e.g., [21]), data mining (e.g., [80,81,97]), and biometric authentication (e.g., [17,45,106]).

For a variety of different SMPC models (determining, e.g., the number of supported parties, the capabilities of the adversary, and bounds on the number of corrupted parties) there exist so-called feasibility results stating under which conditions any functionality can be implemented as an SMPC protocol (e.g., [14,60,99,127]). In a sense, these results provide SMPC frameworks which allow the generic design of an SMPC protocol for a given functionality. However, depending on the target functionality, the generic constructions may result in impractical protocols. Thus, there is a second line of work which aims at designing efficient SMPC protocols for specific functionalities (see, e.g., [2,53,79,80]). In this context, the primary goal is to exploit the properties of a specific functionality in order to come up with a customized protocol design. In this thesis, we focus on the specific design approach.

4.2 General Security Model

Before formally defining what it means to securely compute a functionality (see Section 4.3 and Section 4.4), we first specify the considered communication model and classify the power of the adversary in whose presence an SMPC protocol is executed.

For our SMPC protocols, we assume the *cryptographic communication model* where the adversary is given access to all exchanged messages but the communication channels are protected such that the adversary cannot omit, modify, duplicate, or generate messages sent between honest parties [29,59]. The communication channels are referred to as *reliable*. Note that the reliability of communication channel is only guaranteed in a cryptographic sense.

If not stated otherwise, we distinguish between the corrupted parties and the adversary. The corrupted parties do actively participate in the execution of an SMPC protocol and are fully controlled by the adversary. Among the most important attributes that have to be specified in order to classify the power of an adversary are: (1) coordination capability, (2) adversary behavior, (3) corruption strategy, (4) adversary structure, (5) computational power (cf., e.g., [29,59]).

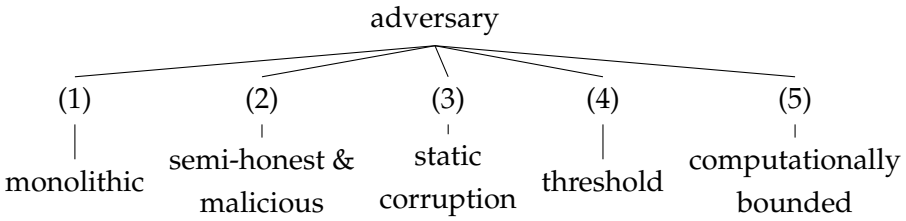


Figure 4.1: Considered adversary power.

Coordination Capability: We consider a *monolithic* adversary who is characterized by the ability to coordinate the actions of corrupted parties and to manage an exchange of information between them.

Adversary Behavior: We consider a *semi-honest* (passive) adversary as well as a *malicious* (active) adversary. A semi-honest adversary instructs the corrupted parties to follow the protocol specification but to record their individual protocol transcripts. The adversary obtains the transcripts of all corrupted parties on the base of which it tries to learn private information on the honest parties (i.e., not corrupted parties) inputs. A malicious adversary additionally has the capability to instruct the corrupted parties to arbitrarily deviate from the protocol specification.

Corruption Strategy: We assume that the adversary is *static*, i.e., the set of corrupted parties is fixed before a protocol execution starts. Once a party gets corrupted, the adversary learns all data held by this party, particularly its input, the sent and received messages as well as its *internal coin tosses* (randomness).

Adversary Structure: The adversary structure Γ is a family of subsets of $\mathscr{P}$ that contains all considered subsets of parties that can be controlled by the adversary. We consider a *threshold* adversary structure that consists of all subsets of parties with a cardinality of $1 \leq t < \iota$.

Computational Power: We consider an adversary that is *computationally bounded*, i.e., it can be represented as a PPT algorithm.

4.3 Semi-Honest Model

The corrupted parties controlled by a semi-honest adversary exactly follow the protocol instructions, but keep record of their protocol view consisting of their input and output,

internal coin tosses, and the messages received from other parties. By analyzing the aggregated protocol views of the corrupted parties, the adversary tries to learn as much as possible from the protocol execution which goes beyond the input and output of the corrupted parties and what can be deduced thereof. Informally, an SMPC protocol is said to be secure if gaining such additional information from protocol participation is impossible. The security model in which a semi-honest adversary is considered is referred to as the *semi-honest model*.

Although the semi-honest model only guarantees security against passive attacks, it is well justified. Providing security against passive attacks is often sufficient for real world applications [77], especially if the essential protocol behavior is controlled and hidden by complex software, where a targeted deviation is much more difficult than just recording and analyzing the contents of communication and registers [59]. Targeted deviations from the protocol specification may require advanced in-depth knowledge which is unattainable for standard users of such software. Furthermore, the associated costs of such attacks may far exceed the potential benefit from cheating [77]. The small probability of being caught cheating, which may result in bad reputation, exclusion, or the initiation of legal actions, may deter users from actively cheating. Conversely, semi-honest behavior never can be detected because it is restricted to the analysis of local data which a party is intended to obtain by the protocol specification. Furthermore, the design of semi-honest protocols is considered to be a first step to achieve security against malicious adversaries because it already provides a basis for a protocol secure in the malicious model where the parties additionally have to be forced to follow the protocol specifications.

The formalization of the semi-honest model which is presented in the following is according to [59,68]. In order to facilitate the introduction of the notion related to secure protocol design, we separately treat the two-party case for the semi-honest model (see Section 4.3.1) and subsequently provide the extension for the multi-party case (see Section 4.3.2).

4.3.1 Two-Party Case

In the two-party case where an SMPC protocol is executed by two parties, the semi-honest adversary controls exactly one party, thus, the adversary and the corrupted party coincide. In the following, we provide the definition of security stated according to the *simulation paradigm* in the semi-honest model for the two-party case.

Let $\mathcal{F} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*, (x_1, x_2) \mapsto (\mathcal{F}_1(x_1, x_2), \mathcal{F}_2(x_1, x_2))$ be a two-party functionality where P_i ($i \in \mathscr{P} := \{1, 2\}$) provides input x_i and obtains output

$\mathcal{F}_i(x_1, x_2)$. Let π be a two-party protocol implementing functionality $\mathcal{F}$. The *view* of a party P_i w.r.t. the execution of protocol π on input (x_1, x_2) and security parameter $k \in \mathbb{N}$ is defined as $\mathsf{VIEW}_i^{\pi}(x_1, x_2, k) := (x_i, \tilde{r}_i, m_1, \ldots, m_n)$, where $\tilde{r}_i$ represents P_i's the internal coin tosses and m_j refers to the j-th message received as part of π. The corresponding protocol output of P_i is denoted as $\mathsf{OUTPUT}_i^{\pi}(x_1, x_2, k)$. The joint output of both parties is denoted as $\mathsf{OUTPUT}^{\pi}(x_1, x_2, k) := (\mathsf{OUTPUT}_1^{\pi}(x_1, x_2, k), \mathsf{OUTPUT}_2^{\pi}(x_1, x_2, k))$. The adversary structure is $\Gamma = \{\{1\}, \{2\}\}$.

Definition 4.1 (Security: Two-Party Case, Semi-Honest Model)**.** *A two-party protocol π is said to* securely implement *a two-party functionality $\mathcal{F}$ in the semi-honest model if there exist PPT simulators S_i $(i \in \{1, 2\})$ such that*

$$\{(\mathsf{S}_i(1^k, x_i, \mathcal{F}_i(x_1, x_2)), \mathcal{F}(x_1, x_2))\}_{x_1, x_2, k} \stackrel{c}{\equiv} \{(\mathsf{VIEW}_i^{\pi}(x_1, x_2, k), \mathsf{OUTPUT}^{\pi}(x_1, x_2, k))\}_{x_1, x_2, k}.$$

Informally, Definition 4.1 states that a protocol π securely implements functionality $\mathcal{F}$ in the semi-honest model if the view of a party (real view) can be simulated (simulated view) from the party's input and output only. When proving the security of a protocol, we enclose the simulated values generated by a simulator with angle brackets $\langle \cdot \rangle$ in order to distinguish them from the values that occur in the real protocol execution.

Remark 4.1 (Gate Functionalities, Gates: Two-Party Case)**.** A *gate functionality $\mathcal{G}$* (resp., *gate ρ*) is a special type of functionality (resp., protocol) which obtains encrypted/shared input and/or returns encrypted/shared output where the plain values are generally not known to any party. When defining a gate functionality $\mathcal{G}$ (resp., a gate ρ) we use the notation $(y) \leftarrow \mathcal{G}(x)$ (resp., $(y) \leftarrow \rho(x)$) to indicate that both parties provide the same input and obtain the same output. A gate functionality (resp., a gate) requires a given *context* within which it is called. The context may, e.g., include access to a public-key infrastructure, to already shared values, to a common trusted random string, or to a trusted list of ciphertexts satisfying a specific property. For a given context, the goal is to show that a gate ρ can be simulated from gate specific input (and output) such that the simulated view is computationally indistinguishable (resp., statistically indistinguishable) from the real view of the protocol execution in order to enable that a gate can be called within a protocol π.

4.3.2 Multi-Party Case

In the multi-party case, an SMPC protocol is executed between ι parties with $\iota \geq 2$. In the following, we provide the definition of security stated according to the simulation paradigm in the semi-honest model for the multi-party case.

Let $\overline{x} := (x_1, \ldots, x_\iota)$ and let $\mathcal{F} : (\{0,1\}^*)^\iota \to (\{0,1\}^*)^\iota, \overline{x} \mapsto (\mathcal{F}_1(\overline{x}), \ldots, \mathcal{F}_\iota(\overline{x}))$ be a multi-party functionality where P_i ($i \in \mathscr{P} := \{1, \ldots, \iota\}$) provides input x_i and obtains output $\mathcal{F}_i(\overline{x})$. Let π be a multi-party protocol implementing functionality $\mathcal{F}$. The view and the output of a party P_i w.r.t. the execution of protocol π on input $\overline{x}$ and security parameter k are defined analogously to the two-party case. We write $\mathsf{OUTPUT}^\pi(\overline{x}, k) := (\mathsf{OUTPUT}_1^\pi(\overline{x}, k), \ldots, \mathsf{OUTPUT}_\iota^\pi(\overline{x}, k))$ in order to refer to the output of protocol π on input $\overline{x}$ and security parameter k. The adversary structure Γ contains all proper subsets of the ι parties. Let $C = \{i_1, \ldots, i_t\} \subset \mathscr{P}$ be the set of indices of a specific subset of corrupted parties. We write $\overline{x}_C$, $\mathcal{F}_C(\overline{x})$, and $\mathsf{VIEW}_C^\pi(\overline{x}, k)$ as a short-hand notation for $(x_{i_1}, \ldots, x_{i_t})$, $(\mathcal{F}_{i_1}(\overline{x}), \ldots, \mathcal{F}_{i_t}(\overline{x}))$, and $(C, \mathsf{VIEW}_{i_1}^\pi(\overline{x}, k), \ldots, \mathsf{VIEW}_{i_t}^\pi(\overline{x}, k))$, respectively.

Definition 4.2 (Security: Multi-Party Case, Semi-Honest Model)**.** *An ι-party protocol π is said to* securely implement *the ι-party functionality $\mathcal{F}$ in the semi-honest model if there exists a PPT simulator* S *such that for every $C \in \Gamma$ it holds that*

$$\{(\mathsf{S}(1^k, C, \overline{x}_C, \mathcal{F}_C(\overline{x})), \mathcal{F}(\overline{x}))\}_{\overline{x},k} \stackrel{c}{\equiv} \{(\mathsf{VIEW}_C^\pi(\overline{x}, k), \mathsf{OUTPUT}^\pi(\overline{x}, k))\}_{\overline{x},k}$$

As for the two-party case, we enclose the simulated values generated by a simulator with angle brackets $\langle \cdot \rangle$.

Remark 4.2 (Gate Functionalities, Gates: Multi-Party Case)**.** For the multi-party case, gate functionalities (resp., gates) can be described analogously to Remark 4.1.

4.3.3 Sequential Modular Composition

In the following, we present a composition theorem for the semi-honest model according to [59] (by utilizing Remark 7.4.5, [59]) which allows the analysis of the security of a protocol which makes use of some subprotocols in a modular way. In the context of this thesis, we only use the composition theorem when calling a decryption operation within a superordinate protocol in order to gain independence of a specific additively homomorphic threshold cryptosystem.

A protocol π implementing funtionality $\mathcal{F}$ is said to be *oracle-functionalities $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-aided* if the parties executing protocol π do have a sequential access to a series of n oracles to which they provide their private input in an established order and from which they receive their private output. The oracle-functionality $\mathcal{F}_i$ ($i \in \mathbb{N}_n$) obtains input and provides output according to the definition of functionality $\mathcal{F}$. An oracle-functionalities $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-aided protocol π is said to *securely reduce* to functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_n$ if π securely computes functionality $\mathcal{F}$ when making use of the oracle functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_n$.

Theorem 4.1 (Sequential Modular Composition: Semi-Honest Model)**.** *Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be a series of ι-party functionalities and let $\pi_1, \ldots, \pi_n$ be a series of ι-party protocols securely computing $\mathcal{F}_1, \ldots, \mathcal{F}_n$, respectively. Let $\mathcal{F}$ be a ι-party functionality and let π' be an ι-party protocol securely reducing $\mathcal{F}$ to $\mathcal{F}_1, \ldots, \mathcal{F}_n$ in the presence of a semi-honest adversary. Then, an ι-party protocol π which securely implements functionality $\mathcal{F}$ in the presence of semi-honest adversaries is derived from π' by replacing the oracle calls for $\mathcal{F}_1, \ldots, \mathcal{F}_n$ by executions of $\pi_1, \ldots, \pi_n$, respectively.*

4.4 Malicious Model

As for a semi-honest adversary, a malicious adversary controlling a subset of parties tries to learn as much as possible from the protocol execution. However, the malicious adversary is additionally allowed to instruct the corrupted parties to arbitrarily deviate from the protocol specification which, e.g., includes the substitution of the actual input and the premature abortion of the protocol execution. The principal approach to design protocols which are secure in the presence of a malicious adversary is to augment a protocol in such a way that semi-honest behavior is enforced. The security model in which a malicious adversary is considered is referred to as the *malicious model*. In this section, we skip the explicit treatment of the two-party case (this case is covered by the definition of security in the malicious model for the multi-party case).

4.4.1 Multi-Party Case

In the following, let $\mathcal{F}$ and π be as specified in Section 4.3.2. The formalization of the malicious model provided in the following is according to [24]. Security is stated in terms of the *ideal world vs. real world paradigm*.[1] More precisely, the capabilities of a *real world adversary* A in a so-called *real world execution* of π are compared to the capabilities of an *ideal world adversary* S interacting in the ideal world where a trusted third party exists. In the ideal world model, the parties send their private input to a trusted third party which computes the target functionality on the obtained inputs and provides each party with its prescribed output. Thus, the computation of $\mathcal{F}$ in the ideal world model achieves the maximum level of security. However, in the real world model no trusted third party exists and the parties execute protocol π on their private inputs in order to obtain their respective outputs. Intuitively, the *ideal world view* (resp., *real world view*)

[1] Note that security in the semi-honest model can also be defined in terms of the ideal vs. real world paradigm. However, the simulation paradigm allows for a more simple definition which cannot be used to define security in the malicious model.

of a party consists of the information (e.g., the received messages) it learns during the evaluation of $\mathcal{F}$ (resp., computation of π).

Definition 4.3 (Ideal World Model)**.** *Let* $\mathsf{ADVR}^{\mathcal{F}}_{\mathsf{S}(a)}(\overline{x}, k)$ *(resp.* $\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{S}(a),i}(\overline{x}, k)$*) denote the output of a malicious ideal world adversary (resp., the output of* P_i*) for functionality* $\mathcal{F}$ *(computed by a trusted third party) on input* $\overline{x}$ *and security parameter* k *under attack of* S *given auxiliary input* a *including* C*. The joint evaluation of* $\mathcal{F}$ *in the ideal world model is denoted as*

$$\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{S}(a)}(\overline{x}, k) := (\mathsf{ADVR}^{\mathcal{F}}_{\mathsf{S}(a)}(\overline{x}, k), \mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{S}(a),1}(\overline{x}, k), \ldots, \mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{S}(a),\iota}(\overline{x}, k)).$$

Definition 4.4 (Real World Model)**.** *Let* $\mathsf{ADVR}^{\pi}_{\mathsf{A}(a)}(\overline{x}, k)$ *(resp.,* $\mathsf{REAL}^{\pi}_{\mathsf{A}(a),i}(\overline{x}, k)$*) denote the output of a malicious real world adversary* A *(resp., the output of* P_i*) for protocol* π *on input* $\overline{x}$ *and security parameter* k *under attack of* A *given auxiliary input* a *including* C*. The joint execution of* π *in the real world model is denoted as*

$$\mathsf{REAL}^{\pi}_{\mathsf{A}(a)}(\overline{x}, k) := (\mathsf{ADVR}^{\pi}_{\mathsf{A}(a)}(\overline{x}, k), \mathsf{REAL}^{\pi}_{\mathsf{A}(a),1}(\overline{x}, k), \ldots, \mathsf{REAL}^{\pi}_{\mathsf{A}(a),\iota}(\overline{x}, k)).$$

Just as a real world adversary, an ideal world adversary is allowed to substitute the input of the corrupted party and, depending on Γ, to instruct the trusted third party to prematurely abort.

Definition 4.5 (Security: Multi-Party Case, Malicious Model)**.** *An* ι*-party protocol* π *is said to securely implement the* ι*-party functionality* $\mathcal{F}$ *in the malicious model if for any PPT real world adversary corrupting parties in* $C \in \Gamma$*, there exists a PPT ideal world adversary* S *such that*

$$\{\mathsf{IDEAL}^{\mathcal{F}}_{\mathsf{S}(a)}(\overline{x}, k)\}_{\overline{x},k} \stackrel{c}{\equiv} \{\mathsf{REAL}^{\pi}_{\mathsf{A}(a)}(\overline{x}, k)\}_{\overline{x},k}.$$

Gate functionalities and gates for the malicious model can be described analogously as it has been done for the semi-honest case in Section 4.3.

4.4.2 Sequential Modular Composition

As for the semi-honest model, there is a composition theorem for the malicious model which is presented in the following according to [24]. This composition theorem is only used when calling a decryption operation within a superordinate protocol. Before presenting the composition theorem, we first define the *hybrid model*.

Definition 4.6 (Hybrid Model)**.** *In the* $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$*-hybrid model, a protocol* π *implementing functionality* $\mathcal{F}$ *is executed in the real world model but is augmented in that an ideal world model access is granted in order to evaluate* $\mathcal{F}_1, \ldots, \mathcal{F}_n$*, i.e., a trusted third party is sequentially invoked in order to compute functionalities* $\mathcal{F}_1, \ldots, \mathcal{F}_n$*.*

Theorem 4.2 (Sequential Modular Composition: Malicious Model)**.** *Let $\mathcal{F}_1, \ldots, \mathcal{F}_n$ be a series of ι-party functionalities and let $\pi_1, \ldots, \pi_n$ be a series of ι-party protocols securely computing $\mathcal{F}_1, \ldots, \mathcal{F}_n$, respectively. Let $\mathcal{F}$ be a ι-party functionality and let π' be an ι-party protocol that securely implements $\mathcal{F}$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_n)$-hybrid model in the presence of malicious adversaries. Then, an ι-party protocol π derived from protocol π' where the ideal world calls of $\mathcal{F}_1, \ldots, \mathcal{F}_n$ are replaced by the executions of protocols $\pi_1, \ldots, \pi_n$ securely implements $\mathcal{F}$ in the presence of malicious adversaries.*

4.4.3 CDN-Framework

In the following, we briefly review the CDN framework [27,28]. The CDN framework allows a set of ι parties to securely compute the output of a protocol π (implementing functionality $\mathcal{F}$) in the presence of a malicious adversary who controls at most a minority of $\tau - 1 = \lceil \iota/2 \rceil - 1$ parties. Security is defined according to Definition 4.5 with the restriction that $|C| < \lceil \iota/2 \rceil$ and the extension that each party is allowed to receive a public input and/or output (see [27]). Requiring a majority of honest parties assures that a protocol execution always succeeds with the correct computation result (for details see [27]). The framework assumes a threshold homomorphic cryptosystem satisfying a set of specific properties. In this thesis, we focus on the threshold variant of the Paillier cryptosystem (see Sections 3.2.1 and 3.2.2) which satisfies the required properties. In the CDN framework, each party has to prove that it follows the protocol specification by making use of Σ-protocols (see Section 3.3.1). Once a corrupted party is found to not follow the protocol specification it is excluded from the further protocol execution and the remaining parties can continue since they know the encrypted input of the excluded party. Note that we will not explicitly address this aspect for our novel gates and protocols designed within the CDN framework.

At the core of the CDN framework is a universal protocol $\mathsf{FuncEval}_{\mathcal{F}}$ which expects the description of an arithmetic circuit composed of some specified basic gates for evaluating functionality $\mathcal{F}$. In [27,28] some basic gates have been proposed. One of these gates which we use in the context of this work is a multiplication gate $(\llbracket a \cdot b \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket a \rrbracket, \llbracket b \rrbracket))$ which enables the set of participating parties to compute the encrypted product $\llbracket a \cdot b \rrbracket$ from two ciphertexts $\llbracket a \rrbracket$ and $\llbracket b \rrbracket$.

In a simplified form, protocol $\mathsf{FuncEval}_{\mathcal{F}}$ can be divided into three phases which are sketched in the following. For simplicity, we omit pre-computation steps such as key distribution (for details see [27]).

Input Phase: Each party encrypts its private input and broadcasts the corresponding ciphertext(s). Additionally, for each broadcasted ciphertext a party runs the POPK protocol (see Section 3.3.2) in order to prove that it knows the corresponding plaintext.

Computation Phase: The parties jointly evaluate the given circuit gate by gate. Each gate obtains an encrypted input and provides an encrypted output which in turn may constitute the input for another gate.

Output Phase: All parties who are still participating in the protocol execution jointly execute a private decryption protocol providing each party with its prescribed output.

Theorem 4.3. ([28].) *The* $\mathsf{FuncEval}_{\mathcal{F}}$ *protocol* (*as sketched above*) *securely evaluates* $\mathcal{F}$ *in the presence of static active adversaries corrupting a minority of parties in the malicious model.*

The following remarks describe adaptions and relaxations of the CDN-Framework we use throughout this thesis.

Remark 4.3. In order to integrate a new gate into the CDN Framework, it is necessary to show that given the encrypted input and encrypted output of a gate, the gate can be simulated in such a way that the simulated view of all parties is statistically indistinguishable from the view of all parties in the real world. However, Schoenmakers et al. argue that it suffices to show that a gate can be simulated in a statistically indistinguishable way for input of a special form (see below) [55,65,106]. This is due to the fact that the proof of Theorem 4.3 in [27] is kept modular and uses an intermediary distribution YAD^b which is a function of an encrypted bit $[\![b]\!]$. The proof then shows that

$$\mathsf{IDEAL}_{\mathsf{S}}^{\mathcal{F}} \stackrel{p}{\equiv} \mathsf{YAD}^0 \stackrel{c}{\equiv} \mathsf{YAD}^1 \stackrel{s}{\equiv} \mathsf{REAL}_{\mathsf{A}}^{\pi},$$

where $\mathsf{IDEAL}_{\mathsf{S}}^{\mathcal{F}} := \{\mathsf{IDEAL}_{\mathsf{S}(a)}^{\mathcal{F}}(\overline{x}, k)\}_{\overline{x},k}$ and $\mathsf{REAL}_{\mathsf{A}}^{\pi} := \{\mathsf{REAL}_{\mathsf{A}(a)}^{\pi}(\overline{x}, k)\}_{\overline{x},k}$. A distinguisher for distributions $\mathsf{IDEAL}_{\mathsf{S}}^{\mathcal{F}}$ and $\mathsf{REAL}_{\mathsf{A}}^{\pi}$ yields a distinguisher for distributions YAD^0 and YAD^1. Since the gap between YAD^0 and YAD^1 depends on the single encrypted bit $[\![b]\!]$, the security of a gate is reduced to the security of the underlying cryptosystem. Thus, for a gate obtaining common input $[\![x]\!]$, it suffices to conduct the simulation for input $[\![\tilde{x}]\!] = [\![(1-b)x^{\langle 0\rangle} + bx^{\langle 1\rangle}]\!]$ where $x^{\langle 0\rangle}$ and $x^{\langle 1\rangle}$ represent x in the YAD^0 and YAD^1 distributions, respectively. Values $x^{\langle 0\rangle}$, $x^{\langle 1\rangle}$, and $[\![b]\!]$ are given to the simulator. For further details see [55,65,106].

Remark 4.4. In [106] it is argued that the CDN framework can be extended to support a dishonest majority (and thus in particular to support the two-party case) at the loss of

robustness (it can no longer be guaranteed that the protocol execution succeeds). In other words, a corrupted party cannot be prevented from prematurely aborting the protocol execution. However, there exist techniques to assure that a party does not gain any advantage from premature abortion [105].

Remark 4.5. Note that the CDN framework assumes that $\mathcal{F}$ is a deterministic functionality. Nevertheless, the framework can also handle probabilistic functionalities as described in [27]. In that case it has to be assured that the random bits influencing the protocol output are jointly generated by the participating parties and remain encrypted throughout the protocol execution.

Remark 4.6. Note that for a gate obtaining symmetric input, i.e., all parties provide the same encrypted input, no additional Σ-protocols have to be used to enforce that all parties provide the correct input. In case that an honest majority can be guaranteed, the correct input will be in the majority. Those parties whose input deviates from the correct input are excluded from the further protocol execution. If an honest majority cannot be guaranteed, then the protocol execution aborts in the case that the parties' inputs differ.

4.5 Miscellaneous

4.5.1 Set-Up Assumptions

In this thesis, we make the following set-up assumptions

- *Pre-distributed Key Material*: Since we make use of a partially homomorphic cryptosystem (the Paillier threshold variant presented in [50]) as an underlying technique for designing SMPC protocols, the question arises how the parties participating in a protocol execution obtain the required key material. Since there are various protocols for a distributed key generation for threshold Paillier (e.g., [69,91]), we do not explicitly address this question in this thesis. Instead, we concentrate on the protocol design and assume the key material to be pre-distributed as it is common in the literature. This also applies for gate/protocols which combine two-party as well as multi-party computations and thus require multiple Paillier keys for different threshold values.

- *Parties can reach each other*: Throughout this thesis, we assume that all parties that wish to jointly execute an SMPC protocol know how to communicate with each other without revealing their identities (e.g., user names or IP addresses). In the literature, there exist various techniques to achieve this requirement (e.g., one-time identifiers [104], Tor [114]).

4.5.2 Threshold Paillier Dependencies

For all gates and protocols we assume a threshold variant of Paillier as the underlying cryptosystem. However, all novel gates and protocols presented in this thesis are independent of (threshold) Paillier but only require a homomorphic threshold encryption scheme providing for homomorphic addition, homomorphic subtraction, homomorphic scalar multiplication, and ciphertext blinding. Note that the other homomorphic properties presented in Section 3.2.3 can be deduced from these four essential properties.

4.5.3 Proving the Security of Gates and Protocols

In the following, we provide some general procedures for proving the security of a protocol or a gate in order to facilitate the understanding of the security proofs that are presented in the following chapters.

Our security proofs implicitly use the property of IND-MULT-CPA security (see Definition 3.18) of threshold Paillier. In particular, this property allows the simulation of multiple (possibly interdependent) ciphertext that are communicated during a protocol execution by ciphertexts that are chosen uniformly at random from the corresponding ciphertext space. This in turn allows us to concentrate on the critical parts of security proofs when plaintexts (e.g., the decryption of jointly generated ciphertexts) have to be simulated.

The presented definitions of security (Definition 4.1, Definition 4.2, and Definition 4.5) also cover the correctness of a protocol because it is postulated that the distribution of the outputs specified by a functionality is computationally indistinguishable from the distribution of the outputs of the protocol implementing this functionality. As it is commonly done in the literature, we separate the actual proof of correctness (i.e., the proof that the protocol computes what it was designed for) from the security proof. Then, in the security proof the correctness of the protocol under consideration is implicitly used. This practice allows the conduction of compact and simulation focused security proofs. We proceed similarly for proving the security of a gate, however, there are some particularities connected with gates as discussed in Remark 4.1 (Section 4.3.1) and Remark 4.3 (Section 4.4.3).

When we say that a simulator S_1 *calls* another simulator S_2, we mean that S_1 performs the steps which are specified for S_2 (on appropriate input and output).

4.5.4 Gate/Protocol Notation

In order to refer to a gate/protocol functionality or to a concrete gate/protocol, we use the templates $\mathcal{G}^{\mathsf{P}}_{\mathsf{Name}}$, $\mathcal{F}^{\mathsf{P}}_{\mathsf{Name}}$, $\rho^{\mathsf{A\text{-}P}}_{\mathsf{Name}}$, and $\pi^{\mathsf{A\text{-}P}}_{\mathsf{Name}}$ where Name specifies the name of the gate

(functionality) and protocol (functionality), respectively. Furthermore, $\mathsf{P} \in \{\mathsf{T}, \mathsf{M}\}$ indicates the underlying party case which is either the two-party case (T) or the multi-party case (M) and $\mathsf{A} \in \{\mathsf{SH}, \mathsf{M}\}$ indicates the underlying adversary model which is either the semi-honest model (SH) or the malicious model (M). Note that a gate functionality and a protocol functionality are indpendent of the adversary model, thus, the corresponding templates do not specify an adversary model A in the superscript. We omit the superscripts in case they are apparent from the context.

4.5.5 Complexity Measures

For determining the theoretical complexity of an SMPC gate/protocol, one generally considers the *communication complexity* and the *round complexity*.

- The communication complexity counts the number of messages and their sizes which are communicated during a gate/protocol execution while a broadcast message is only counted once.
- The round complexity measures the number of sequential communication steps (i.e., a gate/protocol step in which a party sends a message) where all steps that can be performed simultaneously are collected to one single step.

In order to refer to the communication complexity or round complexity of a gate ρ_{Name} or protocol π_{Name} we write O_{Name}. Which of the two complexities is meant will be apparent from the context. Furthermore, we add a superscript in order to indicate the magnitude of a specific parameter determining the complexity of the considered gate/protocol. Consider, for example, a gate ρ_{Y} with communication complexity in $O_{\mathsf{Y}} := \mathcal{O}(y)$. This gate is used as a building block in gate ρ_{X}. The communication complexity of gate ρ_{X} solely depends on the complexity of ρ_{Y} and the magnitude of y corresponds to a parameter x which occurs in ρ_{X}. Then, the communication complexity of ρ_{X} is written as $O_{\mathsf{X}} := \mathcal{O}(O_{\mathsf{Y}}^{y \in \mathcal{O}(x)})$.

CHAPTER 5

Privacy-Preserving Building Blocks

In this chapter, we introduce novel gates and review existing gates which previously have been proposed in the literature. These gates are used as building blocks for designing new gates or for designing one of the novel bartering protocols and their extensions (see Chapters 6 and 7). The SMPC setting (i.e., the party case and the adversary model) for which a gate is designed depends on the context in which it is used. For example, this means that if we need a specific gate only for the two-party case and to provide security in the semi-honest model, we present this gate only for this setting. However, in case that we require a gate functionality independently of a specific setting, we (at least) present a gate for the multi-party case and the malicious model since it also covers the two-party case and, in general, it is straight-forward to extract a gate providing security in the semi-honest model by performing minor modifications (such as removing the used zero knowledge proofs).

The gates presented in this chapter can be divided into three categories: secure (1) basic operations, (2) comparison operations, and (3) selection operations. For each of the three categories, we first introduce the gate functionalities followed by presenting one or more gates implementing each of these functionalities. In case that a gate functionality has been implemented before, we either review existing gates (resp., a single gate if, to the best of our knowledge, there exist no further, more efficient, gates) implementing that functionality or devise a new gate that corrects privacy-leakages of existing gates. In case that we introduce a gate functionality that, to the best of our knowledge, has not been implemented before, we devise a gate (resp., multiple gates for different SMPC settings) implementing that functionality. In the latter case, we provide an overview of related work when it is not obvious why existing (similar) gates cannot be directly used instead.

Since the gates in this chapter are presented according to their general functional-

Category	Gate				Security Model	Party Case
	Operation	Symbol	Func.-Def.	Gate Spec.		
Basic	Multiplication	$\rho_{\text{Mult}}^{\text{SH-M}}$	Def. 5.1	Spec. 1	Semi-Honest	Multi
		$\rho_{\text{Mult}}^{\text{M-M}}$	Def. 5.1	Spec. 1	Malicious	Multi
	Bit-Decomp.	$\rho_{\text{BD}}^{\text{M-M}}$	Def. 5.2	Spec. 2	Malicious	Multi
	Modulo	$\rho_{\text{Mod}}^{\text{M-M}}$	Def. 5.3	Spec. 3	Malicious	Multi
Comparison	Less-Than	$\underline{\rho_{\text{LT-Bin-SO}}^{\text{SH-M}}}$	Def. 5.4	Spec. 4	Semi-Honest	Multi
		$\underline{\rho_{\text{LT-SO}}^{\text{SH-M}}}$	Def. 5.5	Spec. 5	Semi-Honest	Multi
		$\rho_{\text{BDI-LT}}^{\text{M-M}}$	Def. 5.6	Spec. 6	Malicious	Multi
		$\rho_{\text{IS-LT-SCOT}}^{\text{SH-T}}$	Def. 5.7	Spec. 7	Semi-Honest	Two
	Equality	$\rho_{\text{ET}}^{\text{SH-M}}$	Def. 5.8	Spec. 8	Semi-Honest	Multi
		$\rho_{\text{ET}}^{\text{M-M}}$	Def. 5.8	Spec. 8	Malicious	Multi
Selection	from Dataset	$\underline{\rho_{\text{ES}}^{\text{M-M}}}$	Def. 5.10	Spec. 9	Malicious	Multi
		$\underline{\rho_{\text{CRS-}i^*}^{\text{SH-M}}}$	Def. 5.11	Spec. 10	Semi-Honest	Multi
		$\underline{\rho_{\text{CRS-}i^*}^{\text{M-M}}}$	Def. 5.11	Spec. 11	Malicious	Multi
		$\underline{\rho_{\text{CRS-C-}i^*}^{\text{SH-M}}}$	Def. 5.11	Spec. 12	Semi-Honest	Multi
	from Interval	$\rho_{\text{RBVG}}^{\text{M-M}}$	Def. 5.12	Spec. 13	Malicious	Multi
		$\underline{\rho_{\text{RSI-}\omega}^{\text{SH-T}}}$	Def. 5.13	Spec. 14	Semi-Honest	Two
		$\underline{\rho_{\text{RIE}}^{\text{M-M}}}$	Def. 5.14	Spec. 15	Malicious	Multi
		$\underline{\rho_{\text{RIE-}\mathcal{D}}^{\text{M-M}}}$	Def. 5.15	Spec. 16	Malicious	Multi
		$\underline{\rho_{\text{IS-}\vartheta}^{\text{M-M}}}$	Def. 5.16	Spec. 17	Malicious	Multi

Table 5.1: Overview of the presented gates. An underlined gate symbol indicates a newly proposed gate; an underlined definition indicates a newly proposed gate functionality.

ity rather than their purpose in the context of a privacy-preserving bartering protocol, this chapter ought to be seen as reference for Chapters 6-7. An overview of the presented gates, including the SMPC setting they are designed for, is given in Table 5.1. An overview of the communication and the round complexities of the presented gates is given at the end of this chapter in Table 5.6.

Contributions: The following list provides an overview of the novel gates that are presented in this chapter.

- A two-party gate secure in the semi-honest model for performing a less than com-

parison on two encrypted bits that provides XOR shared output. This gate has been published in [119].

- A multi-party gate secure in the semi-honest model for performing a less than comparison on two encrypted integers that provides XOR shared output. This contribution is joint work with Fabian Förg (PhD Student at Stevens Institute of Technology at that time).

- A multi-party gate secure in the malicious model for obliviously selecting one encrypted entry out of a list of ciphertexts. This gate is joint work with Wadim Pessin (graduate student at RWTH Aachen University at that time) and has been published in [124].

- Several multi-party gates that securely implement the primitive of conditional random selection (and variants) that, depending on the gate, provide security in the semi-honest model or the malicious model. An initial two-party gate (not presented in this thesis) providing security in the semi-honest model is joint work with Fabian Förg and has been published in [119]. A multi-party variant providing security in the malicious model has been published in [122].

- Two-party and multi-party gates for sampling from a private (unknown) interval that is given by encrypted bounds that, depending on the gate, provide security in the semi-honest model and the malicious model. The most important gate allows multiple parties to sample from a private interval according to an arbitrary discrete distribution and provides security in the malicious model. The gates have been published in [48,121,122,124].

- A multi-party gate secure in the malicious model for the oblivious shrinking of a private (unknown) interval given by encrypted bounds in case that the interval width is beyond a given public threshold. This gate has been published in [121, 122].

Outline: In Section 5.1, we present several secure basic gates, e.g., allowing for the multiplication of two encrypted integers. Gates which implement secure comparison operations like testing the equality of two encrypted integers are introduced in Section 5.2. Finally, in Section 5.3 a series of gates implementing various secure selection operations are presented.

5.1 Basic Operations

5.1.1 Multiplication

A multiplication gate is used to allow a set of parties to securely compute $[\![xy]\!]$ from $[\![x]\!]$ and $[\![y]\!]$ while keeping x and y private.

Definition 5.1 ($\mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}$: Multiplication)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![x]\!]$ and $[\![y]\!]$. Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}$ is given by $([\![x \cdot y]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}(([\![x]\!], [\![y]\!]))$.*

Based on $\mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}$, we define $([\![x_1 \cdot \ldots \cdot x_n]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{UFI-Mult}}(([\![x_1]\!], \ldots, [\![x_n]\!]))$ (Unbounded Fan In Multiplication) to be a shorthand notation for computing $[\![x_1 \cdot \ldots \cdot x_n]\!]$ by means of subsequently executing an implementation of $\mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}$ $n-1$ times:

$$\begin{aligned}([\![y_1]\!]) &\leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}(([\![x_1]\!], [\![x_2]\!])),\\ ([\![y_2]\!]) &\leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}(([\![y_1]\!], [\![x_3]\!])),\\ &\ldots,\\ ([\![y_{n-1}]\!]) &\leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}(([\![y_{n-2}]\!], [\![x_n]\!])).\end{aligned}$$

Furthermore, let $(([\![x_1]\!], [\![x_1 \cdot x_2]\!], \ldots, [\![x_1 \cdot \ldots \cdot x_n]\!])) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{S-Mult}}(([\![x_1]\!], \ldots, [\![x_n]\!]))$ (Staggered Multiplication) denote a call of $\mathcal{G}^{\mathrm{M}}_{\text{UFI-Mult}}$ on input $[\![x_1]\!], \ldots, [\![x_n]\!]$ where all intermediate computation results (including $[\![x_1]\!]$) are added to the output. We write $([\![x_1 \cdot \ldots \cdot x_i]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{S-Mult-i}}(([\![x_1]\!], \ldots, [\![x_n]\!]))$ with $i \in \mathbb{N}^0_n$ to address the i-th component of the output of $\mathcal{G}^{\mathrm{M}}_{\text{S-Mult}}$. For technical reasons, we define $([\![1]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{S-Mult-0}}(([\![x_1]\!], \ldots, [\![x_n]\!]))$.

Gate Specification 1 ($\rho^{\text{M-M}}_{\mathrm{Mult}}$ and $\rho^{\text{SH-M}}_{\mathrm{Mult}}$)**.** A gate $\rho^{\text{M-M}}_{\mathrm{Mult}}$ implementing $\mathcal{G}^{\mathrm{M}}_{\mathrm{Mult}}$ providing security against malicious adversaries has been proposed in [27,28]. A gate $\rho^{\text{SH-M}}_{\mathrm{Mult}}$ providing security against semi-honest adversaries can be implemented analogously by omitting the ZKPOKs.

Complexity. The communication complexity of the multiplication gate from [27,28] is in $\mathcal{O}(\iota s)$ and the round complexity is in $\mathcal{O}(1)$.[1]

5.1.2 Bit-Decomposition

Bit-decomposition denotes an operation that converts an unsigned m-bit integer x into its binary representation $x_0, \ldots, x_{m-1}$. In the following, we present the definition of the gate functionality for a secure bit-decomposition of Paillier ciphertexts which allows the conversion of an encrypted integer $[\![x]\!]$ into the corresponding encrypted bit representation such that x as well as the actual bit length of x remains private.

[1] Recall that ι refers to the number of all parties participating in a protocol execution and s refers to the security parameter of Paillier.

Definition 5.2 ($\mathcal{G}_{\text{BD}}^{\text{M}}$: Bit-Decomposition)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![x]\!]$. Then, gate functionality $\mathcal{G}_{\text{BD}}^{\text{M}}$ is given by $(([\![x_0]\!], \ldots, [\![x_{m-1}]\!])) \leftarrow \mathcal{G}_{\text{BD}}^{\text{M}}([\![x]\!])$ where $x_{bin} := (x_0, \ldots, x_{m-1})$ is the bit representation of x with $0 \leq x < 2^m$.*

Secure bit-decomposition is an operation of practical importance in the context of designing efficient SMPC protocols. This is due to the fact that depending on the gate functionality either an integer or a binary arithmetic-oriented implementation might be more efficient. When designing complex SMPC protocols consisting of various different gates, it is possible to benefit from both worlds by using a gate implementing the bit-decomposition functionality [101]. Note that a secure bit-composition, i.e., computing $[\![x]\!]$ from $[\![x_0]\!], \ldots, [\![x_{m-1}]\!]$, can be computed as follows:

$$[\![x]\!] := [\![x_0]\!] +_h 2^1 \times_h [\![x_1]\!] +_h \ldots +_h 2^{m-1} \times_h [\![x_{m-1}]\!].$$

Gate Specification 2 ($\rho_{\text{BD}}^{\text{M-M}}$)**.** A gate $\rho_{\text{BD}}^{\text{M-M}}$ implementing $\mathcal{G}_{\text{BD}}^{\text{M}}$ providing security against malicious adversaries has been introduced in [101].

Complexity. The communication complexity of the bit-decomposition gate from [101] is in $\mathcal{O}(m \cdot O_{\text{Mult}})$ and has a round complexity in $\mathcal{O}(1)$ assuming that the required secure multiplication operations run in parallel.

5.1.3 Modulo

The following gate functionality defines a modulo reduction operation allowing multiple parties to compute $[\![x \ mod \ b]\!]$ for a given ciphertext $[\![x]\!]$ (of an unknown plaintext x) and a public modulus b.

Definition 5.3 ($\mathcal{G}_{\text{Mod}}^{\text{M}}$: Modulo Reduction)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold a ciphertext $[\![x]\!]$ with $|x| = m_x$ and a public modulus b with $2^{m_b - 1} < b \leq 2^{m_b}$ and $m_b \leq m_x$. Then, gate functionality $\mathcal{G}_{\text{Mod}}^{\text{M}}$ is given by $([\![x \ mod \ b]\!]) \leftarrow \mathcal{G}_{\text{Mod}}^{\text{M}}(([\![x]\!], b))$.*

Gate Specification 3 ($\rho_{\text{Mod}}^{\text{M-M}}$)**.** A gate $\rho_{\text{Mod}}^{\text{M-M}}$ implementing gate functionality $\mathcal{G}_{\text{Mod}}^{\text{M}}$ has been introduced in [65]. The security of $\rho_{\text{Mod}}^{\text{M-M}}$ against a malicious adversary has been proven in the context of the CDN framework.

Complexity. The communication and round complexities of gate $\rho_{\text{Mod}}^{\text{M-M}}$ from [65] are in $\mathcal{O}(\iota s m_b)$ and $\mathcal{O}(\iota)$, respectively.

5.2 Comparison Operations

5.2.1 Less Than Comparison

In the following, we present four gate functionalities for computing different flavors of secure less than comparisons. The first two gate functionalities assume the input of the bits (resp., integers) to be compared are only available in encrypted form and provide XOR shared (see Section 3.1.6.1) output. The third gate functionality assumes that the integers to be compared are only available in a bit decomposed encrypted form and provides the comparison result in encrypted form. The fourth gate functionality defines a special kind of an oblivious transfer protocol where the secret the receiver learns depends on the computation result of a secure less than comparison.

Note that it depends on the SMPC setting and the protocol/gate that requires a secure less than comparison, which of the gates implementing one of the following gate functionalities is more suitable.

Definition 5.4 ($\mathcal{G}^{\mathrm{M}}_{\text{LT-Bin-SO}}$: Less Than Comparison for Binary Values with Shared Output)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![b]\!]$ and $[\![b']\!]$ with $b, b' \in \{0,1\}$. Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\text{LT-Bin-SO}}$ is given by $(o_1, \ldots, o_\iota) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{LT-Bin-SO}}(([\![b]\!], [\![b']\!]))$ with $o_1, \ldots, o_{\iota-1} \leftarrow_{\$} \{0,1\}$ s.t. $o_1 \oplus o_2 \oplus \ldots \oplus o_\iota = [b < b']$.*

Definition 5.5 ($\mathcal{G}^{\mathrm{M}}_{\text{LT-SO}}$: Less Than Comparison with Shared Output)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![x]\!]$ and $[\![y]\!]$ with $x, y \in \mathbb{P}$. Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\text{LT-SO}}$ is given by $(o_1, \ldots, o_\iota) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{LT-SO}}(([\![x]\!], [\![y]\!]))$ with $o_1, \ldots, o_{\iota-1} \leftarrow_{\$} \{0,1\}$ s.t. $o_1 \oplus o_2 \oplus \ldots \oplus o_\iota = [x < y]$.*

Definition 5.6 ($\mathcal{G}^{\mathrm{M}}_{\text{BDI-LT}}$: Bit-Decomposed Input Less Than Comparison)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![x_{bin}]\!] := ([\![x_0]\!], \ldots, [\![x_{m-1}]\!])$ and $[\![y_{bin}]\!] := ([\![y_0]\!], \ldots, [\![y_{m-1}]\!])$ where x_{bin} (resp., y_{bin}) is the binary representation of x with $0 \leq x < 2^m$ (resp., of y with $0 < y \leq 2^m$). Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\text{BDI-LT}}$ is given by $([\![b]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{BDI-LT}}(([\![x_{bin}]\!], [\![y_{bin}]\!]))$ where $b = [x < y]$.*

Definition 5.7 ($\mathcal{G}^{\mathrm{T}}_{\text{IS-LT-SCOT}}$: Input-Symmetric Less Than Strong Conditional Oblivious Transfer)**.** *Let party P_1 hold integer x_1 and $[\![v_1]\!]$ and let party P_2 hold integer x_2 and $[\![v_2]\!]$. Then, gate functionality $\mathcal{G}^{\mathrm{T}}_{\text{IS-LT-SCOT}}$ is given by*

$$\left.\begin{array}{ll} (\mathsf{Blind}([\![v_1]\!]), \perp) & \textit{if } x_1 < x_2 \\ (\mathsf{Blind}([\![v_2]\!]), \perp) & \textit{otherwise} \end{array}\right\} \leftarrow \mathcal{G}^{\mathrm{T}}_{\text{IS-LT-SCOT}}((x_1, [\![v_1]\!]), (x_2, [\![v_2]\!])),$$

where $\perp$ denotes the empty string.

Given a gate implementing one of the less-than based comparison operations from above (Definitions 5.4-5.7) for a specific SMPC setting (i.e., party case and adversary

Gate 5.1. Specification of $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$.

1 All parties jointly compute $(\llbracket c \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket 1 \rrbracket -_h \llbracket b \rrbracket, \llbracket b' \rrbracket))$

2 All parties set $\llbracket c_0 \rrbracket := \llbracket c \rrbracket$

3 Each party P_ℓ (ℓ from 1 to $\iota - 1$):

3.1 Select $o_\ell \leftarrow_\$ \{0, 1\}$

3.2 Set $\llbracket c_\ell \rrbracket := \begin{cases} \mathsf{Blind}(\llbracket c_{\ell-1} \rrbracket) & \text{if } o_\ell = 0 \\ \llbracket 1 \rrbracket -_h \llbracket c_{\ell-1} \rrbracket & \text{otherwise} \end{cases}$

3.3 Send $\llbracket c_\ell \rrbracket$ to party $P_{\ell+1}$

4 All parties jointly compute $o_\iota = \mathsf{Dec}(\llbracket c_{\iota-1} \rrbracket)$ s.t. only P_ι learns the result

5 Each party P_ℓ outputs o_ℓ ($\forall \ell \in \mathscr{P}$)

model) it is straight-forward to implement the corresponding less-than or equal (LTE), greater-than (GT), and greater-than or equal (GTE) variants (see, e.g., [118]). For example, a gate implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{LT-SO}}$ can be used to implement the LTE, GT, and GTE variants by defining $\mathcal{G}^{\text{M}}_{\text{LTE-SO}}((\llbracket x \rrbracket, \llbracket y \rrbracket)) := \mathcal{G}^{\text{M}}_{\text{LT-SO}}((\llbracket x \rrbracket, \llbracket y \rrbracket +_h \llbracket 1 \rrbracket))$, $\mathcal{G}^{\text{M}}_{\text{GT-SO}}((\llbracket x \rrbracket, \llbracket y \rrbracket)) := \mathcal{G}^{\text{M}}_{\text{LT-SO}}((\llbracket y \rrbracket, \llbracket x \rrbracket))$, and $\mathcal{G}^{\text{M}}_{\text{GTE-SO}}((\llbracket x \rrbracket, \llbracket y \rrbracket)) := \mathcal{G}^{\text{M}}_{\text{LT-SO}}((\llbracket y \rrbracket, \llbracket x \rrbracket +_h \llbracket 1 \rrbracket))$. Furthermore, if a gate $\rho_{\text{LT-SO}}$ implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{LT-SO}}$ is required to provide the computation result as a single encrypted bit, a gate ρ_{LT} can be derived from $\rho_{\text{LT-SO}}$ by defining $\rho_{\text{LT}}(\llbracket x \rrbracket, \llbracket y \rrbracket)$ as $(\llbracket o_1 \oplus o_2 \oplus \ldots \oplus o_\iota \rrbracket) \leftarrow \rho_{\text{LT-SO}}(\llbracket x \rrbracket, \llbracket y \rrbracket)$.

Gate Specification 4 ($\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$)**.** In the following, we devise a gate $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{LT-Bin-SO}}$ (see Definition 5.4).

As the first step of $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ (see Gate 5.1), all parties jointly compute $\llbracket c \rrbracket \leftarrow \rho_{\text{Mult}}(\llbracket 1 \rrbracket -_h \llbracket b \rrbracket, \llbracket b' \rrbracket)$ such that c indicates whether or not $b < b'$. The remaining steps are intended to share c between the parties such that all parties have to aggregate their shares in order to obtain the comparison result. Next, all parties set $\llbracket c_0 \rrbracket := \llbracket c \rrbracket$. Parties $P_1, \ldots, P_{\iota-1}$ successively compute $\llbracket c_\ell \rrbracket$ from $\llbracket c_{\ell-1} \rrbracket$ where P_ℓ ($\forall \ell \in \mathscr{P} \backslash \{\iota\}$) first selects $o_\ell \leftarrow_\$ \{0, 1\}$, computes $\llbracket c_\ell \rrbracket := \mathsf{Blind}(\llbracket c_{\ell-1} \rrbracket)$ if $o_\ell = 0$ and $\llbracket c_\ell \rrbracket := \llbracket 1 \rrbracket -_h \llbracket c_{\ell-1} \rrbracket$ otherwise, and sends $\llbracket c_\ell \rrbracket$ to $P_{\ell+1}$. Party P_1 initiates this staggered computation. Subsequently, all parties jointly decrypt $\llbracket c_{\iota-1} \rrbracket$ such that only P_ι learns the result $o_\iota = \mathsf{Dec}(\llbracket c_{\iota-1} \rrbracket)$. Finally, each party P_ℓ outputs o_ℓ ($\ell \in \mathscr{P}$).

The correctness of gate $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ follows directly from Table 5.2 which shows all possible combinations of $(b, b') \in \{0, 1\} \times \{0, 1\}$ and the random bits $o_1, \ldots, o_{\iota-1}$ selected

c	b	b'	o'	o_ι	$o' \oplus o_\iota$
0	0	0	0	0	0
1	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	0
0	0	0	1	1	0
1	0	1	1	0	1
0	1	0	1	1	0
0	1	1	1	1	0

Table 5.2: Truth table for $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ with $c := [b < b']$ and $o' := o_1 \oplus \ldots \oplus o_{\iota-1}$.

by $P_1, \ldots, P_{\iota-1}$, respectively, which are captured by $o' := o_1 \oplus \ldots \oplus o_{\iota-1}$. In particular, Table 5.2 shows that $o_1 \oplus \ldots \oplus o_\iota$ is equal to 1 if $b < b'$ and otherwise equal to 0—exactly as prescribed by Definition 5.4. Furthermore, the output distribution of $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ is as prescribed by Definition 5.4.

Theorem 5.1. *Let the input $(\llbracket b \rrbracket, \llbracket b' \rrbracket)$ and the partial output $(o_{i_1}, \ldots, o_{i_t})$ with $\{i_1, \ldots, i_t\} = C \subset \mathscr{P}$ and $t < \tau$ be given to simulator S. Gate $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ can be simulated such that the simulated view is computationally indistinguishable from the real view.*

Proof. First, assume that parties $P_1, \ldots, P_t$ are corrupted, i.e., $C = \{1, \ldots, t\}$. For all corrupted parties, simulator S sets $\langle \llbracket c \rrbracket \rangle \leftarrow_\$ \mathbb{C}$ and calls the simulator of ρ_{Mult} on $(\llbracket 1 \rrbracket -_h \llbracket b \rrbracket, \llbracket b' \rrbracket)$ and $\langle \llbracket c \rrbracket \rangle$. In order to simulate $\llbracket c_{\ell-1} \rrbracket$ which P_ℓ ($\ell \in C$) receives from $P_{\ell-1}$, simulator S computes Steps 3.1-3.3 (Gate 5.1) in place of $P_{\ell-1}$ (using $P_{\ell-1}$'s given output $o_{\ell-1}$) and sets $\langle \llbracket c_{\ell-1} \rrbracket \rangle$ accordingly.

In case that $(\ell - 1) \notin C$, simulator S simulates $\llbracket c_{\ell-1} \rrbracket$ which P_ℓ receives from $P_{\ell-1}$ as $\langle \llbracket c_{\ell-1} \rrbracket \rangle \leftarrow_\$ \mathbb{C}$. Furthermore, in case that $\iota \in C$, simulator S sets $\langle \mathsf{Dec}(\llbracket c_{\iota-1} \rrbracket) \rangle := o_\iota$. □

Complexity. The communication and round complexities of gate $\rho^{\text{SH-M}}_{\text{LT-Bin-SO}}$ are in $\mathcal{O}(\iota s + O_{\text{Mult}})$ and $\mathcal{O}(\iota)$, respectively.

Gate Specification 5 ($\rho^{\text{SH-M}}_{\text{LT-SO}}$). In the following, we introduce a new gate $\rho^{\text{SH-M}}_{\text{LT-SO}}$ implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{LT-SO}}$ presented in Definition 5.5. This gate is based on a variant of the two-party privacy-preserving less than comparison protocol from [76]. The protocol variant provides shared output and is secure in the semi-honest model [86]. To the best of our knowledge, there exists no previously presented gate implementing multi-party functionality $\mathcal{G}^{\text{M}}_{\text{LT-SO}}$.

The underlying approach of our novel gate (see Gate 5.2) is the same as in [76,86] which is based on the following observation. For $x, y \in \mathbb{N}$, it holds that $x - y < 0$ iff $[x < y] = 1$ and $y - x - 1 < 0$ iff $[x \geq y] = 1$.

In the following, the plaintext space of the used additively homomorphic threshold cryptosystem is associated with interval $[0, n)$ ($n := N$ in case of threshold Paillier). Analogously to [76,86], we require that $x, y \in [l, h] =: \mathbb{D} \subset [0, n)$, for some $l, h \in \mathbb{N}, h > l$. Furthermore, interval $[0, n)$ has to be split in order to represent negative integers: Zero as well as positive integers are represented by the lower half $[0, \lceil n/2 \rceil - 1]$ of $[0, n)$ while negative integers are represented by the upper half $[\lceil n/2 \rceil, n - 1]$ of $[0, n)$.

In $\rho^{\text{SH-M}}_{\text{LT-SO}}$, P_1 first obliviously computes the differences $[\![x-y]\!]$ and $[\![y-x-1]\!]$ on the encrypted input values $[\![x]\!]$, $[\![y]\!]$. In order to prevent the leakage of any information beyond the sign when one of those encrypted differences is decrypted (cf. Step 4, Gate 5.2), both differences are randomized by letting P_1 perform a homomorphic scalar multiplication on $[\![x - y]\!]$ (resp., $[\![y - x - 1]\!]$) with a random scalar $r \leftarrow_{\$} \mathbb{D}_r = [1, h_r := (h - l)^2] \subset [0, n)$ followed by homomorphically adding a random shift $r' \leftarrow_{\$} \mathbb{D}_{r'} := [0, r)$ as it is proposed in [76]:

$$[\![d_0]\!] := (([\![x]\!] -_h [\![y]\!]) \times_h r) +_h [\![r']\!]$$
$$[\![d'_0]\!] := (([\![y]\!] -_h [\![x]\!] -_h [\![1]\!]) \times_h r) +_h [\![r']\!].$$

As $d_0, d'_0 \in [0, n)$, it holds that $d_0 \in [\lceil n/2 \rceil, n - 1]$, $d'_0 \in [0, \lceil n/2 \rceil - 1]$ if $x < y$ and $d_0 \in [0, \lceil n/2 \rceil - 1]$, $d'_0 \in [\lceil n/2 \rceil, n - 1]$ otherwise. Next, parties $P_1, \ldots, P_{\iota-1}$ successively update $([\![d_0]\!], [\![d'_0]\!])$ where P_ℓ ($\forall \ell \in \mathscr{P} \backslash \{\iota\}$) selects $o_\ell \leftarrow_{\$} \{0, 1\}$ and swaps the entries of $([\![d_{\ell-1}]\!], [\![d'_{\ell-1}]\!])$ iff $o_\ell = 1$ (see Steps 2.1-2.2, Gate 5.2). Independent of the value o_ℓ, P_ℓ blinds the possibly swapped tuple entries and sends the resulting tuple $([\![d_\ell]\!], [\![d'_\ell]\!])$ to $P_{\ell+1}$. After P_ι receives $[\![d_{\iota-1}]\!]$, it sets $[\![d_\iota]\!] := [\![d_{\iota-1}]\!]$, broadcasts $[\![d_\iota]\!]$, and initiates the joint decryption of $[\![d_\iota]\!]$ such that only P_ι learns the result. Party P_ι sets o_ι according to

$$o_\iota := \begin{cases} 1 & \text{if } d_\iota \geq \lceil n/2 \rceil \\ 0 & \text{otherwise} \end{cases}.$$

Finally, each party P_ℓ ($\ell \in \mathscr{P}$) outputs o_ℓ.

For the two-party protocol in [76], the authors observe that their protocol has a leakage which also occurs in Gate 5.2: In case that $d_\iota = 0$, P_ι learns that $x = y$ (Step 4, Gate 5.2). According to [76], the probability of revealing that x is equal to y is given by

$$\epsilon = Pr[d_\iota = 0 | r \leftarrow_{\$} \mathbb{D}_r, r' \leftarrow_{\$} \mathbb{D}_{r'}] \approx \frac{ln\, h_r}{h_r}.$$

The authors of [76] argue that for a large h_r, probability ϵ is negligible and provide the example that for comparing two 160-bit integers, it holds that $\epsilon < 2^{-314}$.

Gate 5.2. Specification of $\rho^{\text{SH-M}}_{\text{LT-SO}}$.

1 Party P_1:

1.1 Select $r \leftarrow_{\$} \mathbb{D}_r$, $r' \leftarrow_{\$} \mathbb{D}_{r'}$

1.2 Compute $[\![d_0]\!] := (([\![x]\!] -_h [\![y]\!]) \times_h r) +_h [\![r']\!]$

1.3 Compute $[\![d'_0]\!] := (([\![y]\!] -_h [\![x]\!] -_h [\![1]\!]) \times_h r) +_h [\![r']\!]$

2 Party P_ℓ for ℓ from 1 to $\iota - 1$:

2.1 Select $o_\ell \leftarrow_{\$} \{0, 1\}$

2.2 Set $([\![d_\ell]\!], [\![d'_\ell]\!]) := \begin{cases} (\mathsf{Blind}([\![d_{\ell-1}]\!]), \mathsf{Blind}([\![d'_{\ell-1}]\!])) & \text{if } o_\ell = 0 \\ (\mathsf{Blind}([\![d'_{\ell-1}]\!]), \mathsf{Blind}([\![d_{\ell-1}]\!])) & \text{otherwise} \end{cases}$

2.3 Send $([\![d_\ell]\!], [\![d'_\ell]\!])$ to party $P_{\ell+1}$

3 Party P_ι:

3.1 Set $[\![d_\iota]\!] := [\![d_{\iota-1}]\!]$

3.2 Broadcast $[\![d_\iota]\!]$

4 All parties jointly compute $d_\iota = \mathsf{Dec}([\![d_\iota]\!])$ s.t. P_ι learns the result

5 Party P_ι sets $o_\iota := \begin{cases} 1 & \text{if } d_\iota \geq \lceil \frac{n}{2} \rceil \\ 0 & \text{otherwise} \end{cases}$

6 Party P_ℓ outputs o_ℓ ($\forall \ell \in \mathscr{P}$)

One further issue related with Gate 5.2 is that P_1 and P_ι may not be corrupted at the same time, otherwise, one of the differences $(x - y)$ or $(y - x - 1)$ can be computed by the adversary from the knowledge of r, r', and o_ι. Note that this restriction can be circumvented by letting all parties jointly compute the hidden differences $[\![d_0]\!]$ and $[\![d_1]\!]$ in an oblivious fashion.

