Privacy Preserving Train Scheduling

Using homomorphic encryption to create train schedules

Masters Thesis

Prakhar Jain

Delft University of Technology

Privacy Preserving Train Scheduling

Using homomorphic encryption to create train schedules

by

Prakhar Jain

Student Number 5800277

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday July 2, 2024 at 1:00 PM.

Supervisor:	Dr. Zekeriya Erkin
Committee Member:	Dr. Annibale Panichella
Advisor:	Tianyu Li
Project Duration:	18th September, 2023 - 2nd July, 2024

Cover image is from UK Travel Guide [1]

[1] https://happytowander.com/uk-train-travel-guide/

TUDelft

Preface

I got inspiration to work on this problem through my work with my supervisor Dr. Z Erkin. He saw my excitement working with the city of Amsterdam on a similar problem, trying to divide passengers between trains and trams to avoid overcrowding. I thought that this problem is down to sub-optimal schedules between various operators and services. I changed my approach to the problem atleast once every two weeks and Zeki pushed me to work hard understanding the real world implications of creating a good solution to the problem.

Through the nine and a half months of working on this thesis, I learnt a lot about myself. I learnt my research techniques, my strengths in communication and weaknesses in precision, something I believe I have improved on, with a good framework to go forth. Overall, this was not the easiest journey and would not have been possible without the absolute support of the entire research group. From my advisor Tianyu continuously helping me see where my logic falls short and having faith in my output from the beginning to Jelle Vos being an absolute expert in the field solving doubts I was carrying for weeks trying to look at every source in a single conversation. The combined meetings within the research group helped me streamline my thinking right from the get go, and helped me focus on what is important. Chelsea Guan, Davis Sterns, Vojta Crha and Juno Jense consistently helped me formalize my ideas. They were the first ones I discussed any novel idea with and their insights helped me improve upon them and understand my hunches better.

Next, I would like to thank my friends Nathan, Dylan, Niklas, Susanne, Nevena and Frank for making my degree in Delft, the most enjoyable time of my life. My friends Anvi, Karan and Aryan though not present in Delft, have always helped me push myself to the limit.

Lastly, my parents in India have been an absolute pillar. They helped and encouraged me to take risks, move to a new country to realize my potential. Knowing that they are always there for me and will support me through thick and thin has been the biggest reason I have been able to keep myself calm more often than not.

Prakhar Jain
Delft, June 2024

Abstract

A substantial number of passengers in Europe rely on trains for transportation, facilitated by a network of high-speed international trains. However, the coordination of train schedules across multiple networks often poses challenges due to incompatible timings. The scheduling of multiple train networks shares similarities with multi-processor task scheduling and airline scheduling but is distinguished by its cooperative nature rather than a competitive one. Cooperative scheduling necessitates the sharing of private information. This information, 'demand', is commercial sensitive information, since it can reveal demographic information like incomes and tax returns. Privacy-preserving protocols can enable the computation of statistics without revealing this demands (in railway systems) to unauthorized parties. Despite the critical role of privacy in multi-party scheduling, research in this domain remains limited due to domain specific constraints. A model supporting such privacy considerations could significantly help Europe achieve its carbon-neutral goals while improving cross-border services. In this research, we propose a system designed to facilitate joint service scheduling, ensuring confidentiality, integrity, and authenticity. We use partial and fully homomorphic encryption techniques that mimic the outcomes achievable with a trusted third party. We conduct a comparative analysis of online and offline approaches, emphasizing how they achieve confidentiality, collusion-resistance, traceability and non-repudiation. Theoretical and experimental evaluations demonstrate the feasibility of the system for real-world applications by creating schedules for upto four parties. Our solution for scheduling seven slots takes ≈ 3 hours, which is a feasible duration to solve a problem of this size.

Contents

1 Introduction

In our interconnected and intricate society, everything, from the utilization of time to energy consumption, operates within the framework of schedules. Among this complexity, the pursuit of simplicity is realized through the creation of schedules, whether in the personal realm for effective time management or within a university setting for structuring lectures and exams. In this perpetual struggle with time, schedules emerge as our singular weapon to manage time. Strategic management of resources and tasks assumes a pivotal role, serving as a linchpin in optimizing efficiency and achieving desired results. Scheduling, as a foundational concept, revolves around the systematic arrangement of tasks or activities over time, considering diverse constraints and objectives.

As our world embraces increasing complexity, our schedules are intricately interwoven with the schedules of others. The evolution of scheduling into a distributed problem, triggered by the advent of multi-core processors necessitating task distribution across various resources, introduces new challenges [47]. Depending on the nature of multi-party schedules, concerns regarding the privacy of data owned by various stakeholders within the schedule surface. This thesis focuses on the exploration of privacy within scheduling algorithms, specifically in the context of railway timetabling. The objective is to formulate privacy definitions for the scheduling problem and propose solutions facilitating distributed railway scheduling while safeguarding the sensitive data of involved parties.