Let $o' := o_1 \oplus \ldots \oplus o_{\iota-1}$. The correctness of gate $\rho^{\text{SH-M}}_{\text{LT-SO}}$ follows directly from Table 5.3 indicating that $o' \oplus o_\iota = 1$ if $x < y$ and equal to 0 otherwise. Furthermore, the output values $o_1, \ldots, o_{\iota-1}$ are uniformly distributed as prescribed by Definition 5.5.

Theorem 5.2. *Let input $([\![x]\!], [\![y]\!])$ and the partial output $(o_{i_1}, \ldots, o_{i_t})$ with $\{i_1, \ldots, i_t\} = C \subset \mathscr{P}$ and $t < \tau$ be given to simulator S. Provided that $\epsilon < negl(s)$ and P_1, P_ι do not collude, gate $\rho^{\text{SH-M}}_{\text{LT-SO}}$ can be simulated such that the simulated view is computational indistinguishable from the real view.*

$[x < y]$	o'	d_ι	o_ι	$o' \oplus o_\iota$
1	0	$d_\iota = d_0 \geq \lceil n/2 \rceil$	1	1
	1	$d_\iota = d'_0 < \lceil n/2 \rceil$	0	1
0	0	$d_\iota = d_0 < \lceil n/2 \rceil$	0	0
	1	$d_\iota = d'_0 \geq \lceil n/2 \rceil$	1	0

Table 5.3: Truth table for $\rho^{\text{SH-M}}_{\text{LT-SO}}$ with $o' := o_1 \oplus \ldots \oplus o_{\iota-1}$.

Proof. First, assume that parties $P_1, \ldots, P_t$ are corrupted, i.e., $C = \{1, \ldots, t\}$. Simulator S simulates $[\![d_0]\!]$ and $[\![d'_0]\!]$ by performing Step 1 (Gate 5.2) in place of P_1. In order to simulate $([\![d_{\ell-1}]\!], [\![d'_{\ell-1}]\!])$, which P_ℓ ($\ell \in C$) receives from $P_{\ell-1}$, simulator S computes Step 2.2 (Gate 5.2) (using $P_{\ell-1}$'s given output $o_{\ell-1}$) in place of $P_{\ell-1}$ and sets $(\langle [\![d_{\ell-1}]\!] \rangle, \langle [\![d'_{\ell-1}]\!] \rangle)$ accordingly. Furthermore, we have to distinguish the following cases:

1. $(\ell - 1) \notin C$: S simulates $([\![d_{\ell-1}]\!], [\![d'_{\ell-1}]\!])$ which P_ℓ receives from $P_{\ell-1}$ as $(\langle [\![d_{\ell-1}]\!] \rangle, \langle [\![d'_{\ell-1}]\!] \rangle) \leftarrow_\$ \mathbb{C} \times \mathbb{C}$.

2. $(\iota - 1) \notin C$: S simulates the ciphertext $[\![d_\iota]\!]$ broadcasted by P_ι in Step 3.2 by setting $\langle [\![d_\iota]\!] \rangle \leftarrow_\$ \mathbb{C}$.

3. $\iota \in C$: S has to distinguish whether $o_\iota = 1$ or $o_\iota = 0$ in order to simulate Step 4 (Gate 5.2). In the former case d_ι is simulated as $\langle d_\iota \rangle \leftarrow_\$ [\lceil n/2 \rceil, n - 1]$ while in the latter case $\langle d_\iota \rangle \leftarrow_\$ [0, \lceil n/2 \rceil - 1]$.

□

Complexity. The communication and round complexities of gate $\rho^{\text{SH-M}}_{\text{LT-SO}}$ are in $\mathcal{O}(\iota s)$ and $\mathcal{O}(\iota)$, respectively.

Gate Specification 6 ($\rho^{\text{M-M}}_{\text{BDI-LT}}$)**.** In [55], the authors propose a gate $\rho^{\text{M-M}}_{\text{BDI-GT}}$ implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{BDI-GT}}$ (cf. Definition 5.6). The security of $\rho^{\text{M-M}}_{\text{BDI-GT}}$ against malicious adversaries has been proven in the context of the CDN framework.

Complexity. The communication and round complexities of gate $\rho^{\text{M-M}}_{\text{BDI-GT}}$ are in $\mathcal{O}(\iota s m)$ and $\mathcal{O}(log(m))$, respectively.

Gate Specification 7 ($\rho^{\text{SH-T}}_{\text{IS-LT-SCOT}}$)**.** A gate $\rho^{\text{SH-T}}_{\text{IS-LT-SCOT}}$ implementing gate functionality $\mathcal{G}^{\text{T}}_{\text{IS-LT-SCOT}}$ (see Definition 5.7) has been proposed in [48,86]. Gate functionality $\mathcal{G}^{\text{T}}_{\text{IS-LT-SCOT}}$ describes a special kind of oblivious transfer referred to as *input-symmetric less than strong*

oblivious transfer [86] where each party P_ℓ ($\ell \in \{1,2\}$) holds a private integer x_ℓ and an encrypted secret $[\![v_\ell]\!]$, respectively. Depending on whether or not $x_1 < x_2$, P_1 learns its own (blinded) encrypted secret or the one provided by P_2. What is special about a gate implementing gate functionality $\mathcal{G}^{\text{T}}_{\text{IS-LT-SCOT}}$ is that no party learns the result of the less than comparison implying that P_1 cannot distinguish whether it obtains its own encrypted secret or the encrypted secret of P_2.

Complexity. The communication and round complexities of gate $\rho^{\text{SH-T}}_{\text{IS-LT-SCOT}}$ are in $\mathcal{O}(O_{\text{LT-SO}})$, respectively, assuming the use of the oblivious transfer protocol from [96].

5.2.2 Equality Test

An equality test gate can be used whenever a set of parties hold common values $[\![x]\!]$ and $[\![y]\!]$ and want to compute $[\![[x = y]]\!]$ while keeping x and y private.

Definition 5.8 ($\mathcal{G}^{\text{M}}_{\text{ET}}$: Equality Test)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![x]\!]$ and $[\![y]\!]$. Then, gate functionality $\mathcal{G}^{\text{M}}_{\text{ET}}$ is given by $([\![o]\!]) \leftarrow \mathcal{G}^{\text{M}}_{\text{ET}}(([\![x]\!], [\![y]\!]))$ where $o = [x = y]$.*

Definition 5.9 ($\mathcal{G}^{\text{M}}_{\text{BDI-ET}}$: Bit-Decomposed Input Equality Test)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold $[\![x_{bin}]\!] := ([\![x_0]\!], \ldots, [\![x_{m-1}]\!])$ and $[\![y_{bin}]\!] := ([\![y_0]\!], \ldots, [\![y_{m-1}]\!])$ where x_{bin} (resp., y_{bin}) is the binary representation of x with $0 \leq x < 2^m$ (resp., of y with $0 \leq y < 2^m$). Then gate functionality $\mathcal{G}^{\text{M}}_{\text{BDI-ET}}$ is given by $([\![b]\!]) \leftarrow \mathcal{G}^{\text{M}}_{\text{BDI-ET}}(([\![x_{bin}]\!], [\![y_{bin}]\!]))$ where $b = [x = y]$.*

Gate Specification 8 ($\rho_{\text{ET}}, \rho_{\text{BDI-ET}}$)**.** Given a gate implementing a less than comparison, it is straight-forward to implement an equality test. For example, a gate ρ_{ET} implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{ET}}$ can be derived from $\rho_{\text{LT-SO}}$ by defining $\rho_{\text{ET}}(([\![x]\!], [\![y]\!]))$ as $([\![o_1 \oplus o_2 \oplus \ldots \oplus o_\iota]\!]) \leftarrow \rho_{\text{LT-SO}}(([\![x]\!] -_h [\![y]\!], [\![1]\!]))$. A gate implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{BDI-ET}}$ can be computed as $([\![b]\!]) \leftarrow \rho_{\text{Mult}}(\rho_{\text{BDI-LTE}}(([\![x_{bin}]\!], [\![y_{bin}]\!])), \rho_{\text{BDI-GTE}}(([\![x_{bin}]\!], [\![y_{bin}]\!])))$. We use these constructions throughout the thesis when equality tests are needed because other existing gates are less efficient or do not provide the required input/output behavior.

Related Work: For the two-party case, a gate for equality testing which provides security against semi-honest adversaries (extendable to provide security against malicious adversaries) has been presented in [113]. This gate is based on a secure evaluation of the Jacobi symbol, supports encrypted input, and provides encrypted output as required by Definition 5.2.2. An inherent disadvantage of the proposed construction is that a false positive, i.e, the gate returns $[\![1]\!]$ in case of $x \neq y$, occurs with a probability of approximately $1/2$. Repeating the execution of the gate on the same inputs multiple times followed by multiplying the encrypted results reduces the probability of false positives

but can results in a less efficient solution compared to, e.g., equality testing gates with an error probability of zero.

In the literature, several protocols for equality testing have been implemented in which each party knows exactly one of the plaintext values which are to be compared and where at least one party learns the plaintext result of the equality test. In [46], Fagin et al. provide a comprehensive overview of several techniques for performing a secure equality test w.r.t. different privacy demands. In [73], the authors propose a two-party protocol (based on an additively homomorphic threshold cryptosystem) for equality testing providing security in the semi-honest model along with the necessary modifications to achieve security in the malicious model. The computation result is an encryption of 0 in case that $x = y$ and an encryption of a random number (drawn uniformly at random from the underlying plaintextspace) otherwise. Both parties jointly decrypt the computation result and output 1 if the result is equal to 0 and output 0 otherwise. None of these protocols can be used in the context of this thesis since we require an encrypted protocol output.

5.3 Selection Operations

5.3.1 Element Selection

Given a series of Paillier ciphertexts $[\![y_1]\!], \ldots, [\![y_n]\!]$, and $[\![w]\!]$ ($w \in \mathbb{N}_n$) of unknown plaintexts, we present a gate which allows the oblivious selection of $[\![y_w]\!]$. The corresponding gate functionality is defined as follows.

Definition 5.10 ($\mathcal{G}_{\text{ES}}^{\text{M}}$: Element Selection for Paillier Ciphertexts)**.** *Let $N = p \cdot q$ be the Paillier modulus and let all parties P_ℓ ($\ell \in \mathscr{P}$) hold encryptions $[\![y_1]\!], \ldots, [\![y_n]\!]$ and $[\![w]\!]$ with $n < min(p, q)$ and $w \in \mathbb{N}_n \subset \mathbb{P}$. Then, gate functionality $\mathcal{G}_{\text{ES}}^{\text{M}}$ is given by* $(\mathsf{Blind}([\![y_w]\!])) \leftarrow \mathcal{G}_{\text{ES}}^{\text{M}}(([\![y_1]\!], \ldots, [\![y_n]\!], [\![w]\!]))$.

Gate Specification 9 ($\rho_{\text{ES}}^{\text{M-M}}$)**.** The core idea of our novel gate $\rho_{\text{ES}}^{\text{M-M}}$ implementing $\mathcal{G}_{\text{ES}}^{\text{M}}$, is to construct a Lagrange like polynomial (cf. Definition 3.4, Section 3.1.3) $L(x) := \sum_{j=1}^{n} y_j l_{j,n}(x)$ over $\mathbb{Z}_N[x]$ (instead of a field $\mathbb{F}$) for the data points $(x_1, y_1), \ldots, (x_n, y_n) := (1, y_1), \ldots, (n, y_n)$. Consequently, the basis polynomials

$$l_{j,n}(x) := \prod_{\substack{k=1 \\ k \neq j}}^{n} (x - x_k) \cdot \underbrace{(x_j - x_k)^{-1}}_{\substack{\text{exists in } \mathbb{Z}_N \\ \text{since } n < min(p,q)}} \in \mathbb{Z}_N[x] \tag{5.1}$$

have a special form:

Gate 5.3. Specification of $\rho_{\text{ES}}^{\text{M-M}}$.

1 All parties jointly compute $((\llbracket w \rrbracket, \ldots, \llbracket w^{n-1} \rrbracket)) \leftarrow \rho_{\text{S-Mult}}((\llbracket w \rrbracket, \overset{n-1}{\ldots}, \llbracket w \rrbracket))$

2 Party P_ℓ ($\forall \ell \in \mathscr{P}$):

2.1 For j from 1 to n:

2.1.1 Locally compute $\llbracket l_{j,n}(w) \rrbracket$ by evaluating $l_{j,n}(\llbracket w \rrbracket)$ using homomorphic operations and the result from Step 1

3 For j from 1 to n:

3.1 All parties jointly compute $(\llbracket l_{j,n}(w) \cdot y_j \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket l_{j,n}(w) \rrbracket, \llbracket y_j \rrbracket))$

4 Party P_ℓ ($\forall \ell \in \mathscr{P}$):

4.1 Locally compute $\llbracket y_w \rrbracket := \llbracket l_{1,n}(w) \cdot y_1 \rrbracket +_h \ldots +_h \llbracket l_{n,n}(w) \cdot y_n \rrbracket$

4.2 Output $\llbracket y_w \rrbracket$

$$
\begin{aligned}
l_{j,n}(x) &= (-1)^{n-j}((j-1)!(n-j)!)^{-1} \prod_{\substack{k=1 \\ k \neq j}}^{n} (x-k) \\
&= (-1)^{n-j}((j-1)!(n-j)!)^{-1} \sum_{k'=0}^{n-1} a_{j,k'} x^{k'},
\end{aligned}
$$

with $a_{j,k'} \in \mathbb{Z}_N$. Intuitively, $\rho_{\text{ES}}^{\text{M-M}}$ works as follows (to simplify matters, we first explain $\rho_{\text{ES}}^{\text{M-M}}$ by just considering plaintext values). Given input $y_1, \ldots, y_n$ and $w \in \mathbb{N}_n$, the Lagrange like polynomial $L(x)$ is constructed for the data points $(1, y_1), \ldots, (n, y_n)$ and evaluated for $x = w$ which yields the desired result $L(w) = y_w$.

Now, turning to ciphertext values $\llbracket y_1 \rrbracket, \ldots, \llbracket y_n \rrbracket$ and $\llbracket w \rrbracket$, it is possible to compute an encryption $\llbracket L(w) \rrbracket$ of $L(w)$ by using

$$
\sum_{j=1}^{n}{}_h \, \rho_{\text{Mult}}((\llbracket y_j \rrbracket, \llbracket l_{j,n}(w) \rrbracket)), \text{ with}
$$

$$
\llbracket l_{j,n}(w) \rrbracket := (-1)^{n-j}((j-1)!(n-j)!)^{-1} \times_h \sum_{k=0}^{n-1}{}_h \, a_{j,k} \times_h \rho_{\text{S-Mult-}k}((\llbracket w \rrbracket, \overset{n-1}{\ldots}, \llbracket w \rrbracket)).
$$

Since only the encryptions of $y_1, \ldots, y_n$ and w are available to the parties, they have to interact in order to compute $\llbracket w^2 \rrbracket, \ldots, \llbracket w^{n-1} \rrbracket$ and $\llbracket y_j \cdot l_{j,n}(w) \rrbracket$ ($\forall j \in \mathbb{N}_n$). The re-

maining operations to evaluate $[\![L(w)]\!]$ can be computed locally by each party. Note that $(-1)^{-n+j}(j-1)!(n-j)!$ divides $\sum_{k'=0}^{n-1} a_{j,k'} w^{k'}$ in $\mathbb{Z}$.

Gate $\rho_{\text{ES}}^{\text{M-M}}$ proceeds as follows (see Gate 5.3): First, all parties jointly compute $\rho_{\text{S-Mult}}(([\![w]\!], \overset{n-1}{\ldots}, [\![w]\!]))$ in order to obtain $[\![w^2]\!], \ldots, [\![w^{n-1}]\!]$. Now, each party can locally compute $[\![l_{j,n}(w)]\!]$ ($\forall j \in \mathbb{N}_n$) as detailed above. Next, all parties jointly compute $\rho_{\text{Mult}}(([\![l_{j,n}(w)]\!], [\![y_j]\!]))$ ($\forall j \in \mathbb{N}_n$) whereupon each party locally computes $[\![L(w)]\!]$ which constitutes the common gate output.

From our assumption that $n < min(p, q)$ (for a given Paillier modulus $N = p \cdot q$) and from the facts that $\iota \ll min(p, q)$ and $x_i = i$ for $i \in \mathscr{P}$, it follows that $(x_j - x_k)$ (see Equation 5.1) has an inverse in $\mathbb{Z}_N$ and, thus, we can derive from the definition of Lagrange Interpolation (see Definition 3.4) that $[\![L(x_i)]\!] := [\![y_i]\!]$. Note that this derivation does not require $L(x)$ to be uniquely determined.

In the following, we show how $\rho_{\text{ES}}^{\text{M-M}}$ can be simulated in the CDN framework (see Section 4.4.3).

Theorem 5.3. *Let $w^{\langle 0\rangle}$, $w^{\langle 1\rangle}$, $y_j^{\langle 0\rangle}$, $y_j^{\langle 1\rangle}$ ($\forall j \in \mathbb{N}_n$), and an encryption $[\![b]\!]$ of $b \in \{0, 1\}$ be given to simulator S. For input $[\![\tilde{w}]\!] := [\![(1-b)\cdot w^{\langle 0\rangle} + b \cdot w^{\langle 1\rangle}]\!]$ and $[\![\tilde{y}_j]\!] := [\![(1-b)\cdot y_j^{\langle 0\rangle} + b \cdot y_j^{\langle 1\rangle}]\!]$ ($\forall j \in \mathbb{N}_n$), gate $\rho_{\text{ES}}^{\text{M-M}}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. To simulate Step 1 of Gate 5.3, simulator S calls the simulator of $\rho_{\text{S-Mult}}$ on input $([\![\tilde{w}]\!], \overset{n-1}{\ldots}, [\![\tilde{w}]\!])$ and output $([\![\tilde{w}^1]\!], [\![\tilde{w}^2]\!], \ldots, [\![\tilde{w}^{n-1}]\!])$ where $[\![\tilde{w}^i]\!]$ is computed as $[\![(1-b)(w^{\langle 0\rangle})^i + b \cdot (w^{\langle 1\rangle})^i]\!]$. For all $j \in \mathbb{N}_n$, simulator S computes $[\![l_{j,n}(\tilde{w})]\!] := [\![(1-b) \cdot l_{j,n}(w^{\langle 0\rangle}) + b \cdot l_{j,n}(w^{\langle 1\rangle})]\!]$ where $l_{j,n}(w^{\langle b\rangle}) = (-1)^{n-j}((j-1)!(n-j)!)^{-1} \prod_{k=1,k\neq j}^{n}(w^{\langle b\rangle} - k)$. To simulate Step 3 of Gate 5.3, simulator S calls the simulator of ρ_{Mult} on input $[\![l_{j,n}(\tilde{w})]\!], [\![\tilde{y}_j]\!]$ and output $[\![l_{j,n}(\tilde{w}) \cdot \tilde{y}_j]\!] := [\![(1-b) \cdot l_{j,n}(w^{\langle 0\rangle}) \cdot y_j^{\langle 0\rangle} + b \cdot l_{j,n}(w^{\langle 1\rangle}) \cdot y_j^{\langle 1\rangle}]\!]$ ($\forall j \in \mathbb{N}_n$). Finally, simulator S computes $[\![\tilde{y}_w]\!] = [\![(1-b) \cdot y_w^{\langle 0\rangle} + b \cdot y_w^{\langle 1\rangle}]\!]$ where $y_w^{\langle b\rangle} := y_1^{\langle b\rangle} \cdot l_{1,n}(w^{\langle b\rangle}) + \ldots + y_n^{\langle b\rangle} \cdot l_{n,n}(w^{\langle b\rangle})$. Since the simulated values are consistent with those of a real gate execution, the simulated view and the real view are statistically indistinguishable. □

Complexity. The communication and round complexities of gate $\rho_{\text{ES}}^{\text{M-M}}$ are in $\mathcal{O}(n \cdot O_{\text{Mult}})$, respectively.

Related Work: In [31], the authors propose a secure protocol for Lagrange interpolation over encrypted values which allows a prover to prove that a series of encrypted points $(1, \mathsf{Enc}(y_1)), \ldots, (n, \mathsf{Enc}(y_n))$ (encrypted with a threshold additively homomorphic encryption scheme) lie on a polynomial of low degree. The protocol requires that the randomization values used for the encryption of $\mathsf{Enc}(y_1), \ldots, \mathsf{Enc}(y_n)$ are chosen in a spe-

cific manner, which eventually requires the prover to know the corresponding plaintexts $y_1, \ldots, y_n$ and, thus, cannot be used to implement gate functionality $\mathcal{G}_{\text{ES}}^{\text{M}}$.

Note that the setting for $\mathcal{G}_{\text{ES}}^{\text{M}}$ given by Definition 5.10 is different from the setting that is required for an oblivious transfer where a sender holds n secrets and a receiver holds the index $w \in \mathbb{N}_n$ of the secret he is interested in. For $\mathcal{G}_{\text{ES}}^{\text{M}}$, $[\![y_1]\!], \ldots, [\![y_n]\!]$, and $[\![w]\!]$ result from previous private computations implying that no party knows any of the corresponding plaintexts. Consequently, an oblivious transfer protocol cannot be used to implement $\mathcal{G}_{\text{ES}}^{\text{M}}$. However, vice versa a gate implementing $\mathcal{G}_{\text{ES}}^{\text{M}}$ can be used (executed between two parties P_1 and P_2) to implement an oblivious transfer assuming that P_1 (sender) knows $y_1, \ldots, y_n$ and P_2 (receiver) knows w.

5.3.2 Conditional Random Selection

For a given private set of data records, the primitive of privacy-preserving conditional random selection (CRS) allows the random selection of one of the data records that satisfy a specified condition without leaking any information on any data entry. The corresponding gate functionality (see Definition 5.11) expects encrypted value vectors and an associated encrypted indicator vector as input. The entries of these vectors at a specific index represent a data record; multiple data records constitute a data table. A binary indicator vector can be used to indicate whether or not the data records satisfy a specified condition. By using an integer valued indicator vector, CRS allows for a prioritization of data records. Accordingly, an indicator vector entry indicates whether the corresponding data record satisfies the specified condition (if greater than zero) and indicates its priority (the higher the value, the higher the priority of the data record).

Definition 5.11 ($\mathcal{G}_{\text{CRS-}i^*}^{\text{M}}$ ($\mathcal{G}_{\text{CRS-C-}i^*}^{\text{M}}$): Conditional Random Selection (with Output Check))**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold m vectors $[\![L_i]\!] := ([\![l_{i,1}]\!], \ldots, [\![l_{i,n}]\!])$ of length n where $i \in \mathbb{N}_m$. Let $[\![L_{i^*}]\!]$ ($i^* \in \mathbb{N}_m$) be an encrypted indicator vector and $\{[\![L_1]\!], \ldots, [\![L_m]\!]\} \backslash [\![L_{i^*}]\!]$ be value vectors.*

1. *Then, functionality $\mathcal{G}_{\text{CRS-}i^*}^{\text{M}}$ is given by $(([\![l_1^*]\!], \ldots, [\![l_m^*]\!])) \leftarrow \mathcal{G}_{\text{CRS-}i^*}^{\text{M}}(([\![L_1]\!], \ldots, [\![L_m]\!]))$ with $[\![l_i^*]\!] := \mathsf{Blind}([\![l_{i,j^*}]\!])$ $(\forall i \in \mathbb{N}_m)$, where $j^* \leftarrow_{\$} \{j \in \mathbb{N}_n : l_{i^*,j} = max(l_{i^*,1}, \ldots, l_{i^*,n})\}$.*
2. *Then, functionality $\mathcal{G}_{\text{CRS-C-}i^*}^{\text{M}}$ is given by $(([\![l_1^*]\!], \ldots, [\![l_m^*]\!])) \leftarrow \mathcal{G}_{\text{CRS-C-}i^*}^{\text{M}}(([\![L_1]\!], \ldots, [\![L_m]\!]))$ with $[\![l_i^*]\!] := \mathsf{Blind}([\![l_{i,j^*}]\!])$ $(\forall i \in \mathbb{N}_m)$, where $j^* \leftarrow_{\$} \{j \in \mathbb{N}_n : l_{i^*,j} = max(l_{i^*,1}, \ldots, l_{i^*,n})\}$ if there is at least one $j \in \mathbb{N}_n$ s.t. $l_{i^*,j} > 0$. Otherwise, $\mathcal{G}_{\text{CRS-C-}i^*}^{\text{M}}$ returns $(\perp, \overset{m}{\ldots}, \perp)$ where $\perp$ denotes the empty string.*

Gate Specification 10 ($\rho_{\text{CRS-}i^*}^{\text{SH-M}}$)**.** Since a gate implementing gate functionality $\mathcal{G}_{\text{CRS-}i^*}^{\text{M}}$ (see Definition 5.11) operates on an encrypted data table (including the encrypted indicator

Gate 5.4. Specification of $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$.

1 Shuffling Phase

1.1 All parties set $(\llbracket L_1^{(0)} \rrbracket, \ldots, \llbracket L_m^{(0)} \rrbracket) := (\llbracket L_1 \rrbracket, \ldots, \llbracket L_m \rrbracket)$

1.2 Party P_ℓ for ℓ from 1 to ι:

1.2.1 Select permutation σ_ℓ of $\{0, \ldots, n\}$ uniformly at random

1.2.2 Compute $\llbracket L_i^{(\ell)} \rrbracket := \mathsf{Blind}(\sigma_\ell(\llbracket L_i^{(\ell-1)} \rrbracket))$ $(\forall i \in \mathbb{N}_m)$

1.2.3 If $\ell = \iota$: Broadcast $(\llbracket L_1^{(\iota)} \rrbracket, \ldots, \llbracket L_m^{(\iota)} \rrbracket)$

1.2.4 Else: Send $(\llbracket L_1^{(\ell)} \rrbracket, \ldots, \llbracket L_m^{(\ell)} \rrbracket)$ to party $P_{\ell+1}$

2 Swapping Phase

2.1 All parties set $(\llbracket L_1'^{(0)} \rrbracket, \ldots, \llbracket L_m'^{(0)} \rrbracket) := (\llbracket L_1^{(\iota)} \rrbracket, \ldots, \llbracket L_m^{(\iota)} \rrbracket)$

2.2 For j' from 2 to n:

2.2.1 All parties jointly compute $(o_1, \ldots, o_\iota) \leftarrow \rho_{\text{GT-SO}}((\llbracket L_{i^*}'^{(0)}[j'-1] \rrbracket, \llbracket L_{i^*}'^{(0)}[j'] \rrbracket))$

2.2.2 Party P_ℓ for ℓ from 1 to ι:

2.2.2.1 $\forall i \in \mathbb{N}_m$ set $(\llbracket L_i'^{(\ell)}[j'-1] \rrbracket, \llbracket L_i'^{(\ell)}[j'] \rrbracket) :=$
$$\begin{cases} \mathsf{Blind}((\llbracket L_i'^{(\ell-1)}[j'-1] \rrbracket, \llbracket L_i'^{(\ell-1)}[j'] \rrbracket)) & \text{if } o_\ell = 0 \\ \mathsf{Blind}((\llbracket L_i'^{(\ell-1)}[j'] \rrbracket, \llbracket L_i'^{(\ell-1)}[j'-1] \rrbracket)) & \text{otherwise} \end{cases}$$

2.2.2.2 If $[\ell = \iota] \wedge [j' \neq n]$: Send $\llbracket L_i'^{(0)}[j'] \rrbracket := \llbracket L_i'^{(\iota)}[j'] \rrbracket$ to party P_1 $(\forall i \in \mathbb{N}_m)$

2.2.2.3 Else If $[\ell = \iota] \wedge [j' = n]$: Broadcast $\llbracket L_i'^{(\iota)}[n] \rrbracket$ $(\forall i \in \mathbb{N}_m)$

2.2.2.4 Else: Send $(\llbracket L_i'^{(\ell)}[j'-1] \rrbracket, \llbracket L_i'^{(\ell)}[j'] \rrbracket)$ to party $P_{\ell+1}$ $(\forall i \in \mathbb{N}_m)$

3 Output Phase

3.1 All parties set $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket) := (\llbracket L_1'^{(\iota)}[n] \rrbracket, \ldots, \llbracket L_m'^{(\iota)}[n] \rrbracket)$

3.2 Party P_ℓ outputs $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket)$ $(\forall \ell \in \mathscr{P})$

vector), it has to obliviously check which data records fulfill or violate the condition of interest. Our approach is to obliviously move a random data record which satisfies the condition of interest (resp., a random data record with the highest priority) to a specific place, more precisely, to the n-th row, in the data table which constitutes the gate output and can be decrypted if desired. Gate $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$ (see Gate 5.4) can be split into the following three phases:

1. *Shuffling Phase:* The primary purpose of the shuffling phase is to ensure that a data record satisfying the condition of interest (resp., the data record with highest priority) is selected uniformly at random from all those data records satisfying the condition (resp., with highest priority). Note that if there is no such data record,

an arbitrary one is returned which is selected uniformly at random from all data records. First, all parties set $(\llbracket L_1^{(0)} \rrbracket, \ldots, \llbracket L_m^{(0)} \rrbracket) := (\llbracket L_1 \rrbracket, \ldots, \llbracket L_m \rrbracket)$. Next, all vectors are shuffled in a staggered fashion such that at the end of the shuffling phase, no party can link any entry of the input vectors to any entry in the shuffled vectors (even if $\iota - 1$ parties are collaborating). To this end, each party P_ℓ ($\ell \in \mathscr{P}$) subsequently shuffles the vectors $(\llbracket L_1^{(\ell-1)} \rrbracket, \ldots, \llbracket L_m^{(\ell-1)} \rrbracket)$ shuffled by $P_{\ell-1}$. The shuffling consists of first selecting a random permutation σ_ℓ of $\{1, \ldots, n\}$ (see Step 1.2.1, Gate 5.4), applying it to each vector by computing $\sigma_\ell(\llbracket L_i^{(\ell-1)} \rrbracket)$ ($\forall i \in \mathbb{N}_m$), followed by blinding the entries of the permuted vectors resulting in $(\llbracket L_1^{(\ell)} \rrbracket, \ldots, \llbracket L_m^{(\ell)} \rrbracket)$ (see Step 1.2.2, Gate 5.4) which is sent to $P_{\ell+1}$. Party P_1 initiates the shuffling phase by operating on $(\llbracket L_1^{(0)} \rrbracket, \ldots, \llbracket L_m^{(0)} \rrbracket)$. Lastly, P_ι broadcasts $(\llbracket L_1^{(\iota)} \rrbracket, \ldots, \llbracket L_m^{(\iota)} \rrbracket)$. Note that a party applies the same permutation on each vector, thus, the data records stay intact.

2. *Swapping Phase:* In Step 2.1 (Gate 5.4), the parties first set $(\llbracket L_1'^{(0)} \rrbracket, \ldots, \llbracket L_m'^{(0)} \rrbracket) := (\llbracket L_1^{(\iota)} \rrbracket, \ldots, \llbracket L_m^{(\iota)} \rrbracket)$. The purpose of the next step is to obliviously re-order pairs of entries from the indicator vector and the value vectors while maintaining their association such that the rightmost data record which meets the condition (resp., with the highest priority) percolates to the rightmost index of the data table. The re-ordering is based on the indicator vector entries which are compared in a bubble sort fashion (Step 2.2.1, Gate 5.4). Depending on the outcome of the comparison, both indicator vector entries are obliviously swapped along with the corresponding value vector entries. The oblivious swap operations are performed in a staggered fashion where each party P_ℓ subsequently and obliviously swaps the current two data records depending on o_ℓ (obtained in Step 2.2.1) and sends the result to $P_{\ell+1}$. Party P_1 initiates each swapping by operating on $(\llbracket L_{i^*}'^{(0)}[j'-1] \rrbracket, \llbracket L_{i^*}'^{(0)}[j'] \rrbracket)$ ($j' \in \{2, \ldots, n\}$). In case that $j' \neq n$, P_ι updates $\llbracket L_i'^{(0)}[j'] \rrbracket := \llbracket L_i'^{(\iota)}[j'] \rrbracket$ and sends the result to P_1 and, otherwise, (i.e., $j' = n$) P_ι broadcasts $\llbracket L_i'^{(\iota)}[n] \rrbracket$ ($\forall i \in \mathbb{N}_m$).

3. *Output Phase:* In Step 3.1 (Gate 5.4) the parties set $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket)$ to the rightmost data record $(\llbracket L_1'^{(\iota)}[n] \rrbracket, \ldots, \llbracket L_m'^{(\iota)}[n] \rrbracket)$ of the re-ordered encrypted data table determined by $(\llbracket L_1'^{(\iota)} \rrbracket, \ldots, \llbracket L_m'^{(\iota)} \rrbracket)$. Subsequently, each party P_ℓ ($\forall \ell \in \mathscr{P}$) outputs $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket)$.

Turning now to the correctness of gate $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$: At the end of the shuffling phase, the data records are randomly permuted. Furthermore, the loop in Step 2.2 (Gate 5.4) propagates the data record with the largest index $j \in \mathbb{N}_n$ among all data records satisfying the specified condition (resp., with the highest priority). This is due to the fact that in each

iteration of the for loop two adjoined data records are swapped iff the associated indicator values are swapped. Specifically, this swap occurs iff the indicator value at index $j'-1$ is strictly greater than the indicator value at index j'. Due to the fact that in the shuffling phase the data records are shuffled uniformly at random, the data record which propagates to index n is chosen uniformly at random from all data records satisfying the specified condition (resp., with the highest priority).

Theorem 5.4. *Let input* $(\llbracket L_1 \rrbracket, \ldots, \llbracket L_m \rrbracket)$ *and output* $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket)$ *be given to simulator* S*. Gate* $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$ *can be simulated such that the simulated view is computationally indistinguishable from the real view.*

Proof. First, assume that parties $P_1, \ldots, P_t$ are corrupted, i.e., $C = \{1, \ldots, t\}$. Simulator S simulates $(\llbracket L_1^{(\ell)} \rrbracket, \ldots, \llbracket L_m^{(\ell)} \rrbracket)$ $(\ell \in C)$ by computing Step 1.2 (Gate 5.4) on simulator input $(\llbracket L_1 \rrbracket, \ldots, \llbracket L_m \rrbracket)$ in place of $P_1, \ldots, P_t$. The broadcast message sent by P_ι in Step 1.2.3 is simulated as $(\langle \llbracket L_1^{(\iota)} \rrbracket \rangle, \ldots, \langle \llbracket L_m^{(\iota)} \rrbracket \rangle) \leftarrow_{\$} \mathbb{C}^{mn}$. In order to simulate $(\llbracket L_i'^{(\ell)}[j'-1] \rrbracket, \llbracket L_i'^{(\ell)}[j'] \rrbracket)$ $(\forall \ell \in C)$, simulator S computes Step 2.2 in place of $P_1, \ldots, P_t$ but simulates the output of each execution of $\rho_{\text{GT-SO}}$ (cf. Gate Specification 5) by setting $o_\ell \leftarrow_{\$} \{0,1\}$ $(\forall \ell \in C)$ and calls the simulator of $\rho_{\text{GT-SO}}$ on input $(\langle \llbracket L_{i^*}^{(0)}[j'-1] \rrbracket \rangle, \langle \llbracket L_{i^*}^{(0)}[j'] \rrbracket \rangle)$ and $(\langle o_1 \rangle, \ldots, \langle o_t \rangle)$. For $j' \neq n$, simulator S simulates the message sent from P_ι to P_1 in Step 2.2.2.2 by selecting $\langle \llbracket L_i'^{(0)}[j'] \rrbracket \rangle \leftarrow_{\$} \mathbb{C}$ $(\forall i \in \mathbb{N}_m)$. For the broadcast message sent by P_ι in Step 2.2.2.3, simulator S sets $\langle \llbracket L_i'^{(\iota)}[n] \rrbracket \rangle := \llbracket l_i^* \rrbracket$ $(\forall i \in \mathbb{N}_m)$. Furthermore, we have to distinguish the following cases:

1. $(\ell - 1) \notin C$: S simulates $(\llbracket L_1^{(\ell-1)} \rrbracket, \ldots, \llbracket L_m^{(\ell-1)} \rrbracket)$ which P_ℓ receives from $P_{\ell-1}$ as $(\langle \llbracket L_1^{(\ell-1)} \rrbracket \rangle, \ldots, \langle \llbracket L_m^{(\ell-1)} \rrbracket \rangle) \leftarrow_{\$} \mathbb{C}^{mn}$ and $(\llbracket L_i'^{(\ell-1)}[j'-1] \rrbracket, \llbracket L_i'^{(\ell-1)}[j'] \rrbracket)$ as $(\langle \llbracket L_i'^{(\ell-1)}[j'-1] \rrbracket \rangle, \langle \llbracket L_i'^{(\ell-1)}[j'] \rrbracket \rangle) \leftarrow_{\$} \mathbb{C}^2$ $(\forall i \in \mathbb{N}_m)$.

2. $\{(\iota - t'), (\iota - (t'-1)), \ldots, \iota\} =: C' \subseteq C$ with $t' < t$: S simulates $(\llbracket L_1^{(\ell')} \rrbracket, \ldots, \llbracket L_m^{(\ell')} \rrbracket)$ (resp., $(\llbracket L_i'^{(\ell')}[j'-1] \rrbracket, \llbracket L_i'^{(\ell')}[j'] \rrbracket)$, $\forall i \in \mathbb{N}_m$) by inverting Step 1.2.2 (resp., Step 2.2.2.1) in place of $P_{\iota-t'}, \ldots, P_\iota$ beginning with P_ι based on $(\langle \llbracket L_1^{(\iota)} \rrbracket \rangle, \ldots, \langle \llbracket L_m^{(\iota)} \rrbracket \rangle) \leftarrow_{\$} \mathbb{C}^{mn}$ (resp., the given output $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket)$).

□

Complexity. The communication and round complexities of gate $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$ are in $\mathcal{O}(\iota mns + n \cdot O_{\text{GT-SO}})$ and $\mathcal{O}(n \cdot O_{\text{GT-SO}} + \iota n)$, respectively.

Gate Specification 11 ($\rho^{\text{M-M}}_{\text{CRS-}i^*}$)**.** A gate $\rho^{\text{M-M}}_{\text{CRS-}i^*}$ securely implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{CRS-}i^*}$ (see Definition 5.11) in the presence of a malicious adversary is given by Gate 5.5. As for $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$, gate $\rho^{\text{M-M}}_{\text{CRS-}i^*}$ is split into a shuffling phase, a swapping phase, and an output phase. Here, a secret shuffle operation performed by a single party has to include a

zero-knowledge proof allowing the other parties to check the correctness of this operation and, thus, to enforce semi-honest behavior. In the following, we assume the efficient proof system of [64] allowing the implementation of a verifiable secret shuffling operation for ciphertexts. We refer to the proof of correct shuffling as POCS. The resulting mix-net (consisting of the staggered shuffling operations together with the corresponding POCSs) ensures that if there is at least one honest party choosing a random permutation, it is impossible to link any encrypted element of the input list to any encrypted element of the output list. Furthermore, the proof system from [64] allows a party to prove that several lists of ciphertexts have been shuffled with the same permutation. In order to enforce semi-honest behavior for the swapping phase, we let all parties jointly compute the subsequent swap operations instead of performing these operations in a staggered fashion as it has been proposed for $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$.

1. *Shuffling Phase:* Successively, all parties verifiably shuffle the entries of the encrypted input vectors. Party P_1 begins by shuffling $[\![L_i^{(0)}]\!] := [\![L_i]\!]$ $(\forall i \in \mathbb{N}_m)$ in a verifiable fashion resulting in $([\![L_1^{(1)}]\!], \ldots, [\![L_m^{(1)}]\!])$ which P_1 sends to the next party P_2. Note that a party shuffles each encrypted vector using the same permutation in order to maintain the data records. After P_ι verifiably shuffled $[\![L_i^{(\iota-1)}]\!]$ $(\forall i \in \mathbb{N}_m)$, each party holds $([\![L'_1]\!], \ldots, [\![L'_m]\!]) := ([\![L_1^{(\iota)}]\!], \ldots, [\![L_m^{(\iota)}]\!])$ (see Step 2.1, Gate 5.5). The successive shuffling is a necessary step in $\rho^{\text{M-M}}_{\text{CRS-}i^*}$ which decouples the gate input from the gate output, i.e., it is neither possible for a single party nor for any set of up to $\iota - 1$ collaborating parties to link an encrypted data record of $([\![L_1]\!], \ldots, [\![L_m]\!])$ to an encrypted data record of $([\![L_1^{(\iota)}]\!], \ldots, [\![L_m^{(\iota)}]\!])$.

2. *Swapping Phase:* The for-loop in Step 2.2 operates on two adjacent entries $l'_{i^*,j'-1}$ and $l'_{i^*,j'}$ $(\forall j' \in \{2, \ldots, n\})$ of the encrypted indicator vector $[\![L'_{i^*}]\!] := ([\![l'_{i^*,1}]\!], \ldots,$ $[\![l'_{i^*,n}]\!])$ in each iteration. First, the encrypted bit representations of both entries are computed (Steps 2.2.1 and 2.2.2) which constitute the input for gate $\rho_{\text{BDI-GT}}$ (see Gate Specification 6) for obliviously determining $[\![o]\!] := [\![[l'_{i^*,j'-1} > l'_{i^*,j'}]]\!]$. Ciphertext $[\![o]\!]$ is used in the inner for-loop (Step 2.2.4) which iterates over all encrypted vectors $[\![L'_1]\!], \ldots, [\![L'_m]\!]$ and obliviously swaps entries $[\![l'_{i,j'-1}]\!]$ and $[\![l'_{i,j'}]\!]$ $(\forall i \in \mathbb{N}_m)$ in the case that $o = 1$ (Steps 2.2.4.1 and 2.2.4.2). The purpose of the nested for-loop of Step 2.2 is to iterate over all data records in a bubble-sort fashion in order to propagate a (due to preceding shuffling) random data record with a maximum indicator vector entry to the rightmost index of the data table, i.e., in Step 3.1, $([\![L'_1[n]]\!], \ldots, [\![L'_m[n]]\!])$ contains the selected data record which constitutes the output of the gate.

3. *Output Phase:* Each party P_ℓ $(\forall \ell \in \mathscr{P})$ outputs $([\![l_1^*]\!], \ldots, [\![l_m^*]\!]) := ([\![l'_{1,n}]\!], \ldots, [\![l'_{m,n}]\!])$.

Gate 5.5. Specification of $\rho^{\text{M-M}}_{\text{CRS-}i^*}$.

1 Shuffle Phase:

1.1 All parties set $(\llbracket L_1^{(0)} \rrbracket, \ldots, \llbracket L_m^{(0)} \rrbracket) := (\llbracket L_1 \rrbracket, \ldots, \llbracket L_m \rrbracket)$

1.2 Party P_ℓ for ℓ from 1 to ι:

1.2.1 Verifiably shuffles $\llbracket L_1^{(\ell-1)} \rrbracket, \ldots, \llbracket L_m^{(\ell-1)} \rrbracket$ with the same random permutation resulting in $\llbracket L_1^{(\ell)} \rrbracket, \ldots, \llbracket L_m^{(\ell)} \rrbracket$ and provides POCS

2 Swapping Phase:

2.1 All parties set $(\llbracket L'_1 \rrbracket, \ldots, \llbracket L'_m \rrbracket) := (\llbracket L_1^{(\iota)} \rrbracket, \ldots, \llbracket L_m^{(\iota)} \rrbracket)$

2.2 For j' from 2 to n:

2.2.1 All parties jointly compute $(\llbracket (l'_{i^*,j'-1})_{bin} \rrbracket) \leftarrow \rho_{\text{BD}}(\llbracket (l'_{i^*,j'-1}) \rrbracket)$

2.2.2 All parties jointly compute $(\llbracket (l'_{i^*,j'})_{bin} \rrbracket) \leftarrow \rho_{\text{BD}}(\llbracket (l'_{i^*,j'}) \rrbracket)$

2.2.3 All parties jointly compute $(\llbracket o \rrbracket) \leftarrow \rho_{\text{BDI-GT}}((\llbracket (l'_{i^*,j'-1})_{bin} \rrbracket, \llbracket (l'_{i^*,j'})_{bin} \rrbracket))$

2.2.4 For i from 1 to m:

2.2.4.1 All parties jointly compute
$\llbracket L'_i[j'-1] \rrbracket := \rho_{\text{Mult}}((\llbracket 1-o \rrbracket, \llbracket l'_{i,j'-1} \rrbracket)) +_h \rho_{\text{Mult}}((\llbracket o \rrbracket, \llbracket l'_{i,j'} \rrbracket))$

2.2.4.2 All parties jointly compute
$\llbracket L'_i[j'] \rrbracket := \rho_{\text{Mult}}((\llbracket 1-o \rrbracket, \llbracket l'_{i,j'} \rrbracket)) +_h \rho_{\text{Mult}}((\llbracket o \rrbracket, \llbracket l'_{i,j'-1} \rrbracket))$

3 Output Phase:

3.1 All parties set $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket) := (\llbracket l'_{1,n} \rrbracket, \ldots, \llbracket l'_{m,n} \rrbracket)$

3.2 Party P_ℓ outputs $(\llbracket l_1^* \rrbracket, \ldots, \llbracket l_m^* \rrbracket)$ $(\forall \ell \in \mathscr{P})$

Considering that a verifiable shuffle requires a zero knowledge proof allowing each party to check whether the shuffling operation has been performed correctly, each party deviating from the gate specification is detected. The correctness of gate $\rho^{\text{M-M}}_{\text{CRS-}i^*}$ can be argued analogously to the correctness of $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$.

Theorem 5.5. *Let $L_i^{\langle 0 \rangle}$, $L_i^{\langle 1 \rangle}$, and $\llbracket b \rrbracket$ $(b \in \{0,1\})$ with $L_i^{\langle b \rangle} := (l_{i,1}^{\langle b \rangle}, \ldots, l_{i,n}^{\langle b \rangle})$ be given to simulator S $(\forall i \in \mathbb{N}_m)$. For input $\llbracket \tilde{L}_i \rrbracket := (\llbracket \tilde{l}_{i,1} \rrbracket, \ldots, \llbracket \tilde{l}_{i,n} \rrbracket)$ with $\llbracket \tilde{l}_{i,j} \rrbracket := \llbracket (1-b) \cdot l_{i,j}^{\langle 0 \rangle} + b \cdot l_{i,j}^{\langle 1 \rangle} \rrbracket$ $(\forall i \in \mathbb{N}_m, \forall j \in \mathbb{N}_n)$, gate $\rho^{\text{M-M}}_{\text{CRS-}i^*}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. W.l.o.g. assume that $P_1, \ldots, P_t$ are corrupted. First, the successive verifiable shuf-

fles (Step 1.2, Gate 5.5) have to be simulated which can be done similarly to [55]. Subsequently, for ℓ from 1 to t, simulator S lets P_ℓ verifiably shuffle $([\![\tilde{L}_1^{(\ell-1)}]\!], \ldots, [\![\tilde{L}_m^{(\ell-1)}]\!])$ and obtains updated lists $[\![\tilde{L}_1^{(\ell)}]\!], \ldots, [\![\tilde{L}_m^{(\ell)}]\!]$. From the respective zero knowledge proofs, simulator S extracts the corresponding permutations which link $([\![\tilde{L}_1]\!], \ldots, [\![\tilde{L}_m]\!])$ and $([\![\tilde{L}_1^{(t)}]\!], \ldots, [\![\tilde{L}_m^{(t)}]\!])$. For the honest parties $P_{t+1}, \ldots, P_\iota$, the simulator chooses random permutations $\sigma_{\ell'}^{\langle 0\rangle}$ and $\sigma_{\ell'}^{\langle 1\rangle}$ ($\forall \ell' \in \{t+1, \ldots, \iota\}$). Up to this point, simulator S thus holds permutations $\sigma_1, \ldots, \sigma_t$ (for the corrupted parties) and permutations $(\sigma_{t+1}^{\langle 0\rangle}, \sigma_{t+1}^{\langle 1\rangle}), \ldots, (\sigma_\iota^{\langle 0\rangle}, \sigma_\iota^{\langle 1\rangle})$ (for the honest parties). What remains to be done is to simulate the zero knowledge proofs for the last $\iota - t$ verifiable shufflings. Let $L_i^{(\ell')\langle b\rangle} := \sigma_{\ell'}^{\langle b\rangle}(L_i^{(\ell'-1)\langle b\rangle})$ ($\forall b, i, \ell' : b \in \{0,1\}, i \in \mathbb{N}_m, \ell' \in \{t+1, \iota\}$), let $[\![\tilde{l}_{i,j}^{(\ell')}]\!] := [\![(1-b) \cdot l_{i,j}^{(\ell')\langle 0\rangle} + b \cdot l_{i,j}^{(\ell')\langle 1\rangle}]\!]$, and let $[\![\tilde{L}_i^{(\ell')}]\!] := ([\![\tilde{l}_{i,1}^{(\ell')}]\!], \ldots, [\![\tilde{l}_{i,n}^{(\ell')}]\!])$ with $j \in \mathbb{N}_n$. For ℓ' from $t+1$ to ι, simulator S calls the simulator of the zero knowledge proof for shuffle on inputs $([\![\tilde{L}_1^{(\ell'-1)}]\!], \ldots, [\![\tilde{L}_m^{(\ell'-1)}]\!])$ and $([\![\tilde{L}_1^{(\ell')}]\!], \ldots, [\![\tilde{L}_m^{(\ell')}]\!])$.

The remaining proof is straight-forward. Next, simulator S sets $([\![\tilde{L}'_1]\!], \ldots, [\![\tilde{L}'_m]\!]) := ([\![\tilde{L}_1^{(\iota)}]\!], \ldots, [\![\tilde{L}_m^{(\iota)}]\!])$. In order to simulate Step 2.2.1, the simulator of ρ_{BD} is called on input $l'^{\langle 0\rangle}_{i^*,j'-1}$, $l'^{\langle 1\rangle}_{i^*,j'-1}$, and $[\![b]\!]$. The output of ρ_{BD} is computed as $[\![(\tilde{l}'_{i^*,j'-1})_{bin}]\!] := [\![(1-b) \cdot (l'^{\langle 0\rangle}_{i^*,j'-1})_{bin} + b \cdot (l'^{\langle 1\rangle}_{i^*,j'-1})_{bin}]\!]$. Step 2.2.2 can be simulated analogously. The call of gate $\rho_{\text{BDI-GT}}$ can be simulated by calling the corresponding simulator on inputs $(l'^{\langle 0\rangle}_{i^*,j'-1})_{bin}$, $(l'^{\langle 1\rangle}_{i^*,j'-1})_{bin}$, $(l'^{\langle 0\rangle}_{i^*,j'})_{bin}$, $(l'^{\langle 1\rangle}_{i^*,j'})_{bin}$, and $[\![b]\!]$. The corresponding output is computed as $[\![\tilde{o}]\!] := [\![(1-b) \cdot o^{\langle 0\rangle} + b \cdot o^{\langle 1\rangle}]\!]$ where $o^{\langle b\rangle} := [l'^{\langle b\rangle}_{i^*,j'-1} > l'^{\langle b\rangle}_{i^*,j'}]$. Furthermore, the four calls of the multiplication gate ρ_{Mult} in Step 2.2.4.1 and Step 2.2.4.2 have to be simulated. The first call of ρ_{Mult} on input $([\![1-o]\!], [\![l'_{i,j'-1}]\!])$ can be simulated by providing the corresponding simulator with input $([\![1]\!] -_h [\![\tilde{o}]\!], [\![\tilde{l}'_{i,j'-1}]\!])$ and output $[\![(1-\tilde{o}) \cdot \tilde{l}'_{i,j'-1}]\!] := [\![(1-b) \cdot (1-o^{\langle 0\rangle}) \cdot l'^{\langle 0\rangle}_{i,j'-1} + b \cdot (1-o^{\langle 1\rangle}) \cdot l'^{\langle 1\rangle}_{i,j'-1}]\!]$. The remaining calls of ρ_{Mult} can be simulated similarly. After updating $[\![l'^{\langle b\rangle}_{i,j'-1}]\!] := [\![(1-o^{\langle b\rangle}) \cdot l'^{\langle b\rangle}_{i,j'-1} + o^{\langle b\rangle} \cdot l'^{\langle b\rangle}_{i,j'}]\!]$ ($b \in \{0,1\}$), $[\![\tilde{L}_i[j'-1]]\!]$ is computed as $[\![(1-b) \cdot l'^{\langle 0\rangle}_{i,j'-1} + b \cdot l'^{\langle 1\rangle}_{i,j'-1}]\!]$. $[\![\tilde{L}_i[j']]\!]$ can be computed analogously. Finally, simulator S sets $[\![\tilde{l}^*_i]\!] := [\![\tilde{l}'_{i,n}]\!]$ ($\forall i \in \mathbb{N}_m$). □

Complexity. The communication and round complexities of gate $\rho^{\text{M-M}}_{\text{CRS-}i^*}$ are in $\mathcal{O}(\iota mns + n \cdot (O_{\text{BD}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-GT}}^{m \in \mathcal{O}(s)}) + nm \cdot O_{\text{Mult}})$ and $\mathcal{O}(\iota + n \cdot (O_{\text{BD}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-GT}}^{m \in \mathcal{O}(s)}) + nm \cdot O_{\text{Mult}})$, respectively.

Gate Specification 12 ($\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$)**.** In case that there exists no data record that satisfies the specified condition (resp., all data records are assigned with priority 0), a gate implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{CRS-}i^*}$ outputs an arbitrary encrypted data record selected uniformly at random from all data records given by $([\![L_1]\!], \ldots, [\![L_m]\!])$. Depending on the application scenario this behavior may be undesirable. In fact, it may be preferable that

output is only provided if there is at least one data record satisfying the condition (resp., with priority greater than 0). This behavior is captured by gate functionality $\mathcal{G}^{\text{M}}_{\text{CRS-C-}i^*}$. A gate $\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$ implementing $\mathcal{G}^{\text{M}}_{\text{CRS-C-}i^*}$ can be derived from $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$ by adding two additional steps after Step 3.1 (Gate 5.4): All parties obliviously check whether or not $l^*_{i^*} > 0$, e.g., by jointly executing $(o_1, \ldots, o_\iota) \leftarrow \rho^{\text{SH-M}}_{\text{LT-SO}}((\llbracket 0 \rrbracket, \llbracket l^*_{i^*} \rrbracket))$ (see Gate Specification 5) followed by combining their shares in order to learn the comparison result. Depending on whether or not $l^*_{i^*} > 0$, the parties output $(\llbracket l^*_1 \rrbracket, \ldots, \llbracket l^*_m \rrbracket)$ or $(\perp, \overset{m}{\ldots}, \perp)$ where $\perp$ denotes the empty string. Note that the involved modifications have no influence on the theoretical complexities of gate $\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$.

Gate $\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$ can be simulated analogously to $\rho^{\text{SH-M}}_{\text{CRS-}i^*}$. The output $(o_1, \ldots, o_\iota)$ of $\rho^{\text{SH-M}}_{\text{LT-SO}}(\llbracket 0 \rrbracket, \llbracket l^*_{i^*} \rrbracket)$ can be simulated in such a way that $o_1 \oplus \ldots \oplus o_\iota = 0$ if the output of $\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$ is $(\perp, \overset{m}{\ldots}, \perp)$ and $o_1 \oplus \ldots \oplus o_\iota = 1$ otherwise.

Note that this extension cannot be applied to gate $\rho^{\text{M-M}}_{\text{CRS-}i^*}$, since the simulation of $o_1, \ldots, o_\iota$ would require knowledge of $b \in \{0, 1\}$ instead of just $\llbracket b \rrbracket$.

Related Work: The primitive of privacy-preserving conditional random selection is akin to oblivious filtering [79] and oblivious sorting (e.g., [66,71,128]).

The oblivious database filtering protocol from [79] (based on secret sharing and providing security in the semi-honest or malicious model depending on the used building blocks) operates on a distributed set of data records where each data record has an additional binary entry indicating whether or not the data record satisfies a specified condition, referred to as *inclusion criterion*. The participating parties jointly determine those data records which satisfy the inclusion criterion in an oblivious fashion. In [79], the authors distinguish between three different types of parties: data donors, miners, and clients. The data donors hold private data records which are secretly shared before being submitted to the miners. The miners first obliviously shuffle the shares of the data records followed by reconstructing the entries containing the binary indicators in order to identify those data records which meet the inclusion criterion. The corresponding shares of these data records are send to the clients which can reconstruct them. The oblivious database filtering protocol from [79] can be modified in order to only reveal a random subset of those data records which meet the inclusion criterion. However, the protocol leaks the number of *all* data records which satisfy the inclusion criterion which makes it unsuitable for implementing conditional random selection.

Protocols for oblivious sorting allow multiple parties to jointly sort their aggregated private inputs without leaking the value and the position of any input to any party. Using an oblivious sorting protocol in order to determine a random data record which satisfies a specified condition (by sorting the data records based on their indicator entries)

results in less efficient implementations compared to the direct approaches presented above. This is due to the fact that, generally, protocols for oblivious sorting involve additional communication and computation overhead since the correct position of each input is computed. Furthermore, the most efficient sorting protocols have to terminate in order to assure that a specific input is at the right position and thus cannot be prematurely aborted in order to extract a data record satisfying the specified condition.

5.3.3 Random Value Generation

The following gate functionality allows multiple parties to jointly compute the encrypted bit representation of an integer r drawn uniformly at random from a publicly known interval such that r remains private for all parties. Designing such a gate constitutes a particularly important building block for the malicious model allowing to force a specific (possibly corrupted) party to really use a random value if demanded by the protocol specification (in the semi-honest model a party can just select a random value on its own).

Definition 5.12 ($\mathcal{G}^{\mathrm{M}}_{\mathrm{RBVG}}$: Random Bitwise Value Generation)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold a public value a such that $2^{m-1} < a \leq 2^m$. Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\mathrm{RBVG}}$ is given by $((\llbracket r_0 \rrbracket, \ldots, \llbracket r_{m-1} \rrbracket)) \leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{RBVG}}(a)$ where $r_{bin} := (r_0, \ldots, r_{m-1})$ is the bit representation of r with $r \leftarrow_{\$} [0, a)$.*

Gate Specification 13 ($\mathcal{G}^{\mathrm{M}}_{\mathrm{RBVG}}$)**.** A gate $\rho^{\text{M-M}}_{\mathrm{RBVG}}$ implementing gate functionality $\mathcal{G}^{\mathrm{M}}_{\mathrm{RBVG}}$ (see Definition 5.12) has been presented in [65].

Complexity. The communication complexity of gate $\rho^{\text{M-M}}_{\mathrm{RBVG}}$ from [65] is in $\mathcal{O}(\iota s m)$ and the round complexity is in $\mathcal{O}(\iota)$.

5.3.4 Random Subinterval Selection

In the following, we present three gate functionalities along with the corresponding gate specifications that allow the sampling from a private interval that is given by its encrypted bounds.

Definition 5.13 ($\mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}$: Random Sub-Interval)**.** *Let P_1 and P_2 both hold the encrypted interval bounds $\llbracket \lambda \rrbracket$, $\llbracket \mu \rrbracket$ of an unknown interval $[\lambda, \mu]$ with $\omega \leq |[\lambda, \mu]| \leq \omega_\Delta$, where ω and ω_Δ are publicly known. Then, gate functionality $\mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}$ is given by $((\llbracket \lambda_r \rrbracket, \llbracket \mu_r \rrbracket)) \leftarrow \mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}((\llbracket \lambda \rrbracket, \llbracket \mu \rrbracket))$, where $\mu_r - \lambda_r = |[\lambda_r, \mu_r]| = \omega$, $\lambda_r \leftarrow_{\$} [\lambda, \mu - \omega]$, and $\mu_r = \lambda_r + \omega$.*

The following two functionalities are more specific in that a single encrypted element (instead of an encrypted subinterval) is sampled from a private interval. However, it is

straight-forward to extend Definition 5.14 and the corresponding gate for selecting a random subinterval.

Definition 5.14 ($\mathcal{G}^{\mathrm{M}}_{\mathrm{RIE}}$: Random Interval Element)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold the encrypted interval bounds $[\![\lambda]\!]$, $[\![\mu]\!]$ of an unknown interval $[\lambda, \mu]$ with $0 \leq |[\lambda, \mu]| \leq \omega_\Delta$ where ω_Δ is publicly known. Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\mathrm{RIE}}$ is given by $([\![e]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{RIE}}(([\![\lambda]\!], [\![\mu]\!]))$, where $[\![e]\!]$ is an encryption of $e \leftarrow_{\$} [\lambda, \mu]$.*

Definition 5.15 ($\mathcal{G}^{\mathrm{M}}_{\mathrm{RIE}\text{-}\mathcal{D}}$: Random Interval Element Sampled According to Discrete Distribution $\mathcal{D}$)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold the encrypted interval bounds $[\![\lambda]\!]$, $[\![\mu]\!]$ of an unknown interval $[\lambda, \mu]$ with $0 \leq |[\lambda, \mu]| \leq \omega_\Delta$. Then, functionality $\mathcal{G}^{\mathrm{M}}_{\mathrm{RIE}\text{-}\mathcal{D}}$ is given by $([\![e]\!]) \leftarrow \mathcal{G}^{\mathrm{M}}_{\mathrm{RIE}\text{-}\mathcal{D}}(([\![\lambda]\!], [\![\mu]\!]))$, where $[\![e]\!]$ is such that $e \leftarrow_{\mathcal{D}} [\lambda, \mu]$ and ω_Δ, $\mathcal{D}$ are publicly known.*

Note that Definition 5.14 and Definition 5.15 coincide in case when in Definition 5.15 distribution $\mathcal{D}$ instantiated with the uniform distribution.

Gate Specification 14 ($\rho^{\text{SH-T}}_{\text{RSI-}\omega}$)**.** The approach of gate $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$ (see Gate 5.6) implementing gate functionality $\mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}$ (see Definition 5.13) is to obliviously compute $[\![\lambda_{r'}]\!]$ such that $\lambda_{r'} \leftarrow_{\$} [0, \Delta]$ where $\Delta := \mu - \lambda - \omega$. Setting $[\![\lambda_r]\!] := [\![\lambda]\!] +_h [\![\lambda_{r'}]\!]$ and $[\![\mu_r]\!] := [\![\lambda_r]\!] +_h [\![\omega]\!]$ we have that $[\lambda_r, \mu_r]$ is an interval of width ω selected uniformly at random from $[\lambda, \mu]$.

First, P_1 computes $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!] -_h [\![\omega]\!]$ as well as an encrypted vector

$$[\![V]\!] := ([\![v_0]\!], \ldots, [\![v_{\omega_\Delta}]\!]) = ([\![0]\!], \ldots, [\![\omega_\Delta]\!]), \tag{5.2}$$

containing the encryption of each possible value for $\lambda_{r'} \in [0, \omega_\Delta]$. Subsequently, P_1 generates a second encrypted vector $[\![U]\!] := ([\![u_0]\!], \ldots, [\![u_{\omega_\Delta}]\!])$ where entry $[\![u_i]\!]$ is associated with entry $[\![v_i]\!]$ from $[\![V]\!]$ ($\forall i \in \mathbb{N}^0_{\omega_\Delta}$). The entries of $[\![U]\!]$ are jointly generated by P_1 and P_2: First, P_2 generates an encrypted lower triangular matrix $[\![A]\!] \in \mathbb{C}^{\omega_\Delta+1 \times \omega_\Delta+1}$ which is composed of row vectors $[\![A_0]\!], \ldots, [\![A_{\omega_\Delta}]\!] \in \mathbb{C}^{1 \times \omega_\Delta+1}$ as

$$[\![A]\!] := ([\![A_0]\!], \ldots, [\![A_{\omega_\Delta}]\!])^T \tag{5.3}$$

where $[\![A_i]\!] := \sigma_i(([\![0]\!], [\![1]\!], \ldots, [\![i]\!], 0, \overset{\omega_\Delta - i}{\ldots}, 0))$ and σ_i is a random permutation of $\{0, \ldots, i\}$ chosen by P_2. In Step 2.2, P_2 sends $[\![A]\!]$ to P_1. For each row i of $[\![A]\!]$, party P_1 selects $j \leftarrow_{\$} \mathbb{N}^0_i$ and sets $[\![u_i]\!] := [\![A_i[j]]\!]$. Subsequently, P_1 re-randomizes the entries of $[\![U]\!]$ and chooses a random permutation σ' of $\{0, \ldots, \omega_\Delta\}$ in order to compute $[\![V']\!] := \sigma'([\![V]\!])$ and $[\![U']\!] := \sigma'(\mathsf{Blind}(U))$ which are sent along with $[\![\Delta]\!]$ to P_2. Analogously to P_1, party P_2 re-randomizes and permutes $[\![V']\!]$ and $[\![U']\!]$ resulting in $[\![V'']\!] := \sigma''(\mathsf{Blind}([\![V']\!]))$ and $[\![U'']\!] := \sigma''(\mathsf{Blind}([\![U']\!]))$ which are sent to P_1 (cf. Step 4). For each entry $[\![v''_i]\!]$ in $[\![V'']\!]$ ($i \in \mathbb{N}^0_{\omega_\Delta}$) both parties jointly execute ρ_{ET} (see Gate Specification 8) followed by a decryption of the result $[\![o_i]\!]$ in order to check whether or not v''_i is equal to Δ (see Step 5).

Gate 5.6. Specification of $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$.

1 Party P_1:

1.1 Compute $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!] -_h [\![\omega]\!]$

1.2 Generate $[\![V]\!]$ according to Eq. 5.2

2 Party P_2

2.1 Generate $[\![A]\!]$ according to Eq. 5.3

2.2 Send $[\![A]\!]$ to party P_1

3 Party P_1:

3.1 For i from 0 to ω_Δ:

3.1.1 Select $j \leftarrow_\$ \mathbb{N}_i^0$

3.1.2 Set $[\![u_i]\!] := [\![A_i[j]]\!]$

3.2 Set $[\![U]\!] := ([\![u_0]\!], \ldots, [\![u_{\omega_\Delta}]\!])$

3.3 Select permutation σ' of $\{0, \ldots, \omega_\Delta\}$ uniformly at random

3.4 Compute $[\![V']\!] := \sigma'([\![V]\!])$ and $[\![U']\!] := \sigma'(\mathsf{Blind}([\![U]\!]))$

3.5 Send $[\![V']\!]$, $[\![U']\!]$, $[\![\Delta]\!]$ to P_2

4 Party P_2 :

4.1 Select permutation σ'' of $\{0, \ldots, \omega_\Delta\}$ uniformly at random

4.2 Compute $[\![V'']\!] := \sigma''(\mathsf{Blind}([\![V']\!]))$ and $[\![U'']\!] := \sigma''(\mathsf{Blind}([\![U']\!]))$

4.3 Send $[\![V'']\!]$, $[\![U'']\!]$ to P_1

5 For i from 0 to ω_Δ:

5.1 Both parties jointly compute:

5.1.1 $([\![o_i]\!]) \leftarrow \rho_{\text{ET}}(([\![\Delta]\!], [\![v_i'']\!]))$

5.1.2 $\mathsf{Dec}([\![o_i]\!])$ s.t. both parties learn the result

6 Both parties:

6.1 Set $[\![\lambda_{r'}]\!] := [\![u_i'']\!]$ for $i^* \in \mathbb{N}^0_{\omega_\Delta}$ s.t. $o_{i^*} = 1$

6.2 Locally compute $[\![\lambda_r]\!] := [\![\lambda_{r'}]\!] +_h [\![\lambda]\!]$ and $[\![\mu_r]\!] := [\![\lambda_r]\!] +_h [\![\omega]\!]$

6.3 Output $([\![\lambda_r]\!], [\![\mu_r]\!])$

In Step 6, both parties set $[\![\lambda_{r'}]\!] := [\![u''_{i^*}]\!]$ for $i^* \in \mathbb{N}^0_{\omega_\Delta}$ such that $o_{i^*} = 1$. Note that by construction, i^* is unequivocally determined. Next, both parties locally compute $[\![\lambda_r]\!] := [\![\lambda]\!] +_h [\![\lambda_{r'}]\!]$, $[\![\mu_r]\!] := [\![\lambda_r]\!] +_h [\![\omega]\!]$ and output $([\![\lambda_r]\!], [\![\mu_r]\!])$.

In order to see that $[\lambda_r, \mu_r]$ is a subinterval of width ω drawn uniformly at random from $[\lambda, \mu]$, first note that $\lambda_r = \lambda + \lambda_{r'}$ is drawn uniformly at random from $[\lambda, \mu - \omega]$ since $\lambda_{r'}$ is equal to $u_\Delta \leftarrow_\$ [0, \Delta]$. By setting $\mu_r = \lambda_r + \omega$, it follows that $[\lambda_r, \mu_r]$ is as prescribed by Definition 5.13.

Theorem 5.6. *Let input* $([\![\lambda]\!], [\![\mu]\!])$ *and output* $([\![\lambda_r]\!], [\![\mu_r]\!])$ *be given to simulator* S*. Gate* $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$ *can be simulated such that the simulated view is computationally indistinguishable from the real view.*

Proof. First, assume that P_1 is corrupted. Simulator S simulates $[\![A]\!] \in \mathbb{C}^{\omega_\Delta+1 \times \omega_\Delta+1}$ which P_1 receives in Step 2.2 (Gate 5.6) by setting

$$\langle [\![A_i]\!] \rangle := \begin{pmatrix} \langle [\![a_{0,0}]\!] \rangle & 0 & & \cdots & 0 \\ \langle [\![a_{1,0}]\!] \rangle & \langle [\![a_{1,1}]\!] \rangle & 0 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ \langle [\![a_{\omega_\Delta,0}]\!] \rangle & \langle [\![a_{\omega_\Delta,1}]\!] \rangle & \langle [\![a_{\omega_\Delta,2}]\!] \rangle & \cdots & \langle [\![a_{\omega_\Delta,\omega_\Delta}]\!] \rangle \end{pmatrix}$$

where $(\langle [\![a_{0,0}]\!] \rangle, \ldots, \langle [\![a_{\omega_\Delta,\omega_\Delta}]\!] \rangle) \leftarrow_\$ \mathbb{C}^{((\omega_\Delta+1)^2+\omega_\Delta+1)/2}$. The encrypted vectors $[\![V'']\!]$ and $[\![U'']\!]$ are simulated by setting $\langle [\![V'']\!] \rangle \leftarrow_\$ \mathbb{C}^{\omega_\Delta+1}$ and $\langle [\![U'']\!] \rangle \leftarrow_\$ \mathbb{C}^{\omega_\Delta+1}$. In order to simulate the i-th execution of ρ_{ET} ($i \in \mathbb{N}^0_{\omega_\Delta}$), simulator S calls the simulator of ρ_{ET} on input $[\![\Delta]\!]$ and $\langle [\![v''_i]\!] \rangle$ and output $\langle [\![o_i]\!] \rangle \leftarrow_\$ \mathbb{C}$. Furthermore, the output of the i^*-th ($i^* \leftarrow_\$ \mathbb{N}^0_{\omega_\Delta}$) decryption operation (cf. Step 5.1.2) is simulated as $\langle 1 \rangle$. The output of all other decryption operations is simulated as $\langle 0 \rangle$.

The view of corrupted party P_2 can be simulated similarly by simulating $\langle [\![V']\!] \rangle \leftarrow_\$ \mathbb{C}^{\omega_\Delta+1}$, $\langle [\![U']\!] \rangle \leftarrow_\$ \mathbb{C}^{\omega_\Delta+1}$, and $\langle [\![\Delta]\!] \rangle \leftarrow_\$ \mathbb{C}$. □

Complexity. The communication complexity of gate $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$ is in $\mathcal{O}(\omega^2_\Delta s + \omega_\Delta \cdot O_{\text{ET}})$ and the round complexity is in $\mathcal{O}(\omega_\Delta \cdot O_{\text{ET}})$.

Gate Specification 15 ($\rho^{\text{M-M}}_{\text{RIE}}$)**.** Similar to the specification of $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$, for $\rho^{\text{M-M}}_{\text{RIE}}$ implementing gate functionality $\mathcal{G}^{\text{M}}_{\text{RIE}}$ (see Definition 5.14), we reduce the problem of selecting $[\![e]\!]$ such that $e \leftarrow_\$ [\lambda, \mu]$ to the problem of selecting an element $[\![e_\Delta]\!]$ such that $e_\Delta \leftarrow_\$ [0, \Delta]$ with $\Delta := \mu - \lambda$. Then, $[\![e]\!]$ is obtained by computing $[\![e_\Delta]\!] +_h [\![\lambda]\!]$. The central ideas of Gate 5.7 are (i) to generate $\omega_\Delta + 1$ encrypted random integers $[\![r_j]\!]$ for $j \in \mathbb{N}_{\omega_\Delta+1}$ with $r_j \leftarrow_\$ [0, j)$ using gate ρ_{RBVG} (see Gate Specification 13) in Step 2 of Gate 5.7 and to then (ii) call gate ρ_{ES} (see Gate Specification 9) on common input $([\![r_1]\!], \ldots, [\![r_{\omega_\Delta+1}]\!], [\![\Delta]\!] +_h [\![1]\!])$

Gate 5.7. Specification of $\rho^{\text{M-M}}_{\text{RIE}}$.

1 Party P_ℓ locally computes $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!]$ $(\forall \ell \in \mathscr{P})$

2 For j from 1 to $\omega_\Delta + 1$:

  2.1 All parties P_ℓ jointly compute $([\![r_j]\!]) \leftarrow \rho_{\text{RBVG}}(j)$

3 All parties P_ℓ jointly compute $([\![e_\Delta]\!]) \leftarrow \rho_{\text{ES}}(([\![r_1]\!], \ldots, [\![r_{\omega_\Delta+1}]\!], [\![\Delta]\!] +_h [\![1]\!]))$

4 Party P_ℓ $(\forall \ell \in \mathscr{P})$:

  4.1 Locally compute $[\![e]\!] := [\![e_\Delta]\!] +_h [\![\lambda]\!]$

  4.2 Output $[\![e]\!]$

which returns $[\![e_\Delta]\!] := [\![r_{\Delta+1}]\!]$ (see Step 3). Finally, each party locally computes $[\![e]\!] := [\![e_\Delta]\!] +_h [\![\lambda]\!]$ which constitutes the common gate output.

Under the assumption that $0 \leq |[\lambda, \mu]| \leq \omega_\Delta$, output $[\![e]\!]$ is indeed such that the corresponding plaintext e is selected uniformly at random from $[\lambda, \mu]$. This is due to the fact that $\rho_{\text{ES}}([\![r_1]\!], \ldots, [\![r_{\omega_\Delta+1}]\!], [\![\Delta]\!] +_h [\![1]\!])$ returns $[\![e_\Delta]\!] := [\![r_{\Delta+1}]\!]$ and according to the specification of ρ_{RBVG}, the corresponding plaintext e_Δ is selected uniformly at random from $[0, \Delta] = [0, \Delta + 1)$. Consequently, $[\![e]\!]$ is an encryption of $e \leftarrow_{\$} [(0 + \lambda), (\Delta + \lambda)] = [\lambda, \mu]$.

Theorem 5.7. *Let ω_Δ, $\lambda^{\langle 0 \rangle}, \lambda^{\langle 1 \rangle}, \mu^{\langle 0 \rangle}$, $\mu^{\langle 1 \rangle}$, and $[\![b]\!]$ $(b \in \{0, 1\})$ be given to simulator S. For input $[\![\tilde{\lambda}]\!] := [\![(1 - b) \cdot \lambda^{\langle 0 \rangle} + b \cdot \lambda^{\langle 1 \rangle}]\!]$ and $[\![\tilde{\mu}]\!] := [\![(1 - b) \cdot \mu^{\langle 0 \rangle} + b \cdot \mu^{\langle 1 \rangle}]\!]$, gate $\rho^{\text{M-M}}_{\text{RIE}}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. First, simulator S computes $[\![\tilde{\Delta}]\!] := [\![(1 - b) \cdot \Delta^{\langle 0 \rangle} + b \cdot \Delta^{\langle 1 \rangle}]\!]$ with $\Delta^{\langle b \rangle} := \mu^{\langle b \rangle} - \lambda^{\langle b \rangle}$. The $\omega_\Delta + 1$ calls of ρ_{RBVG} can be simulated by calling the simulator of ρ_{RBVG} on input $[\![b]\!]$, j, and $r_j^{\langle 0 \rangle}$, $r_j^{\langle 1 \rangle}$, where $r_j^{\langle 0 \rangle}, r_j^{\langle 1 \rangle} \leftarrow_{\$} [0, j)$. Step 3 of Gate 5.7 is simulated by calling the simulator of ρ_{ES} on $\Delta^{\langle 0 \rangle} + 1$, $\Delta^{\langle 1 \rangle} + 1$, $r_j^{\langle 0 \rangle}$, $r_j^{\langle 1 \rangle}$, and $[\![b]\!]$ $(\forall j \in \mathbb{N}_{\omega_{\Delta+1}})$. The output of ρ_{ES} is computed as $[\![\tilde{e}_\Delta]\!] := [\![(1 - b) \cdot r^{\langle 0 \rangle}_{\Delta^{\langle 0 \rangle}+1} + b \cdot r^{\langle 1 \rangle}_{\Delta^{\langle 1 \rangle}+1}]\!]$. Finally, simulator S computes $[\![\tilde{e}]\!] := [\![(1 - b) \cdot (r^{\langle 0 \rangle}_{\Delta^{\langle 0 \rangle}+1} + \lambda^{\langle 0 \rangle}) + b \cdot (r^{\langle 1 \rangle}_{\Delta^{\langle 1 \rangle}+1} + \lambda^{\langle 1 \rangle})]\!]$. □

Complexity. The communication and round complexities of gate $\rho^{\text{M-M}}_{\text{RIE}}$ is in $\mathcal{O}(\omega_\Delta \cdot O^{m \in \mathcal{O}(log(\omega_\Delta))}_{\text{RBVG}} + O^{n \in \mathcal{O}(\omega_\Delta)}_{\text{ES}})$, respectively.

Gate Specification 16. Gate $\rho^{\text{M-M}}_{\text{RIE-}\mathcal{D}}$ implementing $\mathcal{G}^{\text{M}}_{\text{RIE-}\mathcal{D}}$ (see Definition 5.15) is presented in Gate 5.8 and allows the random sampling of values from a private interval $I = [\lambda, \mu]$

Gate 5.8. Specification of $\rho^{\text{M-M}}_{\text{RIE-}\mathcal{D}}$.

1 All parties set $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!]$

2 For i from 0 to ω_Δ:

2.1 All parties jointly compute $([\![x_i]\!]) \leftarrow \rho_{\text{RBVG}}(|\mathcal{R}_i|)$

2.2 All parties jointly compute $([\![y_i]\!]) \leftarrow \rho_{\text{ES}}(([\![r_{i,0}]\!], \ldots, [\![r_{i,|\mathcal{R}_i|-1}]\!], [\![x_i]\!]))$

3 All parties jointly compute $([\![e_\Delta]\!]) \leftarrow \rho_{\text{ES}}(([\![y_0]\!], \ldots, [\![y_{\omega_\Delta}]\!], [\![\Delta]\!]))$

4 All parties set $[\![e]\!] := [\![e_\Delta]\!] +_h [\![\lambda]\!]$

5 Party P_ℓ outputs $[\![e]\!]$ ($\forall \ell \in \mathscr{P}$)

with interval width $\Delta := \mu - \lambda$ that are distributed according to a (arbitrary) given discrete probability distribution $\mathcal{D}$ defined by a *probability mass function* $f_{\mathcal{D},I}(l; \Delta) = Pr(x = I[l+1])$ ($l \in \mathbb{N}^0_\Delta$) where $I[l+1]$ refers to the $(l+1)$-st smallest element in I. It is important to note that $\mathcal{D}$ describes the distribution of the elements in I relatively to the interval bounds of I.

The problem of computing $[\![e]\!]$ with $e \leftarrow_{\mathcal{D}} [\lambda, \mu]$ is reduced to the problem of computing an element $e_\Delta \leftarrow_{\mathcal{D}} [0, \Delta]$. Then, $[\![e]\!]$ is obtained by computing $[\![e_\Delta]\!] +_h [\![\lambda]\!]$. The key idea is to generate $\omega_\Delta + 1$ encrypted lists $[\![\mathcal{R}_0]\!], \ldots, [\![\mathcal{R}_{\omega_\Delta}]\!]$. List $[\![\mathcal{R}_i]\!]$ consists of entries

$$([\![r_{i,0}]\!], [\![r_{i,1}]\!] \ldots, [\![r_{i,|\mathcal{R}_i|-1})]\!] := ([\![0]\!], \overset{j_0}{\ldots}, [\![0]\!], [\![1]\!], \overset{j_1}{\ldots}, [\![1]\!], \ldots, [\![i]\!], \overset{j_i}{\ldots}, [\![i]\!])$$

where $i \in \mathbb{N}^0_{\omega_\Delta}$ and $j_l, |\mathcal{R}_i|$ ($l \in \mathbb{N}^0_i$) are such that $j_l/|\mathcal{R}_i| = f_{\mathcal{D}}(l; i)$. For each of these encrypted lists the parties obliviously select a single entry by calling $[\![x_i]\!] \leftarrow \rho_{\text{RBVG}}(|\mathcal{R}_i|)$ (see Gate Specification 13). A subsequent call of ρ_{ES} (see Gate Specification 9) on input $([\![r_{i,0}]\!], \ldots, [\![r_{i,|\mathcal{R}_i|-1}]\!], [\![x_i]\!])$ allows the parties to obliviously obtain the encrypted x_i-th entry of $\mathcal{R}_i$. The resulting ciphertexts $[\![y_0]\!], \ldots, [\![y_{\omega_\Delta}]\!]$ are such that $y_i \leftarrow_{\$} \mathcal{R}_i$ and thus $y_i \leftarrow_{\mathcal{D}} [0, i]$. Next, the parties jointly call gate ρ_{ES} (see Gate Specification 9) on input $([\![y_0]\!], \ldots, [\![y_{\omega_\Delta}]\!], [\![\Delta]\!])$ and obtain $[\![e_\Delta]\!]$ where $e_\Delta \leftarrow_{\mathcal{D}} [0, \Delta]$. Finally, each party locally computes $[\![e]\!] := [\![e_\Delta]\!] +_h [\![\lambda]\!]$ which constitutes the output of the gate.

Assuming that for a given gate the lists $\mathcal{R}_0, \ldots, \mathcal{R}_{\omega_\Delta}$ are fixed, $[\![\mathcal{R}_0]\!], \ldots, [\![\mathcal{R}_{\omega_\Delta}]\!]$ can be considered to be publicly known pre-computed gate input. In the following, we provide an example for gate $\rho^{\text{M-M}}_{\text{RIE-}\mathcal{D}}$ with private input interval $[3, 5]$ where $\mathcal{D}$ is instantiated with the *binomial distribution* $\mathcal{B}$.

i	$f(l;i)$ $(\forall l \in [0,i])$	$\mathcal{R}_i$
0	$Pr[x{=}0] = 1$	$\mathcal{R}_0 = (0)$
1	$Pr[x{=}0] = Pr[x{=}1] = 1/2$	$\mathcal{R}_1 = (0,1)$
2	$Pr[x{=}0] = Pr[x{=}2] = 1/4; Pr[x{=}1] = 1/2$	$\mathcal{R}_2 = (0,1,1,2)$
3	$Pr[x{=}0] = Pr[x{=}3] = 1/8; Pr[x{=}1] = Pr[x{=}2] = 3/8$	$\mathcal{R}_3 = (0,1,1,1,2,2,2,3)$
4	$Pr[x{=}0] = Pr[x{=}4] = 1/16; Pr[x{=}1] = Pr[x{=}3] = 1/4;$ $Pr[x{=}2] = 3/8$	$\mathcal{R}_4 = (0,1,1,1,1,2,2,2,2,2,2,3,3,3,3,4)$

Table 5.4: Construction of $\mathcal{R}_i$ for $\mathcal{D} := \mathcal{B}$ and $\omega_\Delta = 4$ ($i \in \mathbb{N}^0_{\omega_\Delta}$).

Example 5.1. Let $I = [\lambda, \mu] = [3,5]$ and let $\omega_\Delta = 4$. Furthermore, let $\mathcal{D}$ be instantiated with the binomial distribution (with success probability 0.5) given by the *probability mass function*

$$f_{\mathcal{B},I}(l;\Delta) = Pr[x = I[l+1]] = \binom{\Delta}{l} 0.5^l (1-0,5)^{\Delta - l}$$

with $l \in \mathbb{N}^0_\Delta$. Then, $\mathcal{R}_0, \ldots, \mathcal{R}_4$ with $|\mathcal{R}_i| = 2^i$ ($i \in \mathbb{N}^0_4$) are as specified in Table 5.4. After computing $[\![\Delta]\!] := [\![2]\!]$, in Step 2 of Gate 5.8, $[\![y_0]\!], \ldots, [\![y_4]\!]$ are jointly computed where $y_i \leftarrow_{\mathcal{B}} [0,i]$. The probabilities for $y_i = l$ ($l \in \mathbb{N}^0_i$) are given in Table 5.4. In Step 3, the parties obliviously compute $[\![e_\Delta]\!]$ with $e_\Delta := y_\Delta$. Consequently, after locally computing $[\![e]\!] := [\![e_\Delta]\!] +_h [\![\lambda]\!]$, it holds that $Pr[e{=}3] = Pr[e{=}5] = 1/4$ and $Pr[e{=}4] = 1/2$.

Due to the construction of $\mathcal{R}_0, \ldots, \mathcal{R}_{\omega_\Delta}$ and under the assumption that $0 \leq \mu - \lambda \leq \omega_\Delta$, it holds that gate output $[\![e]\!]$ is selected as $e \leftarrow_{\mathcal{D}} [\lambda, \mu]$.

Theorem 5.8. *Let ω_Δ, $\mathcal{R}_0, \ldots, \mathcal{R}_{\omega_\Delta}$, $\lambda^{\langle 0 \rangle}$, $\lambda^{\langle 1 \rangle}$, $\mu^{\langle 0 \rangle}$, $\mu^{\langle 1 \rangle}$, and $[\![b]\!]$ be given to simulator S. For input $[\![\tilde{\lambda}]\!] := [\![(1-b)\cdot\lambda^{\langle 0 \rangle} + b \cdot \lambda^{\langle 1 \rangle}]\!]$ and $[\![\tilde{\mu}]\!] := [\![(1-b)\cdot\mu^{\langle 0 \rangle} + b \cdot \mu^{\langle 1 \rangle}]\!]$, gate $\rho^{\text{M-M}}_{\text{RIE-}\mathcal{D}}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. First, simulator S computes $[\![\tilde{\Delta}]\!] := [\![(1-b)\cdot\Delta^{\langle 0 \rangle} + b \cdot \Delta^{\langle 1 \rangle}]\!]$ with $\Delta^{\langle 0 \rangle} := \mu^{\langle 0 \rangle} - \lambda^{\langle 0 \rangle}$ and $\Delta^{\langle 1 \rangle} := \mu^{\langle 1 \rangle} - \lambda^{\langle 1 \rangle}$. The $\omega_\Delta + 1$ calls of ρ_{RBVG} and ρ_{ES} can be simulated by calling the simulator of ρ_{RBVG} on input $[\![b]\!]$, $|\mathcal{R}_i|$, $x_i^{\langle 0 \rangle}$, $x_i^{\langle 1 \rangle}$ where $x_i^{\langle 0 \rangle}, x_i^{\langle 1 \rangle} \leftarrow_{\$} [0, |\mathcal{R}_i|)$ and by calling the simulator of ρ_{ES} on input $[\![b]\!]$, $x_i^{\langle 0 \rangle}$, $x_i^{\langle 1 \rangle}$, $\mathcal{R}_i = (r_{i,0}, r_{i,1}, \ldots, r_{i,|\mathcal{R}_i|-1})$. The output of ρ_{ES} is set to $[\![\tilde{y}_i]\!] := [\![(1-b)\cdot y_i^{\langle 0 \rangle} + b \cdot y_i^{\langle 1 \rangle}]\!]$ where $y_i^{\langle 0 \rangle} := r_{i,x_i^{\langle 0 \rangle}}$ and $y_i^{\langle 1 \rangle} := r_{i,x_i^{\langle 1 \rangle}}$. Step 3 of Gate 5.8 can be simulated by calling the simulator of ρ_{ES} on inputs $[\![b]\!]$, $\Delta^{\langle 0 \rangle}$, $\Delta^{\langle 1 \rangle}$, $y_i^{\langle 0 \rangle}$, $y_i^{\langle 1 \rangle}$ $(\forall i \in \mathbb{N}^0_{\omega_\Delta})$. The output of ρ_{ES} is computed as $[\![\tilde{e}_\Delta]\!] := [\![(1-b)\cdot y^{\langle 0 \rangle}_{\Delta^{\langle 0 \rangle}} + b \cdot y^{\langle 1 \rangle}_{\Delta^{\langle 1 \rangle}}]\!]$. Finally, simulator S computes $[\![\tilde{e}]\!] := [\![(1-b)\cdot(y^{\langle 0 \rangle}_{\Delta^{\langle 0 \rangle}} + \lambda^{\langle 0 \rangle}) + b \cdot (y^{\langle 1 \rangle}_{\Delta^{\langle 1 \rangle}} + \lambda^{\langle 1 \rangle})]\!]$. Since the simulated values are consistent with those of a real gate execution, the simulated view and the real view are statistically indistinguishable. □

Complexity. The communication and round complexity of gate $\rho^{\text{M-M}}_{\text{RIE-}\mathcal{D}}$ is in $\mathcal{O}(\omega_\Delta \cdot (O^{m \in \mathcal{O}(log(|\mathcal{R}_{\omega_\Delta}|))}_{\text{RBVG}} + O^{n \in \mathcal{O}(|\mathcal{R}_{\omega_\Delta}|)}_{\text{ES}}))$, respectively.

Related Work: A first protocol implementing the functionality corresponding to gate functionality $\mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}$ for plaintext input and plaintext output where each party holds its own private interval and the random subinterval is sampled from the corresponding private overlap interval has been proposed in [86]. This protocol can trivially be modified to a gate implementing $\mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}$ and is based on rejection sampling. However, this protocol (and the gate that would result from it) entails a major leakage which is due to its lack of providing data obliviousness. This leakage results from a loop where the number of iterations depend on the width of the private overlap interval and can disclose information on the size of this interval. Other related approaches presented in [86] have the limitations that they either have a (small) error probability and also an information leakage or there is a chance of non-termination. More recently, a gate that is more efficient than $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$ (see Gate Specification 14) has been presented in [118] which has a communication and round complexity in $\mathcal{O}(\omega_\Delta \cdot O_{\text{LTE-SO}} + O^{m\in\mathcal{O}(1);n\in\mathcal{O}(\omega_\Delta)}_{\text{CRS-}i^*})$, respectively, and provides security against semi-honest adversaries. Thus, whenever utilizing a gate implementing $\mathcal{G}^{\mathrm{T}}_{\text{RSI-}\omega}$ we use the gate from [118]. However, gate $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$ (presented above) is a contribution to this thesis since it is the first proposed gate which can be used to design protocols achieving security under Definition 4.1.

5.3.5 Interval Shrinking

In the following, we present a gate functionality, referred to as *interval shrinking*, describing a gate that allows multiple parties to obliviously shrink an unknown interval (given by its encrypted bounds) in case that the interval width is beyond a given public threshold. Otherwise, the interval remains unchanged. At the end of the computation, no party learns whether or not the private interval shrunk.

Definition 5.16 ($\mathcal{G}^{\mathrm{M}}_{\text{IS-}\vartheta}$: Interval Shrinking)**.** *Let all parties P_ℓ ($\ell \in \mathscr{P}$) hold the encrypted interval bounds $[\![\lambda]\!]$, $[\![\mu]\!]$ of an unknown interval $[\lambda, \mu]$ and a publicly known even integer ϑ. Let $\Delta := \mu - \lambda$ and let $m := \lceil \Delta/2 \rceil + \lambda$. Then, gate functionality $\mathcal{G}^{\mathrm{M}}_{\text{IS-}\vartheta}$ is given by $(([\![\lambda_o]\!], [\![\mu_o]\!])) \leftarrow \mathcal{G}^{\mathrm{M}}_{\text{IS-}\vartheta}(([\![\lambda]\!], [\![\mu]\!]))$, where*

$$([\![\lambda_o]\!], [\![\mu_o]\!]) := \begin{cases} ([\![\lambda]\!], [\![\mu]\!]) & \text{if } \Delta \leq \vartheta \\ ([\![m - \vartheta/2]\!], [\![m + \vartheta/2]\!]) & \text{otherwise} \end{cases}$$

Gate Specification 17. In order to devise a gate $\rho^{\text{M-M}}_{\text{IS-}\vartheta}$ which securely implements $\mathcal{G}^{\mathrm{M}}_{\text{IS-}\vartheta}$ it is necessary to hide whether or not the private input interval is shrunken. Furthermore, one needs to devise an approach to compute the encrypted midst $[\![m]\!]$ of an encrypted interval given by $[\![\lambda]\!]$ and $[\![\mu]\!]$. For the case that the interval width is even, $[\![m]\!]$ can simply

Gate 5.9. Specification of $\rho^{\text{M-M}}_{\text{IS-}\vartheta}$.

1 Each party P_ℓ ($\ell \in \mathscr{P}$) locally computes $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!]$

2 All parties jointly compute $([\![w]\!]) \leftarrow \rho_{\text{Mod}}(([\![\Delta]\!], 2))$

3 Each party P_ℓ ($\ell \in \mathscr{P}$) locally computes $[\![m]\!] := ([\![\Delta]\!] +_h [\![w]\!])/_h 2 +_h [\![\lambda]\!]$

4 All parties jointly compute :

4.1 $([\![\Delta_{\text{bin}}]\!]) \leftarrow \rho_{\text{BD}}([\![\Delta]\!])$

4.2 $([\![o]\!]) \leftarrow \rho_{\text{BDI-GT}}(([\![\Delta_{\text{bin}}]\!], [\![\vartheta_{\text{bin}}]\!]))$

4.3 $([\![a_1]\!]) \leftarrow \rho_{\text{Mult}}(([\![o]\!], ([\![m]\!] -_h [\![\vartheta/2]\!])))$

4.4 $([\![a_2]\!]) \leftarrow \rho_{\text{Mult}}((([\![1]\!] -_h [\![o]\!]), [\![\lambda]\!]))$

4.5 $([\![b_1]\!]) \leftarrow \rho_{\text{Mult}}(([\![o]\!], ([\![m]\!] +_h [\![\vartheta/2]\!])))$

4.6 $([\![b_2]\!]) \leftarrow \rho_{\text{Mult}}((([\![1]\!] -_h [\![o]\!]), [\![\mu]\!]))$

5 Each party P_ℓ ($\ell \in \mathscr{P}$):

5.1 Locally compute $[\![\lambda_o]\!] := [\![a_1]\!] +_h [\![a_2]\!]$

5.2 Locally compute $[\![\mu_o]\!] := [\![b_1]\!] +_h [\![b_2]\!]$

5.3 Output $([\![\lambda_o]\!], [\![\mu_o]\!])$

be computed as $(([\![\mu]\!] -_h [\![\lambda]\!])/_h\ 2) +_h [\![\lambda]\!]$. If the interval width is odd, it is not possible to apply this homomorphic computation since the homomorphic division operation will yield an unexpected result. Thus, the challenge is to compute $[\![m]\!]$ without any information on $|[\lambda, \mu]|$.

Steps 1 - 3 in Gate 5.9 demonstrate how to compute $[\![m]\!]$ for the private interval $[\lambda, \mu]$ given by $[\![\lambda]\!]$, $[\![\mu]\!]$. First, both parties locally compute the interval width $[\![\Delta]\!] := [\![\mu]\!] -_h [\![\lambda]\!]$ and then determine $[\![w]\!] := [\![\Delta\ mod\ 2]\!]$ in Step 2. Note that w is a bit that indicates whether or not Δ is even. With the knowledge of $[\![w]\!]$ each party can now locally compute the encrypted interval midst $[\![m]\!] := ([\![\Delta]\!] +_h [\![w]\!])/_h\ 2 +_h [\![\lambda]\!]$. Note that homomorphically adding $[\![w]\!]$ ensures that $[\![\Delta]\!] +_h [\![w]\!]$ is even and thus the homomophic division operation in Step 3 yields the expected result. The remaining steps of Gate 5.9 obliviously compute the encrypted interval bounds of the target interval $[\lambda_o, \mu_o]$. In order to determine whether or not the interval has to be shrunken, it is necessary to compute $[\![o]\!] := [\![[\Delta > \vartheta]]\!]$ (cf. Step 4.2). Since the comparison gate which is used in Step 4.2 to de-

	$\|[\lambda,\mu]\|$	w	m	o	λ_o	μ_o
$\|[\lambda,\mu]\| \leq \vartheta$	even	0	$\Delta/2+\lambda$	0	λ	μ
	odd	1	$(\Delta+1)/2+\lambda$			
$\|[\lambda,\mu]\| > \vartheta$	even	0	$\Delta/2+\lambda$	1	$m-\vartheta/2$	$m+\vartheta/2$
	odd	1	$(\Delta+1)/2+\lambda$			

Table 5.5: Correctness of $\rho_{\text{IS-}\vartheta}^{\text{M-M}}$.

termine $[\![o]\!]$ expects encrypted binary input, it is necessary to first compute the encrypted bit representation of Δ (see Step 4.1). The encrypted left bound of the output interval is computed as $[\![\lambda_o]\!] := [\![o(m-\vartheta/2)+(1-o)\lambda]\!]$ (see Steps 4.3, 4.4, 5.1). Analogously, the encrypted right bound of the output interval can be computed (see Steps 4.5, 4.6, 5.2).

The correctness of gate $\rho_{\text{IS-}\vartheta}^{\text{M-M}}$ follows immediately from Table 5.5 which lists the (decrypted) gate output for $|[\lambda,\mu]| \leq \vartheta$ and $|[\lambda,\mu]| > \vartheta$.

Theorem 5.9. *Let $\lambda^{\langle 0\rangle}$, $\lambda^{\langle 1\rangle}$, $\mu^{\langle 0\rangle}$, $\mu^{\langle 1\rangle}$, ϑ, and $[\![b]\!]$ ($b \in \{0,1\}$) be given to simulator* S*. For inputs $[\![\tilde{\lambda}]\!] := [\![(1-b)\cdot\lambda^{\langle 0\rangle}+b\cdot\lambda^{\langle 1\rangle}]\!]$ and $[\![\tilde{\mu}]\!] := [\![(1-b)\cdot\mu^{\langle 0\rangle}+b\cdot\mu^{\langle 1\rangle}]\!]$, gate $\rho_{\text{IS-}\vartheta}^{\text{M-M}}$ can be simulated such that the simulated view is statistically indistinguishable from the real view.*

Proof. First, simulator S computes $[\![\tilde{\Delta}]\!] := [\![(1-b)\cdot\Delta^{\langle 0\rangle}+b\cdot\Delta^{\langle 1\rangle}]\!] = [\![(1-b)\cdot(\mu^{\langle 0\rangle}-\lambda^{\langle 0\rangle})+b\cdot(\mu^{\langle 1\rangle}-\lambda^{\langle 1\rangle})]\!]$. To simulate Step 2 of Gate 5.9, the simulator of ρ_{Mod} (see Gate Specification 3) is called on $\Delta^{\langle 0\rangle}, \Delta^{\langle 1\rangle}, 2, [\![b]\!]$ and the output is set to $[\![\tilde{w}]\!] := [\![(1-b)\cdot(\Delta^{\langle 0\rangle} \bmod 2)+b\cdot(\Delta^{\langle 1\rangle} \bmod 2)]\!]$. Next, simulator S computes $[\![\tilde{m}]\!] := [\![(1-b)\cdot m^{\langle 0\rangle}+b\cdot m^{\langle 1\rangle}]\!]$ where $m^{\langle b\rangle} = (\Delta^{\langle b\rangle}+w^{\langle b\rangle})/2+\lambda^{\langle b\rangle}$. To simulate Step 4 of Gate 5.9, the simulator of ρ_{BD} (see Gate Specification 2) is called on input $\Delta^{\langle 0\rangle}, \Delta^{\langle 1\rangle}, [\![b]\!]$, and the simulator of $\rho_{\text{BDI-GT}}$ (cf. Gate Specification 6) is called on input $\Delta_{\text{bin}}^{\langle 0\rangle}, \Delta_{\text{bin}}^{\langle 1\rangle}, \vartheta_{\text{bin}}, [\![b]\!]$. The output of ρ_{BD} (resp., $\rho_{\text{BDI-GT}}$) is simulated as $[\![\tilde{\Delta}_{\text{bin}}]\!] := [\![(1-b)\cdot\Delta_{\text{bin}}^{\langle 0\rangle}+b\cdot\Delta_{\text{bin}}^{\langle 1\rangle}]\!]$ (resp., $[\![\tilde{o}]\!] := [\![(1-b)\cdot o^{\langle 0\rangle}+b\cdot o^{\langle 1\rangle}]\!]$ where $o^{\langle b\rangle} := [\Delta^{\langle b\rangle} > \vartheta]$). Furthermore, the first call of ρ_{Mult} (see Gate Specification 1) is simulated by calling the simulator of ρ_{Mult} on input $([\![\tilde{o}]\!], [\![(1-b)\cdot(m^{\langle 0\rangle}-\vartheta/2)+b\cdot(m^{\langle 1\rangle}-\vartheta/2)]\!])$ and output $[\![\tilde{a}_1]\!] := [\![(1-b)\cdot a_1^{\langle 0\rangle}+b\cdot a_1^{\langle 1\rangle}]\!]$ where $a_1^{\langle b\rangle} := o^{\langle b\rangle}\cdot(m^{\langle b\rangle}-\vartheta/2)$. $[\![\tilde{a_2}]\!]$, $[\![\tilde{b_1}]\!]$, and $[\![\tilde{b_2}]\!]$ are simulated analogously. Finally, simulator S computes $[\![\tilde{\lambda}_o]\!] := [\![(1-b)\cdot(a_1^{\langle 0\rangle}+a_2^{\langle 0\rangle})+b\cdot(a_1^{\langle 1\rangle}+a_2^{\langle 1\rangle})]\!]$ and $[\![\tilde{\mu}_o]\!] := [\![(1-b)\cdot(b_1^{\langle 0\rangle}+b_2^{\langle 0\rangle})+b\cdot(b_1^{\langle 1\rangle}+b_2^{\langle 1\rangle})]\!]$. □

Complexity. The communication and round complexities of gate $\rho_{\text{IS-}\vartheta}^{\text{M-M}}$ are in $\mathcal{O}(O_{\text{Mod}}^{m_b\in\mathcal{O}(1)}+O_{\text{BD}}^{m\in\mathcal{O}(s)}+O_{\text{BDI-GT}}^{m\in\mathcal{O}(s)}+O_{\text{Mult}})$, respectively.

5.4 Summary

In this chapter, we presented several gates that are used as building blocks for designing our novel privacy-preserving bartering protocols. For newly proposed gates we dis-

cussed related work (if any). Table 5.1 gives an overview of all gates presented in this chapter. Furthermore, Table 5.6 summarizes the communication complexities and the round complexities of these gates.

Gate	Communication Complexity	Round Complexity
$\rho_{\text{Mult}}^{\text{SH-M}}$	$\mathcal{O}(\iota s)$	$\mathcal{O}(1)$
$\rho_{\text{Mult}}^{\text{M-M}}$	$\mathcal{O}(\iota s)$	$\mathcal{O}(1)$
$\rho_{\text{BD}}^{\text{M-M}}$	$\mathcal{O}(m \cdot O_{\text{Mult}})$	$\mathcal{O}(1)$
$\rho_{\text{Mod}}^{\text{M-M}}$	$\mathcal{O}(\iota s m_b)$	$\mathcal{O}(\iota)$
$\rho_{\text{LT-Bin-SO}}^{\text{SH-M}}$	$\mathcal{O}(\iota s + O_{\text{Mult}})$	$\mathcal{O}(\iota + O_{\text{Mult}})$
$\rho_{\text{LT-SO}}^{\text{SH-M}}$	$\mathcal{O}(\iota s)$	$\mathcal{O}(\iota)$
$\rho_{\text{BDI-LT}}^{\text{M-M}}$	$\mathcal{O}(\iota s m)$	$\mathcal{O}(log(m))$
$\rho_{\text{IS-LT-SCOT}}^{\text{SH-T}}$	$\mathcal{O}(O_{\text{LT-SO}})$	$\mathcal{O}(O_{\text{LT-SO}})$
$\rho_{\text{ET}}^{\text{SH-M}}$	$\mathcal{O}(\iota s + O_{\text{LT-SO}})$	$\mathcal{O}(\iota + O_{\text{LT-SO}})$
$\rho_{\text{BDI-ET}}^{\text{M-M}}$	$\mathcal{O}(O_{\text{Mult}} + O_{\text{BDI-LT}}^{m \in \mathcal{O}(s)})$	$\mathcal{O}(O_{\text{Mult}} + O_{\text{BDI-LT}}^{m \in \mathcal{O}(s)})$
$\rho_{\text{ES}}^{\text{M-M}}$	$\mathcal{O}(n \cdot O_{\text{Mult}})$	$\mathcal{O}(n \cdot O_{\text{Mult}})$
$\rho_{\text{CRS-}i^*}^{\text{SH-M}}$	$\mathcal{O}(\iota m n s + n \cdot O_{\text{GT-SO}})$	$\mathcal{O}(n \cdot O_{\text{GT-SO}} + \iota n)$
$\rho_{\text{CRS-}i^*}^{\text{M-M}}$	$\mathcal{O}(\iota m n s + n \cdot (O_{\text{BD}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-GT}}^{m \in \mathcal{O}(s)}) + nm \cdot O_{\text{Mult}})$	$\mathcal{O}(\iota + n \cdot (O_{\text{BD}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-GT}}^{m \in \mathcal{O}(s)}) + nm \cdot O_{\text{Mult}})$
$\rho_{\text{CRS-C-}i^*}^{\text{SH-M}}$	$\mathcal{O}(\iota m n s + n \cdot O_{\text{GT-SO}})$	$\mathcal{O}(n \cdot O_{\text{GT-SO}} + \iota n)$
$\rho_{\text{RBVG}}^{\text{M-M}}$	$\mathcal{O}(\iota s m)$	$\mathcal{O}(\iota)$
$\rho_{\text{RSI-}\omega}^{\text{SH-T}}$	$\mathcal{O}(\omega_\Delta \cdot O_{\text{LTE-SO}} + O_{\text{CRS-}i^*}^{m \in \mathcal{O}(1); n \in \mathcal{O}(\omega_\Delta)})$	$\mathcal{O}(\omega_\Delta \cdot O_{\text{LTE-SO}} + O_{\text{CRS-}i^*}^{m \in \mathcal{O}(1); n \in \mathcal{O}(\omega_\Delta)})$
$\rho_{\text{RIE}}^{\text{M-M}}$	$\mathcal{O}(\omega_\Delta \cdot O_{\text{RBVG}}^{m \in \mathcal{O}(log(\omega_\Delta))} + O_{\text{ES}}^{n \in \mathcal{O}(\omega_\Delta)})$	$\mathcal{O}(\omega_\Delta \cdot O_{\text{RBVG}}^{m \in \mathcal{O}(log(\omega_\Delta))} + O_{\text{ES}}^{n \in \mathcal{O}(\omega_\Delta)})$
$\rho_{\text{RIE-}\mathcal{D}}^{\text{M-M}}$	$\mathcal{O}(\omega_\Delta \cdot (O_{\text{RBVG}}^{m \in \mathcal{O}(log(\lvert\mathcal{R}_{\omega_\Delta}\rvert))} + O_{\text{ES}}^{n \in \mathcal{O}(\lvert\mathcal{R}_{\omega_\Delta}\rvert)}))$	$\mathcal{O}(\omega_\Delta \cdot (O_{\text{RBVG}}^{m \in \mathcal{O}(log(\lvert\mathcal{R}_{\omega_\Delta}\rvert))} + O_{\text{ES}}^{n \in \mathcal{O}(\lvert\mathcal{R}_{\omega_\Delta}\rvert)}))$
$\rho_{\text{IS-}\vartheta}^{\text{M-M}}$	$\mathcal{O}(O_{\text{Mod}}^{m_b \in \mathcal{O}(1)} + O_{\text{BD}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-GT}}^{m \in \mathcal{O}(s)} + O_{\text{Mult}})$	$\mathcal{O}(O_{\text{Mod}}^{m_b \in \mathcal{O}(1)} + O_{\text{BD}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-GT}}^{m \in \mathcal{O}(s)} + O_{\text{Mult}})$

Table 5.6: Gate complexities. Note that the role of parameters m and n vary depending on the gate.

CHAPTER 6

Privacy-Preserving Two-Party Bartering

In this chapter, we concentrate on privacy-preserving bartering between two parties as a preliminary study for the much more sophisticated multi-party case treated in the following chapter. Furthermore, the separate consideration of two-party and the multi-party case facilitates the presentation of the privacy-preserving multi-party bartering protocols since a subset of challenges are already addressed by the two-party protocols.

Parts of this Chapter are based on [48,121,122,124].

Contributions: The main contribution of this chapter is a privacy-preserving two-party bartering protocol which provides security in the malicious model. This protocol is joint work with Wadim Pessin (a graduate student at RWTH Aachen University at that time) and has been published in [124]. It is based on the privacy-preserving two-party bartering protocol (secure in the semi-honest model) from [48] which is also presented in this chapter. The latter protocol is the work of Fabian Förg and a contribution to the joint research project on privacy-preserving bartering between RWTH Aachen University and Stevens Institute of Technology but not a contribution of this thesis. A further contribution of this chapter is the introduction of several mechanisms that allow a privacy-preserving and unbiased negotiation of the quantities for the commodities to be traded. These mechanisms can also be used in the context of privacy-preserving multi-party bartering protocols and have been published in [121,122,124].

Outline: First, we provide a formalization of two-party bartering (see Section 6.1). In Section 6.2, we present a two-party privacy-preserving bartering protocol which is secure in the semi-honest model. Based on that, we devise a corresponding protocol pro-

viding security in the malicious model (see Section 6.3). Finally, we detail how to negotiate the actual quantities in a privacy-preserving and unbiased fashion in Section 6.4.

6.1 Setting and Notation

A trade between two parties generically indicates which quantity of which commodity a party receives from (resp., sends to) the other party. More precisely, we consider two parties P_0 and P_1 (i.e., $\mathscr{P} := \{0,1\}$) and a public finite set $\mathscr{C}$ of divisible commodities $c_1, \ldots, c_{|\mathscr{C}|}$. Each party P_ℓ ($\ell \in \mathscr{P}$) specifies exactly one *quote* $\mathbf{q}^{(\ell)} := (\mathbf{o}^{(\ell)}, \mathbf{d}^{(\ell)})$ where $\mathbf{o}^{(\ell)}$ and $\mathbf{d}^{(\ell)}$ is P_ℓ's *offer* and *demand*, respectively. We model $\mathbf{o}^{(\ell)}$ as a 2-tuple $\mathbf{o}^{(\ell)} := (c_o^{(\ell)}, \overline{q}_o^{(\ell)})$ where $c_o^{(\ell)} \in \mathscr{C}$ specifies the commodity offered by P_ℓ and $\overline{q}_o^{(\ell)} \in \mathbb{N}$ denotes the maximum quantity of $c_o^{(\ell)}$ offered. Similarly, we model $\mathbf{d}^{(\ell)} := (c_d^{(\ell)}, \underline{q}_d^{(\ell)})$ with $c_d^{(\ell)} \in \mathscr{C}$ and $\underline{q}_d^{(\ell)} \in \mathbb{N}$ which denotes the minimum quantity of $c_d^{(\ell)}$ desired. The *quantity ranges* of the offered and desired commodities of a party P_ℓ are thus defined as $Q_o^{(\ell)} := [1, \overline{q}_o^{(\ell)}]$ and $Q_d^{(\ell)} := [\underline{q}_d^{(\ell)}, \infty)$ indicating that P_ℓ is willing to send any quantity in $Q_o^{(\ell)}$ of $c_o^{(\ell)}$ in return for receiving any quantity in $Q_d^{(\ell)}$ of $c_d^{(\ell)}$. We exclude the case that a party offers the same commodity as it desires from our bartering setting.

Definition 6.1 (Potential Trade, Two-Party Case)**.** *A* potential trade *between two parties P_0 and P_1 is a trade such that ($\forall \ell \in \mathscr{P} = \{0,1\}$) P_ℓ receives a specific quantity $q_{c_o^{(1-\ell)}}^{(1-\ell,\ell)} \in Q_d^{(\ell)} \cap Q_o^{(1-\ell)}$ of commodity $c_d^{(\ell)}$ from $P_{1-\ell}$ and sends a specific quantity $q_{c_o^{(\ell)}}^{(\ell,1-\ell)} \in Q_o^{(\ell)} \cap Q_d^{(1-\ell)}$ of $c_o^{(\ell)}$ to $P_{1-\ell}$.*

Informally, a potential trade between two parties is a trade both parties are mutually satisfied with. One or more potential trades exist if the parties' quotes are *compatible* which is the case if the following condition holds:

$$(c_d^{(0)} = c_o^{(1)}) \wedge (c_d^{(1)} = c_o^{(0)}) \wedge (Q_d^{(0)} \cap Q_o^{(1)} \neq \emptyset) \wedge (Q_d^{(1)} \cap Q_o^{(0)} \neq \emptyset) \tag{6.1}$$

Definition 6.2 (Actual Trade, Two-Party Case)**.** *An* actual trade *between two parties is a specific potential trade where the actual quantities of the commodities to be traded, referred to as $q_{c_o^{(1-\ell)}}^{*(1-\ell,\ell)}$ ($\forall \ell \in \mathscr{P} = \{0,1\}$), are selected according to a specific preselected discrete distribution $\mathcal{D}$, written as $q_{c_o^{(1-\ell)}}^{*(1-\ell,\ell)} \leftarrow_{\mathcal{D}} Q_d^{(\ell)} \cap Q_o^{(1-\ell)}$.*

The question arises of how to choose a discrete distribution $\mathcal{D}$ in order to leak as little information as possible from a party's protocol output on the other party's specified quantity ranges as well as to determine the actual quantities in an unbiased and fair fashion; we return to this question later in Section 6.4.

$\mathscr{P}$		offer	demand	
P_0	$\mathbf{q}^{(0)} = ($	$(\mathtt{A}, 6),$	$(\mathtt{B}, 3)$	$)$
P_1	$\mathbf{q}^{(1)} = ($	$(\mathtt{B}, 5),$	$(\mathtt{A}, 5)$	$)$

Table 6.1: Compatible input quotes.

$\mathscr{P}$		offer	demand	
P_0	$\mathbf{q}^{(0)} = ($	$(\mathtt{A}, 5),$	$(\mathtt{B}, 3)$	$)$
P_1	$\mathbf{q}^{(1)} = ($	$(\mathtt{B}, 5),$	$(\mathtt{A}, 6)$	$)$

Table 6.2: Incompatible input quotes.

Definition 6.3 ($\mathcal{F}^{\mathrm{T}}_{\text{Barter-}\mathcal{D}}$: Bartering, Two-Party Case)**.** *Let P_0 hold private input $\mathbf{q}^{(0)}$ and let P_1 hold private input $\mathbf{q}^{(1)}$. Furthermore, let $\mathcal{D}$ be a publicly known fixed discrete distribution. Then, functionality $\mathcal{F}^{\mathrm{T}}_{\text{Barter-}\mathcal{D}}$ is given by $((q^{*(0,1)}_{c_o^{(0)}}, q^{*(1,0)}_{c_o^{(1)}})) \leftarrow \mathcal{F}^{\mathrm{T}}_{\text{Barter-}\mathcal{D}}(\mathbf{q}^{(0)}, \mathbf{q}^{(1)})$ where $q^{*(0,1)}_{c_o^{(0)}} \leftarrow_{\mathcal{D}} Q_o^{(0)} \cap Q_d^{(1)}$ and $q^{*(1,0)}_{c_o^{(1)}} \leftarrow_{\mathcal{D}} Q_o^{(1)} \cap Q_d^{(0)}$ if Condition 6.1 holds and $q^{*(0,1)}_{c_o^{(0)}} = q^{*(1,0)}_{c_o^{(1)}} = 0$ otherwise.*

The definition of $\mathcal{F}^{\mathrm{T}}_{\text{Barter-}\mathcal{D}}$ assures that when implemented as an SMPC protocol, two parties can securely determine whether or not they are willing to trade. During the protocol execution, the quotes of both parties remain private. In case that no potential trade exists, no further information is revealed. Otherwise, a party merely learns at which quantity it receives (resp., sends) its desired (resp., offered) commodity.

Example 6.1. To clarify the bartering specific notation for the two-party case, in the following, we consider two examples of possible input quotes (see Tables 6.1 and 6.2). In the first example (see Table 6.1), P_0 is willing to give away at most six units of commodity $\mathtt{A}$ for at least three units of commodity $\mathtt{B}$ while P_1 is willing to give at most five units of commodity $\mathtt{B}$ for at least five units of commodity $\mathtt{A}$. Overall, there are six potential trades between P_0 and P_1 where P_0 receives commodity $\mathtt{B}$, P_1 receives commodity $\mathtt{A}$, and $(q^{(0,1)}_{c_o^{(0)}}, q^{(1,0)}_{c_o^{(1)}}) \in Q := (Q_o^{(0)} \cap Q_d^{(1)}) \times (Q_o^{(1)} \cap Q_d^{(0)}) = \{(5,3),(6,3),(5,4),(6,4),(5,5),(6,5)\}$. In order to determine the actual trade, according to Definition 6.3, $(q^{*(0,1)}_{c_o^{(0)}}, q^{*(1,0)}_{c_o^{(1)}})$ has to be selected from Q according to distribution $\mathcal{D}$.
The second example (see Table 6.2) is a slight variation of the first example where $\overline{q}_o^{(0)}$ is reduced by one and $\underline{q}_d^{(1)}$ is increased by one. This modification of the input quotes implies that there is no potential trade between P_0 and P_1. The reason for this is that $Q_o^{(0)} \cap Q_d^{(1)} = \emptyset$ and, thus, the quotes of the two parties do not satisfy Condition 6.1.

6.2 Two-Party Bartering Secure in the Semi-Honest Model

In this section, we review protocol $\pi^{\text{SH-T}}_{\text{Barter-}\mathcal{D}}$ from [48] which securely implements functionality $\mathcal{F}^{\mathrm{T}}_{\text{Barter-}\mathcal{D}}$ (see Definition 6.3) in the semi-honest model. In the following, we represent a commodity $c_o^{(\ell)}$ offered by P_ℓ as a $|\mathscr{C}|$-bit vector $c_o^{(\ell)} := (c_{o,1}^{(\ell)}, \ldots, c_{o,|\mathscr{C}|}^{(\ell)})$ where

$c_{o,i}^{(\ell)} = 1$ if $c_o^{(\ell)} = c_i \in \mathscr{C}$ and $c_{o,i}^{(\ell)} = 0$ otherwise ($i \in \mathbb{N}_{|\mathscr{C}|}$). The same notation is used for desired commodities.

6.2.1 Intuition

An SMPC protocol securely implementing functionality $\mathcal{F}_{\text{Barter-}\mathcal{D}}^{\text{T}}$ (in the semi-honest model) must assure that a party only learns it prescribed output and what can be deduced from its output in combination with its private input. Concretely, for $\mathcal{F}_{\text{Barter-}\mathcal{D}}^{\text{T}}$ this means that on providing its private input quote, a party merely learns whether or not Condition 6.1 is satisfied and in case that it is satisfies, a party additionally learns at which quantities the specified commodities are to be exchanged. However, a party is not allowed to learn any intermediate evaluation result of Condition 6.1. Thus, it is not sufficient to just evaluate the predicates of Condition 6.1 one by one and to combine the plain results by an AND operation. Instead the conjunction has to be evaluated in an intertwined fashion not leaking any intermediate results. This requirement can be achieved by subsequently evaluating the predicates of Condition 6.1 but including the encrypted result of the already evaluated predicates in the computation of predicates which still have to be evaluated. In order to securely evaluate whether or not the offered and desired commodities of P_0 and P_1 mutually match, we compute the scalar product on the encrypted vectors representing the parties commodities while gate $\rho_{\text{GT-SO}}$ (cf. Gate Specification 5), respectively, gate $\rho_{\text{LT-SO}}$ (see Gate Specification 5) securely evaluates whether or not the quantity ranges of the parties do mutually overlap.

After the secure evaluation of Condition 6.1, both parties jointly decrypt the result indicating whether or not the quotes are compatible. The remaining steps of $\pi_{\text{Barter-}\mathcal{D}}^{\text{SH-T}}$ depend on the parties' private input, i.e., the encrypted result determines whether the protocol terminates or the quantities of the commodities to be exchanged have yet to be securely computed. However, this dependency is not a privacy issue since the protocol flow is an implicit part of the parties' private output. In case that the parties' quotes are incompatible, $\pi_{\text{Barter-}\mathcal{D}}^{\text{SH-T}}$ terminates and both parties output $(0, 0)$ indicating that no commodities are to be exchanged. Otherwise, both parties have to participate in two executions of protocol $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) allowing them to determine the quantities of the commodities to be exchanged in a privacy-preserving fashion.

6.2.2 Specification of Protocol $\pi_{\text{Barter-}\mathcal{D}}^{\text{SH-T}}$

In the following, we provide the details of protocol $\pi_{\text{Barter-}\mathcal{D}}^{\text{SH-T}}$ (see Protocol 6.1) which securely implements $\mathcal{F}_{\text{Barter-}\mathcal{D}}^{\text{T}}$ (see Definition 6.3) in the semi-honest model. Protocol $\pi_{\text{Barter-}\mathcal{D}}^{\text{SH-T}}$ is split up into the following three phases:

Protocol 6.1. Specification of $\pi^{\text{SH-T}}_{\text{Barter-}\mathcal{D}}$.