A schedule serves as a plan delineating a sequence of activities, events, or tasks, along with their corresponding start times, durations, and end times. Scheduling involves the methodical arrangement and allocation of resources to accomplish specific objectives within a defined timeframe. Schedules find application in diverse contexts, spanning project management, production planning, transportation, events, and daily routines [23]. The field of scheduling has witnessed increased research activities, particularly with advancements in cloud computing and resource sharing. The Journal of Scheduling, now in its 20th year of publication, underscores the significance of the field and the notable developments it has undergone [1].

Schedules can vary in length, spanning milliseconds for scheduling tasks on a processor to yearly schedules for a stadium. Following this timeframe, a schedule is categorized as either a ‘Periodic Schedule’ [74], a ‘Cyclic Schedule’ [4], or a ‘Non-Periodic Schedule’, where the previous iteration provides no information aiding the formation of the subsequent schedule. The complexity of forming schedules varies depending on the number and intricacy of individual

[1]https://link.springer.com/journal/10951

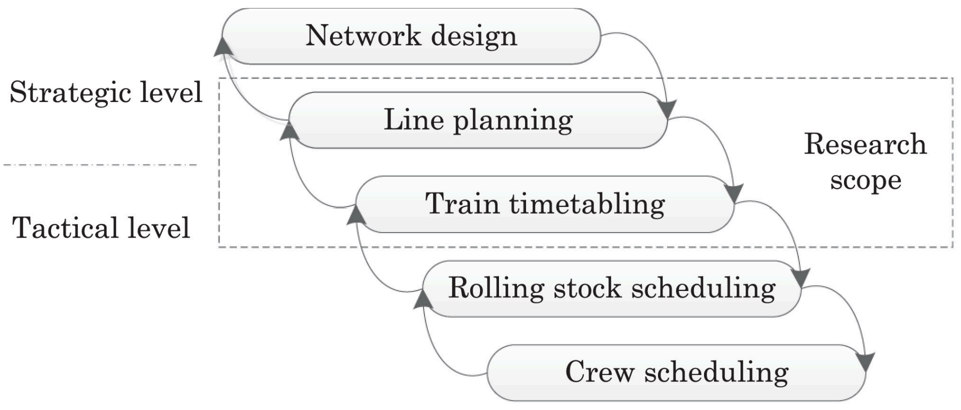


Figure 1.1: Research Gap [8]

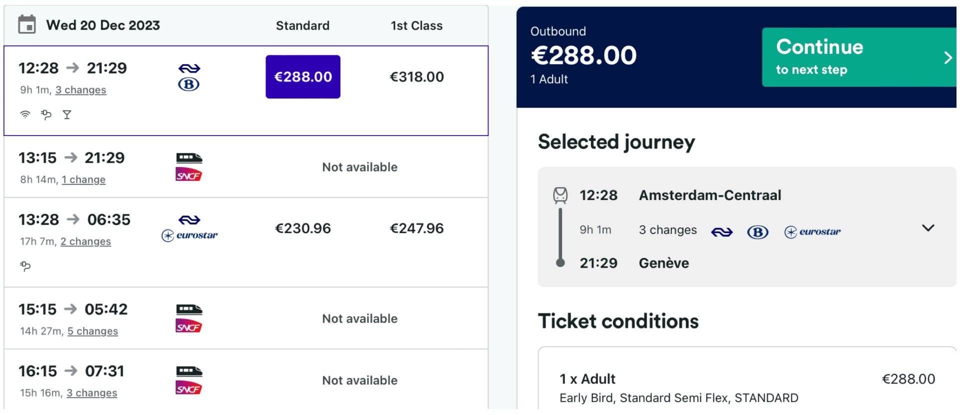
constraints and the amount of resources. The intricacies of different scheduling algorithms are explored in Section 2.1.

1.1. Train Scheduling

Trains, recognized for their speed and eco-friendliness, serve as a prominent mode of transportation across Europe, accommodating journeys of various durations. The railway landscape in Europe spans from trams transporting 50 individuals at an average speed of 20 km/h to high-speed trains with capacities of 1000 passengers, reaching speeds of 300 km/h.

The train schedule plays an important role in the operation of a railway line. Scheduling or Timetabling is the process of providing a definite departure and arrival time for a set of trains in a railway network. It reports absolute values of arrival-departure times of all trains and at all stations and also arrival-departure orders of all trains at each station. Broadly, there are two types of schedule intended for passenger traffic—One is cyclic/periodic and the other is a non-cyclic/aperiodic schedule [80]. Cyclic schedule caters mainly to suburban traffic, where the travel times of trains are uniform e.g., train from Delft to Eindhoven is every 30 minutes. Noncyclic timetables are developed for long-distance passenger trains to achieve certain performance objectives, e.g., the train to Paris from Rotterdam is 1700 to accommodate the rush hour traffic. A periodic schedule is simple to create. A single period can be scheduled, and thereafter, the same period can be copied across a day to create the daily schedule. In recent years, periodic train schedules with a single operation period, such as a metro system, have been popular. Given its convenience, it is used in countries where rail transport acts as a backbone. The regularity of the schedule helps passengers get into a schedule themselves [83]. It is worth noting that aperiodic schedules apply to fewer connections, these are usually long-distance overnight journeys.