1 Evaluation Phase

1.1 Party P_0:

1.1.1 Compute $[\![E]\!] := ([\![c^{(0)}_{d,1}]\!], \ldots, [\![c^{(0)}_{d,|\mathscr{C}|}]\!])$

1.1.2 Send $[\![E]\!]$ to party P_1

1.2 Party P_1:

1.2.1 Set $[\![e]\!] := [\![E[i]]\!]$ for $i \in \mathbb{N}_{|\mathscr{C}|}$ s.t. $c^{(1)}_{o,i} = 1$

1.2.2 Compute $[\![l_i]\!] := \begin{cases} [\![0]\!] & \text{if } c^{(1)}_{d,i} = 0 \\ \mathsf{Blind}([\![e]\!]) & \text{if } c^{(1)}_{d,i} = 1 \end{cases}, \ \forall i \in \mathbb{N}_{|\mathscr{C}|}$

1.2.3 Send $[\![L]\!] := ([\![l_1]\!], \ldots, [\![l_{|\mathscr{C}|}]\!])$ to party P_0

1.3 Party P_0 sets $[\![l]\!] := [\![L[i]]\!]$ for $i \in \mathbb{N}_{|\mathscr{C}|}$ s.t. $c^{(0)}_{o,i} = 1$

1.4 Party P_0 and P_1 jointly compute $(b_0, b_1) \leftarrow \rho_{\text{GT-SO}}(([\![\underline{q}^{(0)}_d]\!], [\![\overline{q}^{(1)}_o]\!]))$

1.5 Party P_0:

1.5.1 Set $([\![m_1]\!], [\![m_2]\!]) := \begin{cases} ([\![0]\!], \mathsf{Blind}([\![l]\!])) & \text{if } b_0 = 0 \\ (\mathsf{Blind}([\![l]\!]), [\![0]\!]) & \text{if } b_0 = 1 \end{cases}$

1.5.2 Send $[\![m_1]\!]$, $[\![m_2]\!]$ to party P_1

1.6 Party P_1 sets $[\![m]\!] := \begin{cases} [\![m_2]\!] & \text{if } b_1 = 0 \\ [\![m_1]\!] & \text{if } b_1 = 1 \end{cases}$

1.7 Party P_0 and P_1 jointly compute $(b'_0, b'_1) \leftarrow \rho_{\text{LT-SO}}(([\![\overline{q}^{(0)}_o]\!], [\![\underline{q}^{(1)}_d]\!]))$

1.8 Party P_1:

1.8.1 Set $([\![w_1]\!], [\![w_2]\!]) := \begin{cases} ([\![0]\!], \mathsf{Blind}([\![m]\!])) & \text{if } b'_1 = 0 \\ (\mathsf{Blind}([\![m]\!]), [\![0]\!]) & \text{if } b'_1 = 1 \end{cases}$

1.8.2 Send $[\![w_1]\!]$, $[\![w_2]\!]$ to party P_0

1.9 Party P_0 computes $[\![w]\!] := \begin{cases} [\![w_2]\!] & \text{if } b'_0 = 0 \\ [\![w_1]\!] & \text{if } b'_0 = 1 \end{cases}$

2 Negotiation Phase

2.1 Party P_0 and P_1 jointly compute $\mathsf{Dec}([\![w]\!])$

2.2 If $w = 0$: Party P_0 and P_1 set $(q^{*(0,1)}_{c^{(0)}_o}, q^{*(1,0)}_{c^{(1)}_o}) = (0, 0)$

2.3 Else: Party P_0 and P_1 jointly compute:

2.3.1 $([\![q^{*(1,0)}_{c^{(1)}_o}]\!]) \leftarrow \rho_{\text{RIE-}\mathcal{D}}(([\![\underline{q}^{(0)}_d]\!], [\![\overline{q}^{(1)}_o]\!]))$ and $([\![q^{*(0,1)}_{c^{(0)}_o}]\!]) \leftarrow \rho_{\text{RIE-}\mathcal{D}}(([\![\underline{q}^{(1)}_d]\!], [\![\overline{q}^{(0)}_o]\!]))$

2.3.2 $\mathsf{Dec}([\![q^{*(1,0)}_{c^{(1)}_o}]\!])$ and $\mathsf{Dec}([\![q^{*(0,1)}_{c^{(0)}_o}]\!])$ s.t. both parties learn the results

3 Output Phase

3.1 Party P_0 and P_1 output $(q^{*(0,1)}_{c^{(0)}_o}, q^{*(1,0)}_{c^{(1)}_o})$

1. *Evaluation Phase*: In this phase, both parties privately compute whether or not Condition 6.1 is satisfied. Since the offered and desired commodities of the parties are represented as $|\mathscr{C}|$-bit vectors with Hamming weight 1, the conjunction of the first two predicates of Condition 6.1 can be computed as

$$(c_d^{(0)} \times c_o^{(1)}) \cdot (c_d^{(1)} \times c_o^{(0)}) = (((c_d^{(0)} \times c_o^{(1)}) \cdot c_d^{(1)}) \times c_o^{(0)}), \tag{6.2}$$

where the result indicates whether or not $(c_d^{(0)} = c_o^{(1)}) \wedge (c_d^{(1)} = c_o^{(0)})$. In order to securely compute Equation 6.2, P_0 first performs a componentwise encryption of its desired commodity and sends the resulting ciphertexts $[\![E]\!] := ([\![c_{d,1}^{(0)}]\!], \ldots, [\![c_{d,|\mathscr{C}|}^{(0)}]\!])$ to P_1. Party P_1 sets $[\![e]\!] := [\![E[i]]\!]$ for the unequivocally determined index $i \in \mathbb{N}_{|\mathscr{C}|}$ for which $c_{o,i}^{(1)} = 1$ (see Step 1.2.1, Protocol 6.1). Due to the fact that the entries of a vector representing a commodity are either 0 or 1 and due to the fact that the Hamming weight of such a vector is 1, we have that $e = c_d^{(0)} \times c_o^{(1)}$. In the next series of steps (Steps 1.2.2-1.3, Protocol 6.1) the parties obliviously combine $[\![e]\!]$ with the computation of the second scalar product $(c_d^{(1)} \times c_o^{(0)})$. First, P_1 computes a list of ciphertexts $[\![L]\!] := ([\![l_1]\!], \ldots, [\![l_{|\mathscr{C}|}]\!])$ where $[\![l_i]\!]$ $(\forall i \in \mathbb{N}_{|\mathscr{C}|})$ corresponds to a fresh encryption of zero if $c_{d,i}^{(1)} = 0$ and to $\mathsf{Blind}([\![e]\!])$ if $c_{d,i}^{(1)} = 1$ such that $L = (c_d^{(0)} \times c_o^{(1)}) \cdot c_d^{(1)}$ (cf. Equation 6.2). In the following step, P_1 sends $[\![L]\!]$ to P_0. Similarly to Step 1.2.1 of Protocol 6.1, P_0 selects $[\![L[i]]\!]$ for index $i \in \mathbb{N}_{|\mathscr{C}|}$ for which $c_{o,i}^{(0)} = 1$ and sets $[\![l]\!] := [\![L[i]]\!]$. Note that $[\![l]\!]$ is an encryption of $l = (c_d^{(0)} \times c_o^{(1)}) \cdot (c_d^{(1)} \times c_o^{(0)})$.

In Step 1.4, both parties jointly compute $\rho_{\text{GT-SO}}$ (cf. Gate Specification 5) on input $[\![\underline{q}_d^{(0)}]\!]$ and $[\![\overline{q}_o^{(1)}]\!]$ (P_0 encrypts $\underline{q}_d^{(0)}$, P_1 encrypts $\overline{q}_o^{(1)}$) returning the XOR shared comparison result to the parties such that at the end of Step 1.4, P_0 learns b_0 and P_1 learns b_1 where $b_0 \oplus b_1 = 0$ iff $\underline{q}_d^{(0)} \leq \overline{q}_o^{(1)}$ implying that $Q_d^{(0)} \cap Q_o^{(1)} \neq \emptyset$. In order to combine the XOR shared result of the comparison operation with the encrypted result of the scalar multiplication operations, P_0 computes $([\![m_1]\!], [\![m_2]\!])$ depending on b_0. More precisely, $([\![m_1]\!], [\![m_2]\!])$ is set to $([\![0]\!], \mathsf{Blind}([\![l]\!]))$ if $b_0 = 0$ and set to $(\mathsf{Blind}([\![l]\!]), [\![0]\!])$ otherwise. Subsequently, P_0 sends $[\![m_1]\!]$ and $[\![m_2]\!]$ to P_1, whereupon P_1 sets $[\![m]\!] := [\![m_2]\!]$ if $b_1 = 0$ and $[\![m]\!] := [\![m_1]\!]$ otherwise. Note that $[\![m]\!]$ is an encryption of $\mathsf{Blind}([\![l]\!])$ if $b_0 \oplus b_1 = 0$ and, otherwise, $[\![m]\!]$ corresponds to a fresh encryption of 0. In order to securely determine whether or not the fourth clause of Condition 6.1 is satisfied as well as to obliviously combine this result with ciphertext $[\![m]\!]$, the same idea as in Steps 1.4 - 1.6 can be applied (cf. Steps 1.7 - 1.9, Protocol 6.1) such that the result of the evaluation phase is a ciphertext $[\![w]\!]$ where $w \in \{0, 1\}$ indicates whether or not Condition 6.1 is satisfied.

2. *Negotiation Phase*: First, both parties jointly decrypt $[\![w]\!]$. In case that $w = 0$, the

quotes of the parties are incompatible and $(q^{*(0,1)}_{c_o^{(0)}}, q^{*(1,0)}_{c_o^{(1)}})$ is set to $(0,0)$. Otherwise, there exists at least one potential trade for P_0 and P_1 and the actual trade is determined by executing $([\![q^{*(1,0)}_{c_o^{(1)}}]\!]) \leftarrow \rho_{\text{RIE-}\mathcal{D}}(([\![\underline{q}_d^{(0)}]\!], [\![\overline{q}_o^{(1)}]\!]))$ and $([\![q^{*(0,1)}_{c_o^{(0)}}]\!]) \leftarrow \rho_{\text{RIE-}\mathcal{D}}(([\![\underline{q}_d^{(1)}]\!], [\![\overline{q}_o^{(0)}]\!]))$ (see Gate Specification 16) followed by the decryption of $[\![q^{*(1,0)}_{c_o^{(1)}}]\!]$ and $[\![q^{*(0,1)}_{c_o^{(0)}}]\!]$. Note that the decryption of $[\![w]\!]$ and the associated protocol flow does not constitute a privacy breach since each party can deduce this information from its own output.

3. *Output Phase*: Both parties P_0 and P_1 output $q^{*(0,1)}_{c_o^{(0)}}$ and $q^{*(1,0)}_{c_o^{(1)}}$.

The correctness and the security (in the semi-honest model) of $\pi^{\text{SH-T}}_{\text{Barter-}\mathcal{D}}$ are proven in [48] where $\rho_{\text{RIE-}\mathcal{D}}$ is replaced by $\rho_{\text{RSI-}\omega}$.

(Complexity.) The communication and round complexities of protocol $\pi^{\text{SH-T}}_{\text{Barter-}\mathcal{D}}$ are in $\mathcal{O}(|\mathscr{C}|s + O_{\text{LT-SO}} + O_{\text{RIE-}\mathcal{D}})$ and $\mathcal{O}(O_{\text{LT-SO}} + O_{\text{RIE-}\mathcal{D}})$, respectively. By assuming the concrete specifications of the gates presented in Section 5 whose complexities are given in Table 5.6 and by setting $\mathcal{D}$, e.g., to the uniform distribution, the theoretical complexities are $\mathcal{O}(|\mathscr{C}|s + \omega_\Delta^2 \iota s)$ and $\mathcal{O}(\omega_\Delta \iota + \omega_\Delta^2)$, respectively, where ω_Δ corresponds to an upper bound on $|Q_d^{(1-\ell)} \cap Q_o^{(\ell)}|$ $(\forall \ell \in \mathscr{P})$.

6.3 Two-Party Bartering Secure in the Malicious Model

In this section, we introduce protocol $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ which securely implements $\mathcal{F}^{\text{T}}_{\text{Barter-}\mathcal{D}}$ (see Definition 6.3) in the malicious model. The specification of protocol $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ is the main contribution of this chapter and is based on [124].

6.3.1 Intuition

The basic approach of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ is similar to $\pi^{\text{SH-T}}_{\text{Barter-}\mathcal{D}}$. The essential differences are reflected, on the one hand, in the design of the used gates (see Chapter 5) and, on the other hand, in the design restrictions associated with the malicious model as well as the additional mechanism required to enforce semi-honest behavior or to catch a cheating party.

Compared to $\pi^{\text{SH-T}}_{\text{Barter-}\mathcal{D}}$, protocol $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ requires an additional input phase (cf. Section 4.4.3). In this phase, each party has to commit to its input and has to prove that it knows the corresponding plaintext(s). Due to the design of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$, a party has to provide its input in integer as well as in binary representation. Thus, a party has to additionally prove that its input w.r.t. both representations is consistent. Furthermore, it is

Protocol 6.2. Specification of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$.

1 Input Phase

1.1 P_0 and P_1 jointly compute $((\llbracket \mathbf{q}^{(0)} \rrbracket, \llbracket \mathbf{q}^{(0)}_{\text{bin}} \rrbracket, \llbracket \mathbf{q}^{(1)} \rrbracket, \llbracket \mathbf{q}^{(1)}_{\text{bin}} \rrbracket)) \leftarrow \mathsf{InputPhase}(\mathbf{q}^{(0)}, \mathbf{q}^{(1)})$

2 Evaluation Phase

2.1 P_0 and P_1 jointly compute:

2.1.1 $(\llbracket b_0 \rrbracket) \leftarrow \rho_{\text{BDI-ET}}((\llbracket (c_d^{(0)})_{\text{bin}} \rrbracket, \llbracket (c_o^{(1)})_{\text{bin}} \rrbracket))$

2.1.2 $(\llbracket b_1 \rrbracket) \leftarrow \rho_{\text{BDI-ET}}((\llbracket (c_d^{(1)})_{\text{bin}} \rrbracket, \llbracket (c_o^{(0)})_{\text{bin}} \rrbracket))$

2.1.3 $(\llbracket b_2 \rrbracket) \leftarrow \rho_{\text{BDI-LTE}}((\llbracket (\underline{q}_d^{(1)})_{\text{bin}} \rrbracket, \llbracket (\overline{q}_o^{(0)})_{\text{bin}} \rrbracket))$

2.1.4 $(\llbracket b_3 \rrbracket) \leftarrow \rho_{\text{BDI-LTE}}((\llbracket (\underline{q}_d^{(0)})_{\text{bin}} \rrbracket, \llbracket (\overline{q}_o^{(1)})_{\text{bin}} \rrbracket))$

2.1.5 $(\llbracket c \rrbracket) \leftarrow \rho_{\text{UFI-Mult}}((\llbracket b_0 \rrbracket, \llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket, \llbracket b_3 \rrbracket))$

3 Negotiation Phase

3.1 P_0 and P_1 jointly compute:

3.1.1 $(\llbracket d_1 \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket \underline{q}_d^{(0)} \rrbracket, \llbracket c \rrbracket))$

3.1.2 $(\llbracket d_2 \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket \overline{q}_o^{(1)} \rrbracket, \llbracket c \rrbracket))$

3.1.3 $(\llbracket d_3 \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket \underline{q}_d^{(1)} \rrbracket, \llbracket c \rrbracket))$

3.1.4 $(\llbracket d_4 \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket \overline{q}_o^{(0)} \rrbracket, \llbracket c \rrbracket))$

3.1.5 $(\llbracket q^{*(1,0)}_{c_o^{(1)}} \rrbracket) \leftarrow \rho_{\text{RIE-}\mathcal{D}}((\llbracket d_1 \rrbracket, \llbracket d_2 \rrbracket))$

3.1.6 $(\llbracket q^{*(0,1)}_{c_o^{(0)}} \rrbracket) \leftarrow \rho_{\text{RIE-}\mathcal{D}}((\llbracket d_3 \rrbracket, \llbracket d_4 \rrbracket))$

4 Output Phase

4.1 P_0 and P_1 jointly compute $((q^{*(1,0)}_{c_o^{(1)}}, q^{*(0,1)}_{c_o^{(0)}})) \leftarrow \mathsf{OutputPhase}((\llbracket q^{*(1,0)}_{c_o^{(1)}} \rrbracket, \llbracket q^{*(0,1)}_{c_o^{(0)}} \rrbracket))$

not possible to hinge the computation of the actual quantities on whether or not Condition 6.1 is satisfied by the parties' input quotes. Instead, two executions of gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) have to be executed (allowing the computation of $\llbracket q^{*(1,0)}_{c_o^{(1)}} \rrbracket$ and $\llbracket q^{*(0,1)}_{c_o^{(0)}} \rrbracket$) regardless of how Condition 6.1 evaluates. This requires additional precaution measures that have to be integrated into the protocol which ensure that in case Condition 6.1 is not satisfied, both executions of gate $\rho_{\text{RIE-}\mathcal{D}}$ return fresh encryptions of zero.

6.3.2 Specification of Protocol $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$

In the following, the details of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ are presented. In particular, we detail the input phase, the computation phase, and the output phase (see Section 4.4.3), where the computation phase is split into an evaluation phase and a negotiation phase.

1. *Input Phase:* At the beginning of the protocol, P_ℓ ($\ell \in \{0,1\}$) holds private input $\mathbf{q}^{(\ell)} = ((c_o^{(\ell)}, \overline{q}_o^{(\ell)}), (c_d^{(\ell)}, \underline{q}_d^{(\ell)}))$. Since $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ is composed of gates expecting encrypted integer input as well as encrypted binary input and because the design strives to reduce the calls of the expensive ρ_{BD} gate (see Gate Specification 2), the protocol requires both parties to commit to their private input as encrypted integers $[\![\mathbf{q}^{(\ell)}]\!] := (([\![c_o^{(\ell)}]\!], [\![\overline{q}_o^{(\ell)}]\!]), ([\![c_d^{(\ell)}]\!], [\![\underline{q}_d^{(\ell)}]\!]))$ as well as encrypted bit decompositions $[\![\mathbf{q}_{\text{bin}}^{(\ell)}]\!] := (([\![(c_o^{(\ell)})_{\text{bin}}]\!], [\![(\overline{q}_o^{(\ell)})_{\text{bin}}]\!]), ([\![(c_d^{(\ell)})_{\text{bin}}]\!], [\![(\underline{q}_d^{(\ell)})_{\text{bin}}]\!]))$. Thus, it is not sufficient that both parties just mutually prove plaintext knowledge of their committed input. Instead, P_ℓ additionally has to prove the consistency of its inputs in order to guarantee the correctness of the protocol output. More precisely, P_ℓ has to perform the following proofs:

   Req 1.) P_ℓ has to prove that it knows the plaintexts of $[\![c_o^{(\ell)}]\!]$, $[\![c_d^{(\ell)}]\!]$, $[\![\overline{q}_o^{(\ell)}]\!]$, and $[\![\underline{q}_d^{(\ell)}]\!]$.

   Req 2.) P_ℓ has to prove that each $[\![(c_o^{(\ell)})_{\text{bin}}]\!]$, $[\![(c_d^{(\ell)})_{\text{bin}}]\!]$, $[\![(\overline{q}_o^{(\ell)})_{\text{bin}}]\!]$, $[\![(\underline{q}_d^{(\ell)})_{\text{bin}}]\!]$ is a tuple of encrypted bits.

   Req 3.) P_ℓ has to prove that $[\![(c_o^{(\ell)})_{\text{bin}}]\!]$, $[\![(c_d^{(\ell)})_{\text{bin}}]\!]$, $[\![(\overline{q}_o^{(\ell)})_{\text{bin}}]\!]$, and $[\![(\underline{q}_d^{(\ell)})_{\text{bin}}]\!]$ are in fact the encrypted bit decompositions corresponding to $[\![c_o^{(\ell)}]\!]$, $[\![c_d^{(\ell)}]\!]$, $[\![\overline{q}_o^{(\ell)}]\!]$, and $[\![\underline{q}_d^{(\ell)}]\!]$, respectively.

   Party P_ℓ can prove Req 1.) by participating in four POPK calls (see Section 3.3.2) where P_ℓ plays the role of the prover while $P_{1-\ell}$ plays the role of the verifier. In order to prove Req 2.), P_l participates in an appropriate number of POPB (see Section 3.3.2) calls in the role of the prover. As required for Req 3.), P_ℓ has to prove that the plaintexts of a commitment $[\![x_{bin}]\!] := ([\![x_0]\!], \ldots, [\![x_m]\!])$ correspond to the bit decomposition of the plaintext of a commitment $[\![x]\!]$ ($\forall x \in \{c_o^{(\ell)}, c_d^{(\ell)}, \overline{q}_o^{(\ell)}, \underline{q}_d^{(\ell)}\}$). First, both parties compute $[\![x']\!] := [\![x_0]\!] +_h 2^1 \times_h [\![x_1]\!] +_h \ldots +_h 2^m \times_h [\![x_m]\!]$, whereupon P_ℓ proves plaintext equality of $[\![x']\!]$ and $[\![x]\!]$ by participating in a single call of POPE (see Section 3.3.2). The input phase of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ is referred to as $(([\![\mathbf{q}^{(0)}]\!], [\![\mathbf{q}_{\text{bin}}^{(0)}]\!], [\![\mathbf{q}^{(1)}]\!], [\![\mathbf{q}_{\text{bin}}^{(1)}]\!])) \leftarrow \text{InputPhase}(\mathbf{q}^{(0)}, \mathbf{q}^{(1)})$.

2. *Evaluation Phase:* This phase obliviously evaluates whether or not Condition 6.1 holds. In Steps 2.1.1 and 2.1.2 it is checked whether or not the desired and offered commodities of the parties match. In Steps 2.1.3 and 2.1.4 it is checked whether or not the corresponding quantity ranges overlap. The AND operation of the intermediate results $[\![b_0]\!], \ldots, [\![b_3]\!]$ is computed in Step 2.1.5 by calling gate $\rho_{\text{UFI-Mult}}$ (see Section 5.1.1) on $[\![b_0]\!], \ldots, [\![b_3]\!]$.

3. *Negotiation Phase:* The goal of the second subphase (Steps 3.1.1 - 3.1.6, Protocol 6.2) is to determine the quantities $q_{c_o^{(1)}}^{*(1,0)}$ and $q_{c_o^{(0)}}^{*(0,1)}$. In order to assure that $q_{c_o^{(1)}}^{*(1,0)} =$

$q^{*(0,1)}_{c_o^{(0)}} = 0$ in case Condition 6.1 does not hold, the specified quantities for the desired and offered commodities of both parties are multiplied with the result of the evaluation in Steps 3.1.1-3.1.4. Finally, in Steps 3.1.5 and 3.1.6 gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) is called on input $([\![\underline{q}_d^{(0)}]\!], [\![\overline{q}_o^{(1)}]\!])$ (resp. $([\![\underline{q}_d^{(1)}]\!], [\![\overline{q}_o^{(0)}]\!])$) in case Condition 6.1 holds and on input $([\![0]\!], [\![0]\!])$ otherwise. The ciphertexts $[\![q^{*(1,0)}_{c_o^{(1)}}]\!]$ and $[\![q^{*(0,1)}_{c_o^{(0)}}]\!]$ constitute the result of the computation phase.

4. *Output Phase:* In the output phase of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$, the parties jointly decrypt $[\![q^{*(1,0)}_{c_o^{(1)}}]\!]$ and $[\![q^{*(0,1)}_{c_o^{(0)}}]\!]$ such that both learn the corresponding plaintext values. The output phase is referred to as $((q^{*(1,0)}_{c_o^{(1)}}, q^{*(0,1)}_{c_o^{(0)}})) \leftarrow \mathsf{OutputPhase}(([\![q^{*(1,0)}_{c_o^{(1)}}]\!], [\![q^{*(0,1)}_{c_o^{(0)}}]\!]))$.

In the following, we show the correctness of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$, i.e., that $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ always computes the correct output (even if one party is corrupted), provided that the protocol is not prematurely aborted. First, note that it is sufficient to check $[\underline{q}_d^{(1)} \leq \overline{q}_o^{(0)}]$ (resp., $[\underline{q}_d^{(0)} \leq \overline{q}_o^{(1)}]$) in order to determine whether or not $Q_d^{(0)} \cap Q_o^{(1)} \neq \emptyset$ (resp., $Q_d^{(1)} \cap Q_o^{(0)} \neq \emptyset$) holds. The encrypted result $[\![c]\!]$ computed in Step 2.1.5 corresponds to an encryption of 1 iff all predicates of Condition 6.1 are satisfied and otherwise corresponds to an encryption of 0. In case that $c = 0$, it holds that $q^{*(1,0)}_{c_o^{(1)}} = q^{*(0,1)}_{c_o^{(0)}} = 0$ since gate $\rho_{\text{RIE-}\mathcal{D}}$ returns a fresh encryption of 0 on input $([\![0]\!], [\![0]\!])$. Otherwise, $\rho_{\text{RIE-}\mathcal{D}}$ is called on the private overlap interval $Q_d^{(0)} \cap Q_o^{(1)} = [\underline{q}_d^{(0)}, \infty) \cap [1, \overline{q}_o^{(1)}] = [\underline{q}_d^{(0)}, \overline{q}_o^{(1)}]$ given by its encrypted bounds (resp., on the private overlap interval $Q_d^{(1)} \cap Q_o^{(0)}$). According to Definition 5.15, gate $\rho_{\text{RIE-}\mathcal{D}}$ returns $q^{*(1,0)}_{c_o^{(1)}} \leftarrow_{\mathcal{D}} Q_d^{(0)} \cap Q_o^{(1)}$ (resp., $q^{*(0,1)}_{c_o^{(0)}} \leftarrow_{\mathcal{D}} Q_d^{(1)} \cap Q_o^{(0)}$) as prescribed by Definition 6.3. Due to the fact that all gates $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ is composed of obtain symmetric input, a party does not have to prove that it provides the correct gate input (see Remark 4.6, Section 4.4.3). Thus, it is only left to be shown that it is sufficient to meet Req 1.) - Req 3.) as specified for the input phase of $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$. First note that by participating in the Σ-protocols for Req 1.) - Req 3.) in the role of the prover, a party P_ℓ ($\ell \in \mathscr{P}$) implicitly proves that it knows the plaintexts of $[\![(c_o^{(\ell)})_{\text{bin}}]\!]$, $[\![(c_d^{(\ell)})_{\text{bin}}]\!]$, $[\![(\overline{q}_o^{(\ell)})_{\text{bin}}]\!]$, $[\![(\underline{q}_d^{(\ell)})_{\text{bin}}]\!]$. Furthermore, a party does not have to prove that $[\![c_o^{(\ell)}]\!]$, $[\![c_d^{(\ell)}]\!]$, $[\![\overline{q}_o^{(\ell)}]\!]$, $[\![\underline{q}_d^{(\ell)}]\!]$ are such that $c_o^{(\ell)}, c_d^{(\ell)} \in \mathscr{C}$ and that $\overline{q}_o^{(\ell)}, \underline{q}_d^{(\ell)} \neq 0$. This is due to the fact that choosing inadmissible input values does not influence the correctness of the protocol output which is detailed in the following. W.l.o.g., let P_0 be corrupted:

Case 1.) Assume that $c_o^{(0)} \notin \mathscr{C}$. When evaluating Condition 6.1, the second equality check (Step 2.1.2, Protocol 6.2) will always fail. Consequently, Condition 6.1 cannot be fulfilled and $\rho_{\text{UFI-Mult}}$ (Step 2.1.5, Protocol 6.2) returns $[\![c]\!]$ with corresponding plaintext $c = 0$. As expected, the protocol output is $(0, 0)$. The case

$c_d^{(0)} \notin \mathscr{C}$ is analogous.

Case 2.) Assume that $\underline{q}_o^{(0)} = 0$. Then, the first LTE check (Step 2.1.3, Protocol 6.2) will fail since $\underline{q}_d^{(1)} > 0$. This results in the same outcome as in Case 1.).

Case 3.) Assume that $\underline{q}_d^{(0)} = 0$. Since $0 \notin Q_d^{(0)} \cap Q_o^{(1)} = [0, \infty) \cap [1, \overline{q}_o^{(1)}]$, selecting $\underline{q}_d^{(0)} = 0$ does not influence the computation of $q_{c_o^{(1)}}^{*(1,0)}$.

Case 4.) Obviously, any combination of this malicious behavior does not influence the correctness of the protocol.

The security of protocol $\pi_{\text{Barter-}\mathcal{D}}^{\text{M-T}}$ directly follows from Theorem 4.3 (see Section 4.4.3):

Corollary 6.1. *Protocol $\pi_{\text{Barter-}\mathcal{D}}^{\text{M-T}}$ securely computes functionality $\pi_{\text{Barter-}\mathcal{D}}^{\text{M-T}}$ in the presence of an active adversaries that corrupts one party.*

(Complexity.) The communication and round complexities of protocol $\pi_{\text{Barter-}\mathcal{D}}^{\text{M-T}}$ are in $\mathcal{O}(O_{\text{BDI-ET}}^{m \in \mathcal{O}(s)} + O_{\text{BDI-LTE}}^{m \in \mathcal{O}(s)} + O_{\text{Mult}} + O_{\text{RIE-}\mathcal{D}})$, respectively. By assuming the concrete specifications of the gates presented in Section 5 whose complexities are given in Table 5.6 and by setting $\mathcal{D}$, e.g., to the uniform distribution, the exact complexities are $\mathcal{O}(\iota s^2 + \omega_\Delta^2 \iota s)$ and $\mathcal{O}(log(s) + \omega_\Delta \iota + \omega_\Delta^2)$, respectively, where ω_Δ corresponds to an upper bound on $|Q_d^{(1-\ell)} \cap Q_o^{(\ell)}|$ $(\forall \ell \in \mathscr{P})$.

6.4 Privacy-Preserving Unbiased Negotiation

Until now, we did not further specify how exactly the actual quantities $q_{c_o^{(1)}}^{*(1,0)}$ and $q_{c_o^{(0)}}^{*(0,1)}$ of the commodities to be traded are computed. Instead, we left open the question of how to instantiate the discrete distribution $\mathcal{D}$ that influences the automatic negotiation of the actual quantities. In the following, we motivate our privacy-preserving negotiation approach and propose two concrete possibilities to instantiate $\mathcal{D}$ (the uniform distribution and the binomial distribution). Furthermore, we present an approach for reducing the chance of imbalances between the parties when determining the actual quantities. It is important to note that the following discussions and results are independent of the adversary model.

Assume that P_0 and P_1 privately identified each other as trade partners, i.e., $\mathbf{q}^{(0)}$ and $\mathbf{q}^{(1)}$ satisfy Condition 6.1 (see Section 6.1). Next, the parties have to negotiate the actual quantities at which they will trade their commodities. Obviously, the parties can negotiate outside of a privacy-preserving bartering protocol after they identified each other. In this case, the negotiation of the quantities is left to the parties and a party which, e.g., has to make the first offer or which has the least bargaining experience may

be discriminated. In order to compensate for such a disadvantage, a party may elect to lie about the quantities it is willing to accept. We address this problem by integrating a negotiation mechanism into our privacy-preserving bartering protocols which enables the parties to negotiate the actual quantities in an unbiased fashion while keeping their specified quantity ranges private. As such, this approach motivates the parties to privately specify their true negotiation ranges. Since the true negotiation ranges remain private even after the protocol execution, our approach additionally ensures that no information on a parties' remaining demand of its desired commodity or a surplus of its offered commodity is leaked which could be exploited in unforeseeable ways.

At first glance, one could determine $q_{c_o^{(1)}}^{*(1,0)}$ (resp., $q_{c_o^{(0)}}^{*(0,1)}$) as the midst of the overlap interval $Q_o^{(1)} \cap Q_d^{(0)}$ (resp., $Q_o^{(0)} \cap Q_d^{(1)}$) in a privacy-preserving fashion by exploiting the homomorphic operations of the underlying cryptosystem. However, this procedure leaks the quantity ranges of a party to its opponent since they can be computed from the midst and the opponent's own quantity ranges. Therefore, our protocols utilize a generic gate $\rho_{\text{RIE-}\mathcal{D}}$ for sampling from a private interval ($Q_o^{(1)} \cap Q_d^{(0)}$ and $Q_o^{(0)} \cap Q_d^{(1)}$) according to an arbitrary discrete distribution $\mathcal{D}$. This gate has been specified for the malicious model in Chapter 5 (see Gate Specification 16), however, a protocol that provides security in the semi-honest model can be designed analogously. By instantiating $\mathcal{D}$ in $\rho_{\text{RIE-}\mathcal{D}}$ with the uniform distribution, i.e., $q_{c_o^{(1)}}^{*(1,0)} \leftarrow_{\$} Q_o^{(1)} \cap Q_d^{(0)}$ (resp. $q_{c_o^{(0)}}^{*(0,1)} \leftarrow_{\$} Q_o^{(0)} \cap Q_d^{(1)}$), the actual quantities are selected in an unbiased fashion honoring the specification of the quantity ranges while at the same time the interval width $|Q_o^{(1)} \cap Q_d^{(0)}|$ (resp., $|Q_o^{(0)} \cap Q_d^{(1)}|$) is kept private (and with that the quantities specified by the parties' quotes). Note that instantiating $\mathcal{D}$ with the uniform distribution results in a communication and round complexity of $\mathcal{O}(\iota s \omega_\Delta^2)$ and $\mathcal{O}(\iota \omega_\Delta + \omega_\Delta^2)$, respectively, for gate $\rho_{\text{RIE-}\mathcal{D}}$. In this case, it is more efficient to replace $\rho_{\text{RIE-}\mathcal{D}}$ by $\rho_{\text{RSI-}\omega}^{\text{SH-T}}$ (see Gate Specification 14) with $\omega = 0$ in the semi-honest model or by $\rho_{\text{RIE}}^{\text{M-M}}$ (see Gate Specification 15) in the malicious model where the communication and round complexities of $\rho_{\text{RSI-}\omega}^{\text{SH-T}}$ are in $\mathcal{O}(\iota s \omega_\Delta)$ and $\mathcal{O}(\iota \omega_\Delta)$ while the complexities of $\rho_{\text{RIE}}^{\text{M-M}}$ are $\mathcal{O}(\iota s \omega_\Delta \cdot log(\omega_\Delta))$ and $\mathcal{O}(\iota \omega_\Delta)$.

However, instantiating $\mathcal{D}$ with the uniform distribution bears the risk of an extremely imbalanced determination of the actual quantities as the following example illustrates. Let $\mathtt{A}, \mathtt{B} \in \mathscr{C}$. Party P_0 holds private input $\mathbf{q}^{(0)} = (\mathbf{o}^{(0)}, \mathbf{d}^{(0)})$ with $\mathbf{o}^{(0)} = (\mathtt{A}, 10)$ and $\mathbf{d}^{(0)} = (\mathtt{B}, 10)$ while P_1 holds private input $\mathbf{q}^{(1)} = (\mathbf{o}^{(1)}, \mathbf{d}^{(1)})$ with $\mathbf{o}^{(1)} = (\mathtt{B}, 20)$ and $\mathbf{d}^{(1)} = (\mathtt{A}, 1)$. There is the chance that the determined actual quantities correspond to $q_{c_o^{(1)}}^{*(1,0)} = 20$ and $q_{c_o^{(0)}}^{*(0,1)} = 1$ which constitutes the best possible trade for P_0 and the worst case trade for P_1 for the given quotes.

One possibility to reduce the risk of extremely imbalanced actual quantities is to instantiate $\mathcal{D}$ with a discrete distribution that concentrates the probability mass around the

midst of the private interval. An example of such a discrete distribution is the binomial distribution (with success probability 0.5) described in Section 5.3.4. This approach ensures that the quantities around the midst of interval $Q_o^{(1)} \cap Q_d^{(0)}$ (resp., $Q_o^{(0)} \cap Q_d^{(1)}$) are selected with a higher probability than quantities near the interval bounds. Since each entry in $Q_o^{(1)} \cap Q_d^{(0)}$ ($Q_o^{(0)} \cap Q_d^{(1)}$) is still possible for $q_{c_o^{(1)}}^{*(1,0)}$ (resp., $q_{c_o^{(0)}}^{*(0,1)}$), the parties are not able to make any reliable assumptions on each others' quantity ranges.[1]

Another possibility to *prevent* the selection of extremely imbalanced actual quantities is to utilize gate $\rho_{\text{IS-}\vartheta}$ (see Gate Specification 17) before executing $\rho_{\text{RIE-}\mathcal{D}}$. More precisely, for a preselected even integer ϑ, the parties execute $((\llbracket\lambda\rrbracket, \llbracket\mu\rrbracket)) \leftarrow \rho_{\text{IS-}\vartheta}((\llbracket\underline{q}_d^{(0)}\rrbracket, \llbracket\overline{q}_o^{(1)}\rrbracket))$ and $((\llbracket\lambda'\rrbracket, \llbracket\mu'\rrbracket)) \leftarrow \rho_{\text{IS-}\vartheta}((\llbracket\underline{q}_d^{(1)}\rrbracket, \llbracket\overline{q}_o^{(0)}\rrbracket))$ followed by executing $q_{c_o^{(1)}}^{*(1,0)} \leftarrow \rho_{\text{RIE-}\mathcal{D}}((\llbracket\lambda\rrbracket, \llbracket\mu\rrbracket))$ and $q_{c_o^{(0)}}^{*(0,1)} \leftarrow \rho_{\text{RIE-}\mathcal{D}}((\llbracket\lambda'\rrbracket, \llbracket\mu'\rrbracket))$. This approach assures that $q_{c_o^{(1)}}^{*(1,0)}$ (resp., $q_{c_o^{(0)}}^{*(0,1)}$) is selected from $[\underline{q}_d^{(0)}, \overline{q}_o^{(1)}] = Q_o^{(1)} \cap Q_d^{(0)}$ (resp., $[\underline{q}_d^{(1)}, \overline{q}_o^{(0)}] = Q_o^{(0)} \cap Q_d^{(1)}$) according to distribution $\mathcal{D}$ if $|Q_o^{(1)} \cap Q_d^{(0)}| \leq \vartheta$ (resp., $|Q_o^{(0)} \cap Q_d^{(1)}| \leq \vartheta$). Otherwise, $q_{c_o^{(1)}}^{*(1,0)}$ (resp., $q_{c_o^{(0)}}^{*(0,1)}$) is selected from $[\lceil(\overline{q}_o^{(1)} - \underline{q}_d^{(0)})/2\rceil + \underline{q}_d^{(0)} - \vartheta/2, \lceil(\overline{q}_o^{(1)} - \underline{q}_d^{(0)})/2\rceil + \underline{q}_d^{(0)} + \vartheta/2]$ (resp., $[\lceil(\overline{q}_o^{(0)} - \underline{q}_d^{(1)})/2\rceil + \underline{q}_d^{(1)} - \vartheta/2, \lceil(\overline{q}_o^{(0)} - \underline{q}_d^{(1)})/2\rceil + \underline{q}_d^{(1)} + \vartheta/2]$). Note that ϑ does not necessarily have to be the same in the two consecutive executions of $\rho_{\text{IS-}\vartheta}$, but rather ϑ should depend on the commodity to be traded.

A further important application of gate $\rho_{\text{IS-}\vartheta}$ in the context of privacy-preserving unbiased negotiation is to determine the upper bound ω_Δ for $\rho_{\text{RIE-}\mathcal{D}}$ (as well as for $\rho_{\text{RSI-}\omega}$ and ρ_{RIE}) independently of all parties' quotes.

6.5 Summary and Future Work

In this chapter, we focused on privacy-preserving bartering between two-parties as a preliminary study for the multi-party case which is covered in the next chapter. We presented privacy-preserving two-party bartering protocols secure in the semi-honest model and in the malicious model, respectively. Subsequently, we introduced novel mechanisms that allow a privacy-preserving and unbiased negotiation of the quantities for the commodities to be traded.

An interesting research direction for future work is to formally analyze the implications of the chosen discrete distribution for determining the quantities for the commodities to be traded on the leakage of a party's private quantities specified in its quote.[2]

[1] However, it is an open question to what extent (depending on the selected distribution) partial information on a parties' quantity ranges are leaked by this approach.

[2] Note that this issue is not covered by the definition of security which underlies our protocols and gates. This is due to the fact that this definition does not consider what partial information a party's output reveals on the private inputs of other parties but instead ensured that a party learns no more information than its prescribed output and what can be deduced from it in combination with its own input.

CHAPTER 7

Privacy-Preserving Multi-Party Bartering

This chapter is dedicated to the problem of how to realize privacy-preserving bartering between multiple parties. While the privacy-preserving two-party bartering protocols presented in the previous chapter can generally be used to find pairwise trades for multiple parties, they are not capable of identifying trade cycles involving more than two parties and can not be generalized. More precisely, privacy-preserving protocols allowing multiple parties to barter their commodities require fundamentally new design approaches that (1) support the identification of trade cycles of arbitrary length, (2) select the "best" trade amongst all potential trades, and (3) restrict the view of a party to its local view of the selected actual trade.

Contributions: The following list provides an overview of the contributions of this chapter:

- We present a sophisticated and intuitive bartering terminology geared towards our privacy requirements that significantly simplifies the description of the complex privacy-preserving multi-party protocols. This terminology has been published in [121,122].[1]
- A privacy-preserving multi-party bartering protocol providing security in the semi-honest model. An initial variant of this protocol has been published in [120,123].
- A privacy-preserving multi-party bartering protocol providing security in the malicious model. This protocol has been published in [121,122].

[1] A previous (less intuitive) terminology has been published in [120,123].

- An optimized privacy-preserving subgraph check that significantly improves the efficiency of our privacy-preserving multi-party bartering protocols (independent of the adversary model). This contribution is joint work with Benjamin Assadsolimani (a graduate student at RWTH Aachen University and a student assistant at the Research Group IT-Security at RWTH Aachen University at that time).
- We elaborate two strategies for determining an actual trade constellation based on different criteria. These strategies have been sketched in [120,123].
- In order to assess the design approaches of our privacy-preserving multi-party bartering protocols, we analyzed an alternative design approach which is based on searching for a maximum weight matching in a bipartite graph. This contribution is joint work with Michael Vu (a graduate student at RWTH Aachen University at that time) and has been published in [117,125].

Parts of this Chapter are based on [120–123,125].

Outline: First, we formalize multi-party bartering (Section 7.1). In Section 7.2 and Section 7.3, we then present our novel privacy-preserving multi-party bartering protocols for the semi-honest and the malicious model, respectively. Subsequently, we present concrete strategies for the selection of an actual trade partner constellation (Section 7.4) and detail how to optimize the privacy-preserving subgraph check which is a main component of both protocols (Section 7.5). Finally, we analyze one further, completely different access to privacy-preserving multi-party bartering and provide a comparison to the underlying approach of our privacy-preserving multi-party bartering protocols (Section 7.6).

7.1 Notation and Terminology

The bartering related notation introduced for the two-party case in Chapter 6 (comprising the symbols for commodities, offers, demands, quotes, and quantity ranges) is reused for the multi-party case. However, we have to extend the bartering related terminology which is—in order to allow for an intuitive visualization—defined in terms of graph theory. In Example 7.1, we provide an overview of how these new terms build on each other.

A *trade partner constellation graph* is a generic directed graph $G^{\text{TPC}} = (V, E)$ whose nodes are identified with the party indices in $\mathscr{P}$ and for each node $\ell \in V := \mathscr{P}$ it holds that it either has exactly one incoming and one outgoing edge $(\ell', \ell), (\ell, \ell'') \in E$ with

G^{C}	Compatibility Graph	Def. 7.2
G^{TPC}	Trade Partner Constellation Graph	Def. 7.1
G^{PTPC}	Potential Trade Partner Constellation Graph	Def. 7.3
G^{ATPC}	Actual Trade Partner Constellation Graph	Def. 7.5
G^{PT}	Potential Trade Graph	Def. 7.6
G^{AT}	Actual Trade Graph	Def. 7.7

Table 7.1: Overview of the main bartering terms.

$\ell', \ell'' \in V$ and $\ell \neq \ell', \ell''$ (i.e, $deg^+(\ell) = deg^-(\ell) = 1$) or otherwise it is isolated (i.e., $deg(\ell) = 0$). Put differently, a trade partner constellation graph either indicates that a party P_ℓ does not actively participate in the encoded constellation or specifies P_ℓ's trade partners. In the latter case, G^{TPC} determines exactly one party $P_{\ell'}$ from which P_ℓ receives some quantity of a commodity (P_ℓ's offerer) as well as exactly on party $P_{\ell''}$ to which P_ℓ has to send some quantity of a commodity (P_ℓ's demander). Let $\mathbb{G}^{\mathrm{TPC}}$ denote a finite set of trade partner constellation graphs $G_1^{\mathrm{TPC}}, \ldots, G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}}$. The definition of a trade partner constellation graph ensures that each party which sends some quantity of a commodity in turn also receives some quantity of another commodity.[2]

Definition 7.1 (Trade Partner Constellation Graph)**.** *A* trade partner constellation graph G^{TPC} *is a directed graph* (V, E) *with* $V := \mathscr{P} = \{1, \ldots, \iota\}$ *and* $\forall \ell \in V : (deg^-(\ell) = 1 \wedge deg^+(\ell) = 1) \vee (deg(\ell) = 0)$.

The following definitions additionally consider (either explicitly or implicitly) the quotes of $P_1, \ldots, P_\iota$. We refer to the set of quotes of all parties as $\mathbf{Q} := \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$. Given $\mathbf{Q}$ we define a directed *compatibility graph* G^{C} whose nodes are (as for a G^{TPC}) associated with the party indices in $\mathscr{P}$, however, the edges are not generic. Instead, a directed edge (ℓ, ℓ') between two nodes ℓ and ℓ' indicates that condition $(c_o^{(\ell)} = c_d^{(\ell')}) \wedge (\overline{q}_o^{(\ell')} \geq \underline{q}_d^{(\ell')})$ is satisfied, i.e., that P_ℓ can satisfy the demand of $P_{\ell'}$.

Definition 7.2 (Compatibility Graph)**.** *Given* $\mathbf{Q} = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}\}$ *with* $\mathbf{q}^{(\ell)} = ((c_o^{(\ell)}, \overline{q}_o^{(\ell)}), (c_d^{(\ell)}, \underline{q}_d^{(\ell)}))$, *a* compatibility graph G^{C} *is a directed graph* (V, E) *with* $V := \{1, \ldots, \iota\}$ *and for any* $\ell, \ell' \in \mathscr{P}$, $\ell \neq \ell'$ *it holds that* $(\ell, \ell') \in E$ *iff* $[(c_o^{(\ell)} = c_d^{(\ell')}) \wedge (\overline{q}_o^{(\ell)} \geq \underline{q}_d^{(\ell')})] = 1$.

An m-cycle in a compatibility graph or in a trade partner constellation graph is referred to as an *m-trade cycle*. To ensure that commodities can be traded according to the direction of the edges in a trade partner constellation graph, it has to be assured

[2]Note that trade partner constellation graphs are generic in the sense that they can be generated without any information on the parties quotes.

that the trade cycles are disjoint (also referred to as *simultaneously executable*). While a compatibility graph may contain multiple trade cycles of which some may not be simultaneously executable (e.g., G^{C} in Figure 7.1 contains one 2-trade cycle and one 3-trade cycle but only one of them is executable at a time because P_1 is involved in both cycles), a potential trade constellation graph (e.g., G_1^{PTPC} and G_2^{PTPC} in Figure 7.1) corresponds to a subgraph of G^{C} which contains either one trade cycle or multiple simultaneously executable trade cycles.

Definition 7.3 (Potential Trade Partner Constellation Graph)**.** *Given G^{C}, a trade partner constellation graph G^{TPC} is referred to as* potential trade partner constellation graph *G^{PTPC} iff G^{TPC} is a subgraph of G^{C}, written as $G^{TPC} \sqsubseteq G^{C}$.*

For given $\mathbb{G}^{\mathrm{TPC}}$ and G^{C}, the set of potential trade partner constellation graphs is denoted as $\mathbb{G}^{\mathrm{PTPC}}$. According to Definition 7.3, it holds that $\mathbb{G}^{\mathrm{PTPC}} \subseteq \mathbb{G}^{\mathrm{TPC}}$ (cf. Figure 7.1).

Definition 7.4 (Welfare Function, Welfare)**.** *A* welfare function $\mathcal{W}(\cdot) : \mathbb{G}^{TPC} \to \mathbb{N}^0$ *maps a trade partner constellation graph in $\mathbb{G}^{TPC}$ to a* welfare $w \in \mathbb{N}^0$ *that measures the overall utility of a trade partner constellation graph.*

Next, we define an *actual trade partner constellation graph* which can be determined from a given set of trade partner constellation graphs and a welfare function.

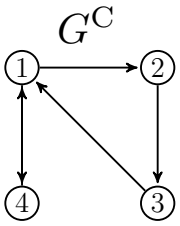
Definition 7.5 (Actual Trade Partner Constellation Graph)**.** *Given $\mathbb{G}^{PTPC}$ and $\mathcal{W}$, an* actual trade partner constellation graph *G^{ATPC} corresponds to a potential trade partner constellation graph G^{PTPC} with maximum welfare drawn uniformly at random from $\mathbb{G}^{PTPC}$, written as $G^{ATPC} \leftarrow_{\mathsf{max}_{\mathcal{W}}} \mathbb{G}^{PTPC}$.*

In Section 7.4, we discuss reasonable possibilities of how to choose $\mathcal{W}(\cdot)$ in order to influence the selection of the actual trade partner constellation graph.

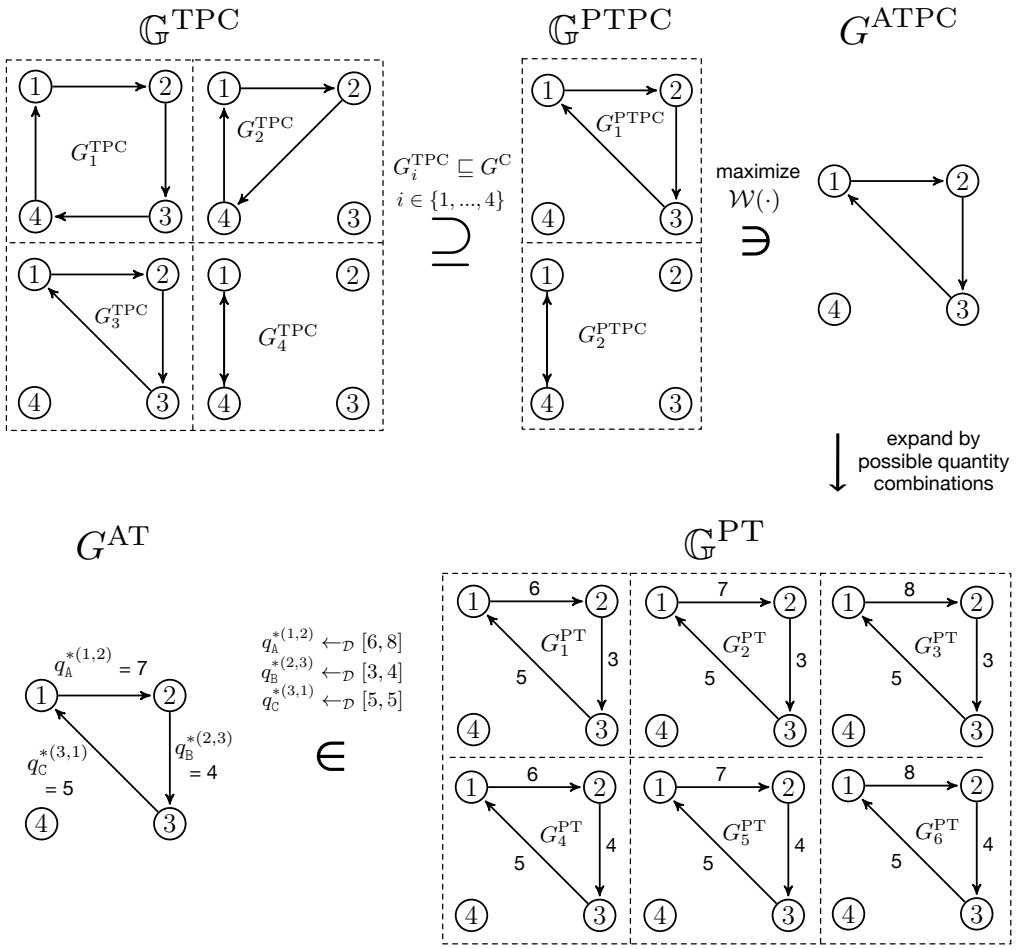
An actual trade partner constellation graph $G^{\mathrm{ATPC}} = (V, E)$ indicates which parties will send (resp., receive) some commodity to (resp., from) which other party in an actual trade. Since the set of quotes $\mathbf{Q}$ implicitly has been considered for determining G^{ATPC}, this implies that when augmenting the edges in G^{ATPC} with a weight $d((\ell, \ell')) \in Q_o^{(\ell)} \cap Q_d^{(\ell')}$ $(\forall (\ell, \ell') \in E)$ (which indicates the quantity of $c_o^{(\ell)}$ party P_ℓ has to send to $P_{\ell'}$) that all parties are satisfied with the corresponding trade. A *potential trade graph* constitutes one possibility for such an augmented actual trade partner constellation graph.

Definition 7.6 (Potential Trade Graph)**.** *Given $\mathbf{Q}$ and G^{ATPC}, a* potential trade graph *$G^{PT} = (V, E)$ corresponds to G^{ATPC} where additionally each edge $(\ell, \ell') \in E$ is associated with a specific weight $q_{c_o^{(\ell)}}^{(\ell,\ell')} \in Q_o^{(\ell)} \cap Q_d^{(\ell')}$ which indicates the quantity of commodity $c_o^{(\ell)}$ P_ℓ has to send to $P_{\ell'}$.*

$\mathscr{P}$	$\mathbf{Q}$	offer	demand
P_1	$\mathbf{q}^{(1)} = ($	$(\mathtt{A}, 8),$	$(\mathtt{C}, 5)$ $)$
P_2	$\mathbf{q}^{(2)} = ($	$(\mathtt{B}, 4),$	$(\mathtt{A}, 6)$ $)$
P_3	$\mathbf{q}^{(3)} = ($	$(\mathtt{C}, 5),$	$(\mathtt{B}, 3)$ $)$
P_4	$\mathbf{q}^{(4)} = ($	$(\mathtt{C}, 8),$	$(\mathtt{A}, 4)$ $)$



(a) Quotes of four parties and the resulting compatibility graph G^{C}.



(b) From $\mathbb{G}^{\mathrm{TPC}}$ to G^{AT}.

Figure 7.1: Bartering terms and their relations.

For a given G^{ATPC}, the set of all potential trade graphs is denoted by $\mathbb{G}^{\text{PT}}$.

Definition 7.7 (Actual Trade Graph)**.** *Given* $\mathbf{Q}$, $\mathbb{G}^{PT}$*, and an arbitrary discrete distribution* $\mathcal{D}$*, an* actual trade graph $G^{AT} = (V, E) \in \mathbb{G}^{PT}$ *is a specific potential trade graph where the edge weights are selected according to* $\mathcal{D}$*, written as* $d((\ell, \ell')) \leftarrow_{\mathcal{D}} Q_o^{(\ell)} \cap Q_d^{(\ell')}$ $(\forall (\ell, \ell') \in E)$*.*

Note that it is straight-forward to extend the discussion of how to choose $\mathcal{D}$ for two-party bartering (see Section 6.4) to multi-party bartering since in both cases each negotiation only affects the output of two parties.

An actual trade graph describes a trade which is designated to be executed. The overall process of determining G^{AT} from $\mathbf{Q}$ and $\mathbb{G}^{\text{TPC}}$ is denoted as $G^{\text{AT}} \xleftarrow{\mathbf{Q}}_{\mathsf{max}_{\mathcal{W},\mathcal{D}}} \mathbb{G}^{\text{TPC}}$.

For matters of convenience, we define

$$N_o^{(\ell)}(G) := \begin{cases} \ell' & \text{if } (\ell', \ell) \in E \\ 0 & \text{otherwise} \end{cases}$$

and

$$N_d^{(\ell)}(G) := \begin{cases} \ell'' & \text{if } (\ell, \ell'') \in E \\ 0 & \text{otherwise} \end{cases}$$

where $G = (V, E)$ is a trade partner constellation graph or a potential/actual trade graph. $N_o^{(\ell)}(G)$ (resp., $N_d^{(\ell)}(G)$) allows an easy access to the offerer (resp., demander) of P_ℓ. Furthermore, we define P_ℓ's *trade partner tuple* w.r.t. G as $N_{o,d}^{(\ell)}(G) := (N_o^{(\ell)}(G), N_d^{(\ell)}(G))$ and P_ℓ's set of all trade partner tuples w.r.t $\mathbb{G}^{\text{TPC}}$ as $\mathbb{N}_{o,d}^{(\ell)}(\mathbb{G}^{\text{TPC}}) := \{N_{o,d}^{(\ell)}(G_j^{\text{TPC}}) | G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}, j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}\}$. Note that $N_{o,d}^{(\ell)}(G) = (0, 0)$ indicates that P_ℓ is excluded from the underlying trade (or trade partner constellation). In Figure 7.1, for example, $N_{o,d}^{(2)}(G_1^{\text{PTPC}}) = (1, 3)$ while $N_{o,d}^{(4)}(G_1^{\text{PTPC}}) = (0, 0)$. In the following, we write $N_o^{(\ell)}$, $N_d^{(\ell)}$, and $N_{o,d}^{(\ell)}$ whenever the corresponding graph G is apparent from the context or the focus lies on concrete trade partners rather than on the graph they are resulting from.

Based on the novel bartering related terminology, now, we can formalize the functionality underlying our privacy-preserving multi-party bartering protocols:

Definition 7.8 ($\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$: Bartering, Multi-Party Case)**.** *Let party* P_ℓ *hold its private input* $\mathbf{q}^{(\ell)}$ $(\forall \ell \in \mathscr{P})$*. Furthermore, let* $\mathbb{G}^{TPC}$ *be a publicly known arbitrary non-empty set of trade partner constellations graphs for* ι *parties, let* $\mathcal{D}$ *be some publicly known discrete distribution, and let* $\mathcal{W}(\cdot) : \mathbb{G}^{TPC} \to \mathbb{N} \cup \{0\}$ *be some publicly known welfare function. Then, functionality* $\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ *is defined as*

$$\left.\begin{array}{ll} (o_1, \ldots, o_\iota) & \textit{if } \mathbb{G}^{PTPC} \neq \emptyset \\ (0) & \textit{otherwise} \end{array}\right\} \leftarrow \mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}(\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)}, \mathbb{G}^{TPC})$$

with $o_\ell := (N_{o,d}^{*(\ell)}(G^{ATPC}), q_{c_o^{(N_o^{*(\ell)})}}^{*(N_o^{*(\ell)},\ell)}, q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})})$, $G^{ATPC} \leftarrow_{\mathsf{max}_{\mathcal{W}}} \mathbb{G}^{PTPC} \subseteq \mathbb{G}^{TPC}$, $q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})} \leftarrow_{\mathcal{D}} Q_o^{(\ell)} \cap Q_d^{(N_d^{*(\ell)})}$. *The aggregated local views of all parties determine a graph* $G^{AT} \xleftarrow{\mathbf{Q}}_{\mathsf{max}_{\mathcal{W},\mathcal{D}}} \mathbb{G}^{TPC}$.

The definition of $\mathcal{F}^{\mathrm{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ assures that when the functionality is implemented as an SMPC protocol, each party learns its local view of the selected actual trade graph G^{AT}. More precisely, a party P_ℓ ($\ell \in \mathscr{P}$) which is excluded from the actual trade learns no further information while a party which can trade learns the party indices of its trade partners as well as the quantities of the commodities to be sent and received (and what can be deduced from this information in combination with $\mathbf{q}^{(\ell)}$). As for the two-party case, a party's quote remains private throughout the protocol execution.