Given the ubiquity and frequency of periodic schedules, this thesis will mainly focus towards a method ideal for periodic scheduling but we will discuss extensions that can accommodate aperiodic schedules. High-quality schedules can accurately meet the travel demands of passengers, avoid a waste of resources, and achieve sustainable development. The field of scheduling is an active field of research [80]. With prevalence of interchanges for reaching regions beyond local areas having increased, especially with the introduction of metros, the concept of changing railway lines to access new destinations is. Due to the substantial upfront cost of track laying and shared infrastructure, multiple railway lines often use the same set of tracks. Consequently, synchronizing the train schedules becomes imperative to optimize the utilization of this limited and shared resource.



Wed 20 Dec 2023		Standard	1st Class
12:28 → 21:29	9h 1m, 3 changes	€288.00	€318.00
13:15 → 21:29	8h 14m, 1 change	Not available	
13:28 → 06:35	17h 7m, 2 changes	€230.96	€247.96
15:15 → 05:42	14h 27m, 5 changes	Not available	
16:15 → 07:31	15h 16m, 3 changes	Not available	

Outbound
€288.00
1 Adult

Continue
to next step

Selected journey

12:28 Amsterdam-Centraal
9h 1m 3 changes eurostar
21:29 Genève

Ticket conditions

1 x Adult €288.00
Early Bird, Standard Semi Flex, STANDARD

Figure 1.2: Amsterdam to Geneva as a tourist

The train scheduling problem encompasses three sub-problems:

1. Determining optimal departure timings along a train's route.
2. Locomotive assignment.
3. Locomotive team assignment.

Although these problems are interrelated, the core lies in solving the train scheduling problem, upon which constraints related to locomotives and teams are subsequently applied [63]. With a history of around 70 years in research [22], train scheduling stands as a complex constraint optimization problem. We refer to the works of Bussieck et al. [8] to gauge where we can improve the scheduling process. The authors present the entire scheduling system and guide us to where the research gap is. This is presented in Figure 1.1. The research gap lies in Train timetabling and line planning. In this work, we will focus on Train timetabling and briefly discuss how the work can be extended to Line planning.

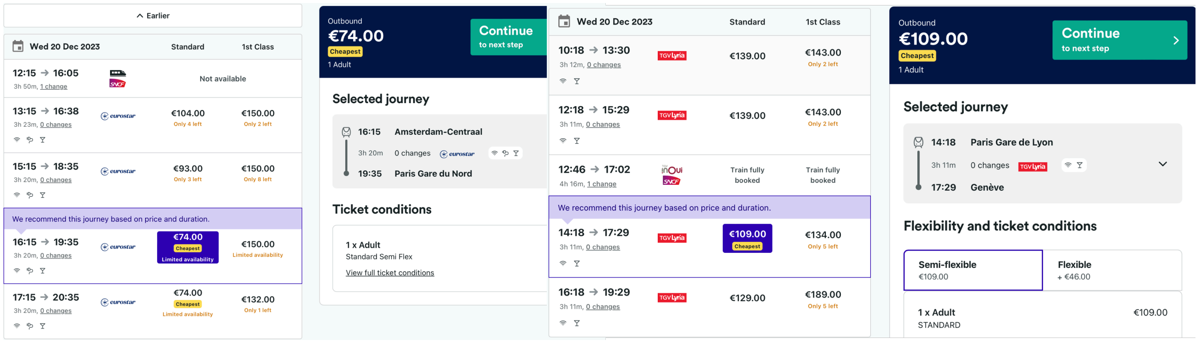
Recent research, influenced by urban transit methods, plays a crucial role in adapting schedules to dynamic conditions, such as unexpected disruptions or changes in demand [58]. Time table scheduling incorporates various methods, including Machine Learning, which provides adaptive solutions by learning from historical data and adapting to evolving operational environments [39]. This is in addition to standard optimization algorithms in Linear Programming and Constraint Optimization [83, 78].

The privatization of rail services in many European countries introduces multiple options for travelers, from state-owned to private entities. The availability of various options theoretically simplifies travel, as passenger demand influences the frequency of scheduled railway services. As illustrated in Figure 1.2, tourists planning a week-long visit to Europe may rely on platforms like Trainline[2] for travel planning. However, discerning travelers can optimize their journeys, focusing on high-speed rail lines to reduce travel times significantly without incurring additional costs, as depicted in Figure 1.3.

1.2. Problems

Since trains are not scheduled to ensure easy transfers between operators, it is up to the passenger to create an optimal schedule for best travel experience. International train stations, by definition, serve trains to multiple countries, which requires the scheduling of independent

[2]https://www.trainline.com


Earlier

Wed 20 Dec 2023		Standard	1st Class
12:15 → 16:05 3h 50m, 1 change	SNCF	Not available	
13:15 → 16:38 3h 23m, 0 changes	eurostar	€104.00 Only 4 left	€150.00 Only 2 left
15:15 → 18:35 3h 20m, 0 changes	eurostar	€93.00 Only 3 left	€150.00 Only 8 left
We recommend this journey based on price and duration.			
16:15 → 19:35 3h 20m, 0 changes	eurostar	€74.00 Cheapest Limited availability	€150.00 Limited availability
17:15 → 20:35 3h 20m, 0 changes	eurostar	€74.00 Cheapest Limited availability	€132.00 Only 1 left

Outbound
€74.00
Cheapest
1 Adult
Continue to next step

Selected journey

16:15 Amsterdam-Centraal
3h 20m 0 changes eurostar
19:35 Paris Gare du Nord

Ticket conditions

1 x Adult
Standard Semi Flex
View full ticket conditions

Wed 20 Dec 2023		Standard	1st Class
10:18 → 13:30 3h 12m, 0 changes	TGV Lyria	€139.00	€143.00 Only 2 left
12:18 → 15:29 3h 11m, 0 changes	TGV Lyria	€139.00	€143.00 Only 2 left
12:46 → 17:02 4h 16m, 1 change	TGV inOui SNCF	Train fully booked	Train fully booked
We recommend this journey based on price and duration.			
14:18 → 17:29 3h 11m, 0 changes	TGV Lyria	€109.00 Cheapest	€134.00 Only 5 left
16:18 → 19:29 3h 11m, 0 changes	TGV Lyria	€129.00	€189.00 Only 5 left

Outbound
€109.00
Cheapest
1 Adult
Continue to next step

Selected journey

14:18 Paris Gare de Lyon
3h 11m 0 changes TGV Lyria
17:29 Genève

Flexibility and ticket conditions

Semi-flexible €109.00	Flexible + €46.00

1 x Adult €109.00
STANDARD

Figure 1.3: Option for a local

rail operators. Knowing transfer options is crucial for planning long-distance journeys, as illustrated in Figure 1.3, where the 12:15 train from Amsterdam connects seamlessly with the 16:18 train from Paris, allowing for a transfer time of under 15 minutes. However, given the common occurrence of delays, a 15-minute buffer renders this option impractical. Despite having five trains departing from Paris, only one aligns with the schedule for travelers originating from Amsterdam. The short connection time poses a risk and missing the connection would require a day-long stay in the city. This issue stems from independent scheduling practices, with SNCF (The French Railway) and NS (The Dutch Railways) optimizing their routes based on constraints tailored to their specific use cases. Eurostar, as a coordinating entity, collaborates with national rails to formulate its schedule.

1.2.1. Distributed Railway Scheduling

When NS and SNCF share the same infrastructure at certain points, they inevitably influence each other's passenger flow, forming part of the same network with weakly linked constraints. An ideal system would involve all railway systems sharing any infrastructure being scheduled collaboratively. The need for a distributed railway scheduling system arises, drawing inspiration from parallel problems such as distributed task scheduling [47].

1.2.2. Privacy in Scheduling

Privacy-preserving scheduling algorithms delicately balance the optimization of resource allocation while protecting sensitive information. This is particularly crucial in railway scheduling, where constraints could potentially expose private information. Therefore, it is essential to protect the data required for constraints or objective functions from external exposure.

The concept of privacy-preserving scheduling is relatively new and its system properties can be compared with similar fields. For instance, the system of multiparty scheduling, which is an abstraction of slot allotment, can be paralleled with the field of auctioning. Privacy-preserving auctions have been extensively researched in [32], [34], and [52]. Consequently, the following properties can be adapted to multiparty privacy-preserving train timetable scheduling:

- **Unforgeability**: Each participant's demands are securely tied to their own identity, preventing any attempt to falsely attribute demands to another participant.
- **Anonymity**: The demands submitted by an operator are detached from their personal identity, ensuring that there is no identifiable link between the two.
- **Pseudonymity**: Operators are represented by pseudonyms, safeguarding their real identities while still allowing for identification within the system.
- **Collusion-Resistance**: The scheduling process is designed to withstand collusion at-

tempts amongst operators even when there is a dishonest majority.

- **Public Verifiability**: Anyone can independently verify the final schedule, enhancing transparency and accountability in the scheduling process.
- **Non-Repudiation**: Once operators submit their demands, they cannot later deny their participation, adding credibility and trust to the scheduling system.
- **Traceability**: Operators who act maliciously by creating wrong schedules or those who have won any slots must be identifiable and traceable.
- **Confidentiality**: While the schedule discloses the allocated slots and their respective operators, it conceals information about other operators, ensuring confidentiality and privacy for all participants except those involved in the specific slot.

The privacy-preserving properties a system provides depend on the underlying protocols. To enable information sharing without compromising privacy, these algorithms employ cryptographic techniques, secure multiparty computation [11], homomorphic encryption [35], differential privacy mechanisms [26], or a combination of these. This ensures the confidentiality of personal or proprietary information throughout the scheduling process.