Example 7.1. In the following, we provide an example where four parties $P_1, \ldots, P_4$ strive to barter their offered commodities for their desired commodities. The parties' private quotes $\mathbf{Q} = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(4)}\}$ as well as the corresponding compatibility graph G^{C} are given in Figure 7.1a. To keep the example simple, we consider a trade partner constellation graph set $\mathbb{G}^{\mathrm{TPC}} = \{G_1^{\mathrm{TPC}}, G_2^{\mathrm{TPC}}, G_3^{\mathrm{TPC}}, G_4^{\mathrm{TPC}}\}$ of only four trade partner constellation graphs.[3] From G^{C} and $\mathbb{G}^{\mathrm{TPC}}$, the set of potential trade partner constellation graphs can be determined by checking whether or not $G_i^{\mathrm{TPC}} \sqsubseteq G^{\mathrm{C}}$ ($\forall i \in \mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|}$). In our example, we have $\mathbb{G}^{\mathrm{PTPC}} = \{G_1^{\mathrm{PTPC}}, G_2^{\mathrm{PTPC}}\} = \{G_3^{\mathrm{TPC}}, G_4^{\mathrm{TPC}}\}$. Next, the actual trade partner constellation graph is selected from $\mathbb{G}^{\mathrm{PTPC}}$ such that G^{ATPC} corresponds to a potential trade partner constellation graph chosen uniformly at random from those with maximum welfare. In the example, we assume that $\mathcal{W}(G = (V, E)) = |E|$ and thus $G^{\mathrm{ATPC}} := G_1^{\mathrm{PTPC}}$. From G^{ATPC} and $\mathbf{Q}$ (more precisely, the quantities specified by the parties quotes), we can deduce six different potential trade graphs $\mathbb{G}^{\mathrm{PT}} = \{G_1^{\mathrm{PT}}, \ldots, G_6^{\mathrm{PT}}\}$ (cf. Figure 7.1b). By selecting $q_{c_o^{(1)}}^{*(1,2)} \leftarrow_{\mathcal{D}} [6, 8] = Q_o^{(1)} \cap Q_d^{(2)}$, $q_{c_o^{(2)}}^{*(2,3)} \leftarrow_{\mathcal{D}} [3, 4] = Q_o^{(2)} \cap Q_d^{(3)}$, and $q_{c_o^{(3)}}^{*(3,1)} \leftarrow_{\mathcal{D}} [5, 5] = Q_o^{(3)} \cap Q_d^{(1)}$ (assuming that $\mathcal{D}$ corresponds to the binomial distribution), G^{AT} is determined which corresponds to $G_5^{\mathrm{PT}} \in \mathbb{G}^{\mathrm{PT}}$ in the example (see Figure 7.1b). For instance, graph G^{AT} indicates that P_1 has to send 7 units of commodity $\mathtt{A}$ to P_2 and receives 5 units of commodity $\mathtt{C}$ from P_3.

[3] Actually, $\mathbb{G}^{\mathrm{TPC}}$ is meant to include all possible trade partner constellations graphs for a fixed size of parties—or for matters of efficiency may be restircted to a subset thereof, e.g., to only include trade cycles of a specific length or to only include constellations in which specific parties get to trade as it is the case in the example.

7.2 Multi-Party Bartering Secure in the Semi-Honest Model

In this section, we introduce protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ which securely implements $\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (see Definition 7.8) in the semi-honest model.

7.2.1 Intuition

In the following, we provide an intuition for protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ which allows the determining of an actual trade for a publicly-known set of trade partner constellations in combination with a given set of parties and their private quotes while maximizing a welfare function that the parties agreed upon prior to executing the protocol. Note that the trade partner constellation set may include all possible constellations for a fixed set of parties. However, for matters of efficiency or due to the specific demands of the commodities traded it may be restricted to a subset thereof, e.g., to only include trade cycles of a specific length or to only include constellations in which specific parties get to trade (e.g., see the example in Figure 7.1). Upon termination of the protocol, a party only learns its local view of the determined actual trade (and what can be derived from it in combination with its private input), i.e., its own trade partners and the actual quantities of the commodities to be traded with them.

Figure 7.2 illustrates the sequence of the main steps of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$.[4] In the first step (Transition 1., Figure 7.2) an encrypted adjacency matrix representing the compatibility graph is obliviously determined and then for each publicly known trade partner constellation graph it is obliviously checked whether it is a subgraph of the compatibility graph by utilizing the encrypted adjacency matrix. This step implicitly determines the set of potential trade partner constellations.

In the second step (Transition 2., Figure 7.2), the protocol first obliviously maps each potential trade partner constellation to its welfare, obliviously determines the actual trade partner constellation uniformly at random amongst those potential trade partner constellations with maximum welfare, and then provides each party P_ℓ with (nothing but) its trade partners of the actual trade partner constellation. The determining of the actual trade uses gate $\rho_{\text{CRS-C}}$ (see Gate Specification 12) implementing the primitive of conditional random selection in a privacy-preserving fashion (see Section 5.3.2).

Lastly, in the third step (Transition 3., Figure 7.2), each party that can barter is involved twice (with each of its trade partners) in executing the two-party gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) for sampling out of a private interval according to the discrete distribution $\mathcal{D}$ to determine the actual quantities of the commodities to be traded. It is

[4] Note that $\mathbb{G}^{\text{PT}}$ is not explicitly determined in our protocol and thus not depicted in Figure 7.2.

$$\mathbb{G}^{\text{TPC}} \xrightarrow{1.} \mathbb{G}^{\text{PTPC}} \xrightarrow[\rho_{\text{CRS-C-1}}]{2.} G^{\text{ATPC}} \xrightarrow[\rho_{\text{RIE-}\mathcal{D}}]{3.} G^{\text{AT}}$$

Figure 7.2: Illustration of substantial steps of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$.

assumed that the parties agreed upon $\mathcal{D}$ prior to the execution of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$. The bartering protocol provides each party with its local view consisting of its trade partners and the quantities of the commodities to be exchanged with both of them (provided that a party can barter), where the cumulated local views induce the actual trade.

In order to assure that each party only learns its own trade partners (i.e., its local view of the actual trade partner constellation), we make use of a novel encoding and decoding operation that is based on the uniqueness of prime factorization. Specifically, each party is assigned a unique set of prime numbers. Each party then selects a mapping between its prime numbers and all of its trade partner tuples in any of the trade partner constellations (including those where the respective trade partner tuple is equal to $(0, 0)$) in the trade partner constellation set. Each party keeps its mapping secret. Subsequently, for all trade partner constellations, all parties jointly compute the product of the corresponding prime numbers (locally mapped to the respective trade partner tuples by the parties) in an oblivious fashion, thus, enabling each party to uniquely encode its participation in the trade partner constellation by contributing a unique prime number in each case. In turn, given the product of prime numbers for the actual trade partner constellation, through trial division each party can locally determine which one of its prime numbers divides the product and then identify the respective trade partner tuple based on its own secret mapping. It is important to note that knowing which set of prime numbers was assigned to which party, anyone can perform the trial division step. However, given the fact that each party keeps its mapping between its prime numbers and its trade partner tuples secret, this does not leak any information. Since also trade partner tuple $(0, 0)$ is mapped to a prime number (individually by each party) and the product of prime numbers includes those prime numbers for parties which neither send nor receive a commodity, the prime number product does not leak which parties participate in the selected actual trade partner constellation. It is also important to note that the size of the prime numbers does not have any influence on the security of our bartering protocol.

7.2.2 Specification of Protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$

In the following, we present the details of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ which is split into the following phases:

1. *Construction Phase:* Based on their quotes, all parties jointly construct the encrypted adjacency matrix $[\![A]\!] := ([\![a_{\ell,\ell'}]\!])_{\iota\times\iota}$ of the compatibility graph G^{C} with $a_{\ell,\ell'} := [(c_o^{(\ell)} = c_d^{(\ell')}) \wedge (\overline{q}_o^{(\ell)} \geq \underline{q}_d^{(\ell')})]$ for $\ell, \ell' \in \mathscr{P}$ and $\ell \neq \ell'$. A ciphertext $[\![[(c_o^{(\ell)} = c_d^{(\ell')}) \wedge (\overline{q}_o^{(\ell)} \geq \underline{q}_d^{(\ell')})]]\!]$ is computed in three steps (see Steps 1.1.1.1-1.1.1.3, Protocol 7.1).

2. *Evaluation Phase:* For each $G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ ($j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$), it is obliviously checked whether or not $G_j^{\text{TPC}} \sqsubseteq G^{\mathsf{C}}$. To this end, for each $G_j^{\text{TPC}} = (V, E)$ all parties jointly execute $\rho_{\text{UFI-Mult}}$ (see Section 5.1.1) on input $([\![a_{\ell_1,\ell_1'}]\!], \ldots, [\![a_{\ell_{|E|},\ell_{|E|}'}]\!])$ where $(\ell_i, \ell_i') \in E$ with $i \in \mathbb{N}_{|E|}$ such that each party learns the computation result $[\![e_j]\!]$. At the end of this phase, each party holds an encrypted vector $[\![L]\!] := ([\![e_1]\!], \ldots, [\![e_{|\mathbb{G}^{\text{TPC}}|}]\!])$ where $e_j = [G_j^{\text{TPC}} \sqsubseteq G^{\mathsf{C}}]$.

3. *Prioritization Phase:* Function $\mathcal{W}(\cdot) : \mathbb{G}^{\text{TPC}} \to \mathbb{N}^0$ maps each trade partner constellation graph to its welfare and can be evaluated by each party independently. Each party locally computes $[\![e_j]\!] \times_h \mathcal{W}(G_j^{\text{TPC}})$ ($\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$) resulting in an encrypted vector $[\![L_1]\!] := ([\![e_1 \cdot \mathcal{W}(G_1^{\text{TPC}})]\!], \ldots, [\![e_{|\mathbb{G}^{\text{TPC}}|} \cdot \mathcal{W}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}})]\!])$. The prioritization phase ensures that in the selection phase (see below), a potential trade partner constellation graph with maximum welfare is determined as actual trade partner constellation graph.

4. *Mapping Phase:* At the beginning of the protocol, each party P_ℓ ($\ell \in \mathscr{P}$) is given an interval $I^{(\ell)}$ of positive integers with at least $|\mathbb{N}_{o,d}^{(\ell)}(\mathbb{G}^{\text{TPC}})|$ prime numbers such that for all $\ell, \ell' \in \mathscr{P}$ ($\ell \neq \ell'$), it holds that $I^{(\ell)} \cap I^{(\ell')} = \emptyset$. Each party P_ℓ constructs a private table mapping each element in $\mathbb{N}_{o,d}^{(\ell)}(\mathbb{G}^{\text{TPC}})$ to a unique prime number randomly chosen from $I^{(\ell)}$. More precisely, each party P_ℓ keeps a set $S^{(\ell)}$ of already assigned prime numbers from $I^{(\ell)}$ which is initialized with $S = \emptyset$. P_ℓ then maps each tuple $N_{o,d}^{(\ell)} \in \mathbb{N}_{o,d}^{(\ell)}(\mathbb{G}^{\text{TPC}})$ to a prime number $p_{N_{o,d}^{(\ell)}}^{(\ell)} \leftarrow_{\$} \mathbf{P}_{I^{(\ell)}} \backslash S^{(\ell)}$. Subsequently, $p_{N_{o,d}^{(\ell)}}^{(\ell)}$ is added to $S^{(\ell)}$ (cf. Step 4.1, Protocol 7.1). Once all parties have established their mapping tables, they engage in the consecutive computation of an encrypted prime number product for each $G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$. Each party P_ℓ contributes a single prime number $p^{(\ell)}$ to the encrypted prime number product associated with $G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$: First, P_ι computes $[\![u_j^{(\iota)}]\!] := [\![p_{N_{o,d}^{(\iota)}}^{(\iota)}]\!]$ for all $j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$ and sends the result to $P_{\iota-1}$. Each party $P_{\ell'}$ ($\ell' \in \mathscr{P}\backslash\{\iota\}$) then computes $[\![u_j^{(\ell')}]\!] := \mathsf{Blind}([\![u_j^{(\ell'+1)}]\!] \times_h p_{N_{o,d}^{(\ell')}(G_j^{\text{TPC}})}^{(\ell')})$ ($\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$) and sends the result to $P_{\ell'-1}$, except P_1 which sets $[\![L_2]\!] := ([\![u_1]\!], \ldots, [\![u_{|\mathbb{G}^{\text{TPC}}|}]\!]) := ([\![u_1^{(1)}]\!], \ldots, [\![u_{|\mathbb{G}^{\text{TPC}}|}^{(1)}]\!])$ and broadcasts $[\![L_2]\!]$ (cf. Steps 4.2- 4.4, Protocol 7.1).

5. *Selection Phase:* From the previous phases, each G_j^{TPC} is associated with two values $[\![e_j]\!]$ and $[\![u_j]\!]$ where $e_j \in \mathbb{N}^0$ indicates the priority of G_j^{TPC} while $u_j \in \mathbb{N}$ is a product of individual prime numbers encoding the trade partners of each party w.r.t. G_j^{TPC}. In this phase, the parties now jointly compute $\rho_{\text{CRS-C-1}}$ (see Gate Specification 12) on the common input $([\![L_1]\!], [\![L_2]\!])$ (see Step 5.1, Protocol 7.1) in order to select an entry of $[\![L_2]\!]$ associated with a potential trade constellation graph which maximizes the welfare function in case that $\mathbb{G}^{\text{PTPC}} \neq \emptyset$. Otherwise, in the case that $\mathbb{G}^{\text{PTPC}} = \emptyset$, the parties learn of this fact. In the former case, $\rho_{\text{CRS-C-1}}$ returns $(c_1^*, c_2^*) \in ([\![L_1]\!], [\![L_2]\!])$ with $c_1^* := [\![l_1^*]\!]$ and $c_2^* := [\![l_2^*]\!]$. In Step 5.3.1 (Protocol 7.1), all parties jointly decrypt $[\![l_2^*]\!]$ such that each of them learns the result. In the latter case where $L_1 = (0, \overset{|\mathbb{G}^{\text{TPC}}|}{\ldots}, 0)$, $\rho_{\text{CRS-C-1}}$ returns $c_1^* = c_2^* = \perp$ (where $\perp$ refers to the empty string) which prompts each party to skip all remaining phases and to output 0.[5]

6. *Reverse Mapping Phase:* Each party P_ℓ checks which prime in $S^{(\ell)}$ divides l_2^*. By reversely applying the mapping table, the unique result determines P_ℓ's trade partners $N_{o,d}^{*(\ell)}$ w.r.t. G^{ATPC}.

7. *Negotiation Phase:* So far, G^{ATPC} has already been determined which implicitly provides $\mathbb{G}^{\text{PT}}$. In order to select G^{AT} in such a way that a party only learns its local view, it engages in the two-party version of gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) with each of its trade partners that were determined in the previous phase. More precisely, for $N_{o,d}^{*(\ell)}(G_j^{\text{TPC}}) = (N_o^{*(\ell)}, N_d^{*(\ell)}) \neq (0,0)$ $(\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|})$, P_ℓ and $P_{N_d^{*(\ell)}}$ execute $([\![q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})}]\!]) \leftarrow \rho_{\text{RIE-}\mathcal{D}}((\underline{[\![q}_d^{(N_d^{*(\ell)})}]\!], [\![\overline{q}_o^{(\ell)}]\!]))$ and $q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})} = \mathsf{Dec}([\![q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})}]\!])$ (s.t. both parties learn the plaintext result) where $q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})}$ indicates the quantity of $c_o^{(\ell)}$ that P_ℓ has to send to $P_{N_d^{*(\ell)}}$. Note that the execution of $\rho_{\text{RIE-}\mathcal{D}}$ additionally requires the two involved parties to share a $(2,2)$-threshold Paillier key which is assumed to be pre-distributed (see the set-up assumptions detailed in Section 4.5.1).

8. *Output Phase:* Each party P_ℓ outputs $o_\ell = (N_{o,d}^{*(\ell)}, q_{c_o^{(N_o^{*(\ell)})}}^{*(N_o^{*(\ell)},\ell)}, q_{c_o^{(\ell)}}^{*(\ell,N_d^{*(\ell)})})$.

In order to show the correctness of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, we distinguish two cases. In the first case where $\mathbb{G}^{\text{PTPC}} = \emptyset$, the output of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ is fixed—each party outputs 0. In the second case where $\mathbb{G}^{\text{PTPC}} \neq \emptyset$, we have to show that $G^{\text{AT}} \overset{\mathbf{Q}}{\leftarrow}_{\max_{\mathcal{W},\mathcal{D}}} \mathbb{G}^{\text{TPC}}$ (see Definition 7.8).

[5] The purpose of this approach is to hide the number of subgraphs of G^{C} (in case that $\mathbb{G}^{\text{PTPC}} \neq \emptyset$) as this could otherwise not be simulated given the inputs and outputs of the corrupted parties only.

Protocol 7.1. Specification of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$.

1 Construction Phase
- 1.1 For each P_ℓ ($\ell \in \mathscr{P}$):
  - 1.1.1 For each $P_{\ell'}$ ($\ell' \in \mathscr{P}\backslash\{\ell\}$):
    - 1.1.1.1 All parties jointly compute $([\![b_1]\!]) \leftarrow \rho_{\text{ET}}(([\![c_o^{(\ell)}]\!], [\![c_d^{(\ell')}]\!]))$
    - 1.1.1.2 All parties jointly compute $([\![b_2]\!]) \leftarrow \rho_{\text{GTE}}(([\![\overline{q}_o^{(\ell)}]\!], [\![\underline{q}_d^{(\ell')}]\!]))$
    - 1.1.1.3 All parties jointly compute $([\![a_{\ell,\ell'}]\!]) \leftarrow \rho_{\text{Mult}}(([\![b_1]\!], [\![b_2]\!]))$

2 Evaluation Phase
- 2.1 For each $G_j^{\text{TPC}} = (V, E) \in \mathbb{G}^{\text{TPC}}$ with $(\ell_i, \ell_i') \in E$ and $j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$, $i \in \mathbb{N}_{|E|}$:
  - 2.1.1 All parties jointly compute $[\![e_j]\!] \leftarrow \rho_{\text{UFI-Mult}}(([\![a_{\ell_1,\ell_1'}]\!], \ldots, [\![a_{\ell_{|E|},\ell_{|E|}'}]\!]))$
- 2.2 All parties set $[\![L]\!] := ([\![e_1]\!], \ldots, [\![e_{|\mathbb{G}^{\text{TPC}}|}]\!])$

3 Prioritization Phase
- 3.1 Each party locally computes $[\![L_1]\!] := ([\![e_1]\!] \times_h \mathcal{W}(G_1^{\text{TPC}}), \ldots, [\![e_{|\mathbb{G}^{\text{TPC}}|}]\!] \times_h \mathcal{W}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}}))$

4 Mapping Phase
- 4.1 Each party P_ℓ ($\ell \in \mathscr{P}$):
  - 4.1.1 Set $S^{(\ell)} := \emptyset$
  - 4.1.2 For each $N_{o,d}^{(\ell)} \in \mathbb{N}_{o,d}^{(\ell)}(\mathbb{G}^{\text{TPC}})$:
    - 4.1.2.1 Select $p^{(\ell)}_{N_{o,d}^{(\ell)}} \leftarrow_{\$} \mathbf{P}_{I^{(\ell)}} \backslash S^{(\ell)}$
    - 4.1.2.2 Update $S^{(\ell)} = S^{(\ell)} \cup \{p^{(\ell)}_{N_{o,d}^{(\ell)}}\}$
- 4.2 Party P_ι:
  - 4.2.1 Set $[\![u_j^{(\iota)}]\!] := [\![p^{(\iota)}_{N_{o,d}^{(\iota)}(G_j^{\text{TPC}})}]\!]$ with $G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ ($\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$)
  - 4.2.2 Send $([\![u_1^{(\iota)}]\!], \ldots, [\![u_{|\mathbb{G}^{\text{TPC}}|}^{(\iota)}]\!])$ to $P_{\iota-1}$
- 4.3 Party $P_{\ell'}$ for ℓ' from $\iota - 1$ to 1:
  - 4.3.1 Compute $[\![u_j^{(\ell')}]\!] := \mathsf{Blind}([\![u_j^{(\ell'+1)}]\!] \times_h p^{(\ell')}_{N_{o,d}^{(\ell')}(G_j^{\text{TPC}})})$ with $G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ ($\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$)
  - 4.3.2 If $\ell' \neq 1$: Send $([\![u_1^{(\ell')}]\!], \ldots, [\![u_{|\mathbb{G}^{\text{TPC}}|}^{(\ell')}]\!])$ to party $P_{\ell'-1}$
- 4.4 Party P_1:
  - 4.4.1 Set $[\![L_2]\!] := ([\![u_1]\!], \ldots, [\![u_{|\mathbb{G}^{\text{TPC}}|}]\!]) := ([\![u_1^{(1)}]\!], \ldots, [\![u_{|\mathbb{G}^{\text{TPC}}|}^{(1)}]\!])$
  - 4.4.2 Broadcast $[\![L_2]\!]$

5 Selection Phase
- 5.1 All parties jointly compute $((c_1^*, c_2^*)) \leftarrow \rho_{\text{CRS-C-1}}(([\![L_1]\!], [\![L_2]\!]))$
- 5.2 If $c_1^* = c_2^* = \perp$:
  - 5.2.1 Skip Steps 6-8
  - 5.2.2 Party P_ℓ outputs 0 ($\forall \ell \in \mathscr{P}$)
- 5.3 Else (i.e., $(c_1^*, c_2^*) = ([\![l_1^*]\!], [\![l_2^*]\!])$):
  - 5.3.1 All parties jointly compute $l_2^* = \mathsf{Dec}([\![l_2^*]\!])$

6 Reverse Mapping Phase
- 6.1 Each party P_ℓ ($\ell \in \mathscr{P}$):
  - 6.1.1 For each $p^{(\ell)}_{N_{o,d}^{(\ell)}} \in S^{(\ell)}$ with $N_{o,d}^{(\ell)} \in \mathbb{N}_{o,d}^{(\ell)}(\mathbb{G}^{\text{TPC}})$:
    - 6.1.1.1 If $p^{(\ell)}_{N_{o,d}^{(\ell)}}$ divides l_2^* then set $N_{o,d}^{*(\ell)} := N_{o,d}^{(\ell)}$ and go to Step 7

7 Negotiation Phase
- 7.1 For each party P_ℓ with $N_{o,d}^{*(\ell)} := (N_o^{*(\ell)}, N_d^{*(\ell)}) \neq (0,0)$ ($\forall \ell \in \mathscr{P}$):
  - 7.1.1 Parties P_ℓ, $P_{N_d^{*(\ell)}}$ jointly compute $([\![q^{*(\ell,N_d^{*(\ell)})}_{c_o^{(\ell)}}]\!]) \leftarrow \rho_{\text{RIE-}\mathcal{D}}(([\![\underline{q}_d^{(N_d^{*(\ell)})}]\!], [\![\overline{q}_o^{(\ell)}]\!]))$
  - 7.1.2 Parties P_ℓ, $P_{N_d^{*(\ell)}}$ jointly compute $q^{*(\ell,N_d^{*(\ell)})}_{c_o^{(\ell)}} = \mathsf{Dec}([\![q^{*(\ell,N_d^{*(\ell)})}_{c_o^{(\ell)}}]\!])$

8 Output Phase
- 8.1 Party P_ℓ outputs $(N_{o,d}^{*(\ell)}, q^{*(N_o^{*(\ell)},\ell)}_{c_o^{(N_o^{*(\ell)})}}, q^{*(\ell,N_d^{*(\ell)})}_{c_o^{(\ell)}})$ ($\forall \ell \in \mathscr{P}$)

Case 1.) In case that $\mathbb{G}^{\text{PTPC}} = \emptyset$, the evaluation phase of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ returns a vector $[\![L]\!] := ([\![e_1]\!], \ldots, [\![e_{|\mathbb{G}^{\text{TPC}}|}]\!])$ with $e_1 = \ldots = e_{|\mathbb{G}^{\text{TPC}}|} = 0$ since there exists no $G^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ such that $G^{\text{TPC}} \sqsubseteq G^{\text{C}}$. Consequently, the prioritization phase does not influence the plaintext values of $[\![L]\!]$ and in the selection phase, gate $\rho_{\text{CRS-C-1}}(([\![L_1]\!], [\![L_2]\!]))$ returns $(\perp, \perp)$ prompting each party to output 0 (cf. Step 5.2.2, Protocol 7.1).

Case 2.) In case that $\mathbb{G}^{\text{PTPC}} \neq \emptyset$, the evaluation phase computes a vector $[\![L]\!] := ([\![e_1]\!], \ldots, [\![e_{\mathbb{G}^{\text{TPC}}}]\!])$ with $e_j = [G^{\text{TPC}}_j \sqsubseteq G^{\text{C}}]$ and Hamming weight $|\mathbb{G}^{\text{PTPC}}|$ for $G^{\text{TPC}}_j \in \mathbb{G}^{\text{TPC}}$. The prioritization phase only modifies those entries of $[\![L]\!]$ where $e_j \neq 0$. Gate $\rho_{\text{CRS-C-1}}(([\![L_1]\!], [\![L_2]\!]))$, called in the selection phase, returns $([\![l^*_1]\!], [\![l^*_2]\!])$ such that $l^*_1 = \mathcal{W}(G^{\text{ATPC}})$ with $G^{\text{ATPC}} \leftarrow_{\mathsf{max}_{\mathcal{W}}} \mathbb{G}^{\text{PTPC}}$ and $l^*_2 = p^{(1)}_{N^{(1)}_{o,d}(G^{\text{ATPC}})} \cdot \ldots \cdot p^{(\iota)}_{N^{(\iota)}_{o,d}(G^{\text{ATPC}})}$. After jointly decrypting $[\![l^*_2]\!]$ in Step 5.3.1 (Protocol 7.1), each party P_ℓ learns l^*_2 and sets $N^{*(\ell)}_{o,d} = N^{(\ell)}_{o,d}(G^{\text{TPC}}_j)$ according to the unequivocally determined prime $p^{(\ell)}_{N^{(\ell)}_{o,d}(G^{\text{TPC}}_j)}$ which divides l^*_2. After each party P_ℓ has determined its trade partners $N^{*(\ell)}_o(G^{\text{ATPC}})$ and $N^{*(\ell)}_d(G^{\text{ATPC}})$, it learns the quantity of the commodity it is going to send (resp., receive) in the negotiation phase. As prescribed by Definition 7.8, these quantities are sampled from $Q^{(\ell)}_o \cap Q^{(N^{*(\ell)}_d)}_d$ (resp., $Q^{(N^{*(\ell)}_o)}_o \cap Q^{(\ell)}_d$) according to distribution $\mathcal{D}$ by utilizing gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16). Overall, it follows that $G^{\text{AT}} \overset{\mathbf{Q}}{\leftarrow}_{\mathsf{max}_{\mathcal{W},\mathcal{D}}} \mathbb{G}^{\text{TPC}}$.

Theorem 7.1. *Let P_ℓ hold $\mathbf{q}^{(\ell)}$ ($\forall \ell \in \mathscr{P}$) and let $\mathbb{G}^{TPC}$, $\mathcal{W}$, and $\mathcal{D}$ be public. Then, protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ securely computes functionality $\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ in the semi-honest model.*

Proof. By separating the different phases of Protocol 7.1, we sketch a simulator S which generates a transcript that is computationally indistinguishable from $\mathsf{VIEW}^{\pi}_{C}(\overline{x})$. W.l.o.g., assume that parties $P_1, \ldots, P_t$ are corrupted, i.e., $C = \{1, \ldots, t\}$. The executions of gates ρ_{ET}, ρ_{GTE}, and ρ_{Mult} (cf. Gate Specification 8, Gate Specification 5, Gate Specification 1) in the construction phase can be simulated by calling the corresponding simulators: The simulator of ρ_{ET} is called on input $[\![c^{(\ell)}_o]\!]$, $[\![c^{(\ell')}_d]\!]$ and output $\langle [\![b_1]\!] \rangle \leftarrow_{\$} \mathbb{C}$ if $\ell, \ell' \in C$. Analogously, the simulator of ρ_{GTE} is called on input $[\![\overline{q}^{(\ell)}_o]\!]$, $[\![\underline{q}^{(\ell')}_d]\!]$ and output $\langle [\![b_2]\!] \rangle \leftarrow_{\$} \mathbb{C}$ if $\ell, \ell' \in C$. In case that $\ell \notin C$ or $\ell' \notin C$, the corresponding input is replaced by a random ciphertext for the simulation of ρ_{ET}, respectively, ρ_{GTE}. The input of the simulator of ρ_{Mult} depends on the simulated results of the two previous gates and is set to $(\langle [\![b_1]\!] \rangle, \langle [\![b_2]\!] \rangle)$ while the output is simulated as $\langle [\![a_{\ell,\ell'}]\!] \rangle \leftarrow_{\$} \mathbb{C}$.

The individual calls of $\rho_{\text{UFI-Mult}}$ in the evaluation phase are simulated by calling the simulator of $\rho_{\text{UFI-Mult}}$ (i.e., multiple calls of the simulator of ρ_{Mult}) on input $\langle [\![a_{\ell_1,\ell'_1}]\!] \rangle, \ldots, \langle [\![a_{\ell_{|E|},\ell'_{|E|}}]\!] \rangle$ and output $\langle [\![e_j]\!] \rangle \leftarrow_{\$} \mathbb{C}$.

For the prioritization phase, S sets $\langle[\![L_1]\!]\rangle := (\langle[\![e_1]\!]\rangle \times_h \mathcal{W}(G_1^{\text{TPC}}), \ldots, \langle[\![e_{|\mathbb{G}^{\text{TPC}}|}]\!]\rangle \times_h \mathcal{W}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}}))$.

The mapping phase can be simulated by letting S perform Steps 4.1- 4.4 of Protocol 7.1 for each party P_ℓ with $\ell \in \mathscr{P}$ (independent of whether or not $\ell \in C$). Utilizing the simulated mapping tables, S simulates $(u_1^{(\ell_C+1)}, \ldots, u_{|\mathbb{G}^{\text{TPC}}|}^{(\ell_C+1)})$ for P_{ℓ_C} as well as $[\![L_2]\!]$ ($\forall \ell_C \in C$). Furthermore, S computes $\langle u_j \rangle$ ($\forall j \in \mathbb{N}_{\mathbb{G}^{\text{TPC}}}$) from the simulated mapping tables (where one of these values is used later to simulate the decryption operation in Step 5.3.1).

The simulation of the selection phase depends on whether or not $(0) \leftarrow \pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}(\overline{x}, \mathbb{G}^{\text{TPC}})$. In case that $(0) \leftarrow \pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}(\overline{x}, \mathbb{G}^{\text{TPC}})$, the simulator of $\rho_{\text{CRS-C-1}}$ (see Gate Specification 12) is called on input $(\langle[\![L_1]\!]\rangle, \langle[\![L_2]\!]\rangle)$ and output $\langle c_1^*\rangle = \langle c_2^*\rangle := \perp$. Otherwise, on input $(\langle[\![L_1]\!]\rangle, \langle[\![L_2]\!]\rangle)$ and output $(c_1^*, c_2^*) \leftarrow_{\$} \mathbb{C} \times \mathbb{C}$. In the latter case, $\mathsf{Dec}(\langle[\![l_2^*]\!]\rangle)$ is called in the hybrid model where $\langle[\![l_2^*]\!]\rangle := c_2^*$. The corresponding output is simulated by setting $\langle l_2^*\rangle := \langle u_j \rangle$ where $j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$ is chosen such that $N_{o,d}^{*(1)} = N_{o,d}^{(1)}(G_j^{\text{TPC}}), \ldots, N_{o,d}^{*(t)} = N_{o,d}^{(t)}(G_j^{\text{TPC}})$. Note that S obtains $N_{o,d}^{*(1)}, \ldots, N_{o,d}^{*(t)}$ as input. This approach is necessary since, otherwise, $\langle l_2^*\rangle$ is not consistent with $\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}(\overline{x}, \mathbb{G}^{\text{TPC}})$.

In order to simulate the negotiation phase for P_{ℓ_C} with $N_{o,d}^{*(\ell_C)} \neq (0,0)$ ($\forall \ell_C \in C$), S has to simulate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) and the decryption operation twice: First, the simulator is called on

- $[\![\underline{q}_d^{(N_d^{*(\ell_C)})}]\!]$ and $[\![\overline{q}_o^{(\ell_C)}]\!]$ if $N_d^{*(\ell_C)} \in C$
- $\langle[\![\underline{q}_d^{(N_d^{*(\ell_C)})}]\!]\rangle \leftarrow_{\$} \mathbb{C}$ and $[\![\overline{q}_o^{(\ell_C)}]\!]$ otherwise

as well as on output $\langle[\![q_{c_o^{(\ell_C)}}^{*(\ell_C, N_d^{(\ell_C)})}]\!]\rangle \leftarrow_{\$} \mathbb{C}$. The second call of the subsimulator can be conducted analogously. The two decryption operations are called in the hybrid model on $\langle[\![q_{c_o^{(\ell_C)}}^{*(\ell_C, N_d^{*(\ell_C)})}]\!]\rangle$ and $q_{c_o^{(\ell_C)}}^{*(\ell_C, N_d^{*(\ell_C)})}$, respectively, on $\langle[\![q_{c_o^{(N_o^{*(\ell_C)})}}^{*(N_o^{*(\ell_C)}, \ell_C)}]\!]\rangle$ and $q_{c_o^{(N_o^{*(\ell_C)})}}^{*(N_o^{*(\ell_C)}, \ell_C)}$. Note that this simulation also captures the corresponding part of $P_{N_d^{*(\ell_C)}}$'s view (resp., $P_{N_o^{*(\ell_C)}}$'s view) in case that $N_d^{*(\ell_C)} \in C$ (resp., $N_o^{*(\ell_C)} \in C$).

Due to the fact that the underlying cryptosystem provides threshold IND-CPA security, we can conclude that the simulated view is computationally indistinguishable from $\mathsf{VIEW}_C^{\pi}(\overline{x})$. □

(Complexity.) The communication and round complexities of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ are in $\mathcal{O}(\iota^2 \cdot (O_{\text{ET}} + O_{\text{GTE}} + O_{\text{Mult}}) + |\mathbb{G}^{\text{TPC}}| \cdot O^{n \in \mathcal{O}(\iota)}_{\text{UFI-Mult}} + O^{m \in \mathcal{O}(1); n \in \mathcal{O}(|\mathbb{G}^{\text{TPC}}|)}_{\text{CRC-C}} + \iota \cdot O_{\text{RIE-}\mathcal{D}})$. By assuming the concrete specifications of the gates presented in Section 5 whose complexities are given in Table 5.6, and by setting $\mathcal{D}$, e.g., to the uniform distribution, the exact theoretical

$$\mathbb{G}^{\text{TPC}} \xrightarrow{1.} \mathbb{G}^{\text{PTPC}} \xrightarrow[\rho_{\text{CRS-1}},\ \rho_{\text{RIE-}\mathcal{D}}]{2.} G^{\text{AT}}$$

Figure 7.3: Illustration of substantial steps of $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$.

complexities are in $\mathcal{O}(\iota^3 s + \iota^2 s \cdot |\mathbb{G}^{\text{TPC}}| + \iota s \omega_\Delta^2)$, respectively, $\mathcal{O}(\iota^3 + \iota \cdot |\mathbb{G}^{\text{TPC}}| + \omega_\Delta^2)$ where ω_Δ corresponds to an upper bound on $\overline{q}_o^{(\ell)} - \underline{q}_d^{(N_d^{*(\ell)})}$, $\forall \ell \in \mathscr{P}$ with $N_{o,d}^{*(\ell)} \neq (0,0)$.[6]

7.3 Multi-Party Bartering Secure in the Malicious Model

In this section, we introduce protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ which securely implements $\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (see Definition 7.8) in the malicious model.

7.3.1 Intuition

Protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ extends protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (see Section 7.2) to provide security against active adversaries. Due to the special purpose design of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, a straightforward transformation to the malicious model is not possible. Instead, new design approaches, e.g., for restricting a party's local view and for the integration of a negotiation mechanism, are required in order to provide the same privacy properties as for $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$. More precisely, we cannot make use of the prime number encoding approach (used for the implementation of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$) in the malicious model since a corrupted party can encode arbitrary information (e.g., the encoding of the corresponding trade partner constellation graph) in its private contribution to the prime number product (for details see Section 7.2.2). Furthermore, the approach of composing a multi-party protocol of multi-party gates as well as two-party gates[7] is not viable in the presence of malicious adversaries as it is not possible to assure that corrupted parties provide consistent input for all protocols/gates they participate in.[8] Instead, it is necessary for all protocol steps to be properly intertwined in order to achieve security in the malicious model.

Figure 7.3 provides an overview of the main steps of $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$. The individual steps leading from a trade partner constellation set to an actual trade are intertwined

[6] Note that all calls of gate $\rho_{\text{RIE-}\mathcal{D}}$ can run in parallel.

[7] Note that a series of two-party gates is used in $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ in order to negotiate the quantities of the commodities to be traded.

[8] For example, a corrupted party cannot be prevented to specify different quantities as input for $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (executed between more than two parties) and $\rho^{\text{M-M}}_{\text{RIE-}\mathcal{D}}$ (executed between two parties) since different threshold keys would be required.

such that the determination of the actual trade partner graph, the set of potential trades, and the actual trade cannot be clearly distinguished anymore.

Intuitively speaking, protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ works as follows. Analogously to $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, in the first step (Transition 1., Figure 7.3) the parties jointly determine the encrypted adjacency matrix representing the compatibility graph as well as an encrypted indicator vector where the j-th entry of this vector indicates whether the j-th trade partner constellation graph of the given publicly known set of trade partner constellation graphs is a subgraph of the private compatibility graph in an oblivious fashion. The step implicitly determines the set of potential trade partner constellation graphs.

Analogously to protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, in the second step (Transition 2., Figure 7.3) of protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, each potential trade partner constellation graph is obliviously associated with its welfare by multiplying the j-th encrypted entry of the indicator vector with the welfare of the j-th trade partner constellation graph by using homomorphic scalar multiplications.

Furthermore, the parties jointly determine an encrypted matrix where an entry represents the quantity at which the party of the corresponding row index would send its offered commodity to the party of the corresponding column index (based on the actual trade graph to be determined) using gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16). Specifically, a quantity of zero indicates that the respective edge is not part of the actual trade graph, i.e., the party of the corresponding row index does not send anything to the party of the corresponding column index. In contrast to $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, gate $\rho_{\text{RIE-}\mathcal{D}}$ has to be executed between each pair of parties and all parties have to participate in each execution in order to ensure input consistency. Subsequently, all parties jointly generate one encrypted vector for each party. The j-th entry of such a vector encodes the trade partners of this party w.r.t. the j-th trade partner constellation graph as well as the corresponding quantities as determined earlier. In a final step, the protocol utilizes gate $\rho_{\text{CRS-1}}$ (see Gate Specification 11) for determining an actual trade graph uniformly at random from amongst those trade constellation graphs that were previously determined to be a subgraph of the compatibility graph.

7.3.2 Specification of Protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$

In the following, we present the details of protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ which is split into the following phases:

1. *Input Phase:* Each party P_ℓ ($\ell \in \mathscr{P}$) holds private input $\mathbf{q}^{(\ell)} = ((c_o^{(\ell)}, \overline{q}_o^{(\ell)}), (c_d^{(\ell)}, \underline{q}_d^{(\ell)}))$. Protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ is composed of gates expecting encrypted integer inputs as well as of gates expecting encrypted binary inputs. In order to reduce the calls of

the expensive ρ_{BD} gate (see Gate Specification 2), we require that all parties commit their private input as encrypted integers $[\![\mathbf{q}^{(\ell)}]\!] := (([\![c_o^{(\ell)}]\!], [\![\overline{q}_o^{(\ell)}]\!]), ([\![c_d^{(\ell)}]\!], [\![\underline{q}_d^{(\ell)}]\!]))$ as well as encrypted bit decompositions $[\![\mathbf{q}_{\text{bin}}^{(\ell)}]\!] := (([\![(c_o^{(\ell)})_{\text{bin}}]\!], [\![(\overline{q}_o^{(\ell)})_{\text{bin}}]\!]), ([\![(c_d^{(\ell)})_{\text{bin}}]\!], [\![(\underline{q}_d^{(\ell)})_{\text{bin}}]\!]))$. Thus, it is not sufficient that all parties just mutually prove plaintext knowledge of their committed input. Instead, P_ℓ also has to prove the consistency of its committed input to guarantee the correctness of the protocol output. More precisely, P_ℓ has to perform the same proofs as required for the privacy-preserving two-party bartering protocol $\pi^{\text{M-T}}_{\text{Barter-}\mathcal{D}}$ (see Req 1-3, Section 6.3). We use the shorthand notation

$$(([\![\mathbf{q}^{(1)}]\!], [\![\mathbf{q}_{bin}^{(1)}]\!], \ldots, [\![\mathbf{q}^{(\iota)}]\!], [\![\mathbf{q}_{bin}^{(\iota)}]\!])) \leftarrow \mathsf{InputPhase}(\mathbf{q}^{(1)}, ..., \mathbf{q}^{(\iota)})$$

to refer to the computation steps required for proving input consistency as well as plaintext knowledge.

2. *Construction Phase* (analogous to $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$): Based on their quotes, all parties jointly construct the encrypted adjacency matrix $[\![A]\!] := ([\![a_{\ell,\ell'}]\!])_{\iota\times\iota}$ of compatibility graph G^{C} with $a_{\ell,\ell'} := [(c_o^{(\ell)} = c_d^{(\ell')}) \wedge (\overline{q}_o^{(\ell)} \geq \underline{q}_d^{(\ell')})]$ for $\ell, \ell' \in \mathscr{P}$, and $\ell \neq \ell'$. The ciphertext $[\![[(c_o^{(\ell)} = c_d^{(\ell')}) \wedge (\overline{q}_o^{(\ell)} \geq \underline{q}_d^{(\ell')})]]\!]$ is computed in three steps (see Steps 2.1.1.1 - 2.1.1.3, Protocol 7.2).

3. *Negotiation Phase:* All parties jointly compute a second encrypted matrix $[\![A']\!] := ([\![a'_{\ell,\ell'}]\!])_{\iota\times\iota}$ where $a'_{\ell,\ell'}$ ($\ell \neq \ell'$) corresponds to the quantity of commodity $c_o^{(\ell)}$ which P_ℓ has to send to $P_{\ell'}$ provided that (ℓ, ℓ') is an edge in the actual trade graph G^{AT} yet to be selected. In order to determine a matrix entry $[\![a'_{\ell,\ell'}]\!]$, the parties jointly execute gate $\rho_{\text{RIE-}\mathcal{D}}$ (see Gate Specification 16) on inputs $\overline{q}_o'^{(\ell)}$ and $\underline{q}_d'^{(\ell')}$ which are computed as $\rho_{\text{Mult}}(([\![\overline{q}_o^{(\ell)}]\!], [\![a_{\ell,\ell'}]\!]))$ and $\rho_{\text{Mult}}(([\![\underline{q}_d^{(\ell')}]\!], [\![a_{\ell,\ell'}]\!]))$ (see Gate Specification 1), respectively, in Step 3.1.1. The two executions of the multiplication gate are necessary in order to prevent that gate $\rho_{\text{RIE-}\mathcal{D}}$ is called on an invalid input which occurs if $\overline{q}_o^{(\ell)} < \underline{q}_d^{(\ell')}$. In this case, $\overline{q}_o'^{(\ell)} = \underline{q}_d'^{(\ell')} = 0$ and $\rho_{\text{RIE-}\mathcal{D}}$ returns a fresh encryption of zero. Note that due to the fact that it is necessary to hide the edges of G^{C}, $\rho_{\text{RIE-}\mathcal{D}}$ has to be executed for each pair of parties. In order to cover the case where there are some parties P_ℓ in a trade partner constellation graph G^{TPC} with $N_{o,d}^{(\ell)}(G^{\text{TPC}}) = (0, 0)$, the parties jointly generate dummy encryptions of zero (see Step 3.1.2) which can be used as a placeholder in the mapping phase. Eventually, these placeholders indicate that for a party without any trade partners there is nothing to exchange.

4. *Evaluation Phase* (analogous to $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$): For each $G_j^{\text{TPC}} = (V, E) \in \mathbb{G}^{\text{TPC}}$ ($j \in \mathbb{N}_{\mathbb{G}^{\text{TPC}}}$), the parties obliviously check whether or not $G_j^{\text{TPC}} \sqsubseteq G^{\text{C}}$ by calling $\rho_{\text{UFI-Mult}}$

(see Section 5.1.1) on input $([\![a_{\ell_1,\ell'_1}]\!],\ldots,[\![a_{\ell_{|E|},\ell'_{|E|}}]\!])$ with $(\ell_i,\ell'_i)\in E$ and $i\in\mathbb{N}_{|E|}$. Note that $[\![a_{\ell_1,\ell'_1}]\!],\ldots,[\![a_{\ell_{|E|},\ell'_{|E|}}]\!]$ are taken from the jointly computed adjacency matrix $[\![A]\!]$ corresponding to G^{C}. At the end of the evaluation phase, each party holds vector $[\![L]\!] := ([\![e_1]\!],\ldots,[\![e_{|\mathbb{G}^{\mathrm{TPC}}|}]\!])$ where $e_j = [G_j^{\mathrm{TPC}} \sqsubseteq G^{\mathrm{C}}]$. The trade partner constellation graphs G_j^{TPC} with $e_j = 1$ constitute the set of potential trade partner constellation graphs $\mathbb{G}^{\mathrm{PTPC}}$.

5. *Prioritization Phase* (analogous to $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$): Function $\mathcal{W}(\cdot):\mathbb{G}^{\mathrm{TPC}}\rightarrow\mathbb{N}^0$ maps each trade partner constellation graph to its welfare. Each party locally computes $[\![e_j]\!]\times_h\mathcal{W}(G_j^{\mathrm{TPC}})$ $(\forall j\in\mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|})$ resulting in an encrypted vector $[\![L_0]\!] := ([\![e_1\cdot\mathcal{W}(G_1^{\mathrm{TPC}})]\!],\ldots,[\![e_{|\mathbb{G}^{\mathrm{TPC}}|}\cdot\mathcal{W}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}})]\!])$.

6. *Mapping Phase:* For each party P_ℓ, an encrypted vector

$$\begin{aligned}[\![L_\ell]\!] &:= ([\![l_{\ell,1}]\!],\ldots,[\![l_{\ell,|\mathbb{G}^{\mathrm{TPC}}|}]\!])\\ &:= ([\![N_{o,d}^{(\ell)}(G_1^{\mathrm{TPC}})]\!]\;||_h\;[\![a'_{N_o^{(\ell)},\ell}]\!]\;||_h\;[\![a'_{\ell,N_d^{(\ell)}}]\!],\ldots,\\ &\quad [\![N_{o,d}^{(\ell)}(G_{|\mathbb{G}^{\mathrm{TPC}}|}^{\mathrm{TPC}})]\!]\;||_h\;[\![a'_{N_o^{(\ell)},\ell}]\!]\;||_h\;[\![a'_{\ell,N_d^{(\ell)}}]\!])\;(\forall\ell\in\mathscr{P})\end{aligned}$$

   is jointly computed such that each party learns all of these encrypted vectors. The j-th entry of a vector L_ℓ contains the trade partners $N_o^{(\ell)}(G_j^{\mathrm{TPC}})$ and $N_d^{(\ell)}(G_j^{\mathrm{TPC}})$ of party P_ℓ as well as the quantities to be received and sent in case that G_j^{TPC} is selected as actual trade graph by protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$. The purpose of the mapping phase is to determine the parties' local views for each trade partner constellation graph.[9]

7. *Selection Phase:* Up to this point, each G_j^{TPC} is associated with an encrypted vector $([\![l_{0,j}]\!],[\![l_{1,j}]\!],\ldots,[\![l_{\iota,j}]\!])$ where $l_{0,j}=0$ if $G_j^{\mathrm{TPC}}\not\sqsubseteq G^{\mathrm{C}}$ and $l_{0,j}=\mathcal{W}(G_j^{\mathrm{TPC}})$ otherwise. Furthermore, $\forall\ell\in\mathscr{P}: l_{\ell,j}=(N_{o,d}^{(\ell)}(G_j^{\mathrm{TPC}}),a'_{N_o^{(\ell)},\ell},a'_{\ell,N_d^{(\ell)}})$ contains P_ℓ's local view of G_j^{TPC}. The parties now jointly call $\rho_{\text{CRS-1}}$ (see Gate Specification 11) on the common input $([\![L_0]\!],\ldots,[\![L_\iota]\!])$. If $\mathbb{G}^{\mathrm{PTPC}}\neq\emptyset$ (i.e., there exists at least one $j\in\mathbb{N}_{|\mathbb{G}^{\mathrm{TPC}}|}$ such that $l_{0,j}\neq 0$), $\rho_{\text{CRS-1}}$ is used to obliviously determine a random actual trade graph G^{AT} which maximizes the welfare function. The first entry of the encrypted output $[\![l_0^*]\!]$ of $\rho_{\text{CRS-1}}$ corresponds to the encrypted welfare of G^{AT}. The remaining encrypted entries $[\![l_1^*]\!],\ldots,[\![l_\iota^*]\!]$ correspond to the parties' local views w.r.t. G^{AT}. Otherwise (i.e., $\mathbb{G}^{\mathrm{PTPC}}=\emptyset$), $\rho_{\text{CRS-1}}$ is used to select $G^{\mathrm{TPC}}\leftarrow_{\$}\mathbb{G}^{\mathrm{TPC}}$ and $l_0^*=0$. The remaining steps of the selection phase deal with the latter case where

[9] Note that in order to perform the homomorphic concatenation operation, it has to be assured that no "wrapping" occurs in $\mathbb{P}$ (see Section 3.2.3). For the given context, however, this issue does not constitute a practical limitation.

$\mathbb{G}^{\text{PTPC}} = \emptyset$. First, the parties obliviously check whether or not $l_0^* = 0$ by jointly calling $(\llbracket b \rrbracket) \leftarrow \rho_{\text{BDI-ET}}((\llbracket (l_0^*)_{bin} \rrbracket, \llbracket 0_{bin} \rrbracket))$ (see Gate Specification 8). Subsequently, the local view $\llbracket l_\ell^* \rrbracket$ of each party P_ℓ is multiplied with the encrypted result $\llbracket b \rrbracket$ of the equality check by jointly calling $(\llbracket o_\ell \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket 1 - b \rrbracket, \llbracket l_\ell^* \rrbracket))$ such that all values l_ℓ^* are obliviously zeroed out in the case that no actual trade graph exists (Step 7.4, Protocol 7.2).

8. *Output Phase:* All parties jointly decrypt $(\llbracket o_1 \rrbracket, \ldots, \llbracket o_\iota \rrbracket)$ such that only P_ℓ obtains o_ℓ $(\forall \ell \in \mathscr{P})$. In case that $\mathbb{G}^{\text{PTPC}} \neq \emptyset$, P_ℓ learns its local view of the selected G^{AT} from o_ℓ. Otherwise, o_ℓ indicates that there exists no G^{AT} for the current set of parties in combination with their private input quotes and the given set of trade partner constellation graphs $\mathbb{G}^{\text{TPC}}$. We use the shorthand notation

$$(o_1, \ldots, o_\iota) \leftarrow \mathsf{OutputPhase}((\llbracket o_1 \rrbracket, \ldots, \llbracket o_\iota \rrbracket))$$

to refer to the output phase.

In the following, we prove the correctness of $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, i.e., that $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ always computes the correct output (if at least one party is honest), provided that the protocol is not prematurely aborted. In order to prove the correctness of the protocol, we have to show that (i) it suffices that each party performs the proofs specified for the input phase and that (ii) $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ in fact computes a random actual trade graph which maximizes the welfare function based on the parties' input quotes and the publicly known set of trade partner constellation graphs iff the set of potential trade partner constellation graphs is not empty. In Section 6.3, (i) has already been proven for the two-party case and can be proven analogously for the multi-party case. Thus, in the following, we focus on showing (ii).

In case that $\mathbb{G}^{\text{PTPC}} = \emptyset$, the result of the evaluation phase is a vector $\llbracket L \rrbracket := (\llbracket e_1 \rrbracket, \ldots, \llbracket e_{|\mathbb{G}^{\text{TPC}}|} \rrbracket)$ with $e_1 = \ldots = e_{|\mathbb{G}^{\text{TPC}}|} = 0$ since there exists no $G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ with $G_j^{\text{TPC}} \sqsubseteq G^{\text{C}}$. Furthermore, the prioritization phase does not change any plaintext value e_j and thus $L_0 = L$ $(j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|})$. This implies that gate $\rho_{\text{CRS-1}}$ returns an encrypted vector $(\llbracket l_0^* \rrbracket, \llbracket l_1^* \rrbracket, \ldots, \llbracket l_\iota^* \rrbracket)$ with $l_0^* = 0$ in the selection phase and in the output phase each party P_ℓ learns its output value $o_\ell = 0$ which indicates that no actual trade exists.

In case that $\mathbb{G}^{\text{PTPC}} \neq \emptyset$, the result of the evaluation phase is a vector $\llbracket L \rrbracket$ such that L has *Hamming weight* $\mathcal{H}(L) = |\mathbb{G}^{\text{PTPC}}|$. The prioritization phase determines the encrypted vector $\llbracket L_0 \rrbracket := (\llbracket l_{0,1} \rrbracket, \ldots, \llbracket l_{0,|\mathbb{G}^{\text{TPC}}|} \rrbracket)$ where $\llbracket l_{0,j} \rrbracket$ is associated with G_j^{TPC} and $l_{0,j} = \mathcal{W}(G_j^{\text{TPC}})$ if $G_j^{\text{TPC}} \sqsubseteq G^{\text{C}}$ and $l_{0,j} = 0$ otherwise. According to the definition of $\mathcal{G}^{\text{M}}_{\text{CRS-}i^*}$, gate $\rho_{\text{CRS-1}}$ returns an encrypted vector $(\llbracket l_0^* \rrbracket, \llbracket l_1^* \rrbracket, \ldots, \llbracket l_\iota^* \rrbracket) := (\llbracket l_{0,j^*} \rrbracket, \llbracket l_{1,j^*} \rrbracket, \ldots, \llbracket l_{\iota,j^*} \rrbracket)$ where $j^* \leftarrow_{\$} \{j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|} : \mathcal{W}(G_j^{\text{TPC}}) = max(\mathcal{W}(G_1^{\text{TPC}}), \ldots, \mathcal{W}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}}))\}$. Since in

Protocol 7.2. Specification of $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$.

1 Input Phase

1.1 $((\llbracket \mathbf{q}^{(1)} \rrbracket, \llbracket \mathbf{q}^{(1)}_{bin} \rrbracket, \ldots, \llbracket \mathbf{q}^{(\iota)} \rrbracket, \llbracket \mathbf{q}^{(\iota)}_{bin} \rrbracket)) \leftarrow \mathsf{InputPhase}(\mathbf{q}^{(1)}, ..., \mathbf{q}^{(\iota)})$

2 Construction Phase

2.1 For each P_ℓ ($\ell \in \mathscr{P}$):

2.1.1 For each $P_{\ell'}$ ($\ell' \in \mathscr{P} \backslash \{\ell\}$):

2.1.1.1 All parties jointly compute
$(\llbracket b_1 \rrbracket) \leftarrow \rho_{\text{BDI-ET}}((\llbracket (c_o^{(\ell)})_{bin} \rrbracket, \llbracket (c_d^{(\ell')})_{bin} \rrbracket))$

2.1.1.2 All parties jointly compute
$(\llbracket b_2 \rrbracket) \leftarrow \rho_{\text{BDI-GTE}}((\llbracket (\overline{q}_o^{(\ell)})_{bin} \rrbracket, \llbracket (\underline{q}_d^{(\ell')})_{bin} \rrbracket))$

2.1.1.3 All parties jointly compute $(\llbracket a_{\ell,\ell'} \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket))$

3 Negotiation Phase

3.1 For each P_ℓ ($\ell \in \mathscr{P}$):

3.1.1 For each $P_{\ell'}$ ($\ell' \in \mathscr{P} \backslash \{\ell\}$):

3.1.1.1 All parties jointly compute $(\llbracket \overline{q}_o'^{(\ell)} \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket \overline{q}_o^{(\ell)} \rrbracket, \llbracket a_{\ell,\ell'} \rrbracket))$

3.1.1.2 All parties jointly compute $(\llbracket \underline{q}_d'^{(\ell')} \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket \underline{q}_d^{(\ell')} \rrbracket, \llbracket a_{\ell,\ell'} \rrbracket))$

3.1.1.3 All parties jointly compute
$\llbracket a'_{\ell,\ell'} \rrbracket := (\llbracket q^{(\ell,\ell')}_{c_o^{(\ell)}} \rrbracket) \leftarrow \rho_{\text{RIE-}\mathcal{D}}((\llbracket \underline{q}_d'^{(\ell')} \rrbracket, \llbracket \overline{q}_o'^{(\ell)} \rrbracket))$

3.1.2 All parties set $\llbracket a'_{\ell,0} \rrbracket, \llbracket a'_{0,\ell} \rrbracket := \llbracket 0 \rrbracket$

4 Evaluation Phase

4.1 For each $G_j^{\text{TPC}} = (V, E) \in \mathbb{G}^{\text{TPC}}$ with $(\ell_i, \ell'_i) \in E$ and $j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}, i \in \mathbb{N}_{|E|}$:

4.1.1 All parties jointly compute
$(\llbracket e_j \rrbracket) \leftarrow \rho_{\text{UFI-Mult}}((\llbracket a_{\ell_1,\ell'_1} \rrbracket, \ldots, \llbracket a_{\ell_{|E|},\ell'_{|E|}} \rrbracket))$

4.2 All parties set $\llbracket L \rrbracket := (\llbracket e_1 \rrbracket, \ldots, \llbracket e_{|\mathbb{G}^{\text{TPC}}|} \rrbracket)$

5 Prioritization Phase

5.1 Each party locally computes
$\llbracket L_0 \rrbracket := (\llbracket e_1 \rrbracket \times_h \mathcal{W}(G_1^{\text{TPC}}), \ldots, \llbracket e_{|\mathbb{G}^{\text{TPC}}|} \rrbracket \times_h \mathcal{W}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}}))$

6 Mapping Phase

6.1 All parties jointly generate $\llbracket L_1 \rrbracket, \ldots, \llbracket L_\iota \rrbracket$ where

$$\begin{aligned}
\llbracket L_1 \rrbracket := (&\llbracket N^{(1)}_{o,d}(G_1^{\text{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(1)},1} \rrbracket \,||_h\, \llbracket a'_{1,N_d^{(1)}} \rrbracket, \ldots, \\
&\llbracket N^{(1)}_{o,d}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(1)},1} \rrbracket \,||_h\, \llbracket a'_{1,N_d^{(1)}} \rrbracket) \\
&\vdots \\
\llbracket L_\iota \rrbracket := (&\llbracket N^{(\iota)}_{o,d}(G_1^{\text{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(\iota)},\iota} \rrbracket \,||_h\, \llbracket a'_{\iota,N_d^{(\iota)}} \rrbracket, \ldots, \\
&\llbracket N^{(\iota)}_{o,d}(G_{|\mathbb{G}^{\text{TPC}}|}^{\text{TPC}}) \rrbracket \,||_h\, \llbracket a'_{N_o^{(\iota)},\iota} \rrbracket \,||_h\, \llbracket a'_{\iota,N_d^{(\iota)}} \rrbracket)
\end{aligned}$$

7 Selection Phase

7.1 All parties jointly compute
$((\llbracket l_0^* \rrbracket, \llbracket l_1^* \rrbracket, \ldots, \llbracket l_\iota^* \rrbracket)) \leftarrow \rho_{\text{CRS-1}}((\llbracket L_0 \rrbracket, \llbracket L_1 \rrbracket, \ldots, \llbracket L_\iota \rrbracket))$

7.2 All parties jointly compute $(\llbracket (l_0^*)_{bin} \rrbracket) \leftarrow \rho_{\text{BD}}(\llbracket l_0^* \rrbracket)$

7.3 All parties jointly compute $(\llbracket b \rrbracket) \leftarrow \rho_{\text{BDI-ET}}((\llbracket (l_0^*)_{bin} \rrbracket, \llbracket 0_{bin} \rrbracket))$

7.4 For ℓ from 1 to ι:

7.4.1 $(\llbracket o_\ell \rrbracket) \leftarrow \rho_{\text{Mult}}((\llbracket 1 - b \rrbracket, \llbracket l_\ell^* \rrbracket))$

8 Output Phase

8.1 $(o_1, \ldots, o_\iota) \leftarrow \mathsf{OutputPhase}((\llbracket o_1 \rrbracket, \ldots, \llbracket o_\iota \rrbracket))$

Step 7.3 (Protocol 7.2) $b = 0$, $[\![o_\ell]\!] := [\![l^*_\ell]\!]$ for all $\ell \in \mathscr{P}$. At the end of the output phase, each party P_ℓ learns its local view of the selected actual trade.

Corollary 7.1. *Protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ securely computes functionality $\mathcal{F}^{\text{M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ in the malicious model.*

Corollary 7.1 and thus the security of protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ directly follows from Theorem 4.3 (Section 4.4.3).

(Complexity.) The communication and round complexities of protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ are in $\mathcal{O}(\iota^2 \cdot (O^{m\in\mathcal{O}(s)}_{\text{BDI-ET}} + O^{m\in\mathcal{O}(s)}_{\text{BDI-GTE}} + O_{\text{Mult}} + O_{\text{RIE-}\mathcal{D}}) + |\mathbb{G}^{\text{TPC}}| \cdot O^{n\in\mathcal{O}(\iota)}_{\text{UFI-Mult}} + O^{n\in\mathcal{O}(1);m\in\mathcal{O}(|\mathbb{G}^{\text{TPC}}|)}_{\text{CRS}} + O^{m\in\mathcal{O}(s)}_{\text{BD}})$. By assuming the concrete specifications of the gates presented in Section 5 whose complexities are given in Table 5.6, and by setting $\mathcal{D}$, e.g., to the uniform distribution, the exact theoretical complexities are in $\mathcal{O}(\iota^3 s^2 + \iota^3 s\omega^2_\Delta + \iota^2 s \cdot |\mathbb{G}^{\text{TPC}}| + \iota s^2 \cdot |\mathbb{G}^{\text{TPC}}|)$, respectively, $\mathcal{O}(\iota^2 \cdot log(s) + \iota^3\omega_\Delta + \iota^2\omega^2_\Delta + \iota \cdot |\mathbb{G}^{\text{TPC}}| + log(s) \cdot |\mathbb{G}^{\text{TPC}}|)$ where ω_Δ corresponds to an upper bound on $\overline{q}'^{(\ell)}_o - \underline{q}'^{(\ell')}_d \; \forall \ell, \ell' \in \mathscr{P}$ with $\ell \neq \ell'$.