1.3. Research Question

The primary objective of this thesis is to formulate a protocol for generating a timetable collaboratively among multiple railway providers while upholding the privacy of each party's confidential information. The key stakeholders in this collaborative effort are the railway operators, the train station and the scheduling party (ProRail) and a notable aspect of this approach is the absence of a third-party mediator to ensure the system's functionality among any number of participating operators.

This research operates within the malicious model, acknowledging the potential for a bounded subset of users to engage in malicious behavior and collusion with other operators. The malicious intent is directed towards uncovering the demands of the operator, which is private to them, to disrupt the integrity and authenticity of the final timetable. In essence, the goal is to prevent any participating party from gaining access to the demand of others or manipulating the submitted data once it is in the system. This thesis tackles the following research question.
How to design a privacy-preserving schedule for a train timetable at International Stations?
How to achieve similar privacy guarantees utilizing a constant number of communication steps between the participating parties?

1.4. Contributions

In this thesis, we propose a data privacy-preserving protocol to schedule trains from multiple operators at an international station, utilizing a set of privately held values in a manner that ensures privacy preservation. More specifically, given a set of operators, each holding private integer values for their passenger demands as private variables, the protocol computes the schedule of all trains in such a way that no party can learn the private values of another user. Only the final schedule is revealed in the end. This final schedule maximizes the overall demand served by combinatorially making all possible schedules. To create the schedules, we use homomorphic encryption techniques. Since we only require additive homomorphism, we create a partially homomorphic encryption-based protocol. We propose an alternative protocol that uses fully homomorphic encryption. This protocol utilizes fewer communication steps, down from exponential to constant while achieving the same schedule. To our knowledge, our protocol is the first to ensure confidentiality of private values even in presence of $< 50\%$ malicious operators. Additionally, our protocol ensures non-repudiation, traceability and collusion resistance with other parties being semi-honest. Note that data poisoning attacks from the users are outside of the scope of this thesis. We supplement our protocol with an improvement. The improvement is to make the system be light and use less bandwidth. We are able to offer all the security properties mentioned in a malicious majority setting and hide the ordering of various demands in this implementation, such that even domain knowledge cannot reveal any information about the privately held values. Finally, we provide theoretical evaluations of the security and performance of our protocols as well as a practical analysis of their performance with a proof-of-concept implementation. Our runtime analysis shows the feasibility of such a protocol for real world scheduling problems in reasonable time $\approx 3hours$ for four independent operators scheduling a total of seven trains to fill in one period of schedule.

1.5. Outline

In Chapter 2, we discuss the essential mathematical and cryptographic concepts required to understand our protocols. We explain basic algorithmics concepts, required to understand the algorithms currently used for train scheduling. Then we describe these algorithms, their salient features and give a motivation for their use. Then we describe all the cryptographic concepts used in similar protocols to ours and give motivation for different homomorphic encryption techniques to develop multiple privacy preserving scheduling protocols for different use-cases. In Chapter 3, we lay out an overview of some related works, these works inspire our research and act as the framework for our approach. In Chapter 4, we describe our assumptions, define the roles and ownership of various parties and variables, and describe our protocol. We present two protocols, one that requires a linear number of communication steps and one that achieves the same train schedule with a constant number of communication steps. In Chapter 5, we provide security analyses of the protocol. We present a theoretical and a practical analysis comparing the runtimes, communication costs, and throughputs of the two designs. In Chapter 6 we discuss the research questions and how our protocol helps answering these. We also discuss future work and limitations of the presented design.

2 Preliminaries

In this chapter, we provide an overview of the concepts of scheduling, and cryptographic techniques required to understand the protocols presented in this thesis, as well as to understand the related literature.

2.1. Periodic Event Scheduling

Since our primary focus is on periodic schedules due to their ubiquity and ease of understanding, we are dealing with the subset of scheduling algorithms which have been categorized as Periodic Event Scheduling. Periodic event scheduling is a method used to organize events that occur at regular intervals. This type of scheduling is commonly used in contexts where regularity and repetition are key. For instance, public transportation systems like buses and trains rely on periodic event scheduling to create timetables that passengers can rely on.

The main reason for using periodic event scheduling is to establish a predictable and efficient routine that optimizes resources and time. It helps in reducing conflicts and overlaps in schedules, ensuring smooth operations, and providing a clear structure for both service providers and users to follow. By having a periodic schedule, organizations can improve service reliability, enhance customer satisfaction, and manage resources more effectively. It's a crucial aspect of operational planning that contributes to the overall stability and predictability of services offered.

The algorithms found in the literature solve different parts of the overall scheduling problems. The complexity of scheduling problems can vary widely and can fall into different complexity classes, including P and NP. The complexity of the algorithms depends on the number and size of the constraints. Even though railway scheduling is a satisfaction problem, depending on the usecase, we can achieve optimality. The feasibility of achieving optimality depends on the complexity of the problem. The following are the complexities the related papers tackle:

- **P (Polynomial Time)**: Scheduling problems that belong to the P class are those for which a solution can be found in polynomial time.
- **NP (Non-deterministic Polynomial Time)**: Scheduling problems that belong to the NP class are those for which a proposed solution can be verified in polynomial time, even though finding a solution grows exponentially with increase in input size.

Algorithms Used

Based on the size of the subproblem in terms of the number of constraints and the domain of these constraints, various techniques can be used for scheduling. Here we provide the background on these techniques to better understand why researchers use these techniques and understand the change in techniques with increasing problem size and complexity.

2.1.1. Branch and Bound Algorithms

A branch and bound algorithm tries to break down a problem into smaller pieces by using a bounding function [15]. It then tries to eliminate these sub-problems which cannot contain the optimal solution. This paradigm can be used for mathematical optimization. This approach is methodical and explainable.

2.1.2. Greedy Algorithms

Greedy algorithms are a class of algorithms that make the best possible decision at each step, aiming for a locally optimal solution in the hope that these local optima will lead to a globally optimal solution [17]. These algorithms can be used for optimization problems, as long as one solution does not break the rest of the solutions. Greedy algorithms are easy to implement and fast to execute, making them great for purposes of rescheduling in case of disruptions in a train network.

2.1.3. Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm that enables an agent to learn optimal actions by interacting with an environment [55]. In RL, an agent takes actions to maximize cumulative rewards over time, guided by trial and error. RL creates its own solutions and iteratively gets better based on the response it gets from the problem upon implementing its solution. An RL definition consists of the state set, the action set, and the corresponding rewards at each time step needed. These states and rewards are then optimized to achieve domain-specific objectives. A policy corresponds to the probability of adopting a particular action from the action set to improve the current state. Improvement is measured by the reward function.

2.1.4. Constraint Optimization

Constraint optimization is a technique used to find the best solution from a set of possible solutions that satisfy a number of constraints [41, 69]. The core idea is to maximize or minimize a certain objective function, such as profit and distance, while adhering to the constraints present in the real world. One special advantage of this class of algorithms is the ease of improvement. An extra constraint doesn't rewrite the entire algorithm and improvements and extra constraints can thus be easily added to a pre-existing solution. A constraint optimization algorithm guarantees a solution, the quality of which can depend on the time the algorithm was allowed to run. This means that the system can produce good results in a time-constrained environment.

2.1.5. Linear Programming

Linear programming (LP) is a mathematical optimization technique used to find the best outcome in a given mathematical model, subject to certain linear constraints. It involves maximizing or minimizing a linear objective function, usually representing costs or demands, while satisfying a set of linear inequality or equality constraints. A typical LP problem takes the following shape:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{m} w_j x_j \\ \text{subject to} \quad & \sum_{j: e_i \in S_j} x_j \geq 1, \qquad i = 1, \ldots, n \\ & \qquad x_j \in \{0,1\}, \quad j = 1, \ldots, m \end{aligned}$$

Where x represents the various constraints and w the respective weights.

LP algorithms tend to converge relatively quickly, especially for small- to medium-sized problems, making them efficient for finding optimal solutions in a timely manner. By formulating the scheduling problem as a linear programming model and solving it using LP techniques, railway operators can make informed decisions to improve the overall performance and reliability of their systems. Moreover, advances in computer and chip technologies have been driven to accelerate LP on a hardware level [44].

2.1.6. Slot Allocation

Slot allocation algorithms are used to assign resources or time slots to various entities or activities. Slot allocation is used in distributed computing for parallel jobs in distributed computing with non-dedicated resources [73]. Research in the field is to have the least down time to utilize the resources optimally.

In transportation slot allocation is a type of traffic planning, of which the key characteristic is that infrastructure users have to reserve a ‘slot’ on the network before departure. The total number of users admitted to each bottleneck per period is limited, depending on its capacity.

Introduction to Airport Coordination by IATA [1] creates a tier list for airport and sets guidelines for airport slot allocation. For tier 3 airports where slot allocation is aggressively used to schedule airlines, the primary duty of the airport is to get the airport into a higher category through maintenance or expansion. The guidelines have been refreshed and reworked over two editions over 15 years, highlighting the importance of the technique for busy airports.

The main objective of slot allocation is to solve capacity conflicts beforehand in planning, and not on the network where congestion would be the result [5]. Slot allocation can be used as an instrument to serve these objectives, for instance, to stimulate competition in the transport market by giving priority to entrants in the traffic market or to reduce externalities of traffic by giving priority to environmentally friendly traffic.

2.2. Adversarial Behaviour

The adversarial behaviour can be described by what an adversary can do within the paradigm described by the protocol. We consider the adversaries commonly used in the literature; semi-honest, malicious and covert.