7.4 Actual Trade Partner Constellation Selection

The selection of the actual trade partner constellation graph (from the set of potential trade partner constellation graphs) of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ is influenced by the choice of the welfare function $\mathcal{W}(\cdot) : \mathbb{G}^{\text{TPC}} \rightarrow \mathbb{N}^0$. While our bartering protocols support any selection strategy as long as it is a function of the considered trade partner constellation graphs in $\mathbb{G}^{\text{TPC}}$, in the following, we propose two concrete strategies and define the corresponding welfare functions.

Assuming that there are multiple potential trade partner constellation graphs of which the actual trade partner constellation graph G^{ATPC} is to be obliviously selected, a concrete selection strategy (relevant for practice) is to maximize the number of parties getting to trade. The number of parties getting to trade w.r.t. a given trade partner constellation graph corresponds to the number of edges in that graph. Let $G^{\text{TPC}} = (V, E)$. We can define $\mathcal{W}$ as

$$\mathcal{W}(G^{\text{TPC}}) = |E|$$

in order to ensure that protocols $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ select G^{ATPC} such that the number of parties getting to trade is maximized. Since the number of edges of each $G^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ is publicly known, $\mathcal{W}(G^{\text{TPC}})$ can be determined by each party such that each potential trade partner constellation graph can be obliviously associated with its welfare by making use of homomorphic scalar multiplication only.

Going one step further, we can also consider the trade cycle length when selecting an actual trade partner constellation graph. Depending on the bartering context (e.g., kidney exchange), a fixed maximum length for trade cycles is required in order to reduce the impact of a dropout as well as to facilitate a simultaneous exchange of commodities. Furthermore, for some applications (e.g., kidney exchange) it is the case that considering trade cycles of lengths larger than 3 does not lead to a significant increase of the number of parties getting to trade [1], but considerably increases the search space. A further reason for concentrating on small trade cycle lengths results from the possible requirement to conduct a simultaneous exchange of commodities and from logistical constraints (cf. [5,6]). Thus, one further goal is to minimize the trade cycle lengths while maximizing the number of parties getting to trade. More precisely, a concrete selection strategy (relevant for practice) is to maximize the number of parties getting to trade and in case that there exist multiple potential trade partner constellation graphs with maximum number of edges, then the one with the most trade cycles is selected. Let $G^{\text{TPC}} = (V, E)$ and let c be the number of cycles in G^{TPC}. We can define $\mathcal{W}'$ as

$$\mathcal{W}'(G^{\text{TPC}}) = \left\lfloor \frac{|E|}{2} \right\rfloor \cdot |E| + c, \tag{7.1}$$

where $\lfloor |E|/2 \rfloor$ corresponds to the maximum number of disjoint cycles a graph with $|E|$ edges can have. In order to see that $\mathcal{W}'(\cdot)$ implements the described selection strategy, consider two arbitrary trade partner constellation graphs G_1^{TPC} and G_2^{TPC} for the same set of parties. In case that both graphs have the same amount of edges and it holds $\mathcal{W}'(G_1^{\text{TPC}}) > \mathcal{W}'(G_2^{\text{TPC}})$, Equation 7.1 obviously implies the number of cycles in G_1^{TPC} is strictly larger than the number of cycles in G_2^{TPC}. To ensure that $\mathcal{W}'(G_1^{\text{TPC}}) > \mathcal{W}'(G_2^{\text{TPC}})$ holds in case that G_1^{TPC} has more edges than G_2^{TPC}, it suffices to show that

$$min\{\mathcal{W}'(G^{\text{TPC}}) | G^{\text{TPC}} \text{ has } |E| \text{ edges}\} > max\{\mathcal{W}'(G^{\text{TPC}}) | G^{\text{TPC}} \text{ has } |E-1| \text{ edges}\}.$$

More precisely, we have to show that the following inequation holds:

$$\left\lfloor \frac{|E|}{2} \right\rfloor \cdot |E| + 1 \overset{!}{>} \left\lfloor \frac{|E|-1}{2} \right\rfloor \cdot (|E|-1) + \left\lfloor \frac{|E|-1}{2} \right\rfloor. \tag{7.2}$$

Proof. In order to show that Equation 7.2 holds, it suffices to consider the case where $|E|$ is odd and thus $\lfloor |E|/2 \rfloor = \lfloor (|E|-1)/2 \rfloor = (|E|-1)/2$:

$$\begin{aligned} & \frac{|E|-1}{2} \cdot |E| + 1 > \frac{|E|-1}{2} \cdot (|E|-1) + \frac{|E|-1}{2} \\ \Leftrightarrow\ & \frac{|E|-1}{2} \cdot |E| + 1 > \frac{|E|-1}{2} \cdot |E| - \frac{|E|-1}{2} + \frac{|E|-1}{2} \\ \Leftrightarrow\ & 1 > 0 \end{aligned}$$

□

Since also the number of cycles in each trade partner constellation graph is publicly known, the welfare function $\mathcal{W}'(\cdot)$ can be integrated analogously to $\mathcal{W}(\cdot)$ into protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$.

It is important to note that the choice of the selection strategy does not influence the communication and round complexity of the privacy-preserving bartering protocols since the involved computations can be computed locally.

7.5 Optimized Privacy-Preserving Subgraph Check

In the following, we present an optimized variant of the subgraph checks that are performed in $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ to obliviously check whether $G_j^{\text{TPC}} \sqsubseteq G^{\text{C}}$ ($\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$). Furthermore, we make the following assumptions:

- $\mathbb{G}^{\text{TPC}}$ covers the complete search space w.r.t. an upper bound m on the trade cycle length which is considered for determining an actual trade partner constellation graph.
- the upper bound m on the trade cycle length is fixed and independent of the number of parties.

As argued in Section 7.4, in some bartering scenarios the maximum cycle length considered for $\iota > 2$ is $m = 3$. In this case, $\mathbb{G}^{\text{TPC}}$ contains all trade partner constellation graphs for ι parties consisting only of 2- and 3-trade cycles. For $\iota = 4$ and $m = 3$, Table 7.3 lists all trade partner constellation graphs in $\mathbb{G}^{\text{TPC}}$.

Based on the above assumptions, it is possible to reduce the number of executions of ρ_{Mult} (see Gate Specification 1) in the evaluation phase of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ as well as $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ from $\mathcal{O}(\iota \cdot |\mathbb{G}^{\text{TPC}}|)$ to $\mathcal{O}(\iota!/(\iota-m)! + |\mathbb{G}^{\text{TPC}}|)$ (cf. Equation 7.3 below).[10] In the following, we assume that the encrypted adjacency matrix $[\![A]\!]$ representing G^{C} which is determined in the construction phase of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, respectively, has already been computed.

The key idea of our approach is to perform the subgraph checks $G_j^{\text{TPC}} \stackrel{?}{\sqsubseteq} G^{\text{C}}$ ($G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$, $\forall j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|}$) in the evaluation phase in a specific order (determined by the number of trade cycles in a trade partner constellation graph G_j^{TPC}) such that some of the encrypted evaluation results can be reused later in the evaluation phase in order to reduce the number of executions of gate ρ_{Mult}. Let $|G_j^{\text{TPC}}|_{\text{c}}$ refer to the number of trade

[10] Note that according to the first assumption $\iota!/(\iota-m)! < (\iota-1) \cdot |\mathbb{G}^{\text{TPC}}|$ for $\iota \geq 3$ and $m \geq 2$.

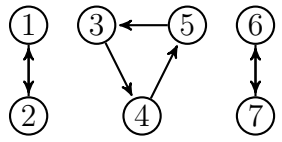
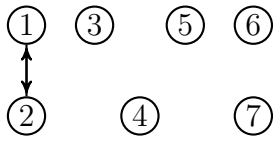
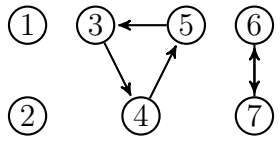
$G_z^{\text{TPC}} \in \mathfrak{C}_3$	$G_x^{\text{TPC}} \in \mathfrak{C}_1$	$G_y^{\text{TPC}} \in \mathfrak{C}_2$
		
$e_z := [G_z^{\text{TPC}} \overset{?}{\sqsubseteq} G^{\text{C}}] = e_x \cdot e_y$	$e_x := [G_x^{\text{TPC}} \overset{?}{\sqsubseteq} G^{\text{C}}]$	$e_y := [G_y^{\text{TPC}} \overset{?}{\sqsubseteq} G^{\text{C}}]$

Table 7.2: Composition of $G_z^{\text{TPC}} \in \mathfrak{C}_3$ from $G_x^{\text{TPC}} \in \mathfrak{C}_1$ and $G_y^{\text{TPC}} \in \mathfrak{C}_2$ for $\iota = 7$.

cycles in G_j^{TPC}.[11] First, $\lfloor \frac{\iota}{2} \rfloor$ disjoint subsets $\mathfrak{C}_i$ ($i \in \mathbb{N}_{\lfloor \frac{\iota}{2} \rfloor}$) of $\mathbb{G}^{\text{TPC}}$ are computed where

$$\mathfrak{C}_i := \{G_j^{\text{TPC}} \in \mathbb{G}^{\text{TPC}} : |G_j^{\text{TPC}}|_{\text{c}} = i\},\ \forall i \in \mathbb{N}_{\lfloor \frac{\iota}{2} \rfloor}.$$

We refer to $i \in \mathbb{N}_{\lfloor \frac{\iota}{2} \rfloor}$ as the *order* of a trade partner constellation graph, i.e., a trade partner constellation graph of order i has i trade cycles. Our approach requires an adjustment of the evaluation phase such that the subgraph checks for the elements in $\mathfrak{C}_i \subset \mathbb{G}^{\text{TPC}}$ are performed before the subgraph checks for the elements in $\mathfrak{C}_{i+1} \subset \mathbb{G}^{\text{TPC}}$. For a trade partner constellation graph in $\mathfrak{C}_1$, the way the subgraph check is performed remains unchanged, i.e., by multiplying the encrypted entries of adjacency matrix $[\![A]\!]$ of G^{C} (using gate ρ_{Mult}) representing the edges of that trade partner constellation graph. For the remaining subgraph checks, the encrypted adjacency matrix $[\![A]\!]$ is not used anymore and only a single execution of ρ_{Mult} is required in order to compute the encrypted evaluation result of the subgraph check (instead of $|E| - 1$ many executions of ρ_{Mult} for $(V, E) =: G^{\text{TPC}} \overset{?}{\sqsubseteq} G^{\text{C}}$ as is the case in $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$). This is due to the following observation: It is possible to decompose a trade partner constellation graph $G_z^{\text{TPC}} \in \mathfrak{C}_{i^*}$ ($i^* \in \mathbb{N}_{\lfloor \frac{\iota}{2} \rfloor} \backslash \{1\}$) into two trade partner constellation graphs $G_x^{\text{TPC}}, G_y^{\text{TPC}} \in \mathbb{G}^{\text{TPC}}$ ($x \neq y$) of a lower order i' (resp., i'') such that $G_z^{\text{TPC}} = G_x^{\text{TPC}} \sqcup G_y^{\text{TPC}}$ and thus $i^* = i' + i''$.[12] An example for such a composition is given in Figure 7.2. By construction, the encrypted evaluation results $[\![e_x]\!]$ and $[\![e_y]\!]$ resulting from the subgraph checks $G_x^{\text{TPC}} \overset{?}{\sqsubseteq} G^{\text{C}}$ and $G_y^{\text{TPC}} \overset{?}{\sqsubseteq} G^{\text{C}}$ already have been computed. Thus, $[\![e_z]\!]$ can be computed as $([\![e_z]\!]) \leftarrow \rho_{\text{Mult}}(([\![e_x]\!], [\![e_y]\!]))$. The number of ρ_{Mult} executions in the adjusted evaluation phase required for all subgraph checks can be determined as

$$\#\rho_{\text{Mult}} = \underbrace{|\mathbb{G}^{\text{TPC}}|}_{T_1} + \underbrace{\sum_{c=3}^{m} (c-2) \cdot \frac{\iota!}{c(\iota - c)!}}_{T_2}. \tag{7.3}$$

[11] Note that by construction these trade cycles are simultaneously executable.

[12] Note that a composition might not be unique.

$\mathfrak{C}_i$	Cycles	ρ_{Mult} calls	$G_j^{\text{TPC}} = (\{1,2,3,4\}, E_j)$	Composition
$\mathfrak{C}_1$	1×2-cycle	1	$E_1 = \{$(1,2),(2,1)$\}$	—
			$E_2 = \{$(1,3),(3,1)$\}$	—
			$E_3 = \{$(1,4),(4,1)$\}$	—
			$E_4 = \{$(2,3),(3,2)$\}$	—
			$E_5 = \{$(2,4),(4,2)$\}$	—
			$E_6 = \{$(3,4),(4,3)$\}$	—
	1×3-cycle	2	$E_7 = \{(1,2),(2,3),(3,1)\}$	—
			$E_8 = \{(1,2),(2,4),(4,1)\}$	—
			$E_9 = \{(1,3),(3,2),(2,1)\}$	—
			$E_{10} = \{(1,3),(3,4),(4,1)\}$	—
			$E_{11} = \{(1,4),(4,3),(3,1)\}$	—
			$E_{12} = \{(1,4),(4,2),(2,1)\}$	—
			$E_{13} = \{(2,3),(3,4),(4,2)\}$	—
			$E_{14} = \{(2,4),(4,3),(3,2)\}$	—
$\mathfrak{C}_2$	2×2-cycle	1	$E_{15} = \{$(1,2),(2,1), (3,4),(4,3)$\}$	$G_1^{\text{TPC}} \sqcup G_6^{\text{TPC}}$
			$E_{16} = \{$(1,3),(3,1), (2,4),(4,2)$\}$	$G_2^{\text{TPC}} \sqcup G_5^{\text{TPC}}$
			$E_{17} = \{$(1,4),(4,1), (2,3),(3,2)$\}$	$G_3^{\text{TPC}} \sqcup G_4^{\text{TPC}}$

Table 7.3: Relationship between trade partner constellation graphs of different order for $\iota = 4$, $m = 3$, $i \in \{1,2\}$, and $j \in \mathbb{N}_{|\mathbb{G}^{\text{TPC}}|=17}$.

For each trade partner constellation graph, the subgraph check requires at least one execution of ρ_{Mult} which is captured in term T_1 in Equation 7.3. The subgraph checks for trade partner constellation graphs with only one trade cycle of length $c \geq 3$ require more than one execution of ρ_{Mult}. In particular, a trade cycle of length c requires $c-1$ executions of ρ_{Mult} of which one is already captured by term T_1. Furthermore, there exists $\iota!/c(\iota-c)!$ trade partner constellation graphs that contain a single trade cycle of length c for ι parties. The number of these additional executions of ρ_{Mult} is captured by term T_2 in Equation 7.3.

It is important to note that the proposed modification of the evaluation phase for protocols $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ are independent of the security model and do not influence the security since only the computation order and the number of ρ_{Mult} executions are involved.

Example 7.2. To illustrate the reuse of evaluation results for a complete search space $\mathbb{G}^{\text{TPC}}$, for $\iota = 4$ and $m = 3$ the space of trade partner constellations corresponds to $\mathbb{G}^{\text{TPC}} = \{G_1^{\text{TPC}}, \ldots, G_{17}^{\text{TPC}}\}$ which divides into $\mathfrak{C}_1 = \{G_1^{\text{TPC}}, \ldots, G_{14}^{\text{TPC}}\}$ and $\mathfrak{C}_2 = \{G_{15}^{\text{TPC}}, G_{16}^{\text{TPC}}, G_{17}^{\text{TPC}}\}$. The corresponding trade partner constellation graphs are specified in Table 7.3. The trade partner constellation graphs in $\mathfrak{C}_2$ can be composed of those trade partner constellation graphs in $\mathfrak{C}_1$ which have one 2-trade cycle. This is illustrated in Table 7.3 by the means of colors. For instance, G_1^{TPC} and G_6^{TPC} consist of a single 2-trade cycle highlighted purple, respectively, brown. These two cycles together constitute G_{15}^{TPC} (as indicated by the reoccurrence of both colors in the set of edges of G_{15}^{TPC}). Consequently, the evaluation results of the subgraph checks for G_1^{TPC} and G_6^{TPC} can be reused for the subgraph check involving G_{15}^{TPC}.

7.6 Multi-Party Bartering based on Maximum Weight Matching

The high-level design approach of both privacy-preserving multi-party bartering protocols presented in Section 7.2 and Section 7.3 has the special feature to define restrictions of the lengths of trade cycles when selecting an actual trade partner constellation graph G^{ATPC}. As discussed in Section 7.4, this feature is of practical importance in the context of bartering. As a side effect, this approach additionally allows the specification of arbitrary selection strategies for G^{ATPC}, provided that the strategy is a function on the considered trade partner constellation graphs. However, the communication and round complexities of both protocols depend on the size of $\mathbb{G}^{\text{TPC}}$ which—when searching for arbitrary trade cycles up to a certain length—grows exponentially in the number of parties. This is due to the fact that as soon as the restriction of the trade cycle length is supported, the underlying decision problem, i.e., deciding whether a given graph has a perfect cycle cover containing cycles of length at most m, is NP-complete (see [1]). Considering all possible trade partner constellations implies a cardinality of $\iota! - 1$ for $\mathbb{G}^{\text{TPC}}$. This fact already indicates that the application of the protocols presented in Section 7.2 and Section 7.3 is targeted at scenarios where either the search space $\mathbb{G}^{\text{TPC}}$ or the number of parties is restricted.

For this reason, we additionally analyzed an alternative design approach which implicitly operates on all possible trade partner constellations while achieving communication and round complexities that are polynomially bounded by the number of parties. This alternative approach allows the determining of an actual trade partner constellation ensuring the same level of privacy as for the first approach. However, the advantage of the reduced complexities comes with the limitation that the trade cycle length cannot be restricted and that it is not possible to consider only some specific trade partner con-

stellations. Furthermore, selection strategies, i.e., welfare functions (see Definition 7.4) for selecting an actual trade partner constellation from the set of potential trade partner constellations can only be specified by the means of edge weights. This reduces the range of practically relevant selection strategies to the maximization of the number of parties getting to trade (other strategies, e.g., the minimization of the cycle lengths (cf. Equation 7.1, Section 7.4), require the access to the edges *and* the nodes). At the core of our alternative approach is a novel privacy-preserving implementation of the *Hungarian Algorithm* [78,88] which allows multiple parties to compute a maximum weight (perfect) matching.

In the following, we briefly sketch the alternative approach and summarize the comparison results between both approaches w.r.t. their practicality. For further details, we refer to [117,125].

The alternative approach is related to an approach sketched in [1] which deals with finding trade cycles in the conventional (non privacy-preserving) bartering setting: Based on $\mathbf{Q}$, a bipartite weighted graph referred to as *barter graph* is constructed such that a maximum weight (perfect) matching encodes an actual trade partner constellation that maximizes the number of parties getting to trade:

Definition 7.9 (Barter Graph). *For a given set of quotes* $\mathbf{Q} = (\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\iota)})$ *with* $\mathbf{q}^{(\ell)} = (\mathbf{o}^{(\ell)}, \mathbf{d}^{(\ell)}) = ((c_o^{(\ell)}, \overline{q}_o^{(\ell)}), (c_d^{(\ell)}, \underline{q}_d^{(\ell)}))$ $(\forall \ell \in \mathscr{P})$*, a* barter graph G^B *is a weighted bipartite graph* $G = (U, W, E, \omega_\epsilon)$ *with* $U := \{\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(\iota)}\}$, $W := \{\mathbf{o}^{(1)}, \ldots, \mathbf{o}^{(\iota)}\}$, $E := \{\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell')}\} : \ell, \ell' \in \mathscr{P}, [(c_d^{(\ell)} = c_o^{(\ell')}) \wedge (\underline{q}_d^{(\ell)} \leq \overline{q}_o^{(\ell')})] = 1\} \cup \{\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell)}\}\}$*, and* $\forall \{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell')}\} \in E$ *we set*

$$\omega_\epsilon(\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell')}\}) := \begin{cases} 2 & \text{if } \ell \neq \ell' \\ 1 & \text{otherwise} \end{cases}.$$

We distinguish between *trivial edges* $\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell)}\}$ and *non-trivial* edges $\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell')}\}$ with $\ell \neq \ell'$. A trivial edge $\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell)}\}$ with $\omega_\epsilon(\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell)}\}) = 1$ indicates that P_ℓ keeps its offered commodity while a non-trivial edge $\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell')}\}$ with $\omega_\epsilon(\{\mathbf{d}^{(\ell)}, \mathbf{o}^{(\ell')}\}) = 2$ indicates that P_ℓ's demand can be satisfied by $P_{\ell'}$'s offer. By construction, a barter graph contains at least one perfect matching (the one in which each party keeps its own commodity). Note that a maximum weight (perfect) matching in a barter graph 'prefers' non-trivial edges, hence the matching encodes an actual trade partner constellation that maximizes the number of parties getting to trade.

One direct approach for computing a maximum weight (perfect) matching in G^{B} is the Hungarian algorithm. In [117,125], we designed a privacy-preserving protocol implementing a slightly adapted variant of the Hungarian algorithm that provides security in the semi-honest model. For an encrypted barter graph (computed as a part of the pro-

tocol based on the parties' private inputs) it privately computes each party's local view of an actual trade partner constellation that maximizes the number of parties getting to trade.

For a fixed security parameter, the communication and round complexities of this protocol (implementing the alternative design approach in a privacy-preserving fashion) are in $\mathcal{O}(\iota^6)$. When considering all possible trade partner constellations for ι parties and considering the same security parameter, the corresponding complexities of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (restricted to the selection of an actual trade constellation, i.e., by skipping the negotiation phase) are in $\mathcal{O}(\iota^2 \cdot \iota!)$ and $\mathcal{O}(\iota \cdot \iota!)$, respectively. By merely comparing the theoretical complexities of both protocols, the protocol from [117,125] outperforms protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ when considering more than 6 parties w.r.t. the communication complexity, respectively, more than 7 parties w.r.t. the round complexity. However, the evaluation of the Java implementations of both approaches lead to a contrary result. The implementation of the alternative approach from [117,125] indicates the protocol's strict practical limitation since for 5 parties (resp., 6 parties) and a key size of 1024 bit, the corresponding protocol already has a run time of more than 20 hours (resp., more than 2 days). For details we refer to [117]. In contrast, the Java implementation of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (see Chapter 8) achieves a run time of about 30 seconds (resp., about 4 minutes) for 5 parties (resp., 6 parties) for a key size of 1024 bit and without a restriction on the trade cycles lengths.[13]

This result is mainly due to the fact that the $\mathcal{O}$-notation does not consider constant factors or summands of lower order. While the protocol flow in the approach from [117, 125] has to be completely hidden from the parties impeding the potential for optimization, protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ allows for significant practical improvements which is due to the fact that the parties process a public set of trade partner constellation graphs. The impossibility of the alternative approach to impose restrictions on the trade cycle lengths as well as its limitations w.r.t. specifying selection strategies are further practical limitations. Consequently, the focus of the remaining chapters lies on the trade partner constellation set based approach used in Section 7.2 and Section 7.3.

7.7 Summary and Future Work

In this chapter, we introduced a novel privacy-preserving multi-party bartering terminology which we used to describe our novel privacy-preserving bartering protocols secure in the semi-honest model and the malicious model, respectively. Subsequently, we

[13]The performance evaluations have been conducted on standard PC hardware (for more details, see Chapter 8).

presented an optimized privacy-preserving subgraph check reducing the complexities of our bartering protocols and specified two examples for welfare functions as strategies for selecting an actual trade constellation. The analysis of the alternative approach for designing a privacy-preserving multi-party bartering protocol based on finding maximum weight matchings in bipartite graphs revealed the advantages and superiority of the first approach presented in this thesis.

An interesting direction for future work is the extension of our bartering setting from $(1:1)$ trades to $(n:n)$ trades ($\iota \geq n \in \mathbb{N}$) where a party's demand can be satisfied by at most n other parties and a party's offer can be used to contribute to satisfy the demands of at most n other parties. This extension involves new challenges w.r.t. the negotiation of the actual quantities and raises the question to what extent the complexities of the privacy-preserving bartering protocols are increased. A further direction for future work is to elaborate how parties can reveal their contact information to their trade partners (but to no other parties) determined by a privacy-preserving bartering protocol in order to allow for further interactions (e.g., for shipping a commodity). One possibility is that each party encrypts its contact information with a symmetric encryption scheme and places the corresponding ciphertext on a public online bulletin board. Then, a privacy-preserving bartering protocol can be modified in such a way that instead of learning the party indices of its trade partners, a party learns their symmetric keys which in turn can be used to obtain their contact information.

CHAPTER 8

Implementation & Performance Evaluation

The theoretical complexities of our privacy-preserving multi-party protocols depend on the cardinality of the set of trade partner constellations which grows exponentially in the number of parties when considering all possible trade partner constellations for trade cycles up to a specific length. Note that this is not a design issue, but can be attributed to the fact that the decision variant of the underlying problem is NP-complete (for details see Section 7.6). This already indicates that the two protocols $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ and $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ will only be practical for a (very) limited number of parties.

The purpose of this chapter is to determine the number of parties that can jointly participate in a privacy-preserving multi-party bartering protocol such that an actual trade is determined in a reasonable amount of time. We focus on the implementation of our multi-party bartering protocol providing security in the semi-honest model for the following reasons: Protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ is more efficient than protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ which is primarily due to (1) the substantially more complex approach to negotiate the actual quantities in protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (for each pair of parties all parties jointly have to determine a pair of quantities) and (2) the fact that protocol $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ also requires the execution of various zero knowledge proofs in order to enforce semi-honest behavior. Furthermore, the semi-honest model is well justified for real-world applications, especially when the protocol behavior is hidden in complex software (see, e.g., [77]) which applies to our privacy-preserving multi-party bartering protocol. Nevertheless, a protocol allowing multiple parties to securely barter in the malicious model is an important milestone and can be of great value in some very sensitive contexts like the exchange of secret information. However, we leave the performance evaluation of $\pi^{\text{M-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ as future work.

For the implementation of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$, we decided to build our own SMPC library since existing libraries do not match all of our requirements (for details see Section 8.1). In the remaining chapter, we refer to $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (secure in the semi-honest model) whenever we use the phrase *the/our privacy-preserving bartering protocol*. Throughout this chapter, when we say "implementation" (resp., "implementing") we actually mean "Java implementation" (resp., "implementing in Java"). For convenience, we do not distinguish between protocols and gates when describing the implementation.

Contributions: The contributions presented in this chapter include a novel SMPC library tailored to some special requirements. Besides a modular communication infrastructure and convenient means to perform comprehensive performance evaluations of implemented protocols, the library provides various fundamental privacy-preserving building blocks as well as components that facilitate the implementation and testing of privacy-preserving bartering protocols. The most important contribution of this chapter is the comprehensive analysis of the performance results of our privacy-preserving multi-party bartering protocol that is implemented on top of the SMPC library. Among other things, the performance analysis shows for how many parties our privacy-preserving bartering protocol can be considered practical which cannot be determined by considering the theoretical complexities only.

The work presented in this chapter has been done in collaboration with Benjamin Assadsolimani (a graduate student at RWTH Aachen University and a student assistant at the Research Group IT-Security at RWTH Aachen University at that time) who implemented the SMPC library.

Outline: In Section 8.1, we introduce a novel SMPC library and its components. Subsequently, we present the performance evaluation of our privacy-preserving multi-party bartering protocol implemented on top of this library (Section 8.2).

8.1 SMPC Library

In this section, we present our SMPC library implemented in Java (using JDK 1.8). The major goal of designing this library is to allow for the implementation of SMPC protocols in order to evaluate their performance which eventually allows the evaluation of the practicality of our privacy-preserving multi-party bartering protocol. This leads us to the following requirements:

- Allowing the implementation of multi-party SMPC protocols for any adversary model.
- Supporting special types of input and output behavior for protocols (e.g., encrypted public input in combination with partywise private plaintext input).
- Providing for additively homomorphic threshold encryption.
- Supporting the specific purpose design of SMPC protocols.

In order to increase the fields of application for our SMPC library (e.g., as a basis for student theses or for other SMPC related projects), we additionally require:

- Supporting a modular protocol design allowing one to concentrate on the implementation of the computation steps of a protocol rather than on library specific details like the coordination of communication.
- Providing for a transparent transition to other security models (e.g., the malicious model).

While there exist various libraries/frameworks facilitating the implementation of SMPC protocols, e.g., VIFF [34], SEPIA [23], Sharemind [20], Fairplay [84], and SMC-MuSe [90], none of them satisfy all of our requirements. Most libraries/frameworks disqualify because of the sole fact that they do not provide homomorphic threshold encryption or because they are restricted to the two-party case. The SMC-MuSe framework which comes closest to our needs, is, however, strongly aligned to privacy-preserving set operations which cannot be used for the implementation of our privacy-preserving multi-party bartering protocol. Consequently, we decided to implement our own SMPC library primarily tailored to our requirements. By abstracting from the implementation details, our SMPC library can be structured into logically separated layers as illustrated in Figure 8.1. The basis of the library is formed by the multi-party communication infrastructure (see Section 8.1.1), by special bartering related data types (e.g., for the representation of quotes, trade partners, or intervals given by encrypted bounds), and a bartering specific quote generator (see Section 8.1.6). The next layer corresponds to the implementation of the threshold variant of the Paillier cryptosystem from [50], where the Java implementation is taken from the SMC-MuSe framework. The next layer corresponds to a framework for providing performance measurements of SMPC protocols which is detailed in Section 8.1.2. The topmost layer provides fundamental SMPC protocols (secure in the semi-honest model) some of which are presented in Section 8.1.5 and are eventually used to implement our privacy-preserving multi-party bartering protocol built on top of this library. Green colored layers are independent of a concrete SMPC setting,

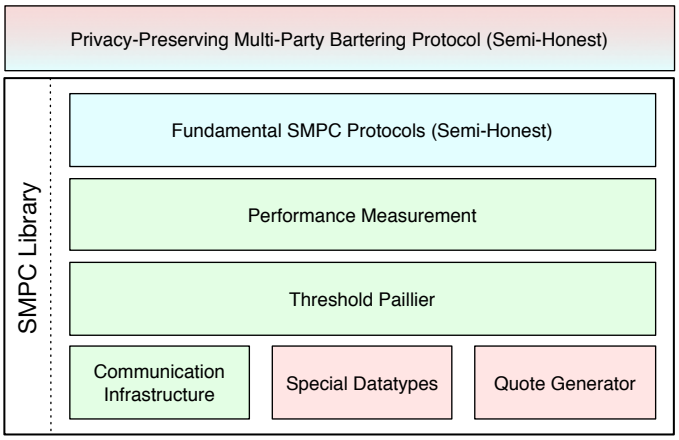


Figure 8.1: High-level overview of the SMPC library.

blue colored layers are semi-honest model specific and contain gates or protocols, and red colored layers depend on our bartering setting (see Figure 8.1).

8.1.1 Multi-Party Communication Infrastructure

In the following, we present a high-level description of the communication infrastructure provided by our SMPC library. The main purpose of the infrastructure is to coordinate the communication of parties that strive to execute an SMPC protocol. The communication infrastructure is built on top of Netty[1] which is an asynchronous event-driven network communication framework designed for developing network applications with Java.

Our library distinguishes between three types of actors each represented by a Java class:

- A *client* corresponds to a party which participates in the execution of a multi-party protocol and is represented by the `Client` class.
- A *protocol* corresponds to the Java implementation of a multi-party protocol and is represented by the `Protocol` class.
- A *server* coordinates the communication between an arbitrary number of clients that jointly execute a multi-party protocol and emulates a protocol's precomputation phase (consisting of, e.g., the generation and distribution of key material). The server is represented by the `Server` class.

[1] `http://netty.io`

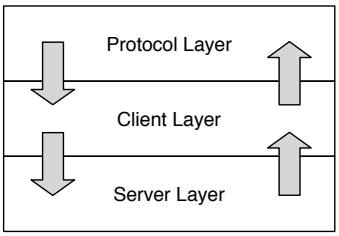


Figure 8.2: Communication layers.

In our communication infrastructure, the server emulates the direct communication between the clients by appropriately forwarding their messages. Furthermore, the server emulates a precomputation phase which is, i.a., responsible for the generation and distribution of cryptographic key material. Each client runs its own instance of the protocol to be executed. The communication between the actors is determined by a hierarchical model consisting of three layers (cf. Figure 8.2): The protocol layer, the client layer, and the server layer. The consideration of the protocol layer allows for a modular processing of subprotocols. Communication can only take place between two adjacent layers. For each actor there exists a separate message type: Client message (`ClientMsg`), protocol message (`ProtocolMsg`), and server message (`ServerMsg`). Abstracting from the details of the implementation, each message type has an ID field and a payload field where the ID specifies the purpose of the message (see below) and the payload contains the actual message to be communicated. Messages of type `ClientMsg` and `ProtocolMsg` additionally provide a source field and a destination field where the former one specifies the sender of the message while the latter specifies the receiver(s) of the message. Furthermore, a message of type `ProtocolMsg` contains a step field which allows the specification of the protocol step this message is intended for (each protocol is divided into logical steps as it is further detailed below).

Our communication infrastructure requires that an actor can only receive a message of its respective type. In case that, e.g., a protocol instance running on client C_1 sends a message to an instance of the same protocol running on client C_2, the message has to traverse all layers of our hierarchical model in both directions: First, the message has to be sent from the protocol instance running on C_1 via C_1 to the server and from the server via C_2 to the protocol instance running on C_2. On each layer, the incoming message has to be wrapped into a message of the appropriate type determined by the next layer.

Since a protocol can rely on subprotocol(s) which in turn can rely on subprotocol(s) and so on, each client has to keep track of which protocol is currently executed. To this end, each party manages its own protocol stack which points to the currently executed (sub)protocol instance and indicates where to return to when the execution of the current

(sub)protocol instance terminates.

The high-level communication between the three actors works as follows. Initially, each client has to connect to the server. It is necessary that each client knows which protocol is to be executed with how many other clients. For the server it is sufficient to know the number of clients participating in a protocol execution.

A1) On receiving a client message from the server, the client checks the ID field specifying the next action:

- `REQUESTID`: The server requests the client to send its identifier (cf. A4).
- `START`: The client locally starts an instance of the protocol which is to be executed.
- `SHUTDOWN`: The client terminates.
- `EXECUTE`: The client extracts the protocol message wrapped in the received client message and forwards it to the protocol instance on top of the protocol stack.

A2) On receiving a protocol message from the client, the protocol instance processes the payload in the designated protocol step.

A3) On receiving a client message from a protocol instance, the client checks the ID field specifying the next action:

- `FORWARD`: It is indicated that the received message is addressed to another protocol instance. The client wraps the received message into a server message with the ID field also set to `FORWARD`.
- `FINISHED`: It is indicated that the protocol instance on top of the protocol stack of the client is finished. The client pops the protocol instance from the stack.
- `SUBPROT`: The execution of a subprotocol is requested. The client creates an instance of the requested subprotocol which is pushed on top of the protocol stack and starts the execution.

A4) On receiving a server message, the server checks the ID field specifying the next action:

- `FORWARD`: The server extracts the client message and forwards it to the intended client(s) specified in the destination field.

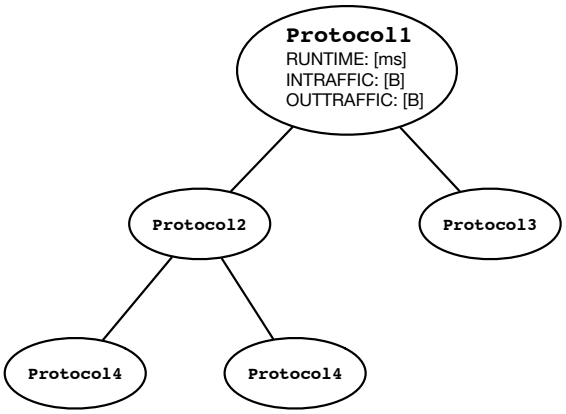


Figure 8.3: Client's protocol tree for `Protocol1`.

- `REGISTER`: The received client message is the response to a preceded client message with the ID flag set to `REQUESTID` sent from the server to the client (see A1). The server registers the client with the transmitted identifier.
- `KEYSETUP`: The client initiates the distribution of a threshold Paillier keypair consisting of the shares of the private key and the public key.

8.1.2 Performance Measurement

In order to evaluate the performance of a multi-party protocol, our SMPC library supports the measurement of the run time and the network traffic of a protocol in a modular way. The run time (measured in milliseconds) and the network traffic (measured in bytes) are determined for each client separately.

Based on its protocol stack, each client manages a protocol tree—represented by the `ProtocolTree` class—which is built during the execution of a protocol. A node in a client's protocol tree represents a protocol instance in its protocol stack. A node representing a subprotocol instance corresponds to a child of the node representing the invoking protocol instance. For each node the run time as well as the network traffic for incoming and outgoing traffic of the associated protocol instance is recorded. The run time of a protocol instance is computed from the time-stamps when it was pushed onto, respectively popped from the stack. The incoming and outgoing traffic is measured for each client's (sub)protocol instance and the results are also stored in the corresponding node of its protocol tree. Note that the incoming and outgoing traffic of a node consists of the incoming and outgoing traffic of all descendant nodes. The protocol tree approach allows for a fine-grained performance evaluation of a protocol and its subprotocols.

Figure 8.3 illustrates a client's protocol tree for `Protocol1` that calls subprotocols

`Protocol2` and `Protocol3` where `Protocol2` calls subprotocol `Protocol4` twice. The performance criteria (i.e., run time and network traffic) are only listed for the `Protocol1` node but are the same for all other nodes.

In order to conduct a performance measurement of a protocol, the protocol has to be embedded in an evaluation context that specifies the parameters used for the performance measurement. This can be accomplished by implementing a `Tester` class for that protocol which extends the abstract `ProtocolTester` class. The `ProtocolTester` stipulates the declaration of general measurement specific parameters like the number of parties jointly executing a protocol, the keysize of the underlying threshold cryptosystem, and the number of repetitions for performance measurements with fixed parameters.

8.1.3 Implementing a New SMPC Protocol

Our SMPC library allows the implementation of new SMPC protocols in a modular way. More precisely, extending the abstract class `SMPCProtocol` allows one to focus on the implementation of the computation steps of an SMPC protocol while the communication infrastructure and the key management remains transparent. Furthermore, the `SMPCProtocol` class already provides access to an implementation of the threshold variant of Paillier from [50].

Extending the abstract class `SMPCProtocol` requires the implementation of the `start()` and the `executeStep()` methods. In the `start()` method, the key distribution is initiated, a variable to record the current position in the protocol execution is initialized, and finally the protocol execution is started. A subprotocol can either initiate an additional key distribution (e.g., in case where a two-party protocol is called from a multi-party protocol) or request the key material from the invoking protocol. The `executeStep()` method has to contain the logic of the SMPC protocol to be implemented which is split into clearly separated computation *steps*. For each step, it is specified which party (represented by a client) has to perform the associated computations. Depending on the SMPC protocol, the computations associated with a step can be performed by a single party, can be performed simultaneously by multiple parties, or can be split between multiple parties. Each step ends with sending one or multiple protocol messages allowing the synchronization of the parties and the initiation of the next step.

8.1.4 Test Environment

In order to conduct the performance evaluations of our privacy-preserving multi-party bartering protocol as well as of the implemented fundamental SMPC protocols used as subprotocols, we set up a cluster of eleven desktop computers (connected via a local area network) equipped with Intel Xeon 5400 series CPUs and 4 GB RAM each running a 64 bit Ubuntu 16.04 LTS. One system represents the server while each of the other systems represents a client, respectively. Depending on the number of parties participating in a protocol execution (we consider protocol execution between two and ten parties), an appropriate number of clients is activated such that each client can run a single party. For the sake of simplicity, in the following, we only speak of parties instead of clients.

For all performance evaluations, the network traffic of a protocol is determined as the accumulated incoming network traffic of all parties. Furthermore, we consider threshold Paillier keys of size 1024 bit and 2048 bit. NIST recommends a key size (i.e., size for an RSA modulus and, thus, for a Paillier modulus) of at least 2048 bit [9] in order to guarantee confidentiality for long time periods. However, a key size of 1024 bit can be considered as sufficient for many bartering scenarios where it is only relevant to keep a commodity private up to the time when the corresponding commodities are traded.

All SMPC protocols that are evaluated in this chapter are data oblivious, i.e., for a fixed parameter setting (including, e.g., the number of parties and the key size) the protocol flow (and thus the run time as well as the network traffic) is independent (up to negligible differences) of a specific protocol input. Thus, in order to conduct the performance evaluations, the protocol input can be chosen arbitrarily as long as it satisfies the constraints resulting from the fixed parameter setting and the definition of the corresponding functionality. However, for a protocol that runs within seconds w.r.t. a fixed parameter setting, the run time may be biased due to the fact that the protocol is not tested in an isolated environment and makes use of network communication. Thus, in case that the run time is below 10 minutes, we perform 10 protocol executions for each test instance and average the run time and the network traffic.[2] Otherwise, the effects on the run time are negligible and only one protocol execution is measured for each test instance.

[2]The small number of repetitions is justified by the low variation of the run time for the evaluated protocols. In general, we determined that the longer the run time of a protocol the lower becomes the variation. For our protocols that run below 10 minutes (for a specific parameter setting) the coefficient of variation ranges between 0.001 and 0.05.

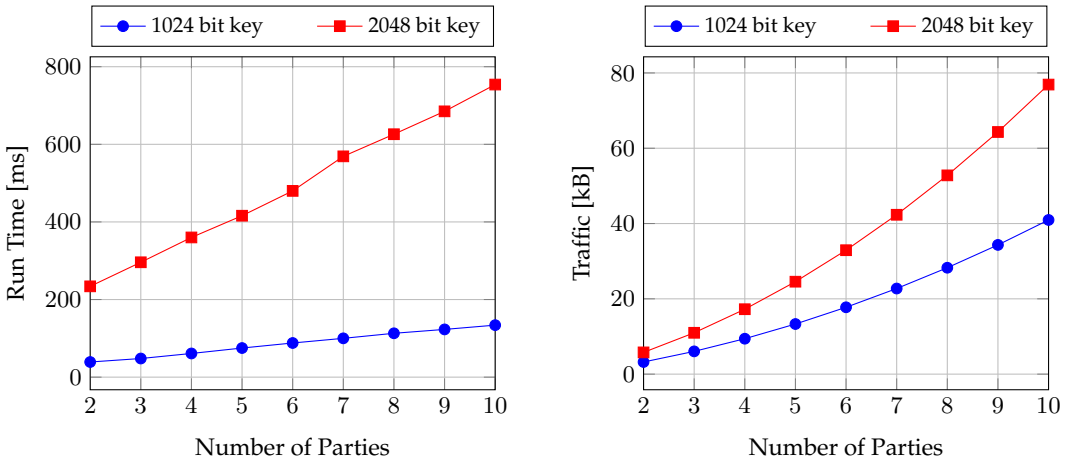


Figure 8.4: Run time (left) and network traffic (right) of `P_MULT_SH_M`.

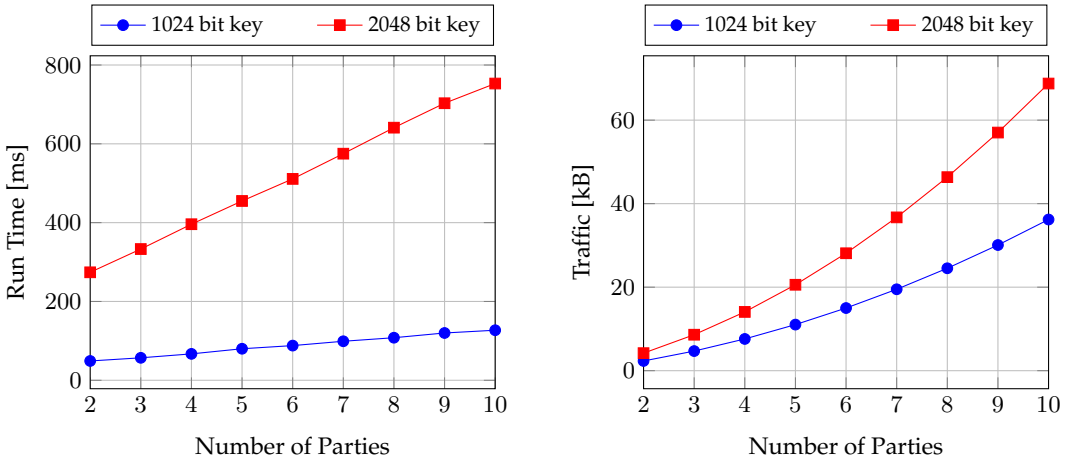


Figure 8.5: Run time (left) and network traffic (right) of `P_LT_SO_SH_M`.

8.1.5 Fundamental SMPC Protocols

In the following, we only present those fundamental SMPC protocols provided by our SMPC library which are used for the implementation of our privacy-preserving multi-party bartering protocol. However, our library provides many more fundamental SMPC protocols (e.g., various two-party privacy-preserving interval operations presented in [118]).

Note that due to the data obliviousness of the considered protocols, we can choose arbitrary plaintext values (satisfying the constraints resulting from the fixed parameter setting and the restrictions placed by the definition of the corresponding functionality) and encrypt these values in order to generate protocol input (for details see Section 8.1.4).

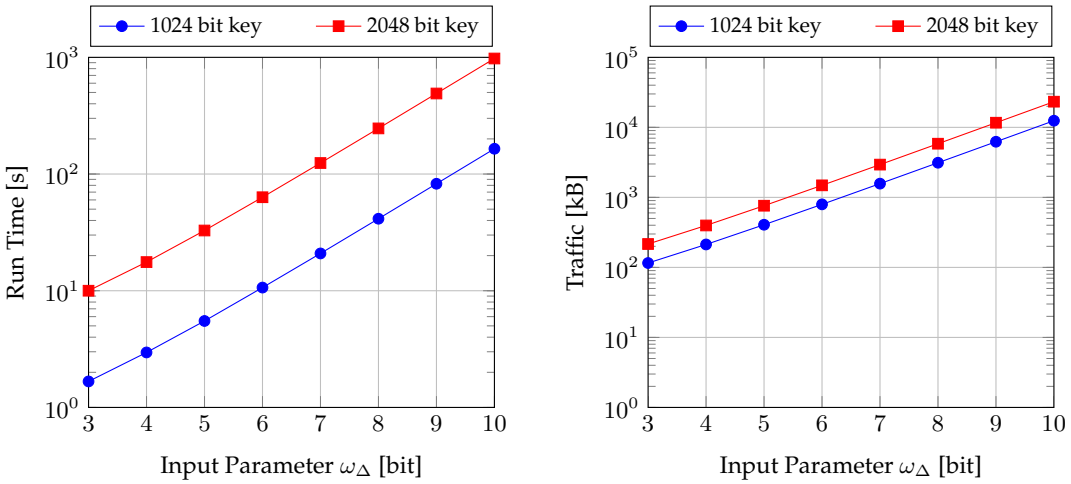


Figure 8.6: Run time (left) and network traffic (right) of `P_RSI_SH_T`.

`P_MULT_SH_M`: Class `P_MULT_SH_M` is implemented according to the specification of $\rho_{\text{Mult}}^{\text{SH-M}}$ derived from [27]. Figure 8.4 shows the performance results for up to ten parties and a 1024 bit (resp., a 2048 bit) threshold Paillier key. For a 1024 bit (resp., 2048 bit) threshold Paillier key, the run time varies between 39 milliseconds (resp., 234 milliseconds) for two parties and 134 milliseconds (resp., 754 milliseconds) for ten parties. Depending on the number of parties, the amount of network traffic varies between 3 kB and 41 kB (resp., between 6 kB and 77 kB) for 1024 bit (resp., 2048 bit) threshold Paillier keys.

`P_LT_SO_SH_M`: The implementation of `P_LT_SO_SH_M` follows the specification of $\rho_{\text{LT-SO}}^{\text{SH-M}}$ (see Gate Specification 5, Section 5.2.1). Figure 8.5 shows the performance results for up to ten parties and a 1024 bit (resp., a 2048 bit) threshold Paillier key. The performance results are similar to the results of `P_MULT_SH_M`. For a 1024 bit (resp., 2048 bit) threshold Paillier key, the run time varies between 49 milliseconds (resp., 274 milliseconds) for two parties and 127 milliseconds (resp., 753 milliseconds) for ten parties. Depending on the number of parties, the amount of network traffic varies between 2 kB and 36 kB (resp., between 4 kB and 69 kB) for 1024 bit (resp., 2048 bit) threshold Paillier keys.

`P_RSI_SH_T`: The implementation of `P_RSI_SH_T` follows the specification of the *two-party* protocol $\rho_{\text{RSI-}\omega}^{\text{SH-T}}$ presented in [118]. Figure 8.6 shows the performance results for an ω_Δ of maximal 2^{10} and a 1024 bit (resp., a 2048 bit) threshold Paillier key. As indicated by

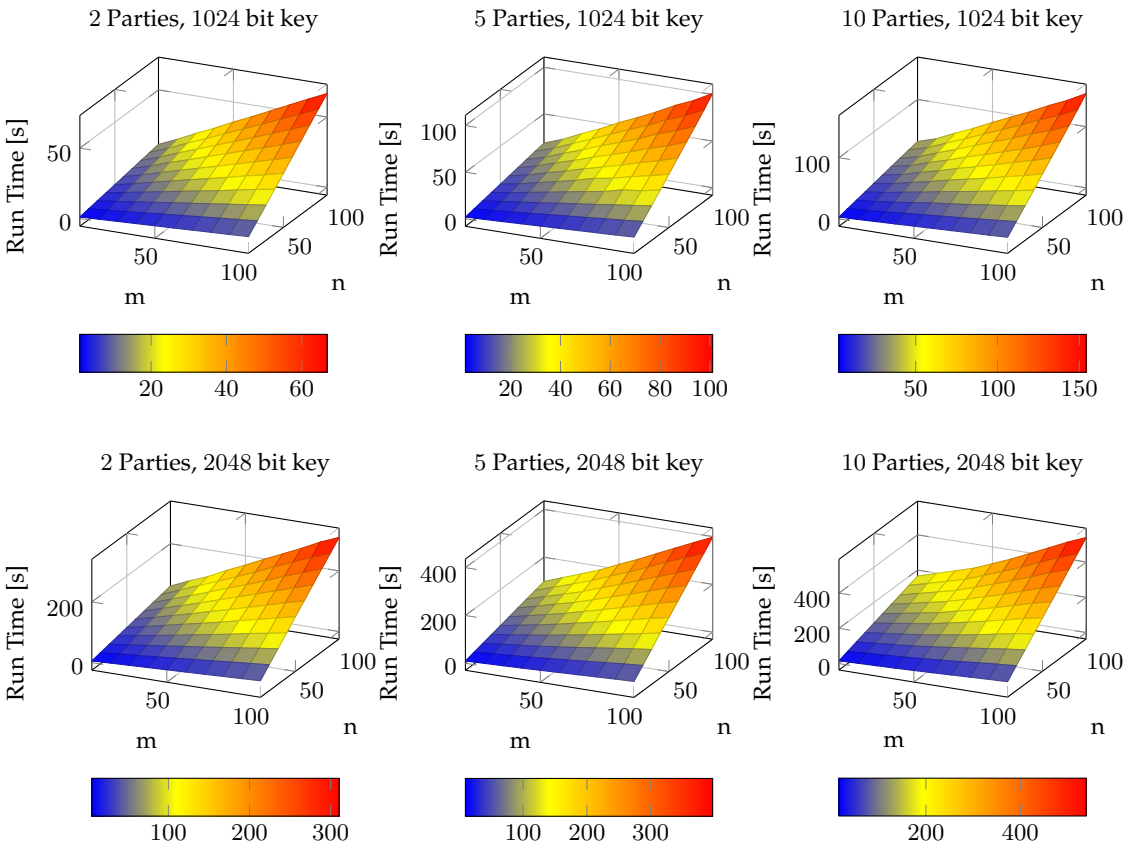


Figure 8.7: Run time of `P_CRS_C_SH_M` for a key size of 1024 bit (above) and 2048 bit (below) for two parties (left), five parties (mid), and ten parties (right) with n, m from 10 to 100.

the theoretical complexities of $\rho^{\text{SH-T}}_{\text{RSI-}\omega}$, the run time and the network traffic substantially depends on the size of ω_Δ, i.e., the maximum supported width of the private interval from which the protocol samples. For a 1024 bit (resp., 2048 bit) threshold Paillier key, the run time of `P_RSI_SH_T` is about 2 seconds (resp., 10 seconds) for $\omega_\Delta := 2^3$, 6 seconds (resp., 33 seconds) for $\omega_\Delta := 2^5$, and 2.5 minutes (resp., 16 minutes) for $\omega_\Delta = 2^{10}$. Depending on ω_Δ, the network traffic varies between 110 kB and 12 MB for 1024 bit threshold Paillier keys and between 210 kB and 25 MB for 2048 bit keys.

`P_CRS_C_SH_M`: Class `P_CRS_C_SH_M` is implemented according to the specification of $\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$ (see Gate Specification 12, Section 5.3.2). Besides the key size and the number of participating parties, there are two additional parameters m and n influencing the run time and the network traffic. Recall that m determines the number of encrypted input vectors and n indicates the number of entries for each vector. We provide the

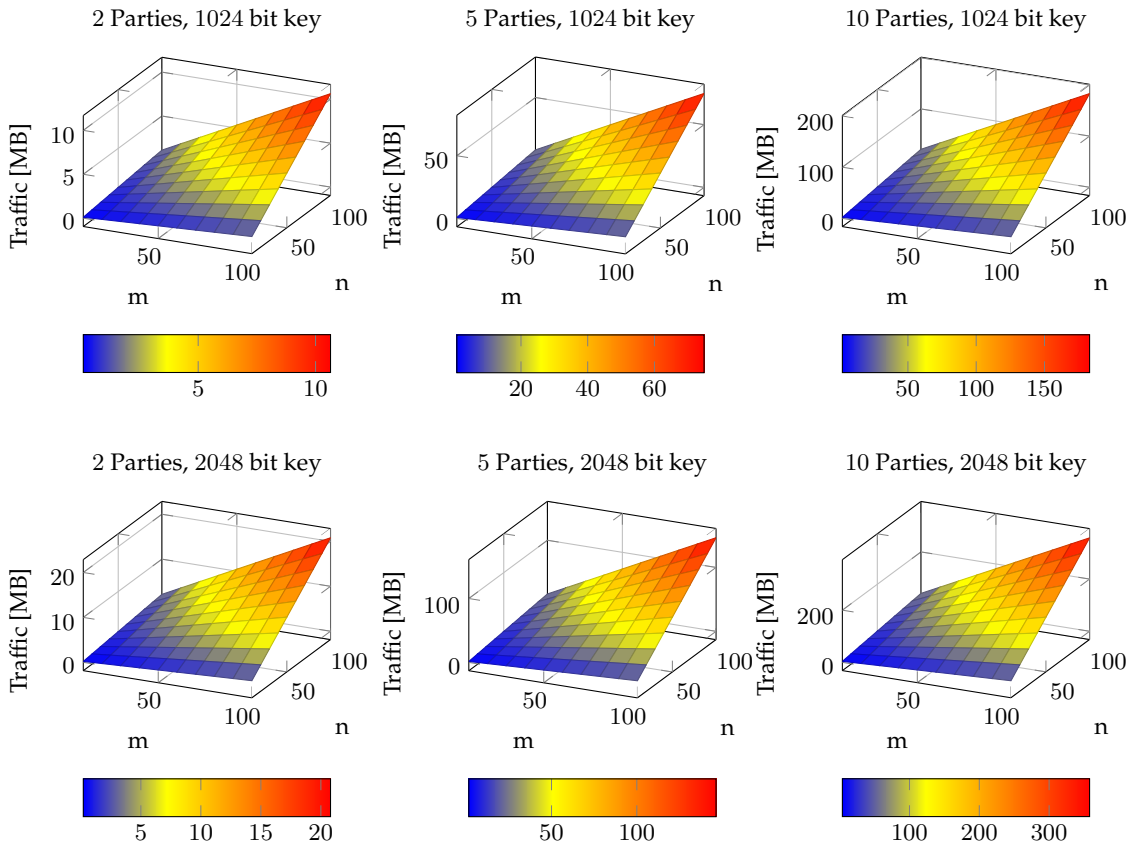


Figure 8.8: Network traffic of `P_CRS_C_SH_M` for a key size of 1024 bit (above) and 2048 bit (below) for two parties (left), five parties (mid), and ten parties (right) with n, m from 10 to 100.

performance results for executions between two, five, and ten parties by increasing m and n in steps of size 10 up to a value of 100, respectively. The measurements are performed for 1024 bit and 2048 bit threshold Paillier keys, respectively. The run time of `P_CRS_C_SH_M` is shown in Figure 8.7 and the network traffic is shown in Figure 8.8. Both Figures indicate that parameter n has a significantly greater impact on the run time and the network traffic than parameter m which is already indicated by the theoretical complexities: While the number of ciphertexts which are to be processed depend linearly on m and n, only parameter n determines the number of executions of gate $\rho_{\text{GT-SO}}$ (cf. Gate Specification 5) used as building block for $\rho^{\text{SH-M}}_{\text{CRS-C-}i^*}$. For a key size of 1024 bit (resp., 2048 bit) and two parties, the run time of `P_CRS_C_SH_M` varies between 1 seconds (resp., 5.7 seconds) with $m = n = 10$ and 67 seconds (resp., 5 minutes) with $m = n = 100$. For five parties the run time varies between 1.8 seconds (resp., 9.4 seconds) with $m = n = 10$ and 1.5 minutes (resp., 6.5 minutes) with $m = n = 100$. For

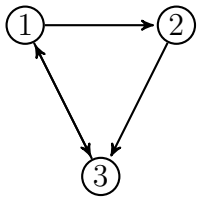


Figure 8.9: Is there any combination of three quotes that induces the depicted compatibility graph?

ten parties it varies between 3.1 seconds (resp., 16 seconds) $m = n = 10$ and 2.5 minutes (resp. 9 minutes) with $m = n = 100$.

Depending on the input parameters and the number of parties, the network traffic of `P_CRS_C_SH_M` varies between 0.1 MB and 183 MB for a key size of 1024 bit and between 0.2 MB and 358 MB for a key size of 2048 bit (see Figure 8.8).

8.1.6 Quote Generator

In order to carry out the performance evaluations of our privacy-preserving bartering protocol, we have to generate quotes (the private protocol input) for a varying number of parties. Furthermore, in order to verify the correctness of the implementation of our privacy-preserving bartering protocol, we want to control the generation of quotes. More precisely, for testing our protocol for special cases we want to generate specific quotes like quotes that induce compatibility graphs with no edges, with no simultaneously executable cycles, or with a specific number of cycles of a specific length. As it turns out, the generation of quotes in such a way that the induced compatibility graph has some specific properties is a challenging task—even for a small number of parties. Consider the following (canonical) example: For three parties, we seek to determine quotes that induce a compatibility graph that consists of two trade cycles that are non-simultaneously executable: one of length 2 and one of length 3 as depicted in Figure 8.9. It turns out that there is no combination of three quotes that induces such a compatibility graph—even if the quantities specified in the quotes are neglected: W.l.o.g. assume that P_1 offers commodity `A` and desires commodity `B`. This implies that P_2 desires commodity `A` and P_3 offers commodity `B`. We further assume w.l.o.g. that P_2 offers commodity `C` and P_3 desires commodity `C`. This setting already induces the required cycle of length 3. However, it is impossible to additionally induce the cycle of length 2 since P_1 offers commodity `A` and P_3 desires commodity `C`.

In order to facilitate the generation of quotes, we introduce a novel tool, referred to as *quote generator* that allows the specification of the most important properties of a

compatibility graph (see below) and to generate a (random) set of quotes that induces a specified compatibility graph. Figure 8.10 depicts the graphical user interface (GUI) of the quote generator.

In order to allow for a fine-grained generation of quotes, the quote generator supports the specification of the following four properties of a compatibility graph (cf. Figure 8.10):

- *Number of components:* The number of components determines the number of disconnected subgraphs in the compatibility graph (see Label 1, Figure 8.10).
- *Number of nodes:* The number of nodes determines how many nodes the compatibility graph shall have (see Label 2, Figure 8.10).
- *Cycles:* The cycles (specified by their lengths) determine the trade cycles in the compatibility graph.
- *Connectivity:* The connectivity value determines the percentage of additional edges which are added to the compatibility graph while still complying with the other three properties. For a connectivity value of 0 (resp., 100) a compatibility graph is generated which has a minimum (resp., maximum) number of edges (for a fixed number of nodes) such that the first three properties are satisfied (see Label 4, Figure 8.10).