Semi-Honest

Also known as honest-but-curious, this form of adversarial behaviour doesn’t disrupt the protocol since the adversary is ‘honest’. However, they are also ‘curious’ to gain as much insight into the protocol as possible. This can come by keeping a mapping of all input, intermediate and output messages. Along with this, they might also keep a track of all the calculations across several runs of a protocol to gain insights into the inner workings of the protocol.

[1] https://www.iata.org/contentassets/4ede2aabfcc14a55919e468054d714fe/wasg-edition-3-english-version.pdf

Malicious

This is a stronger form of adversarial behavior. The adversary might deviate from the protocol. This can cause the protocol to malfunction and produce unexpected or incorrect results. To prevent this behaviour, protocols can install overheads to ensure validity of messages, calculations and results for every actor.

Covert

A covert adversary adversary lies somewhere in between a malicious and a semi-honest adversaries. Aumann et al. [3] argue how this is a more realistic type of adversary. Here, the adversary may act maliciously if they can benefit from this malicious act. It is essentially a trade-off between getting caught with some probability and what monetary/social advantages they can achieve with this malicious behavior, e.g., cheating in an online game can have exceeding benefits with low change of getting caught.

Capabilities

The capability of an adversary can be described by analyzing the computing power they have to try and break the protocol. These can be broadly divided into bounded and unbounded.

Bounded Adversary

A Bounded adversary possesses only a finite amount of computational power to compromise a cryptographic system. This means that the adversary can compromise a system with absolute certainty. This usually means being able to decode an encrypted message. A computationally secure protocol is resilient against such adversaries. Given no domain knowledge or information about the message, the effort required to crack a cryptographic system can be quantified as 2^{80} based on current computation speeds.

Unbounded Adversary

An unbounded adversary boasts infinite computational resources, necessitating a more stringent criterion for cryptographic security. A system that can withstand attacks from such adversaries is labeled **unconditionally secure** or **information-theoretically secure**. An unconditionally secure system ensures no information leakage, achieving what is known as **perfect secrecy**. The **One Time Pad** is an exemplary cryptographic system that achieves perfect secrecy [50]. A one time pad as the name suggests provides perfect secrecy by encrypting every single bit with a different one time use key. This key is generated randomly and changes with every run of a protocol using a one-time pad.

2.3. Cryptography building blocks

The system presented in this thesis and related work use public key cryptography, also known as asymmetric cryptography. This is because the system requires a pair of keys, one for locking and one for unlocking. Public and private keys are used to, respectively, encrypt and decrypt data. As a simple example, each user i generates their own key pair pk_i and sk_i. If Alice (A) now wants to send a message m to Bob (B), they require Bob's public key pk_B. Alice can then encrypts her message m under the encryption function E with input pk_B, resulting in the ciphertext c.

The intriguing aspect of public key cryptography lies in the dynamic between public and secret keys. This relationship operates as a one-way street, symbolized by the trapdoor function, where establishing the link in one direction is effortless, yet challenging in reverse. While the secret key generates the public key, the latter remains cryptic about its counterpart. This asymmetry, wherein deriving any insight into the secret key from the public key proves daunting, forms the bedrock of public key cryptography's security. Prime examples of such cryptographic challenges include factoring and the discrete logarithm problem.

The discrete logarithm problem solves a lot of concerns due to the underlying math: given a generator g of a group G with prime order q, determining x such that $g^x = y \pmod{q}$ for a specified y is difficult, despite the straightforward nature of the inverse operation through repeated multiplication. Based on the discrete logarithm, the decisional Diffie-Hellman assumption (DDH) is defined as the following: Given g^a, g^b for some random $a, b \in Z_q$ and suitable generator g in Group G with prime order q, one cannot distinguish between g^{ab} and g^c for random $c \in Z_q$.

Computationally Secure Pseudo-Random Number Generator

Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) are a class of Pseudo-Random Number Generators (PRNGs) that are suitable for use in cryptography due to their enhanced security properties [72]. They are designed to be secure against serious attacks, even when part of their initial or running state becomes available to an attacker [72]. CSPRNGs are essential in various cryptographic applications, such as generating nonces, initialization vectors, or cryptographic keying materials [72].

In the Rust programming language, the 'rand' crate provides several CSPRNGs [24]. For instance, 'StdRng' is a CSPRNG chosen for good performance and trust of security [24]. It uses the HC-128 algorithm, which is one of the recommendations by ECRYPT's eSTREAM project [24]. All RNGs in Rust implement the 'RngCore' trait, and secure RNGs may additionally implement the 'CryptoRng' trait [24].

In a multi-party setting, CSPRNGs can be used to create a shared symmetric key. Each party can generate a random number using a CSPRNG, and these numbers can be combined (for example, by XORing them together) to create a shared key that is known only to the parties involved. This key can then be used for secure communication between the parties [21].

2.4. Secure Multiparty Computation

Secure multiparty computation (MPC) is a cryptographic technique where two or more parties perform a joint computation that results in a meaningful output without disclosing the input provided by either party [75], [14]. In 1982, Yao [81] proposed the millionaire problem, which officially launched the study of secure multi-party computation, in which several participants with secret inputs work together to compute a function and get their own output, but none of the participants get any information about the inputs.