In the following, we provide a detailed description of how to operate the quote generator. A new trade cycle can be added to the cycle list (see Label 3, Figure 8.10) by entering the trade cycle length in the textbox underneath (see Label 3.1, Figure 8.10) and pressing the plus-button (left button, Label 3.3, Figure 8.10). By additionally activating the associated checkbox (see Label 3.2, Figure 8.10) it is assured that the added trade cycle is simultaneously executable (indicated with an *executable* mark in the cycle list) w.r.t. all other trade cycles with an *executable* mark. A trade cycle can be deleted from the cycle list by first selecting it followed by pressing the minus-button (right button, Label 3.3, Figure 8.10). The cycle list in Figure 8.10 contains two trade cycles of length three and one trade cycle of length four. The trade cycle of length four and one of the trade cycles of length three are guaranteed to be simultaneously executable.

By pressing the generate button (see Label 5, Figure 8.10) the quote generator tries to generate quotes inducing the specified compatibility graph. In case of success, the compatibility graph is displayed together with the parties' quotes see, e.g., Label 8 and Label 9 in the interactive JPanel (see Label 7, Figure 8.10) which allows one to move around the nodes of the graph. The textbox below the generate button displays log messages concerning the quote generation (e.g., an error message is displayed in case that

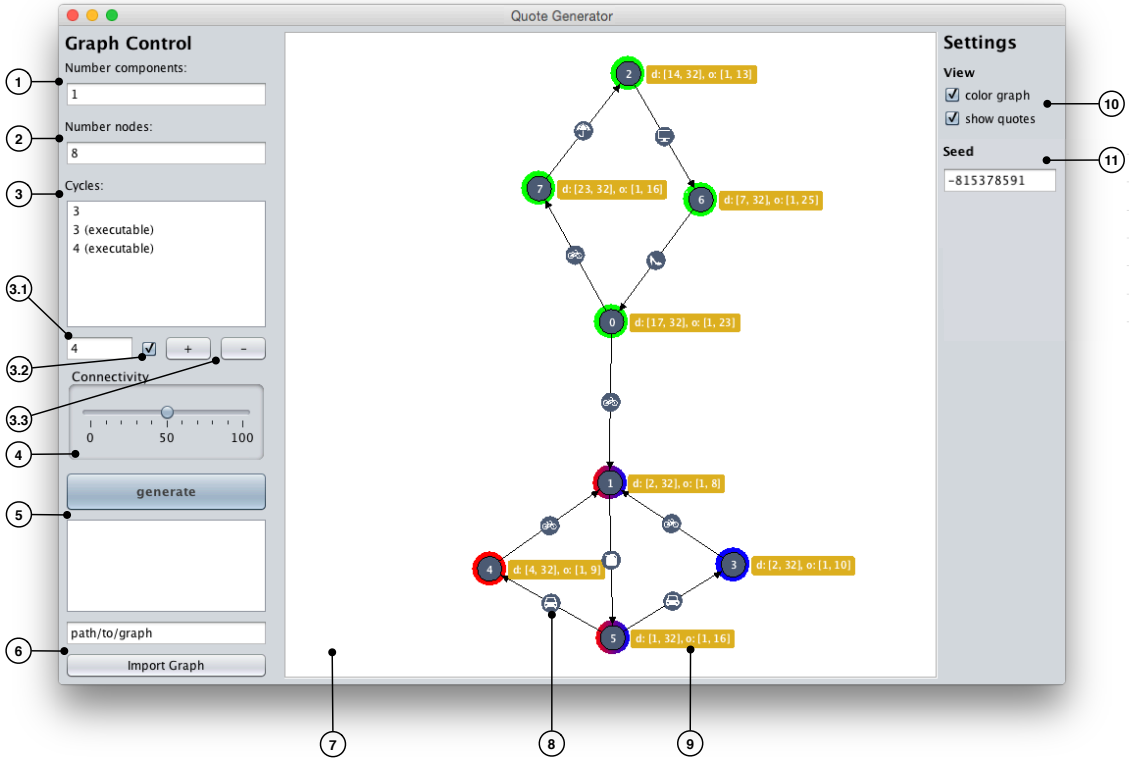


Figure 8.10: GUI of the quote generator.

for the specified parameters the quotes can not be generated). A previously generated set of quotes can be imported (see Label 6, Figure 8.10) in order to display the induced compatibility graph in the interactive JPanel.

On the right-hand side of the quote generator window, it is possible to control further settings influencing the quote generation and the illustration of the compatibility graph. The checkbox labeled "color graph" (see Label 10, Figure 8.10) allows one to color the trade cycles in the compatibility graph with different colors. The labels representing the parties' quotes can be removed from the compatibility graph by deactivating the "show quotes" checkbox (see Label 10, Figure 8.10). The seed (for the initialization of the underlying pseudorandom number generator) which has been used to generate the quotes inducing a compatibility graph satisfying the specified properties is given in the textbox below the "Seed" label (see Label 11, Figure 8.10). Knowing the seed allows the reconstruction of specific previously generated quotes which can be used for debugging purposes.

Based on the specified properties, the compatibility graph as well as the quotes that induce that graph are computed in a brute-force fashion: Simply put, starting with a

graph that only consists of an appropriate number of nodes, the quote generator iteratively adds edges in conjunction with quotes inducing that edges until a compatibility graph is generated that satisfies the specified properties or until at least one of the specified properties is violated. In the latter case, choices for quotes and edges are reversed by using backtracking techniques. If backtracking fails, there exist no combination of quotes that induce a compatibility graph having the specified properties.

In Figure 8.10, a compatibility graph is specified that consists of only one component with 8 nodes and two trade cycles of length 3 and one trade cycle of length 4. One 3-trade cycle and the 4-trade cycle are simultaneously executable. The seed which has been used to generate the quotes inducing the depicted compatibility graph corresponds to -815378591. The three trade cycles are colored green, red, and blue, respectively. Nodes 1 and 5 are two-colored because they are part of the two 3-trade cycles at the same time of which only one can be executed.

8.2 Implementation and Performance Evaluation of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$

In the following, we first present implementation details of our privacy-preserving bartering protocol (Section 8.2.1) followed by a comprehensive performance evaluation (Section 8.2.2).

8.2.1 Implementation

The Java implementation of class `P_Barter_SH_M` for protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ (see Section 7.2.2), follows the protocol specification presented in Section 7.2 and additionally integrates the optimized subgraph check for the evaluation phase (see Section 7.5). The welfare function implemented for the performance evaluation maximizes the number of parties involved in a trade (see Section 7.4). Class `P_Barter_SH_M` does not include the negotiation phase since all executions of $\rho_{\text{RSI-}\omega}$ (see Gate Specification 14) can be performed simultaneously and independently of each other once the actual trade partner constellation has been determined.[3] In order to access the essential building blocks protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ is composed of, the classes `P_MULT_SH_M`, `P_LT_SO_SH_M`, and `P_CRS_C_SH_M` (see Section 8.1.5) have to be imported into class `P_Barter_SH_M`.

The optimized privacy-preserving subgraph check (see Section 7.5) is implemented as follows: For each trade partner constellation graph an array `arr` is created that stores the length of each trade cycle in ascending order. For example, a trade partner constellation graph with two 2-trade cycles and one 3-trade cycle is associated with ar-

[3] Note that the performance of the implementation of $\rho_{\text{RSI-}\omega}$ has already been evaluated in Section 8.1.5.

ray `arr = {2,2,3}`. For a given trade partner constellation graph of order greater than 1 (see Section 7.5), two trade partner constellation graphs of lower order are determined by first splitting array `arr` into two arrays `arr1` and `arr2`, where `arr1` contains the first $\lfloor$`(arr.length + 1)/2`$\rfloor$ elements of `arr` while `arr2` contains the last $\lfloor$`(arr.length)/2`$\rfloor$ elements of `arr`. Then, the first two trade partner constellation graphs in the set of all trade partner constellation graphs are selected that (1) are associated with an array of trade cycle lengths that corresponds to `arr1` (resp., to `arr2`) and that (2) are subgraphs of the considered higher order trade partner constellation graph.

In order to increase the performance of `P_Barter_SH_M`, we exploit the fact that the privacy-preserving subgraph checks for trade partner constellation graphs of the same order are independent of each other (for details see Section 7.5) and that ρ_{Mult} (see Gate Specification 1) remains secure under parallel composition (see [27]). Thus, we assign the instances of `P_MULT_SH_M` used for the subgraph checks for trade partner constellation graphs of the same order to separate threads. The communication infrastructure (see Section 8.1.1) supports the parallel execution of multiple instances of the same protocol by multiplexing and demultiplexing messages. More precisely, protocol messages resulting from multiple instances of the same protocol running in parallel on the same client are multiplexed into one single protocol message. On the receiver side, the client extracts the protocol messages from the received client message, demultiplexes the protocol message, and forwards each single protocol message to the intended protocol instance.

8.2.2 Performance Evaluation

The performance results of `P_Barter_SH_M` are shown in Figure 8.11. We used the quote generator (see Section 8.1.6) to generate valid test input. The measurements comprise the run time and the network traffic of `P_Barter_SH_M` for trade partner constellation sets that contain *all* trade partner constellations (for a fixed size of parties) where (1) the trade cycles are restricted to a length of at most 3 (referred to as $(2,3)$-*cycles*)[4] and (2) where the trade cycle length is not restricted (referred to as *all cycles*). More precisely, in the latter case we consider trade cycles of lengths $2, 3, \ldots, n$ for an execution of `P_Barter_SH_M` with n parties. The measurements for `P_Barter_SH_M` were performed for a key size of 1024 bit and 2048 bit, respectively. For a specified parameter setting (comprising the number of parties, the cardinality of the trade partner constellation set[5], and the key size), we consider `P_Barter_SH_M` as practical if the run time is below 30 minutes.

[4]Note that the consideration of $(2,3)$-cycles is motivated in Section 7.4.

[5]The cardinality of the trade partner constellation set depends on whether $(2,3)$-*cycles* or *all cycles* are considered.

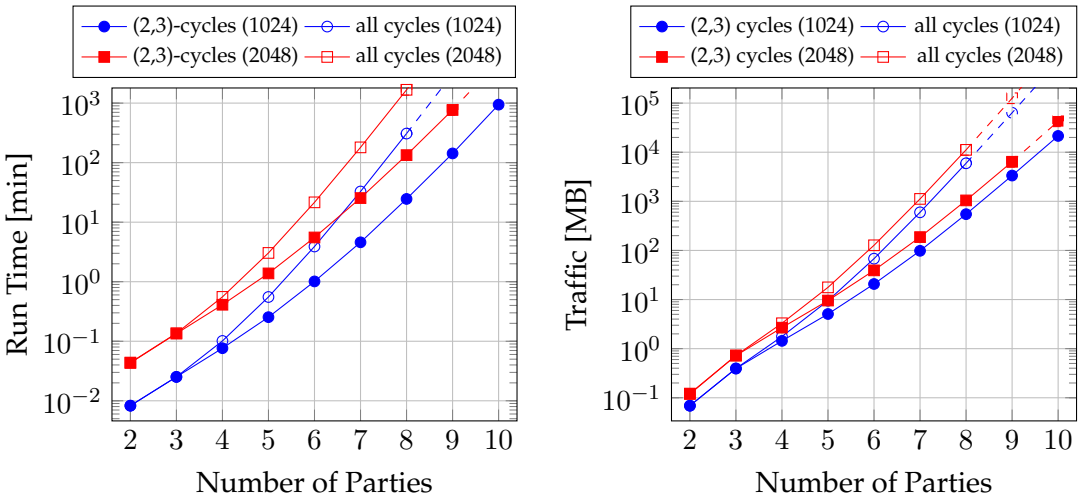


Figure 8.11: Run time (left) and network traffic (right) of `P_Barter_SH_M`.

Table 8.1 and Table 8.2 summarize the measurement results for `P_Barter_SH_M` while Table 8.3 provides the cardinalities of the trade partner constellation sets which were considered (see above). For 1024 bit threshold Paillier keys and for $2, \ldots, 6$ parties, the run time of `P_Barter_SH_M` is below one minute for trade cycles of a maximum length of 3 (no more than 275 trade partner constellations have to be checked) and below four minutes when there is no restriction on the trade cycle length (no more than 719 trade partner constellations have to be checked). For more than 6 parties, the run time increases rapidly in accordance with the cardinalities of the corresponding trade partner constellation sets (cf. Table 8.3). For 9, respectively, 8 and more parties (depending on the trade cycle restrictions) `P_Barter_SH_M` cannot be considered to be practical anymore because the run time is above 2 hours. In the case where the trade cycle length is at most 3, `P_Barter_SH_M` takes more than two hours for 9 parties, while it takes more than five hours for 8 parties when there are no restrictions on the trade cycle length. In the latter case, the run time could not be determined for 9 and 10 parties because our testing machines ran out of memory. For 2048 bit keys, `P_Barter_SH_M` is already impractical for 7 parties when the trade cycle length is not restricted (see Table 8.1).

These results are also reflected by the measured network traffic (see Table 8.2). For 1024 bit threshold Paillier keys and for 9, respectively, 8 and more parties (depending on the trade cycle restrictions), the accumulated network traffic is in the magnitude of gigabytes. When the trade cycle length is at most 3 (resp., not restricted), the network traffic is more than 3 GB for 9 parties (resp., about 6 GB for 8 parties). When using 2048 bit threshold Paillier keys, already more than 1 GB of network traffic arises for 7 parties when the trade cycle length is not restricted.

Parties:	$2, \ldots, 6$	7	8	9	10
1024 bit, $(2,3)$-cycles	$<$ 1 min	$\approx$ 5 min	$\approx$ 25 min	$>$ 2 h	$>$ 15 h
1024 bit, all cycles	$<$ 4 min	$\approx$ 30 min	$>$ 5 h	—	—
2048 bit, $(2,3)$-cycles	$<$ 6 min	$\approx$ 25 min	$>$ 2 h	$>$ 12 h	—
2048 bit, all cycles	$<$ 25 min	$\approx$ 3 h	$>$ 27 h	—	—

Table 8.1: Overview of the run times for `P_Barter_SH_M`.

Parties:	$2, \ldots, 6$	7	8	9	10
1024 bit, $(2,3)$-cycles	$<$ 25 MB	$\approx$ 100 MB	$\approx$ 550 MB	$>$ 3 GB	$>$ 21 GB
1024 bit, all cycles	$<$ 70 MB	$\approx$ 600 MB	$\approx$ 6 GB	—	—
2048 bit, $(2,3)$-cycles	$<$ 40 MB	$\approx$ 200 MB	$\approx$ 1 GB	$>$ 6 GB	—
2048 bit, all cycles	$<$ 130 GB	$>$ 1 GB	$>$ 11 GB	—	—

Table 8.2: Overview of the network traffic for `P_Barter_SH_M`.

Parties	$(2,3)$-cycles	all cycles
2	1	1
3	5	5
4	17	23
5	65	119
6	275	719
7	1211	5039
8	5915	40319
9	31067	362879
10	171575	3628799

Table 8.3: Cardinalities of different complete trade partner constellation sets.

Next, we analyze how the run time of `P_Barter_SH_M` is distributed over the individual phases of $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$. The three most complex phases of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ are the construction phase, the evaluation phase and the selection phase (see Section 7.2). Recap that in the construction phase the encrypted adjacency matrix of the private compatibility graph is computed, in the evaluation phase the subgraph relationship between each trade partner constellation graph and the compatibility graph is checked in a privacy-preserving fashion, and in the selection phase gate $\rho_{\text{CRS-C}}$ (see Gate Specification 12) is

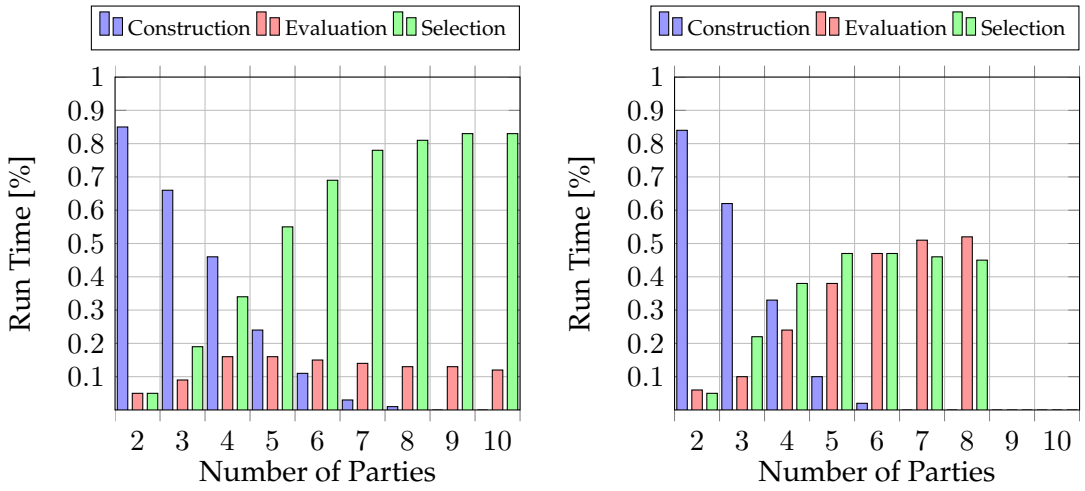


Figure 8.12: Percentage distribution of the run time for the most complex phases of `P_Barter_SH_M` and a key size of 1024 bit. Left: Trade cycles are restricted to a length of at most three. Right: No restrictions on the trade cycle length.

used to determine an actual trade. For `P_Barter_SH_M` and a key size of 1024 bit, Figure 8.12 illustrates the percentage distribution of the run time for these three phases. In case that the trade cycle length is at most 3 (see the left histogram of Figure 8.12) it is remarkable that the percentage of the run time for the evaluation phase is only marginally influenced by the number of parties and remains below 16 %, although the number of trade partner constellation graphs to be checked grows exponentially in the number of parties. This is mainly due to the optimized privacy-preserving subgraph check (see Section 7.5) which is implemented in `P_Barter_SH_M`. In particular, in the case where the trade cycle length is at most 3, the ratio of the number of trade partner constellation graphs with only one cycle and the number of those trade partner constellation graphs that contain more than one trade cycle rapidly decreases when the number of parties increases (cf. Table 8.4). Thus, the more parties participate in the privacy-preserving bartering protocol, the more the evaluation phase benefits from reusing results from privacy-preserving subgraph checks for trade partner constellation graphs of lower order for subgraph checks for trade partner constellation graphs of higher order (for details see Section 7.5). Furthermore, the left histogram of Figure 8.12 indicates that for 2 to 4 parties, the run time of the construction phase dominates the run time of the entire protocol while for 5 and more parties the percentage of the run time for the selection phase increases until the percentage of the run time for the construction phase becomes negligible.

In case that the trade cycle length is not restricted (see the right histogram of Fig-

Parties	$(2,3)$-cycles			all cycles		
	1 cycle	>1 cycles	ratio	1 cycle	>1 cycles	ratio
2	1	0	—	1	0	—
3	5	0	—	5	0	—
4	14	3	4.7	20	3	6.6
5	30	35	0.86	84	35	2.4
6	55	220	0.25	409	310	1.32
7	91	1120	0.081	2365	2674	0.88
8	140	5775	0.024	16064	24255	0.66
9	204	30863	0.0066	125664	237215	0.53
10	285	171290	0.0017	1112073	2516726	0.44

Table 8.4: Number of trade partner constellations containing only one trade cycle and containing more than one trade cycle with and without restrictions on the trade cycle length. The ratio is computed as the number of trade partner constellation graphs with one trade cycle divided by the number of graphs containing more than one trade cycle.

ure 8.12) the evaluation phase dominates the run time of the privacy-preserving bartering protocol for more than 6 parties. This is due to the fact that in this case, the ratio of the number of trade partner constellation graphs with only one trade cycle and the number of those trade partner constellation graphs that contain more than one trade cycle does not decrease that fast compared to the case where the trade cycles are restricted to a length of at most 3 (see Table 8.4). Consequently, the benefit of reusing results of privacy-preserving subgraph checks is less substantial such that with a growing number of parties the run time of the evaluation phase exceeds the run time of the selection phase. While the percentage of the run time for the construction phase dominates the entire run time for 2 and 3 parties, it becomes negligible for more than 5 parties.

For 2048 bit keys, the results are virtually identical and are thus not discussed in detail here.

In summary, the performance evaluation of `P_Barter_SH_M` shows that in practice protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ can only be used for a very limited number of parties. In particular, for a key size of 1024 bit and a trade cycle length of at most 3 (resp., no restrictions of the trade cycle length) protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ can be considered to be practical for at most 7 or 8 parties (resp., 6 or 7 parties) depending on the bartering context. For a key size of 2048 bit, these thresholds are reduced by one party. At first glance, these results seem to greatly limit the applications of protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$. However, we refute this im-

pression in the next chapter by proposing a practical bartering system that—using protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ as a key component—allows an arbitrary number of parties to barter their commodities in a privacy-preserving fashion.

8.3 Summary and Future Work

In this chapter, we introduced a novel SMPC library along with its essential components and presented the performance evaluation of our novel privacy-preserving multi-party bartering protocol (secure in the semi-honest model). Depending on the Paillier key size and the restriction put on the length of the supported trade cycles, we determined that the bartering protocol is practical for at most between 6 and 8 parties.

The performance analysis of the privacy-preserving multi-party bartering protocol providing security in the semi-honest model allows the conclusion that the performance of the protocol variant providing security in the malicious model will be similar due to the related constructions of both protocols. A direct implementation of this variant, however, allows one to make more precise statements about its practicality. The implementation of our privacy-preserving multi-party bartering protocol providing security in the malicious model on top of our framework is left for future work.

CHAPTER 9

Privacy-Preserving Bartering System

The performance evaluation of the implementation of our privacy-preserving multi-party bartering protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ presented in the previous chapter revealed that the protocol is only practical for at most 8 parties (for a 1024 bit threshold Paillier key and a trade cycle length of at most 3). This raises the question whether it is possible to increase the number of parties that can barter their commodities in a privacy-preserving fashion by building on our previous results.

In this chapter, we answer this open question in the affirmative. We introduce the first bartering policy that defines a privacy-preserving bartering model. This model in turn corresponds to an abstraction of an imaginary privacy-preserving bartering system allowing the analysis of the system's properties and behavior which itself is too complex to analyze. The modeled privacy-preserving bartering system allows an arbitrary number of parties (arriving at the system over time) to efficiently barter their commodities in a privacy-preserving fashion. The essential approach is to distribute the parties over simultaneous executions of multiple instances of our privacy-preserving bartering protocol which are managed by the modeled bartering system.

In order to analyze and to optimize our newly proposed bartering model as well as to compare it to the most prominent conventional (i.e., non privacy-preserving) bartering models in the literature, we make use of simulation techniques.

In the remaining chapter, we refer to protocol $\pi^{\text{SH-M}}_{\text{Barter-}(\mathcal{W},\mathcal{D})}$ when we speak of *the/our privacy-preserving bartering protocol*.

Contributions: The following list provides an overview of the contributions of this chapter.

- The introduction of a novel bartering policy defining a privacy-preserving barter-

ing model that in turn represents a bartering system that allows an arbitrary number of parties to barter their commodities in a privacy-preserving fashion. In this model our privacy-preserving bartering protocol is used as a key component.

- A generic framework for the simulation of bartering models which is built on top of an existing framework for discrete event simulation.
- The implementation of our newly proposed privacy-preserving bartering model as well as two prominent conventional bartering models on top of the generic bartering model simulation framework.
- A comprehensive analysis of our novel privacy-preserving bartering model as well as a comparison to existing conventional bartering models.

The work presented in this chapter has been done in collaboration with Benjamin Assadsolimani (a graduate student at RWTH Aachen University and a student assistant at the Research Group IT-Security at RWTH Aachen University at the time of writing of this thesis) who implemented the generic bartering model simulation framework and the considered bartering models. Some results presented in this chapter have been published in his master's thesis [8].

Outline: First, we review two prominent conventional bartering models followed by presenting a novel model representing a privacy-preserving bartering system (Section 9.1). In Section 9.2, we describe our simulation framework used to simulate the considered bartering models and discuss the results.

9.1 Bartering Models

In the following, we describe two models for conventional bartering systems and introduce a novel model that represents a privacy-preserving bartering system. Our modeling approach draws on the literature on conventional bartering models (e.g., [5–7]).

The essential components required for modeling a bartering system are a *bartering pool* and a *bartering policy*. A bartering pool is a means for collecting parties that are willing to trade. The *execution* of a bartering pool refers to the process of determining and selecting trade cycles between the parties which are currently in the pool. A bartering policy defines a bartering model and corresponds to a set of rules that determine under which conditions a bartering pool is to be executed and how the trade cycles are to be selected.

The arrival of parties at a bartering system is modeled by considering discrete periods t_i ($i \in \mathbb{N}$) representing fixed and equal spans of *model time* where at the beginning of each period (i.e., at a discrete point in model time) exactly one *new* party arrives at the modeled system.[1] The model time is fictitious and represents the time whose passing is modeled [93]. In particular, the model time is independent of the *real time* (i.e., wall clock time) the simulation of a model takes (e.g., one minute in model time can be simulated in one second of real time). In the following, we only use the term "model time" explicitly in such cases where both times can be confused. Throughout this chapter we define that one unit of model time models one second.

In the bartering models we consider, a party's quote remains abstract. Instead, we consider a homogeneous and stochastic demand structure defining that every party is willing to trade its offered commodity (at a specific quantity) for any other party's offered commodity (at a specific quantity) with probability p (cf., e.g., [5,6]). More details on this structure and how it is used for simulation purposes are provided in Section 9.2.2.

For the reasons already discussed in Section 7.4, we concentrate on the identification of trade cycles of length at most 3. This also facilitates the comparison of our simulation results to the results presented in the literature on conventional bartering models presented in [5–7] which also consider trade cycles of length at most 3.

In order to evaluate and compare different bartering models, an important measure is the average waiting time of a party which is defined as the difference between its time of departure (corresponding to the time when suitable trade partners were identified for it) and its time of arrival at the modeled bartering system (cf. [6]). Another important measure is the overall number of parties that can trade up to a certain point in time. In case that a model is studied w.r.t. a finite time horizon (involving end-of-period effects, e.g., there still can be trade cycles that, however, are not determined because the end of the simulation is reached) it is necessary to simultaneously consider both measures [6]. In case that an infinite time horizon is considered, a model can be quantified by measuring the average waiting time only. In general, the goal of a bartering policy is to minimize the average waiting time and to maximize the number of parties that can trade (in the following, we refer to these parties as *matched parties*). It is important to note that there is a tight correspondence between both measures: Artificially increasing the waiting time, i.e., collecting a specific number of parties before determining the trade cycles to be executed may also increase the number of parties that can trade and vice-versa.

[1] Note that the length of a period determines the frequency at which the parties arrive at the modeled bartering system.

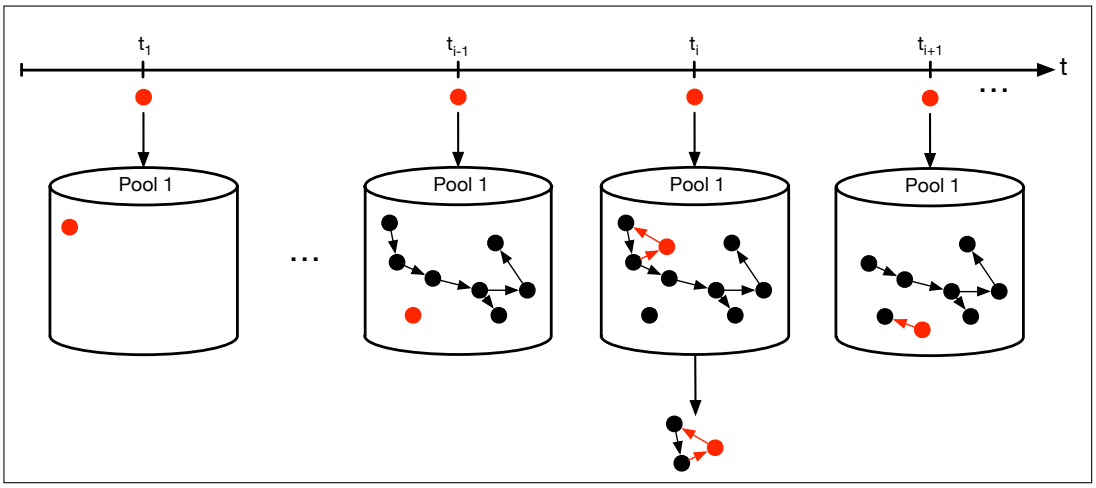


Figure 9.1: Illustration of the conventional greedy model. The period labels are positioned at the beginnings of the periods.

9.1.1 Conventional Greedy Model

The *conventional greedy model* is illustrated in Figure 9.1. It has one single bartering pool of arbitrary size and is defined by the *greedy policy*.

Definition 9.1 (Greedy Policy, based on [6])**.** *The initial condition for the beginning of each time period is that the compatibility graph (induced by the remaining parties in the bartering pool) does not contain any trade cycle of length smaller than or equal to m. When a new party enters the bartering system and joins the bartering pool, it is checked whether or not trade cycles of length at most m (including the new party) can be formed. If this is the case, the longest trade cycle up to length m (corresponding to the actual trade partner constellation) is selected. If there is no unique such cycle, one of them is chosen uniformly at random. All parties involved in the selected trade cycle leave the bartering system; the other parties remain in the bartering pool.*

The greedy policy implies that an actual trade is executed as soon as possible while implicitly assuming that the computation (that is performed by the system) of an actual trade partner constellation runs within one time period. At the beginning of each period a new party (red dot in Figure 9.1) enters the system and joins the bartering pool. In case that the new party forms a new trade cycle of length at most m it is immediately selected (see period t_i in Figure 9.1) and the involved parties leave the modeled bartering system. If the new party forms multiple (necessarily non-disjoint) trade cycles, the longest cycle (up to length m) is determined as the actual trade partner constellation. Furthermore, the policy prescribes that such a cycle has to be chosen uniformly at random in case there is more than one such cycle. For the process of determining the actual trade partner

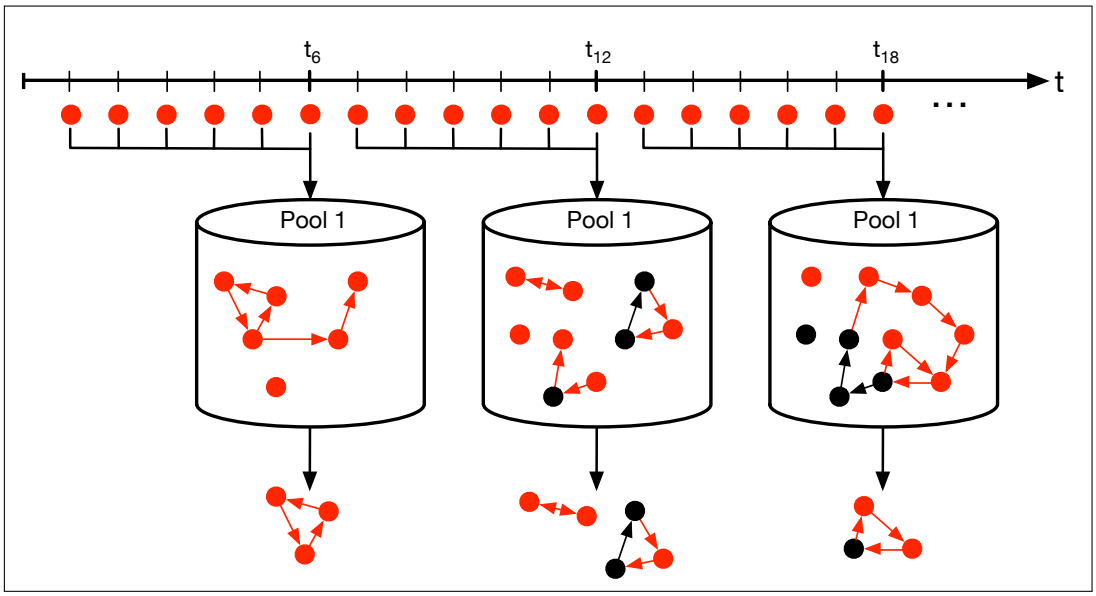


Figure 9.2: Illustration of the conventional batching model. The period labels are positioned at the beginnings of the periods.

constellation, the current compatibility graph is known to the modeled bartering system. In Figure 9.1, this is indicated by the transparent body of the bartering pool.

The main motivation for the conventional greedy model is to reduce the average waiting time of parties.

9.1.2 Conventional Batching Model

The *conventional batching model* also has one single bartering pool and is defined by the *batching policy*. This model is illustrated in Figure 9.2.

Definition 9.2 (Batching Policy, based on [6])**.** *For batch size $k \in \mathbb{N}$, the initial condition for the beginning of a time period $t_{1+n \cdot k}$ ($n \in \mathbb{N}^0$) is that the compatibility graph (induced by the remaining parties in the bartering pool) does not contain any trade cycle of length smaller than or equal to m. The bartering system waits until k* new *parties arrive at the bartering system and enter the bartering pool. Thereupon, a set of disjoint trade cycles of length at most m is identified such that the sum of the trade cycle lengths is maximized. In case that some of the elements in this set are not uniquely determined, the corresponding trade cycles are selected uniformly at random. This resulting set determines the actual trade partner constellation. All parties involved in the selected trade cycle(s) leave the system. The other parties remain in the bartering pool and are considered for the next batch.*

The batching policy[2] implicitly assumes that the computation of an actual trade partner constellation (performed by the system) runs within one time period and prescribes that before the pool is executed, the system has to wait for k new arrivals. This policy allows for a more sustainable selection of trade cycles compared to the greedy policy by benefiting from the foresight: Using the greedy policy it can happen that in a period t_i the newly arrived party forms a 2-trade cycle (which is executed immediately) whereas an additional party could have found a trade partner if the system had waited for the next party arriving at the beginning of time period t_{i+1}. This happens, for example, when the party arriving at the beginning of period t_{i+1} forms a 3-trade cycle involving one of the parties from the 2-trade cycle formed in period t_i. For $k > 1$, the batching policy bypasses shortsighted choices to some extent depending on parameter k.

Analogously to the conventional greedy model, in the conventional batching model the modeled bartering system has complete knowledge of the current compatibility graph.

Figure 9.2 illustrates the conventional batching model for $k = 6$. After collecting 6 parties, in period t_6 the modeled bartering system searches for trade cycles of a maximum length of 3. One 3-trade cycle is identified and the involved parties leave the system while the other three parties remain in the pool. Then, 6 additional parties are collected until the next search starts at the beginning of period t_{12}.

The goal of the batching policy is to increase the overall number of matched parties.

9.1.3 Privacy-Preserving DTP Model

Devising a privacy-preserving bartering system based on our privacy-preserving bartering protocol involves the recognizing of design requirements which do not occur in the context of conventional bartering systems. These requirements result from the fact that a single protocol execution is only practical for at most 8 parties and it is necessary to prevent that a party can identify other parties with which it previously executed an instance of the privacy-preserving bartering protocol (otherwise, a party can adapt its private input based on previous protocol outputs). Note that this privacy requirement is beyond the context of SMPC and has to be addressed by statistical measures.

In the following, we introduce a new policy referred to as DTP policy (Distribution among Threshold Pools) which considers the involved design requirements by making use of multiple so-called *threshold pools*. A threshold pool refers to a bartering pool with a limited capacity of x parties it can accommodate. Once the threshold is reached, the parties in that pool jointly execute an instance of our privacy-preserving bartering protocol. An illustration of the privacy-preserving DTP model is given in Figure 9.3.

[2]Note that for $k = 1$, the greedy policy and the batching policy coincide.

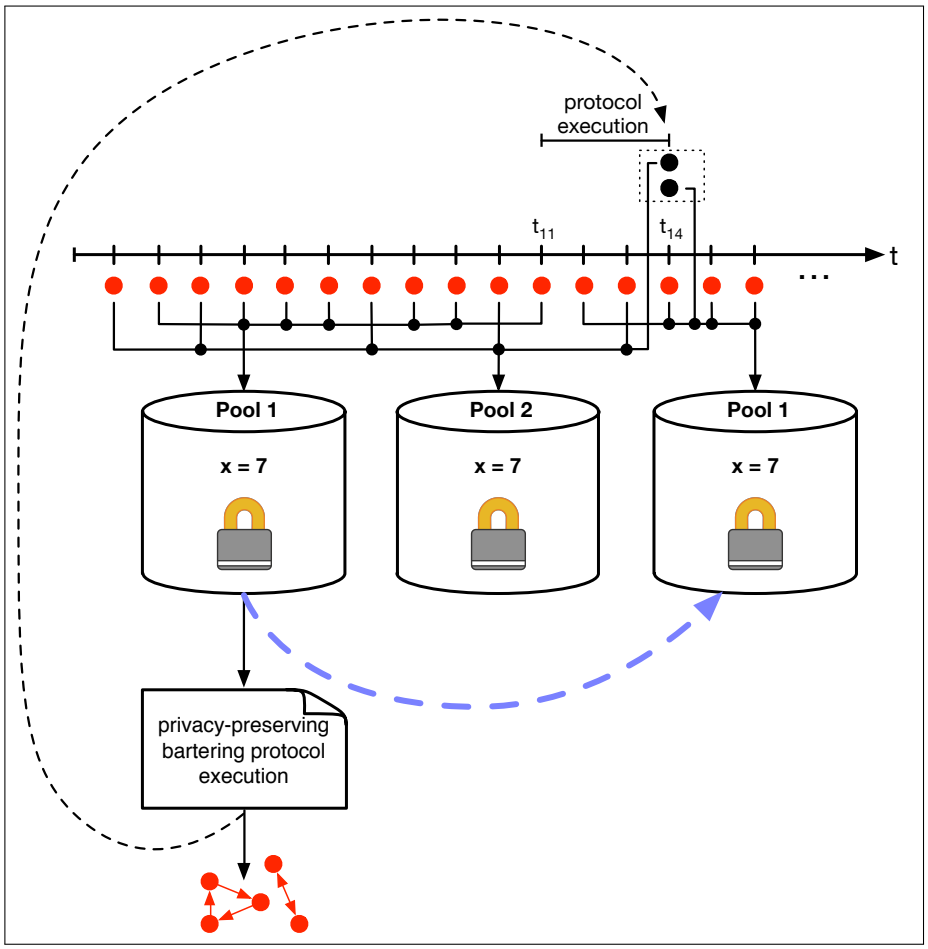


Figure 9.3: Illustration of the privacy-preserving DTP model. The period labels are positioned at the beginnings of the periods.

Definition 9.3 (DTP Policy)**.** *At the beginning of each time period, all threshold pools hold less than x parties. An arriving party is assigned uniformly at random to one threshold pool. As soon as a pool's threshold is reached, the corresponding parties leave the pool and jointly execute an instance of the privacy-preserving multi-party bartering protocol. The empty pool is reused to collect x further parties. All parties that are involved in the selected trade cycle(s) determined by the execution of the privacy-preserving bartering protocol leave the bartering system while the other parties reenter the system and each of them is assigned uniformly at random to one of the available threshold pools.*

Figure 9.3 illustrates the privacy-preserving DTP model consisting of two threshold pools (Pool 1 and Pool 2) with $x = 7$. Arriving parties are distributed among both pools uniformly at random. In period t_{11}, the threshold of Pool 1 is reached, whereupon Pool 1

is emptied and immediately available again. Furthermore, the parties that were collected in Pool 1 jointly execute the privacy-preserving bartering protocol which selects an actual trade consisting of one 3-trade cycle and one 2-trade cycle. Assuming a protocol run time of three periods, the remaining two parties (black dots in Figure 9.3) immediately reenter the bartering system at the beginning of period t_{14}.

In the privacy-preserving DTP model, the compatibility graphs remain private at all time and for any threshold pool. In Figure 9.3, this is indicated by the non-transparent body of the threshold pools marked with a padlock. In contrast to the conventional greedy model and the conventional batching model, in the privacy-preserving DTP model the computation of the actual trade partner constellations is distributed between the parties which bring their own computing power. More precisely, those parties that were collected in one pool jointly execute the privacy-preserving bartering protocol in order to determine the actual trade partner constellation (assuming it exists) in a privacy-preserving fashion. Consequently, arbitrary many protocol executions can run in parallel.

Note that for the described model the corresponding bartering system learns the number of matched parties w.r.t. a specific pool execution. This is due to the fact that the parties that were not matched reenter the bartering system at the beginning of the next period after the protocol execution finished. A first approach to circumventing this issue is to let the unmatched parties draw a waiting time uniformly at random from an appropriate time interval. An unmatched party reenters the system when its waiting time elapsed. This strategy allows to obfuscate the number of matched parties for a specific protocol execution. The details of this strategy are beyond the scope of this thesis and left for future work.

9.2 Simulation of the Bartering Models

In order to draw conclusions about the novel privacy-preserving DTP model and to compare it to the presented conventional bartering models, our goal is to simulate these models for a large number of parties. For the simulation of the considered bartering models we use the DESMO-J simulation framework [37] which we briefly introduce in Section 9.2.1. Building on DESMO-J, we present a generic framework extension that is targeted to simulate a variety of different bartering models sharing some predefined components (Section 9.2.2). Subsequently, we describe the implementation of the conventional greedy model, the conventional batching model, and the privacy-preserving DTP model in that framework extension (Section 9.2.3). Finally, we simulate these models w.r.t. different parameter settings and compare the simulation results based on the

average waiting time and the number of matched parties (Section 9.2.4).

9.2.1 Discrete Event Simulation using DESMO-J

DESMO-J (Discrete Event Simulation Modelling in Java) is a simulation framework for the implementation of discrete event simulation models in Java [58,93]. The framework provides the user with model components (e.g., queues and stochastic distributions), interfaces to specify model specific behavior (e.g., entities and events), and a simulation infrastructure (e.g., an event scheduler and a model time clock).

There are two main concepts for the implementation of the model logic in DESMO-J: The *process oriented view* and the *event oriented view*. The process oriented view requires the implementation of the model logic in terms of processes that interact with each other. A process can be considered as an entity that specifies its own properties and behavior comprising, e.g., the modification of queues, the creation or interruption of other processes, and sleeping for a certain period of time. In contrast, the event oriented view requires to describe the model logic by means of sequentially executed event routines specifying the manipulation of entities. In the following, we focus on the event oriented view because this concept best reflects the logic of a bartering system and some of its predefined components like a party's arrival (an event) that affects the state of a bartering pool (an entity).

An important concept of DESMO-J is to separate modeling aspects from simulation experiments [93]. This approach has the advantage that it allows one to use the same simulation experiment for different models. The `Model` class of DESMO-J allows the specification of the model logic underlying a discrete event simulation. The model specification includes the declaration and description, i.a., of the following components [38,93]:

- *Entities* represent real-world objects by specifying their properties, behavior, and relationships.
- *Events* are descriptions of changes to the state of one or multiple entities at a specific discrete point in time. An event is said to be *scheduled* if it is planned to occur at a specific point in model time. In particular, it is distinguished between *internal events* and *external events*. The scheduling of internal events is caused by state changes of entities, while external events are caused from the outside of the model. The `Model` class allows the specification of events which are initially scheduled.
- *Queues* are data structures for managing entities to be processed in a FIFO fashion (alternatively in a LIFO fashion or by assigning priorities to the enqueued entities).
- *Stochastic distributions* enable the modeling of random behavior.

The `Experiment` class allows the specification of simulation experiments by encapsulating, i.a., the following components:

- An *event list* is a data structure for carrying pending events that have been scheduled.
- The *scheduler* controls the execution of the model. It controls the *event list* from which it selects the next event to be processed.
- The *clock* controls and tracks the model time.
- The *report generator* can be used to generate simulation reports containing, e.g., the state of queues at the end of the simulation or the average waiting time of entities before getting processed.

Furthermore, the `Experiment` class manages simulation specific parameters like the termination time or the termination condition.

Important applications DESMO-J has been used for comprise the modeling of (harbor) logistics and business processes [58]. In this thesis, we extend the scope of application by using DESMO-J for simulating conventional and privacy-preserving bartering models.

9.2.2 Generic Bartering Model Simulation Framework

In the following, we write `Class` in order to refer to an instance a of class `Class` when it is not explicitly stressed that we refer to the class as such. However, when referring to multiple instances of a class `Class`, we explicitly mention that we refer to instances of that class.

Among the most fundamental functionalities that have to be specified for the implementation of a bartering model are the following: (1) controlling the arrival time of parties, (2) assigning parties to pools, (3) deciding when a pool has to be executed, (4) selecting an actual trade constellation between parties in the same pool, and (5) deciding how to handle these parties after an actual trade constellation is determined. In order to represent these generic functionalities in DESMO-J, we implement a new abstract class `Actor` from which we deduce four abstract subclasses `PartyGenerator`, `Distributor`, `PoolExecutor`, and `CalculationManager` (in the following referred to as *actors*). While the `Actor` class captures some DESMO-J specific functionalities for the documentation of a simulation experiment, the four subclasses specify abstract methods targeted to capture functionalities 1-5 which are assumed to be specified in order to implement a concrete bartering model.

- The primary task of a `PartyGenerator` is to model the arrival of parties according to the model specification, e.g., a new party arrives at the bartering system every ten minutes (in model time).

- The purpose of a `Distributor` is to distribute parties to the available bartering pools. For instance, in the conventional greedy model there exists only one pool and the `Distributor` directly assigns a party to this pool.

- A `PoolExecutor` checks whether a pool is ready to be executed. For instance, a threshold pool with threshold x is ready to be executed as soon as x parties are collected.

- As soon as a pool is ready to be executed, a `CalculationManager` initiates an execution of an algorithm or a protocol for determining an actual trade partner constellation between all parties in that pool. Furthermore, a `CalculationManager` is responsible for processing the result of an algorithm or a protocol execution, i.e., for handling matched and unmatched parties.

The concept of actors facilitates the implementation of different bartering models since it allows to provide a generic framework with fixed components such that a modeler only has to extend the abstract classes by implementing the corresponding abstract methods. Besides providing abstract methods, an actor implements a single event allowing it to schedule itself. This feature can be used to implement more complex bartering models, e.g., those comprising special threshold pools that additionally have a minimum threshold value x' and are executed after a specified time period y (in model time) if the minimum threshold value has been reached or exceeded. In this case, a `PoolExecutor` schedules itself after time period y to check whether a pool is ready to be executed.

Our generic bartering model simulation framework provides a `GlobalCompGraph` class for the generation and representation of a *global compatibility graph*. The nodes of a global compatibility graph represent all parties that arrive at the modeled bartering system for a specific simulation experiment while the directed edges between the nodes represent the trade capabilities of the corresponding parties analogously to a compatibility graph (see Definition 7.2, Section 7). A homogeneous and stochastic demand structure (see Section 9.1) is used to specify the edge probability p of the global compatibility graph.[3]

[3]Note that the generation of a global compatibility graph based on a homogeneous and stochastic demand structure is a model simplification. More precisely, such a global compatibility graph may contain combinations of non-simultaneously executable trade cycles that do not occur in practice when assuming that each party offers and desires exactly one commodity (for details see Section 8.1.6). We use this model

Furthermore, we consider six entities that are specified in the following classes: `Party`, `ArrivedPartiesQueue`, `DepartedPartiesQueue`, `Pool`, `PoolQueue`, and `Calculation`.

- A `Party` represents a party that arrived at the modeled bartering system.
- An `ArrivedPartiesQueue` represents a FIFO queue that maintains `Party` instances that are not assigned to a bartering pool yet.
- A `DepartedPartiesQueue` represents a FIFO queue that maintains `Party` instances that departed the modeled bartering system.
- A `Pool` models a single pool of a bartering system.
- A `PoolQueue` is a FIFO queue that maintains `Pool` instances.
- A `Calculation` specifies an algorithm or protocol for the computation of an actual trade constellation for a given set of parties.

Two separate events we consider are provided by the `PartyArrivalEvent` class and the `CalculationFinishedEvent` class.

- The `PartyArrivalEvent` is scheduled for one specific party at the point in model time when this party arrives at the modeled bartering system.
- The `CalculationFinishedEvent` is scheduled at the point in model time when the computation of an actual trade constellation for a given set of parties is finished. This point in model time depends on the algorithm or protocol used to compute the actual trade partner constellation.

Figure 9.4 illustrates the components and their interaction of our generic bartering model simulation framework. Blue components represent actors, green components represent entities, events are colored red, and other components (i.e., the global compatibility graph) are colored orange. An event is illustrated with an extra interface allowing to pass one or multiple objects. A diamond indicates a split or a merge of the program flow. The black dot pointing to the `PartyGenerator` indicates the entry point of the framework.

simplification, as global compatibility graphs that occur in practice cannot be efficiently generated for a large number of parties as it is required for the simulation of bartering models. Since our focus is on determining how many parties are matched on average by the bartering models and on exploring how these models compare to each other w.r.t. the number of matched parties, the model simplification does not bias the conclusions we draw from the simulation results.

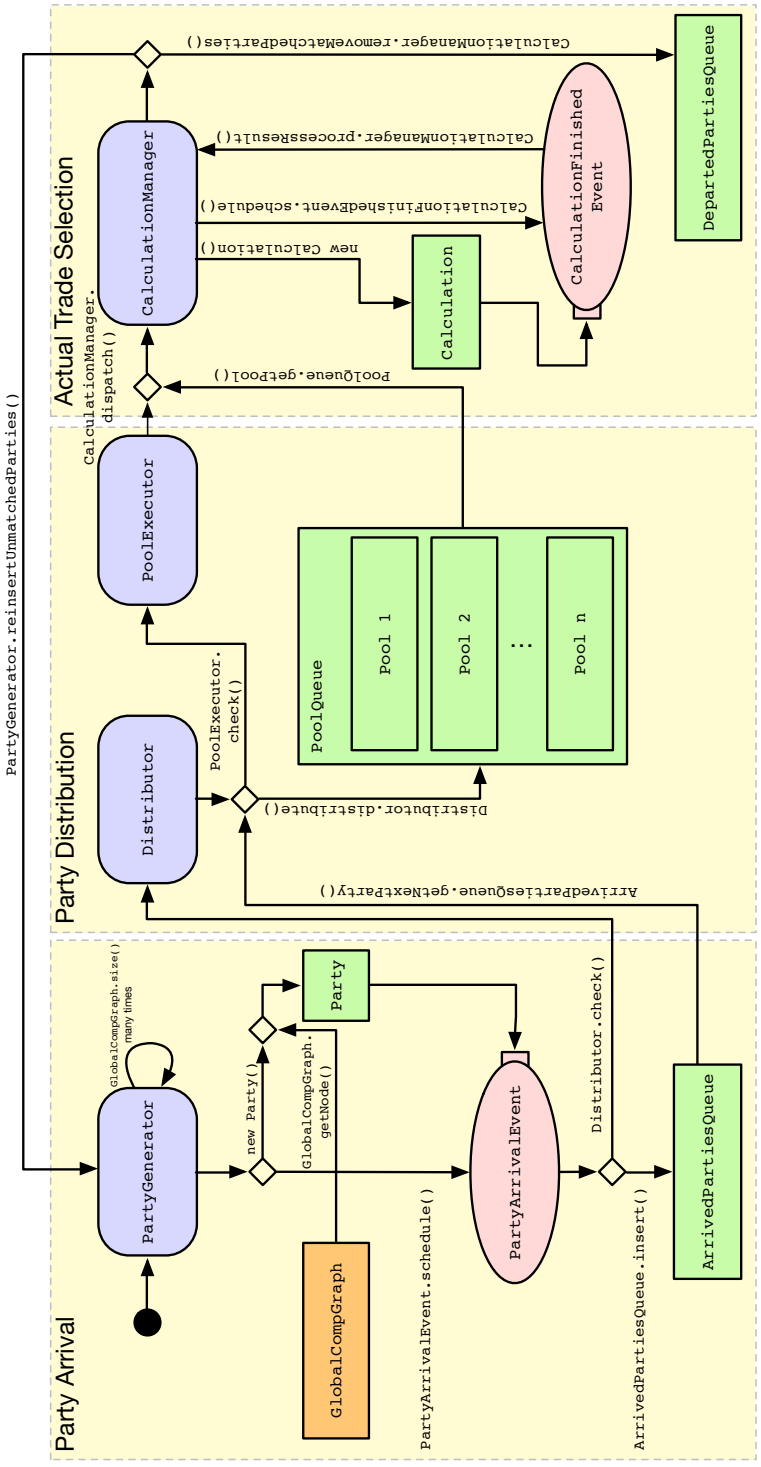


Figure 9.4: Overview of our bartering model simulation framework.

The PartyGenerator expects a global compatibility graph which is generated during the initialization of a simulation experiment. In order to simulate the arrival of the parties, the PartyGenerator creates a new Party for each node in GlobalCompGraph such that the i-th arriving party is associated with the i-th node in the global compatibility graph. For each party, the PartyGenerator schedules a new PartyArrivalEvent by passing the corresponding Party and the model time at which the party is planned to arrive.

The occurrence of a PartyArrivalEvent entails that the Party associated with that event is inserted to the ArrivedPartiesQueue. Furthermore, the Distributor.check() method is called which initiates the Distributor to verify whether or not a specified condition is met before calling the Distributer.distribute() method that takes the first Party from the ArrivedPartiesQueue and distributes it according to a specified distribution strategy to the Pool instances that are maintained by the PoolQueue. Additionally, the Distributor calls the PoolExecutor.check() method that initiates the PoolExecutor to check whether a Pool is ready to be executed. If this is the case, the PoolExecutor calls the CalculationMananger.dispatch() method on the Party instances associated with this Pool. Thereupon, the CalculationManager instantiates a Calculation which represents the computation of the actual trade constellation and schedules a new PartyArrivalEvent which obtains the Calculation object and the model time that the calculation takes. Note that the simulated computation of an actual trade constellation is performed by a conventional algorithm solving a linear program independent of the bartering model (for details see [8]). The CalculationFinishedEvent calls the CalculationManager.processResult() method which initiates the CalculationManager to process the result of the Calculation. The CalculationManager calls the CalculationManager.removeMatchedParties() to insert the Party instances that can trade to the DepartedPartiesQueue and, depending on the bartering model, the other Party instances either stay in their original Pool or are reinserted into the modeled bartering system by calling PartyGenerator.reinsertUnmatchedParties(). It is important to note that a reinserted party represents the same node of the global compatibility graph as before.

9.2.3 Implementation of the Bartering Models

In the following, we present the implementation of the three bartering models presented in Section 9.1 by using our bartering model simulation framework which is built on top of DESMO-J.

9.2.3.1 Implementation of the Conventional Greedy Model

The implementation of the conventional greedy model requests the specification of the following model parameters for a simulation experiment:

- `period` stores the length of a period in model time for a simulation experiment which represents the frequency at which new parties arrive at the modeled bartering system.
- `periodsToSimulate` stores the number of periods to be simulated in a simulation experiment.
- `p` stores the edge probability for the global compatibility graph. The edge probability determines the probability for the existence of an incoming (resp., outgoing) edge between each pair of nodes in the global compatibility graph.

For each of the abstract classes `PartyGenerator`, `Distributor`, `PoolExecutor`, and `CalculationManager` we implemented the subclasses `FixedPartyGenerator`, `GreedyDistributor`, `GreedyPoolExecutor`, and `ConvCalculationManager`, respectively, that capture the properties of the conventional greedy model. During the initialization of a simulation experiment, i.a., the `GlobalCompGraph` is generated for `periodsToSimulate` parties and edge probability `p`. Furthermore a single instance of the `Pool` class is created.

- The `FixedPartyGenerator` is used to schedule the `PartyArrivalEvent` `periodsToSimulate` (= `GlobalCompGraph.size()`) times where the i-th event is scheduled at model time $i \times$ `period`.
- The `GreedyDistributor` directly assigns a `Party` enqueued to the `Arrived-PartiesQueue` to one of the available `Pool` instances uniformly at random. In case of the conventional greedy model, a `Party` is assigned to the only available `Pool`.
- Whenever a `Party` is assigned to a `Pool`, the `GreedyPoolExecutor` calls the `dispatch()` method of the `ConvCalculationManager` on that `Pool`.
- Triggered by the `dispatch()` method, the `ConvCalculationManager` determines an actual trade constellation by using a conventional (nondistributed) algorithm (for details see [8]), measures the duration of the corresponding computation, and schedules the `CalculationFinishedEvent` accordingly. `Party` instances that can trade are inserted into the `DepartedPartiesQueue` while the other `Party` instances remain in their original `Pool`.

9.2.3.2 Implementation of the Conventional Batching Model

The implementation of the conventional batching model is analogous to the implementation of the conventional greedy model. The only differences are that an additional model parameter (i.e., the `batchSize`) has to be specified for a simulation experiment and that the `GreedyPoolExecutor` is replaced by the `BatchingPoolExecutor`. The `BatchingPoolExecutor` class is a subclass of the abstract `PoolExecutor` class. The implementation of the `check()` method of the `BatchingPoolExecutor` class ensures that a number of `batchSize` new `Party` instances have to be assigned to a `Pool` before the `ConvCalculationManager.dispatch()` method is called on the corresponding `Pool`.

9.2.3.3 Implementation of the Privacy-Preserving DTP Model

Compared to the implementation of the conventional greedy model, the implementation of the privacy-preserving DTP Model requests the specification of two further model parameters for a simulation experiment:

- `numPools` stores the number of simultaneously available `Pool` instances.
- `poolSize` stores the maximum number of `Party` instances a `Pool` can accommodate. A `Pool` in combination with a specified `poolSize` yields a threshold pool where the threshold corresponds to `poolSize`.

While the implementation of the privacy-preserving DTP model uses the `GreedyDistributor` just like both conventional bartering models, additional subclasses `FlushingPartyGenerator`, `ThresholdPoolExecutor`, and `PrivCalculationManager` implementing the abstract classes `PartyGenerator`, `PoolExecutor`, and `CalculationManager`, respectively, are required.

- Compared to the `FixedPartyGenerator`, the `FlushingPartyGenerator` additionally supports the simultaneous reinserting of multiple `Party` instances. For each of these `Party` instances, a `PartyArrivalEvent` is scheduled for the same point in model time.
- The `ThresholdPoolExecutor` ensures that a `Pool` has collected a number of exactly `poolSize` `Party` instances before the `PrivCalculationManager.dispatch()` method is called on the corresponding `Pool`.
- The `PrivCalculationManager` determines an actual trade constellation by using the privacy-preserving bartering protocol. From the performance evaluation of

Bartering Model	Special Model Parameter	Joint Model Parameter
conventional greedy	—	numParties, period, p
conventional batching	batchSize	
privacy-preserving DTP	poolSize, numPools	

Table 9.1: Bartering model parameters.

our privacy-preserving bartering protocol (see Section 8.2), we can extract the run time for the protocol w.r.t. the model parameters specified for a simulation experiment. The `CalculationFinishedEvent` is scheduled accordingly to the determined run time. For reasons of fairness w.r.t. the comparison with the conventional bartering models, we add a delay of two minutes (model time) in order to capture DTP specific model behavior like the reentering of parties, the key generation, and the use of anonymization techniques to hide a party's identity from the bartering system and other parties. The delay time is set somewhat arbitrary; deviations from this values do not have significant impacts on the simulation results. `Party` instances representing parties that can trade are inserted into the `DepartedPartiesQueue` while the other `Party` instances are reinserted by calling the `FlushingPartyGenerator.reinsertUnmatchedParties()` method.

9.2.4 Simulation Experiments

For the simulations of the bartering models, we stipulate that one unit of model time models one second. We have to distinguish between model independent parameters (referred to as *joint model parameters*) and model specific parameter (referred to as *special model parameters*). An overview of the model parameters is given in Table 9.1.

The joint model parameters are `numParties`, `period`, and `p`. The number of parties we consider for a simulation (represented by `numParties`) is fixed to 1000 (further increasing the number of considered parties entails disproportionately high simulation times for the DTP model). Furthermore, we consider periods of 600, 1800, and 3600 units in model time corresponding to 10 minutes, 30 minutes, and 1 hour, respectively.[4] For the choice of practically relevant values for parameter `p` (i.e., the edge probability of the

[4]Note that the number of matched parties in the conventional greedy model and the conventional batching model are independent of period lengths. This does not hold for the DTP model: The longer the period, the more time there is for executing instances of our privacy-preserving bartering protocol on varying sets of parties. Thus, to allow for a fair comparison between the considered bartering models, we do not consider periods of more than one hour.

Model Parameter	Values
numParties	1000
period	600/1800/3600
p	0.01/0.02/0.04/0.06/0.08/0.1

Table 9.2: Joint model parameter and their considered values.

global compatibility graph) we draw on the simulations presented in [6]. The authors of [6] consider small values for the edge probability (i.e., $\mathtt{p} = 0.04$, 0.06, 0.08, and 0.1) which are justified by the fact that in various practical contexts, a party only demands a small fraction of the commodities offered by the other parties [6]. We additionally consider even smaller values for the edge probability (i.e., $\mathtt{p} = 0.01$ and 0.02) which may occur in barter markets where a lot of different commodities can be traded. The values of the joint model parameters considered for the simulation are summarized in Table 9.2.

In the following, we perform two series of simulations. In the first series, we simulate the bartering models which have special model parameters (i.e., the conventional batching model and the privacy-preserving DTP model) in isolation in order to determine optimal values for these parameters (w.r.t. the considered parameter ranges) that maximize the number of matched parties and minimize the average waiting time (see Section 9.2.4.1). In a subsequent series of simulations (see Section 9.2.4.2), the special model parameters are fixed to the determined optimal values and the bartering models are simulated for varying joint model parameters according to Table 9.2. For all simulations we consider trade cycles of length at most 3 and for the simulations of the privacy-preserving DTP model we use a 1024 bit Paillier threshold key. Each simulation experiment is repeated ten times and each time a different global compatibility graph is used. To allow for a fair comparison, we use the same global compatibility graphs for the simulation of the different bartering models. The average waiting time and the number of matched parties are averaged over the ten repetitions for each simulation experiment. Due to the fact that our simulation experiments are very time consuming (the performed simulations for the DTP model alone took thirteen days), we chose ten repetitions for each simulation experiment as a trade-off between the accuracy of a simulation result and the number of different parameter combinations we can consider in the scope of this thesis. However, for the simulations where the number of matched parties and the average waiting time are subject to the highest variation (i.e., the simulations of the privacy-preserving DTP model with $\mathtt{p} = 0.01$), we additionally performed 100 repeti-

Model Parameter	Values
`numparties`	1000
`period`	600/1800/3600
`p`	0.01/0.05/0.1
`batchSize`	$10, 20, \ldots, 100, 200, 300, 400$

Table 9.3: Model parameters and their considered values for the conventional batching model.

tions of the simulation experiments in order to draw a conclusion of how representative the measured values are for simulations with only ten repetitions. These additional simulations reveal that ten repetitions for each simulation experiment are absolutely sufficient for the evaluation and the comparison of the bartering models.

The simulations were executed on a Dell OptiPlex 980 with an Intel i7 CPU at 3.2 GHz and 16 GB RAM running Ubuntu 16.04 LTS (64 bit).

9.2.4.1 Special Model Parameter Optimization

Parameter Optimization for the Conventional Batching Model: For the simulation of the conventional batching model, we consider batch sizes of $10, 20, \ldots, 100, 200, 300, 400$. Considering batches of sizes larger than 400 can lead to the following problems: On one hand, in case that there are many trade cycles in the global compatibility graph, we experienced the problem that sometimes we cannot determine an actual trade partner constellation.[5] On the other hand, when considering batches of sizes greater than 200 for 1000 parties, the end of period effect significantly affects the evaluation results (cf. Figure 9.4, left plot). In particular, towards the end of the simulation, many parties may pile up for which it cannot be determined if some of them can barter because the simulation terminates before the next batch is filled. This makes the results hard to compare to other bartering models for which this effect has less of an impact. In order to optimize the `batchSize` parameter it suffices to consider the edge probabilities 0.01, 0.05, and 0.1 in order to get an intuition on the dependency of the number of matched parties (resp., the average waiting time) and the batch size. Table 9.3 summarizes the model parameters of the conventional batching model and the values of the model parameter which were considered for the simulation. For each parameter combination, ten simulations

[5]Note that this can be attributed to the NP-completeness of the underlying decision problem (and our computational limitations).

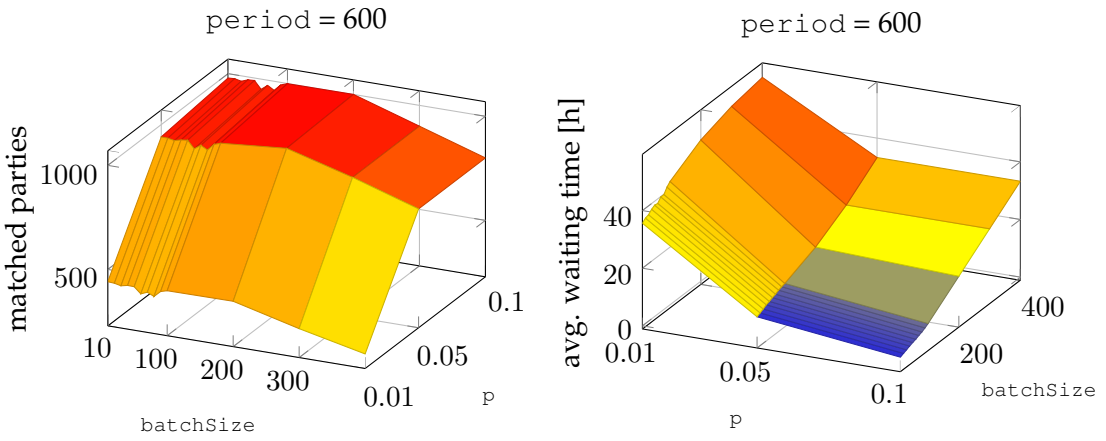


Figure 9.5: Number of matched parties (left) and average waiting time (right) for the conventional batching model for $\texttt{period} = 600$.

were conducted and averaged; for each repetition a different global compatibility graph was used.

The simulation results are presented in Figure 9.5 for $\texttt{period} = 600$. The left plot (resp., right plot) in Figure 9.5 illustrates the dependency of the number of matched parties (resp., the average waiting time) and the special model parameters `batchSize` and $\texttt{p}$. To enhance readability, the x and the z axes are interchanged in the right plot. It is important to note that the period significantly affects the average waiting time but not the number of matched parties. The plots for the average waiting time for different periods look identical, only the axis labels vary. In the following, we do not consider batch sizes greater than 200 since the simulation results are biased due to the end of period effect.

Table 9.4 lists the concrete simulation results for the number of matched parties and the average waiting time for periods of 600, 1800, and 3600 and batch sizes of 10, 100, and 200 w.r.t. different edge probabilities. The variation of the number of parties and the average waiting time w.r.t. the repetitions essentially depends on the edge probability. For $\texttt{p} = 0.01$, the coefficient of variation (i.e., the ratio of the standard deviation to the mean) is below 0.05 for the measured number of matched parties and the average waiting time, respectively. For $\texttt{p} = 0.05$ and $\texttt{p} = 0.1$, the coefficient of variation is below 0.01 for the number of matched parties and below 0.02 for the average waiting time.

Figure 9.5 (left plot) and Table 9.4 (see the entries of the matched parties column for a fixed edge probability) indicate that the batch size only marginally influences the number of matched parties. However, it has a major impact on the average waiting time. For example, for an edge probability of 0.01, a period of 600, and a batch size of 10,

p	batch size	matched parties	period	approx. average waiting time [h]
0.01	10	441	600	35.5
			1800	107
			3600	213.5
	100	445	600	43
			1800	136.5
			3600	256.5
	200	447	600	49
			1800	147
			3600	294
0.05	10	941	600	9.5
			1800	29
			3600	58.5
	100	958	600	14
			1800	42
			3600	84
	200	986	600	18.5
			1800	55.5
			3600	111
0.1	10	976	600	4
			1800	12
			3600	24
	100	999	600	8.5
			1800	25
			3600	50
	200	1000	600	16.5
			1800	50
			3600	99.5

Table 9.4: Excerpt of the simulation results for the conventional batching model.

overall 441 parties are matched which on average have to wait 35.5 hours. By increasing the batch size to 100 only 4 more parties are matched while the average waiting time increases by 7.5 hours. This relation gets even worse for higher edge probabilities. For an edge probability of 0.1, a period of 600 and a batch size of 10, overall 976 parties are matched which on average have to wait approximately 4 hours. For a batch size of 100,

numParties	1000
period	600/1800/3600
p	0.01/0.05/0.1
poolSize	$3, \ldots, 10$
numPools	$3, \ldots, 10$

Table 9.5: Model parameters and their considered values for the privacy-preserving DTP model.

only 23 more parties are matched while the average waiting time is more than doubled. Consequently, for our bartering setting, the consideration of large batch sizes does not bring any advantage over a batch size of 10. In Section 9.2.4.2, it is even argued that the batching policy does not bring any advantage compared to the greedy policy.[6]

Parameter Optimization for the Privacy-Preserving DTP Model: For the simulation of the privacy-preserving DTP model, we consider pool sizes and the number of simultaneously available pools between 3 and 10, respectively. The minimum pool size of 3 is justified by the fact that we support the identification of trade cycles of length 3. The upper bound of 10 for the maximum supported pool size is deduced from the performance evaluation of our privacy-preserving bartering protocol presented in Chapter 8 indicating that the protocol cannot be executed for more than 10 parties on conventional computers. The main motivation behind the consideration of multiple simultaneously available pools are privacy reason (see Section 9.1.3). More precisely, we want to prevent that any party can learn that it is in same pool with any other party it has executed the privacy-preserving bartering protocol before. This has the advantage that a party cannot gain any benefit by modifying its protocol input based on gained knowledge from the *output* of previous protocol executions. As for the conventional batching model, for the optimization of the poolSize parameter and the numPools parameter it suffices to consider edge probabilities 0.01, 0.05, and 0.1 in order to get an intuition on the dependency of the number of matched parties (resp., the average waiting time) and the pools size as well as the number of simultaneously available pools. The values for the model parameters considered for the simulation are summarized in Table 9.5. For each parameter combination, we performed ten simulations and averaged the number of matched parties and the average waiting time. For each repetition, a different global compatibility graph was used.

The simulation results are presented in Figures 9.6-9.8. In particular, Figure 9.6 illus-

[6]Recall that the greedy policy corresponds to the batching policy with a batch size of one.

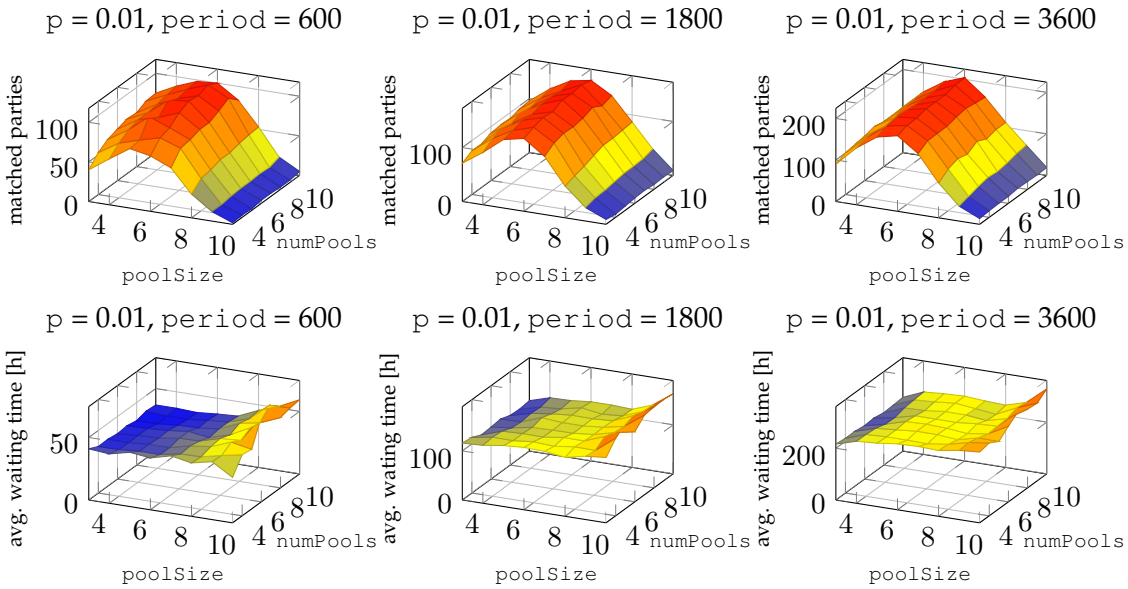


Figure 9.6: Number of matched parties (above) and average waiting time (below) for the privacy-preserving DTP model with $\mathtt{p} = 0.01$ and $\mathtt{period} = 600$ (resp., 1800 and 3600).

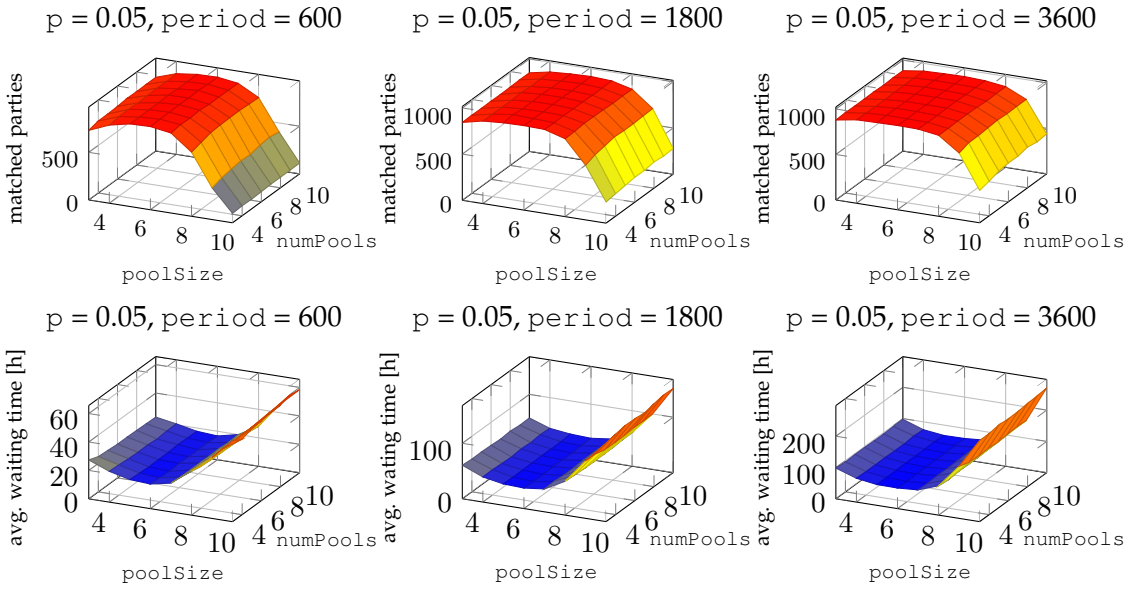


Figure 9.7: Number of matched parties (above) and average waiting time (below) for the privacy-preserving DTP model with $\mathtt{p} = 0.05$ and $\mathtt{period} = 600$ (resp., 1800 and 3600).

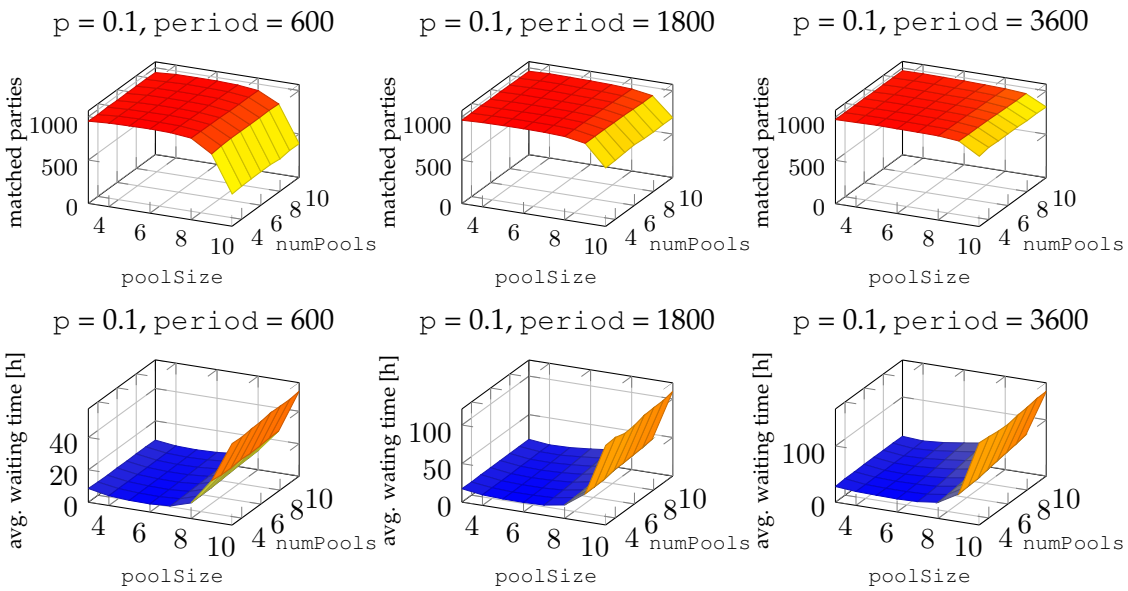


Figure 9.8: Number of matched parties (above) and average waiting time (below) for the privacy-preserving DTP model with $\texttt{p} = 0.1$ and $\texttt{period} = 600$ (resp., 1800 and 3600).

trates the number of matched parties and the average waiting time depending on the `poolSize` parameter and the `numPools` parameter for $\texttt{p} = 0.01$ and for a period of 600, 1800, and 3600, respectively. Analogously, Figure 9.7 and Figure 9.8 illustrate the simulation results for $\texttt{p} = 0.05$ and $\texttt{p} = 0.1$, respectively. At first glance one can see that—independent of the edge probability and the period—the pool size has a high impact on both the number of matched parties and the average waiting time. In contrast, the number of simultaneously available pools has no significant influence on these measures for most of the parameter combinations. Furthermore, it can be derived that the pool size achieving a maximum number of matched parties and the pool size achieving the lowest average waiting time are in fact correlated. Thus, we can consider the number of matched parties and the average waiting time separately.

Since the number of simultaneously available pools (in the range of 3 to 10) has no significant influence on the number of matched parties and on the average waiting time for most of the considered parameter combinations, we have set $\texttt{numPools} := \texttt{poolSize}$. This choice ensures that even in the case that the execution of the privacy-preserving bartering protocol terminates and no party can trade, there are sufficiently many pools such that no two parties are forced to be allocated to the same pool by construction. Otherwise, in case that $\texttt{numPools} < \texttt{poolSize}$, an adversary controlling all but one party in a protocol execution could exploit this design issue in the sense that it

can derive that at least one of the corrupted parties will participate in the same protocol execution as the honest party.

In the following, we identify the pool size that maximizes the number of matched parties w.r.t the considered parameter combinations. For pool sizes from 8 to 10, the number of matched parties strongly decreases which can be attributed to the fact that the run time of the privacy-preserving bartering protocol grows exponentially in the number of parties and thus only a few instances of the protocol can be executed during the duration of the simulation. For each combination of the considered edge probabilities and periods, the three pool sizes achieving the best results w.r.t. the number of matched parties are summarized in Table 9.6. Furthermore, for these combinations, we highlight the maximum average number of matched parties and the minimum average waiting time for each row of the table by using bold font. Evaluating the simulation results presented in Table 9.6, we can conclude that, in general, the best results w.r.t. the number of matched parties (and the average waiting time) are achieved for a pool size of 5 or 6. Note that pool sizes of 5 and 6 achieve better results w.r.t. the number of matched parties than pools of size 3 and 4 which can be attributed to the fact that the run time differences of the privacy-preserving bartering protocol for up to 6 parties are in the order of milliseconds and in the long run there is a greater benefit from covering slightly more parties in a protocol execution than performing slightly more protocol executions but with less parties. In the following, we set $\texttt{poolSize} := 6$ because the simulations for a pool size of 6 yield the best results slightly more often than the simulations for a pool size of 5 (when counting the number of bold entries in Table 9.6 for both pool sizes).

For $\texttt{p} = 0.01$, the coefficient of variation is on average 0.12 for the number of matched parties and 0.1 for the average waiting time. Considering higher edge probabilities, the variation is much lower. For $\texttt{p} = 0.05$ and 0.1, the coefficient of variation is below 0.01 for the number of matched parties and below 0.04 for the average waiting time. Due to the fact that for $\texttt{p} = 0.01$ the number of matched parties and the average waiting time are subject to a high variation, we additionally performed 100 repetitions of the simulation experiments for $\texttt{p} = 0.01$, $\texttt{poolSize} = 6$, and for the different periods (see Table 9.6) in order to draw a conclusion of how representative the measured values are for simulations with only ten repetitions. Since the deviations are minor, we can conclude that ten repetitions are sufficient for the evaluation and the comparison of the considered bartering models.

p	period	best pool sizes	matched parties	approx. average waiting time [h]
0.01	600	5	**98**	38.5
		6	**98** (95)	40.5 (39)
		7	81	**38**
	1800	5	137	120
		6	**146** (146)	**119.5** (121)
		7	127	120.5
	3600	5	186	242.5
		6	**197** (194)	250 (247.5)
		7	168	**242**
0.05	600	5	870	**17**
		6	**884**	**17**
		7	849	19.5
	1800	5	910	38
		6	**916**	**37.5**
		7	905	42
	3600	5	**927**	**66.5**
		6	924	**65.5**
		7	925	71
0.1	600	4	963	6
		5	**965**	**5.5**
		6	**965**	**5.5**
	1800	4	968	14
		5	**972**	**13**
		6	971	13.5
	3600	4	**975**	25
		5	**975**	**24**
		6	973	26

Table 9.6: Excerpt of the simulation results of the privacy-preserving DTP model. The values in brackets result from simulations with 100 repetitions (instead of 10 repetitions).

Bartering Model	Special Model Parameter	Joint Model Parameter
conventional greedy	—	numparties $= 1000$ period $= 600/1800/3600$
privacy-preserving DTP	poolSize $= 6$	p=0.01/0.02/0.04/0.06/0.08/0.1
	numPools $= 6$	

Table 9.7: Model parameters and their considered values for the comparison between the conventional greedy model and the privacy-preserving DTP model.

9.2.4.2 Comparison of the Bartering Models

As already argued in Section 9.2.4.1, considering large batch sizes (in the order of hundreds) does not bring any advantage over a batch size of 10. Comparing the simulation results for the batching model and the greedy model shows that for the considered bartering setting, the batching policy does not bring any advantage over the greedy policy. For example, for p $= 0.01$ and period $= 600$ for the batching model (with batchSize = 10) only 3 more parties are matched than for the greedy model, however, the average waiting time is increased by half an hour (cf. Table 9.4 and Table 9.8). This finding is in accordance with the results of [5,6] which show that the greedy policy is approximately optimal compared to other policies like the batching policy when considering an infinite time horizon. Thus, we only compare the conventional greedy model and privacy-preserving DTP model in order to draw conclusions on the *price of privacy* of the DTP model. In particular, we compare the number of matched parties as well as the average waiting time between both models for different joint model parameter values (see Table 9.7). The special model parameters (poolSize and numPools) of the privacy-preserving DTP model are both set to 6 as justified in Section 9.2.4.1.

As for the first series of simulations, we performed ten simulations for each parameter combination and averaged the simulation results. For each repetition, a different compatibility graph was used while each model and each parameter combination is simulated with the same global compatibility graphs in order to allow for a fair comparison.

As for the conventional batching model, for the conventional greedy model the number of matched parties is independent of the period. Furthermore, the variation of the number of parties and the average waiting time are similar in both models. The simulation results are presented in Figure 9.9 and Figure 9.10. Figure 9.9 illustrates the dependencies of the number of matched parties for the conventional greedy model as well as for the privacy-preserving DTP model and the considered edge probabilities and periods.

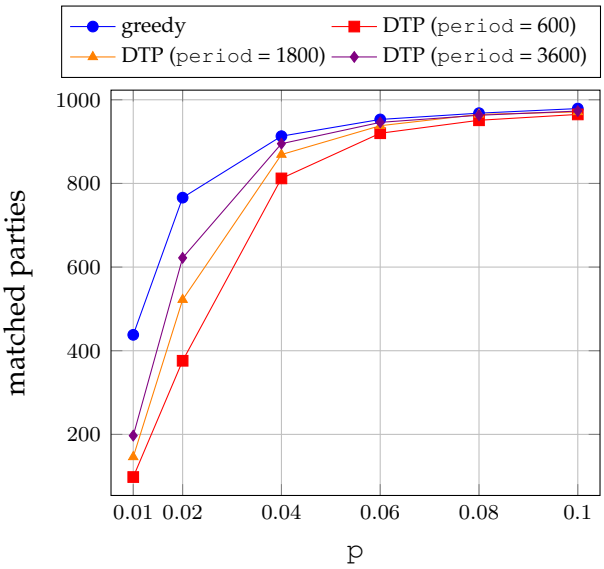


Figure 9.9: Comparison between the conventional greedy model and the privacy-preserving DTP model w.r.t. the number of matched parties for $\texttt{p} = 0.01/0.02/0.04/0.06/0.08/0.1$ and $\texttt{period} = 600/1800/3600$.

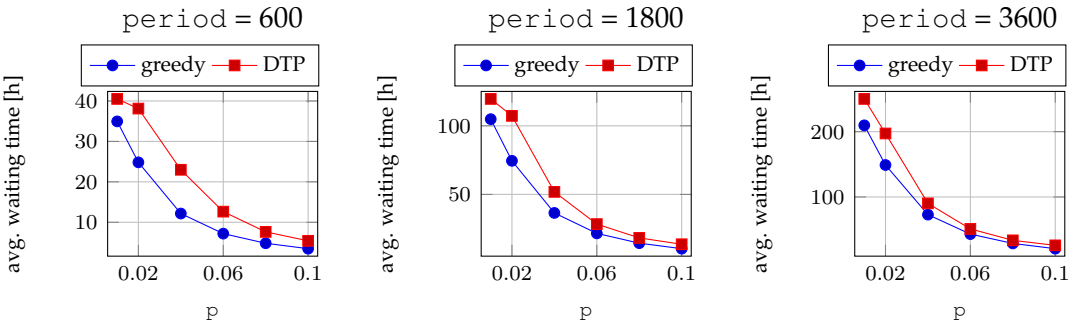


Figure 9.10: Comparison between the conventional greedy model and the privacy-preserving DTP model w.r.t. the average waiting time for $\texttt{p} = 0.01/0.02/0.04/0.06/0.08/0.1$ and $\texttt{period} = 600/1800/3600$.

By comparing the curves for both models, we can make two interesting observations:

1. The longer the period, the better the number of matched parties in the privacy-preserving DTP model approximates the number of matched parties in the conventional greedy model (compare the blue curve for the conventional greedy model and the purple curve for the privacy-preserving DTP model with $\texttt{period} = 3600$). For example, the conventional greedy model matches 766 parties for $\texttt{p} = 0.02$, 953 parties for $\texttt{p} = 0.06$, and 979 parties for $\texttt{p} = 0.1$ while for $\texttt{period} = 3600$

p	period	conventional greedy model		privacy-preserving DTP model	
		matched parties	avg. waiting time	matched parties	avg. waiting time
0.02	600	766	25	376	38
	1800		74.5	522	107.5
	3600		149	622	197
0.06	600	953	7	920	12.5
	1800		21.5	938	28.5
	3600		43	946	51
0.1	600	979	3.5	965	5.5
	1800		10.5	971	13.5
	3600		21	973	26

Table 9.8: Excerpt of the simulation result for the comparison between the conventional greedy model and the privacy-preserving DTP model.

the privacy-preserving DTP model matches 0.5 to 19 percent less parties depending on the edge probability (cf. Table 9.8). Although not explicitly addressed in this thesis, it is obvious that when further increasing the period, the number of matched parties in both models continue to converge. This can be attributed to the fact that the longer the period, the more time there is for executing instances of our privacy-preserving bartering protocol on varying sets of parties.

2. The higher the edge probability the better the number of matched parties in the privacy-preserving DTP model approximates the number of matched parties in the conventional greedy model (see also Table 9.8). This relationship can be attributed to the fact that the higher the edge probability the more trade cycles in the global compatibility graph a party will be involved in. This in turn increases the probability that such a trade cycle is identified in the privacy-preserving DTP model as part of an actual trade partner constellation. For example, the conventional greedy model matches 979 parties for $\mathtt{p} = 0.1$ while the privacy-preserving DTP model matches only 0.5 to 1.1 percent less parties depending on the period for the same edge probability (cf. Table 9.8).

Similar relations also hold for the average waiting time in both models (see the plots in Figure 9.10). For $\mathtt{period} = 3600$, the conventional greedy model achieves an average waiting time of 149 hours for $\mathtt{p} = 0.02$, 43 hours for $\mathtt{p} = 0.06$, and 21 hours for $\mathtt{p} = 0.1$ while the privacy-preserving DTP achieves an average waiting time of 197 hours for $\mathtt{p} = 0.02$, 51 hours for $\mathtt{p} = 0.06$, and 26 hours for $\mathtt{p} = 0.1$ (cf. Table 9.8).

The comparison of the simulation results between the conventional greedy model and the privacy-preserving DTP model yields the following central result:

> Privacy-preserving bartering under the DTP policy is practical.

It furthermore achieves comparably good results w.r.t. the number of matched parties and the average waiting time as it is the case for conventional bartering under the greedy policy that in turn achieves approximately optimal results. The DTP policy is particularly well-suited for bartering settings with periods greater than or equal to 1800 (i.e., 30 minutes) and/or edge probabilities greater than or equal to 0.04 (cf. Figure 9.9 and Figure 9.10). The major difference between the conventional greedy model and the privacy-preserving DTP Model—besides the fact that the latter model is privacy-preserving—is that in the privacy-preserving DTP model there is a continuous, distributed, and highly parallel computation of actual trade partner constellations (i.e., the privacy-preserving bartering protocol is executed between small groups of parties which bring their own computing power) while in the conventional greedy model only one central computation of an actual trade constellation is executed for each new party entering the modeled bartering system.

9.3 Summary and Future Work

In this chapter, we presented a novel bartering policy referred to as the DTP policy. The resulting privacy-preserving bartering model as well as two prominent existing conventional bartering models were implemented on top of our generic bartering model simulation framework. The evaluation of the simulation results of the three considered bartering models showed that bartering under the newly proposed DTP policy is practical and achieves comparable good results w.r.t. the number of matched parties and the average waiting time as it is the case for conventional bartering under the greedy policy. Furthermore, we determined that the novel bartering policy is particularly suited for bartering settings with periods greater than or equal to 30 minutes and/or edge probabilities greater than or equal to 0.04. While these results were determined based on our privacy-preserving multi-party bartering protocol that is secure in the semi-honest model, we expect similar results for the malicious model variant.

A direction for future research is to devise and study further privacy-preserving bartering policies that comprise advanced mechanisms like timers that, for example, can be used to propose bartering policies making use of threshold pools that are executed after the timer expires even if the threshold has not been reached. The goal of these privacy-

preserving bartering policies would be to further increase the number of matched parties and to reduce the average waiting time, especially for bartering settings with short periods and small edge probabilities. One further interesting research direction is to formally study obfuscation mechanisms for parties that reenter the bartering system as it was briefly discussed in Section 9.1.3.

CHAPTER 10

Conclusion

The main goal of this thesis was to design privacy-preserving multi-party bartering protocols based on which it is possible to design a practical bartering system that allows an arbitrary number of parties (arriving at the system over time) to barter their commodities in a privacy-preserving fashion.

We achieved this goal by first approaching the problem from a theoretical point of view. After providing an overview of some preselected state-of-the-art privacy-enhancing technologies and the essential functionalities of conventional bartering systems, we elaborated on the functionalities and requirements of our envisioned privacy-preserving bartering system and identified suitable cryptographic techniques for its implementation. As a preliminary study, we first considered the two-party bartering setting and designed a protocol that allows two parties to privately barter their commodities in the presence of a malicious adversary that can corrupt one party. Next, we designed privacy-preserving multi-party bartering protocols for both the semi-honest and the malicious model. Compared to the two-party bartering setting, the multi-party bartering setting is much more sophisticated and requires fundamentally new design approaches. Our bartering protocols make use of several novel privacy-preserving building blocks which are of general interest beyond the context of (privacy-preserving) bartering and are contributions to this thesis in their own.

In order to get an idea of between how many parties our privacy-preserving multi-party bartering protocols can be executed in practice, we representatively implemented the multi-party protocol providing security in the semi-honest model. Based on the performance evaluation of this protocol, we devised a novel bartering policy describing a privacy-preserving bartering model that in turn underlies a privacy-preserving bartering system allowing an arbitrary number of parties to privately barter their commodities. In order to determine *the price of privacy* for our novel bartering policy, we compared the

resulting privacy-preserving bartering model to the two most prominent conventional (non privacy-preserving) bartering models in the literature. In order to allow for a fair comparison, we implemented the three models on top of a simulation framework, simulated the models for a variety of different values of the model parameters, and compared the simulation results w.r.t. the number of parties that get to trade until the end of a simulation run (i.e., the number of matched parties) and the average waiting time of these parties.

The contributions of this thesis represent advances in the research areas of secure multi-party computation, privacy in the context of e-commerce, and modeling and simulating bartering systems.

We contribute to the area of secure multi-party computation in that we propose and design novel privacy-preserving selection operations that can be used as secure building blocks in more complex SMPC protocols. Furthermore, we propose novel techniques for designing special purpose SMPC protocols (e.g., an efficient encoding and decoding technique that is based on the uniqueness of prime factorization).

By proposing a tight bartering specific privacy setting along with a new bartering specific terminology and by designing privacy-preserving bartering protocols for the semi-honest as well as the malicious adversary model that provide prioritization and negotiation mechanisms, we solve a series of open problems in the context of e-commerce.

We contribute to the area of modeling and simulating dynamic bartering systems by proposing the first privacy-preserving bartering policy. By comparing the resulting privacy-preserving bartering model with the most prominent conventional bartering models by making use of simulation techniques, we show that our privacy-preserving bartering model (representing our envisioned privacy-preserving bartering system) is just as practicable as conventional bartering models and achieves comparable good results w.r.t. the number of matched parties as well as the average waiting time.

In a greater context, the synthesis of the provided contributions constitute a solution that allows an arbitrary number of parties to barter their commodities in a privacy-preserving fashion in such a way that unfairness, manipulation, and corruption can be excluded.

Three important directions for future work aiming at broadening the applicability of privacy-preserving bartering are (1) tapping into specialized barter markets, (2) supporting a more general bartering setting, and (3) examining further privacy-preserving bartering policies:

One important specialized barter market where privacy is a matter of concern is the

kidney exchange market. In such a market, a party corresponds to a patient with kidney disease who knows a living donor who is willing to donate one of his healthy kidneys to him. However, the patient and the donor are incompatible (w.r.t. their medical characteristics). A kidney exchange market allows patients to swap their incompatible donors to obtain a compatible donor [1]. In this context, a direction of future research is to adapt our privacy-preserving multi-party bartering protocols for privately checking whether (resp., to what extent) a patient and a donor are compatible.

In this thesis, we focus on a bartering setting where each party receives some quantity of its desired commodity from at most one party and sends some quantity of its offered commodity to at most one other party. As part of future work, it would be desirable to adapt the proposed privacy-preserving bartering protocols in such a way that the demand of a party can be simultaneously satisfied by multiple other parties as well as to allow the specification of multiple quotes in combination with complex bartering conditions that, e.g., can be used to express that a party is willing to barter commodity `A` for either commodity `B` or commodity `C`.

The third important direction for future work is to devise and study further privacy-preserving bartering policies that comprise advanced mechanisms. For example, timers could be used to propose bartering policies making use of threshold pools that are executed after the timer expired even if the threshold has not yet been reached. The related question is whether the number of matched parties (resp., the average waiting time) can be further increased (resp., further reduced).

Notation

Notation	Description
Binary Numbers	
$\wedge$	AND operation
$\vee$	OR operation
$\oplus$	XOR operation
x_{bin}	binary representation of $x \in \mathbb{N}^0$
$\|x\|$	bit length of $x \in \mathbb{N}$
Number Theory	
$\mathbb{F}$	finite field
$\mathsf{gcd}(a, b)$	greates common divisor of a and b
$\lambda(\cdot)$	Carmichael function
min	minimum
max	maximum
$n!$	faculty of n
$\mathbb{N}$	set of natural numbers excluding zero
$\mathbb{N}^0$	set of natural numbers including zero
$\mathbb{N}_u$	set of natural numbers bounded by integer u ($\mathbb{N}_u := \{1, \ldots, u\}$)
$\mathbf{P}$	set of prime numbers
$\mathbf{P}_I$	set of prime numbers in interval I
σ	permutation
σ^{-1}	inverse permutation
$\mathbb{Z}$	integers
$\mathbb{Z}_n$	residue class ring of the integers with ideal n
$\mathbb{Z}_n^*$	$\{a \in \mathbb{Z}_n \backslash \{0\} \mid \mathsf{gcd}(a, n) = 1\}$
Algebra & Analysis	
$\lfloor x \rfloor$	greatest integer less than or equal to x
$\lceil x \rceil$	smallest integer greater than or equal to x
$\mathcal{H}(V)$	Hamming weight of vector V
ln	log_2
$\mathbb{R}$	rational numbers
$\mathbb{R}^{>0}$	positive rational numbers
$\mathbb{R}^{\geq 0}$	rational numbers greater than or equal 0

Sets & Intervals	
$[\lambda, \mu]$	non-negative closed integer interval with $\lambda, \mu \in \mathbb{N}^0$, $\lambda \leq \mu$
$\|[\lambda, \mu]\|$	interval width $\mu - \lambda$ of $[\lambda, \mu]$
$\|S\|$	cardinality of set S
$e \leftarrow_{\$} S$	e is drawn from set (resp., interval) S uniformly at random
$e \leftarrow_{\mathcal{D}} S$	e is drawn from set (resp., interval) S according to discrete distribution $\mathcal{D}$
Graph Theory	
$\sqsubseteq$	subgraph relationship
$deg(\cdot)$	degree of a node
$deg^{+}(\cdot)$	incoming edges of a node
$deg^{-}(\cdot)$	outgoing edges of a node
G	graph
M	matching in a graph
$\omega_e(\cdot)$	edge weight function
Cryptography	
$[\![m]\!]$	encryption of m
$+_h$	homomorphic addition
$-_h$	homomorphic subtraction
$\times_h$	homomorphic scalar multiplication
$/_h$	homomorphic scalar division
$\oplus_h$	homomorphic scalar XOR
$\|\|_h$	homomorphic concatenation
$\mathsf{Blind}(\cdot)$	ciphertext blinding
$\mathbb{C}$	ciphertext space
N	Paillier modulus
$\mathbb{P}$	plaintext space
s	security parameter

Protocols	
$\perp$	empty string
C	index set of corrupted parties
$\mathcal{F}$	protocol functionality
$\mathcal{G}$	gate functionality
Γ	adversary structure
ι	number of all parties
k	security parameter (input length)
P_i	i-th party
$\mathscr{P}$	index set of all parties
π	protocol symbol
ρ	gate symbol
τ	threshold value of parties ($\tau \leq \iota$)
$\overline{x}$	cumulated protocol input of all parties

Bartering	
$\mathscr{C}$	set of commodities
$c_d^{(i)}$	desired commodity of party i
$c_o^{(i)}$	offered commodity of party i
$\mathbf{d}^{(i)}$	demand of party i
$\mathbf{o}^{(i)}$	offer of party i
$\mathbf{q}^{(i)}$	quote of party i
$\underline{q}_d^{(i)}$	minimum quantity desired by party i
$Q_d^{(i)}$	$Q_d^{(i)} := [\underline{q}_d^{(i)}, \infty)$
$\overline{q}_o^{(i)}$	maximum quantity offered by party i
$Q_o^{(i)}$	$Q_o^{(i)} := [1, \overline{q}_o^{(i)}]$

Misc	
$\forall$	universal quantifier
$\overset{c}{\equiv}$	computational indistinguishable
$\overset{s}{\equiv}$	statistical indistinguishable
$\overset{p}{\equiv}$	perfect indistinguishable
$a := b$	a is defined as b
$negl$	negligible function

List of Publications

- S. Wüller, M. Vu, U. Meyer, S. Wetzel: Using Secure Graph Algorithms for the Privacy-Preserving Identification of Optimal Bartering Opportunities. In *Proceedings of the 2017 ACM Workshop on Privacy in the Electronic Society (WPES), co-located with the 24th ACM Conference on Computer and Communications Security (CCS)*, 2017.

- S. Wüller, U. Meyer, S. Wetzel: Privacy-Preserving Multi-Party Bartering Secure Against Active Adversaries. In *Fifteenth Annual International Conference on Privacy, Security and Trust (PST)*, 2017.

- S. Wüller, U. Meyer, S. Wetzel: Privacy-Preserving Multi-Party Bartering Secure Against Active Adversaries (Extended Version), Cryptology ePrint Archive, Report 2017/377, 2017.

- S. Wüller, U. Meyer, S. Wetzel: Towards Privacy-Preserving Multi-Party Bartering. In *Financial Cryptography and Data Security: FC 2017 International Workshops, BITCOIN, TA, VOTING, WAHC, and WTSC*, 2017.

- S. Wüller, D. Mayer, F. Förg, S. Schüppen, B. Assadsolimani, U. Meyer, S. Wetzel: Designing Privacy-Preserving Interval Operations Based on Homomorphic Encryption and Secret Sharing Techniques, *Journal of Computer Security*, 2017.

- S. Wüller, U. Meyer, S. Wetzel: Towards Privacy-Preserving Multi-Party Bartering (Extended Version), RWTH Aachen University, Technical Report, AIB-2016-10, 2016.

- S. Wüller, W. Pessin, U. Meyer, S. Wetzel: Privacy-Preserving Two-Party Bartering Secure Against Active Adversaries. In *Fourteenth Annual International Conference on Privacy, Security and Trust (PST)*, 2016.

- S. Wüller, M. Kühnel, U. Meyer: Information Hiding in the Public RSA Modulus. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security (IH & MMSEC)*, 2016.

- S. Wüller, M. Kühnel, U. Meyer: Information Hiding in the Public RSA Modulus (Extended Version), RWTH Aachen University, Technical Report, AIB-2015-11, 2015.

- S. Wüller, U. Meyer, F. Förg, S. Wetzel: Privacy-Preserving Conditional Random Selection. In *Thirteenth Annual International Conference on Privacy, Security and Trust (PST)*, 2015.
- F. Förg, D. Mayer, S. Wetzel, S. Wüller, U. Meyer: A Secure Two-Party Bartering Protocol Using Privacy-Preserving Interval Operations. In *Twelfth Annual International Conference on Privacy, Security and Trust (PST)*, 2014.
- K. H. Krempels, C. Terwelp, S. Wüller, T. Frosch, S. Gökay: Communication Reduced Interaction Protocol between Customer, Charging Station, and Charging Station Management System. In *3rd International Conference on Smart Grids and Green IT Systems (SMARTGREENS)*, 2014.
- P. Heiniz, W. Kluth, K. H. Krempels, C. Terwelp, S. Wüller: Continuous Information Provisioning for the Conference Participation Process. In *10th International Conference on E-Business (ICE-B)*, 2013.
- W. Kluth, K. H. Krempels, C. Terwelp, S. Wüller: Increase of Travel Safety for Public Transport by Mobile Applications. In *10th International Conference on E-Business (ICE-B)*, 2013.
- P. Heiniz, K. H. Krempels, C. Terwelp, S. Wüller: Landmark-based Navigation in Complex Buildings. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2012.

Bibliography

[1] D. J. Abraham, A. Blum, and T. Sandholm. Clearing Algorithms for Barter Exchange Markets: Enabling Nationwide Kidney Exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, pages 295–304. ACM, 2007.

[2] G. Aggarwal, N. Mishra, and B. Pinkas. Secure Computation of the kth-Ranked Element. In *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, pages 40–55. Springer Berlin Heidelberg, 2004.

[3] E. Aïmeur, G. Brassard, and F. Mani Onana. Blind Sales in Electronic Commerce. In *Proceedings of the 6th International Conference on Electronic Commerce*, pages 148–157. ACM, 2004.

[4] E. Aïmeur, G. Brassard, and F. Mani Onana. Blind Negotiation in Electronic Commerce. In *Montreal Conference on eTechnologies*, pages 1–9, 2005.

[5] R. Anderson, I. Ashlagi, D. Gamarnik, and Y. Kanoria. A Dynamic Model of Barter Exchange. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1925–1933, 2014.

[6] R. Anderson, I. Ashlagi, D. Gamarnik, and Y. Kanoria. Efficient Dynamic Barter Exchange. Technical report, 2015.

[7] I. Ashlagi, P. Jaillet, and V. H. Manshadi. Kidney Exchange in Dynamic Sparse Heterogenous Pools. *CoRR*, abs/1301.3509, 2013.

[8] B. Assadsolimani. Bringing Privacy-Preserving Multi-Party Bartering to Practice. Master's thesis, RWTH Aachen University, 2018.

[9] E. Barker. NIST Special Publication 800-57: Recommendation for Key Management, Part 1: General, 2016.

[10] BarterOnly. `www.barteronly.com`.

[11] BarterQuest. `www.barterquest.com`.

[12] M. Bellare and P. Rogaway. Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[13] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A Fair Protocol for Signing Contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.

[14] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, 1988.

[15] D. Bernhard, O. Pereira, and B. Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *Advances in Cryptology - ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security*, pages 626–643. Springer Berlin Heidelberg, 2012.

[16] BizXchange. `www.bizxtrading.com`.

[17] M. Blanton and P. Gasti. Secure and Efficient Protocols for Iris and Fingerprint Identification. In *Computer Security – ESORICS 2011: 16th European Symposium on Research in Computer Security*, pages 190–209. Springer Berlin Heidelberg, 2011.

[18] P. Bocheck. Method and System for Managing Multi-Party Barter Transaction, 2008. US Patent 2008/0103987 A1.

[19] P. Bocheck. System and Method for Finding Potential Trading Partners in Both Two-Party and Multi-Party Scenarios, 2010. US Patent 2010/0161597 A1.

[20] D. Bogdanov, S. Laur, and J. Willemson. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *Computer Security - ESORICS 2008: 13th European Symposium on Research in Computer Security*, pages 192–206. Springer Berlin Heidelberg, 2008.

[21] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure Multiparty Computation Goes Live. In *Financial Cryptography and Data Security: 13th International Conference, FC 2009*, pages 325–343. Springer Berlin Heidelberg, 2009.

[22] F. Brandt. A Verifiable, Bidder-Resolved Auction Protocol. In *5th International Workshop on Deception, Fraud and Trust in Agent Societies*, pages 18–25, 2002.

[23] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos. SEPIA: Privacy-preserving Aggregation of Multi-domain Network Events and Statistics. In *Proceedings of the 19th USENIX Conference on Security*, pages 15–31. USENIX Association, 2010.

[24] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[25] W.-T. Chu and F.-C. Chang. A Privacy-Preserving Bipartite Graph Matching Framework for Multimedia Analysis and Retrieval. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 243–250. ACM, 2015.

[26] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In *Advances in Cryptology – CRYPTO 2011: 31st Annual Cryptology Conference*, pages 487–504. Springer Berlin Heidelberg, 2011.

[27] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. Technical report, 2000.

[28] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation from Threshold Homomorphic Encryption. In *Advances in Cryptology - EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques*, pages 280–300. Springer Berlin Heidelberg, 2001.

[29] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty Computation, an Introduction. `http://www-cs.ccny.cuny.edu/~fazio/F15-csc85030/readings/CDN09.pdf`, 2009.

[30] R. Cramer, R. Gennaro, and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *Advances in Cryptology - EUROCRYPT '97: International Conference on the Theory and Application of Cryptographic Techniques*, pages 103–118. Springer Berlin Heidelberg, 1997.

[31] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung. Secure Efficient Multiparty Computing of Multivariate Polynomials and Applications. In *Applied Cryptography and Network Security: 9th International Conference, ACNS 2011*, pages 130–146. Springer Berlin Heidelberg, 2011.

[32] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Verlag Berlin Heidelberg, 2008.

[33] T. Dalenius. Towards a Methodology for Statistical Disclosure Control. *Statistik Tidskrift*, 15:429–444, 1977.

[34] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous Multiparty Computation: Theory and Implementation. In *Public Key Cryptography – PKC 2009:*

12th International Conference on Practice and Theory in Public Key Cryptography, pages 160–179. Springer Berlin Heidelberg, 2009.

[35] I. Damgård and M. Jurik. A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001*, pages 119–136. Springer Berlin Heidelberg, 2001.

[36] I. Damgård and M. Jurik. A Length-Flexible Threshold Cryptosystem with Applications. In *Information Security and Privacy: 8th Australasian Conference, ACISP 2003*, pages 350–364. Springer Berlin Heidelberg, 2003.

[37] DESMO-J. `http://desmoj.sourceforge.net/`.

[38] DESMO-J API. API Documentation. `http://desmoj.sourceforge.net/doc/index.html`.

[39] R. Diestel. *Graphentheorie*. Springer Spektrum, 2010.

[40] P. Drucker. *Managing in the Next Society*. Truman Talley Books, 2002.

[41] C. Dwork. Differential Privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006*, pages 1–12. Springer Berlin Heidelberg, 2006.

[42] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006*, pages 265–284. Springer Berlin Heidelberg, 2006.

[43] J. Dyer, M. Dyer, and J. Xu. Practical Homomorphic Encryption Over the Integers. *CoRR*, abs/1702.07588, 2017.

[44] Encyclopedia Britannica. `www.britannica.com`.

[45] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-Preserving Face Recognition. In *Privacy Enhancing Technologies: 9th International Symposium, PETS 2009*, pages 235–253. Springer Berlin Heidelberg, 2009.

[46] R. Fagin, M. Naor, and P. Winkler. Comparing Information Without Leaking It. *Communications of the ACM*, 39(5):77–85, 1996.

[47] A. Fiat and A. Shamir. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology - CRYPTO'86*, pages 186–194. Springer Berlin Heidelberg, 1987.

[48] F. Förg, D. Mayer, S. Wetzel, S. Wüller, and U. Meyer. A Secure Two-Party Bartering Protocol Using Privacy-Preserving Interval Operations. In *Twelfth Annual International Conference on Privacy, Security and Trust (PST)*, pages 57–66, 2014.

[49] P.-A. Fouque and D. Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In *Advances in Cryptology - ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 351–368. Springer Berlin Heidelberg, 2001.

[50] P.-A. Fouque, G. Poupard, and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In *Financial Cryptography: 4th International Conference, FC 2000*, pages 90–104. Springer Berlin Heidelberg, 2001.

[51] M. Franklin and G. Tsudik. Secure Group Barter: Multi-party Fair Exchange with Semi-trusted Neutral Parties. In *Financial Cryptography: Second International Conference, FC '98*, pages 90–102. Springer Berlin Heidelberg, 1998.

[52] M. K. Franklin and M. K. Reiter. Fair Exchange with a Semi-trusted Third Party (Extended Abstract). In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 1–5. ACM, 1997.

[53] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–19. Springer Berlin Heidelberg, 2004.

[54] K. Frikken and L. Opyrchal. PBS: Private Bartering Systems. In *Financial Cryptography and Data Security: 12th International Conference, FC 2008*, pages 113–127. Springer Berlin Heidelberg, 2008.

[55] J. Garay, B. Schoenmakers, and J. Villegas. Practical and Secure Solutions for Integer Comparison. In *Public Key Cryptography - PKC 2007: 10th International Conference on Practice and Theory in Public-Key Cryptography*, pages 330–342. Springer Berlin Heidelberg, 2007.

[56] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009.

[57] C. Gentry and S. Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In *Advances in Cryptology - EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 129–148. Springer Berlin Heidelberg, 2011.

[58] J. Göbel, P. Joschko, A. Koors, and B. Page. The Discrete Event Simulation Framework DESMO-J: Review, Comparison To Other Frameworks And Latest Development. In *Proceedings of the 27th European Conference on Modelling and Simulation*, pages 100–109, 2013.

[59] O. Goldreich. *Foundations of Cryptography - Volume II Basic Applications*. Cambridge Unversity Press, 2004.

[60] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM, 1987.

[61] O. Goldreich and R. Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *Journal of the ACM*, 43(3):431–473, 1996.

[62] S. Goldwasser and S. Micali. Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM, 1982.

[63] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based Encryption for Fine-grained Access Control of Encrypted Data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[64] J. Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. In *Public Key Cryptography - PKC 2003: 6th International Workshop on Practice and Theory in Public Key Cryptography*, pages 145–160. Springer Berlin Heidelberg, 2002.

[65] J. Guajardo, B. Mennink, and B. Schoenmakers. Modulo Reduction for Paillier Encryptions and Application to Secure Statistical Analysis. In *Financial Cryptography and Data Security: 14th International Conference, FC 2010*, pages 375–382. Springer Berlin Heidelberg, 2010.

[66] K. Hamada, R. Kikuchi, D. Ikarashi, K. Chida, and K. Takahashi. Practically Efficient Multi-party Sorting Protocols from Comparison Sort Algorithms. In *Information Security and Cryptology – ICISC 2012: 15th International Conference*, pages 202–216. Springer Berlin Heidelberg, 2013.

[67] B. Hayes. Alice and Bob in Cipherspace. *American Scientist*, 100(5):362–367, 2012.

[68] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols*. Springer Heidelberg, 2010.

[69] C. Hazay, G. L. Mikkelsen, T. Rabin, and T. Toft. Efficient RSA Key Generation and Threshold Paillier in the Two-Party Setting. In *Topics in Cryptology - CT-RSA 2012*, pages 313–331. Springer Berlin Heidelberg, 2012.

[70] J. Hsu, Z. Huang, A. Roth, T. Roughgarden, and Z. S. Wu. Private Matchings and Allocations. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, pages 21–30. ACM, 2014.

[71] K. V. Jónsson, G. Kreitz, and M. Uddin. Secure Multi-Party Sorting and Applications. Cryptology ePrint Archive, Report 2011/122, 2011. `http://eprint.iacr.org/2011/122`.

[72] S. Kannan, J. Morgenstern, R. Rogers, and A. Roth. Private Pareto Optimal Exchange. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 261–278. ACM, 2015.

[73] M. Kantarcioglu and O. Kardes. Privacy-Preserving Data Mining in the Malicious Model. *International Journal of Information and Computer Security*, 2(4):353–375, 2008.

[74] J. Katz and Y. Lindell. *Introduction to Modern Cryptgraphy*. Chapman & Hall/CRC, 2nd edition, 2015.

[75] M. Kearns, M. Pai, A. Roth, and J. Ullman. Mechanism Design in Large Games: Incentives and Privacy. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, pages 403–410. ACM, 2014.

[76] F. Kerschbaum and O. Terzidis. Filtering for Private Collaborative Benchmarking. In *Emerging Trends in Information and Communication Security: International Conference, ETRICS 2006*, pages 409–422. Springer Berlin Heidelberg, 2006.

[77] V. Kolesnikov. *Secure Two-party Computation and Communication*. PhD thesis, University of Toronto, 2006.

[78] H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics*, 2(1-2):83–97, 1955.

[79] S. Laur, J. Willemson, and B. Zhang. Round-Efficient Oblivious Database Manipulation. In *Information Security: 14th International Conference, ISC 2011*, pages 262–277. Springer Berlin Heidelberg, 2011.

[80] Y. Lindell and B. Pinkas. Privacy Preserving Data Mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[81] Y. Lindell and B. Pinkas. Secure Multiparty Computation for Privacy-Preserving Data Mining. *Journal of Privacy and Confidentiality*, 1(1):59–98, 2009.

[82] N. López, M. Núñez, I. Rodríguez, and F. Rubio. A Multi-agent System for e-Barter Including Transaction and Shipping Costs. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 587–594. ACM, 2003.

[83] S. Lu and R. Ostrovsky. Distributed Oblivious RAM for Secure Two-Party Computation. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013*, pages 377–396. Springer Berlin Heidelberg, 2013.

[84] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay—a Secure Two-party Computation System. In *Proceedings of the 13th Conference on USENIX Security Symposium*, pages 20–37. USENIX Association, 2004.

[85] S. Mathieu. Match-Making in Bartering Scenarios. Master's thesis, University of New Brunswick, 2006.

[86] D. Mayer. *Design and Implementation of Efficient Privacy-Preserving and Unbiased Reconciliation Protocols*. PhD thesis, Stevens Institute of Technology, 2012.

[87] S. Micali. Simple and Fast Optimistic Protocols for Fair Electronic Exchange. In *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, pages 12–19. ACM, 2003.

[88] J. Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[89] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[90] G. Neugebauer and U. Meyer. SMC-MuSe: A Framework for Secure Multi-Party Computation on MultiSets. Technical Report AIB-2012-16, RWTH Aachen University, 2012.

[91] T. Nishide and K. Sakurai. Distributed Paillier Cryptosystem without Trusted Dealer. In *Information Security Applications: 11th International Workshop, WISA 2010*, pages 44–60. Springer Berlin Heidelberg, 2011.

[92] J. Nzouonta, M.-C. Silaghi, and M. Yokoo. Secure Computation for Combinatorial Auctions and Market Exchanges. In *Proceedings of the Third International Joint*

Conference on Autonomous Agents and Multiagent Systems, pages 1398–1399. IEEE Computer Society, 2004.

[93] B. Page and W. Kreutzer. *The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java*. Shaker, 2005.

[94] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology - EUROCRYPT '99: International Conference on the Theory and Application of Cryptographic Techniques*, pages 223–238. Springer Berlin Heidelberg, 1999.

[95] J. M. Park, E. K. P. Chong, and H. J. Siegel. Constructing Fair-exchange Protocols for E-commerce via Distributed Computation of RSA Signatures. In *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, pages 172–181. ACM, 2003.

[96] C. Peikert, V. Vaikuntanathan, and B. Waters. *A Framework for Efficient and Composable Oblivious Transfer*, pages 554–571. Springer Berlin Heidelberg, 2008.

[97] B. Pinkas. Cryptographic Techniques for Privacy-preserving Data Mining. *SIGKDD Explorations Newsletter*, 4(2):12–19, 2002.

[98] M. O. Rabin. How to Exchange Secrets with Oblivious Transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[99] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, pages 73–85. ACM, 1989.

[100] readitswapit. `www.readitswapit.co.uk`.

[101] T. Reistad and T. Toft. Linear, Constant-Rounds Bit-Decomposition. In *Information Security and Cryptology - ICISC 2009*, pages 245–257. Springer Berlin Heidelberg, 2010.

[102] B. Rosenberg, editor. *Handbook of Financial Cryptography and Security*. Chapman and Hall/CRC, 2010.

[103] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *Advances in Cryptology – EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer Berlin Heidelberg, 2005.

[104] B. Sarikaya, editor. *Geographic Location in the Internet*. Springer US, 2002.

[105] B. Schoenmakers and P. Tuyls. Practical Two-Party Computation Based on the Conditional Gate. In *Advances in Cryptology - ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security*, pages 119–136. Springer Berlin Heidelberg, 2004.

[106] B. Schoenmakers and P. Tuyls. Efficient Binary Conversion for Paillier Encrypted Values. In *Advances in Cryptology - EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 522–537. Springer Berlin Heidelberg, 2006.

[107] A. Shamir. How to Share a Secret. *Communcations of the ACM*, 22(11):612–613, 1979.

[108] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology: Proceedings of CRYPTO '84*, pages 47–53. Springer Berlin Heidelberg, 1985.

[109] Y. Shen and S. Pearson. Privacy Enhancing Technologies: A Review. Technical report, HP Laboratories, 2011.

[110] R. Smith and J. Shao. Privacy and E-Commerce: A Consumer-Centric Perspective. *Electronic Commerce Research*, 7(2):89–116, 2007.

[111] SwapRight. `www.swapright.com`.

[112] L. Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[113] T. Toft. Sub-Linear, Secure Comparison with Two Non-colluding Parties. In *Public Key Cryptography - PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography*, pages 174–191. Springer Berlin Heidelberg, 2011.

[114] Tor. `www.torproject.org`.

[115] U-Exchange. `www.u-exchange.com`.

[116] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *Advances in Cryptology — EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer Berlin Heidelberg, 2010.

[117] M. Vu. Designing an Efficient Privacy-Preserving Multi-Party Protocol for the Identification of Bartering Opportunities by Means of Secure Graph Algorithms. Master's thesis, RWTH Aachen University, 2017.

[118] S. Wüller, D. Mayer, F. Förg, S. Schüppen, B. Assadsolimani, U. Meyer, and S. Wetzel. Designing Privacy-Preserving Interval Operations Based on Homomorphic Encryption and Secret Sharing Techniques. *Journal of Computer Security*, 25(1):59–81, 2017.

[119] S. Wüller, U. Meyer, F. Förg, and S. Wetzel. Privacy-Preserving Conditional Random Selection. In *Thirteenth Annual International Conference on Privacy, Security and Trust (PST)*, pages 44–53, 2015.

[120] S. Wüller, U. Meyer, and S. Wetzel. Towards Privacy-Preserving Multi-Party Bartering (Extended Version). Technical Report AIB-2016-10, RWTH Aachen University, 2016.

[121] S. Wüller, U. Meyer, and S. Wetzel. Privacy-Preserving Multi-Party Bartering Secure Against Active Adversaries. In *Fifteenth Annual International Conference on Privacy, Security and Trust (PST)*, 2017.

[122] S. Wüller, U. Meyer, and S. Wetzel. Privacy-Preserving Multi-Party Bartering Secure Against Active Adversaries (Extended Version). Cryptology ePrint Archive, Report 2017/377, 2017. `http://eprint.iacr.org/2017/377`.

[123] S. Wüller, U. Meyer, and S. Wetzel. Towards Privacy-Preserving Multi-Party Bartering. In *Financial Cryptography and Data Security: FC 2017 International Workshops, BITCOIN, TA, VOTING, WAHC, and WTSC*, pages 19–34. Springer International Publishing, 2017.

[124] S. Wüller, W. Pessin, U. Meyer, and S. Wetzel. Privacy-Preserving Two-Party Bartering Secure Against Active Adversaries. In *Fourteenth Annual International Conference on Privacy, Security and Trust (PST)*, pages 229–238, 2016.

[125] S. Wüller, M. Vu, U. Meyer, and S. Wetzel. Using Secure Graph Algorithms for the Privacy-Preserving Identification of Optimal Bartering Opportunities. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 123–132. ACM, 2017.

[126] A. C. Yao. Protocols for Secure Computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[127] A. C. Yao. How to Generate and Exchange Secrets. In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[128] B. Zhang. Generic Constant-Round Oblivious Sorting Algorithm for MPC. In *Provable Security: 5th International Conference, ProvSec 2011*, pages 240–256. Springer Berlin Heidelberg, 2011.