These techniques are being used more readily in various domains owing to the increase in value of data [1]. MPC enables computation on encrypted data since all parties only receive the output of a function while keeping the input data private [67], [14]. MPC is particularly useful in a distributed computing scenario where multiple parties(organizations or individuals) would like to cooperate by computing a function together and obtaining more valuable information without leaking their confidential data [82]. MPC can be applied to real-life problems using secret sharing, garbled circuits, and homomorphic encryption [64]. This thesis uses homomorphic encryption, but we will shortly outline all techniques to highlight the advantages of homomorphic encryption and why this technique is used.

2.4.1. Secret Sharing

Secret sharing hinges on dividing information into small fragments and giving ownership of these fragments to different parties. The combination of these fragments finally unlocks the

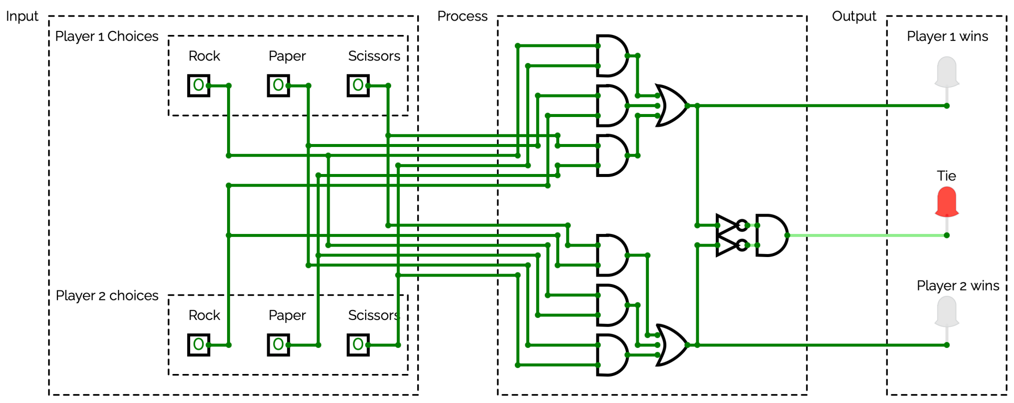


Figure 2.1: Circuit for Rock Paper Scissors

secret. Secret-sharing techniques are often denoted by (t, n), where n refers to the number of shares created and t refers to the minimum number of shares needed to recover the original secret. The idea is to divide the secret up so corruption of one party is not detrimental to privacy of others. To learn $f(x_1, ..., x_i)$, each party applies the function f on the received shares. Combining the shares yields the function applied to the inputs.

2.4.2. Garbled Circuits

Garbled circuit is a technique useful for a **two party system** where two parties want to compute a function over their private inputs without leaking information about their inputs [51]. Of the two parties, one has the circuit which represents the function they want to compute with the other party. We call these parties 'obfuscator' and 'evaluator'. This circuit is a set of OR, AND and XOR gates. Now this circuit is garbled, which in simple terms means encrypted such that it only reveals the correct results when decrypted using the correct key. Any mathematical function can be built into a garbled circuit. The only downside is that this process requires communication and works in a two party setting. The core idea of using bits and gates for complex functions is also utilized in the TFHE fully homomorphic encryption technique which we introduce later in this section is a core technique used in this thesis.

It is clear that the input can also not be in plaintext since inputs themselves are private. Therefore, the inputs are sent via Oblivious Transfer (OT). Let's take a game of rock/paper/scissors to understand this succinctly. There are exactly two inputs, which can take three values here. This is represented in a circuit as shown in Figure 2.1 [2].

Homomorphic Encryption

Homomorphic Encryption enables the execution of basic mathematical operations on encrypted values. The types of operations depends on the underlying math of a protocol. The use of this technique allows for computation over private data, and is thus majorly seen in industries that deal with private data such as medicine [54], big data [40] and biometrics [16]. We can further divide homomorphic encryption into two subtypes; Partially Homomorphic Encryption and Fully Homomorphic Encryption. Each type defines operations that can be applied to the ciphertext. As we go from Partially to Fully Homomorphic Encryption, these operations get stronger in terms of mathematical properties they offer, allowing for an array of problem solutions. The caveat is that as we go up with the type of operations allowed, the computational overhead increases. This can sometimes make the system infesible. This feasibility depends

[2]https://circuitverse.org/users/20039/projects/rock-paper-scissors-18ba61f1-e805-472d-97f9-10071b4bfe8a

on the computation power and the type of solution for which homomorphic encryption is utilized. One benefit of fully homomorphic encryption is the fewer messages sent between the parties [36], this helps with systems that usually have long computation times and otherwise will not require communication like federated learning [43]. This means that a similar security guarantee can be achieved for a protocol using both techniques, while differing in terms of computation and communication complexity.

2.5. Partial Homomorphic Encryption

Partial Homomorphic Encryption (PHE) schemes only support one homomorphic operation, such as addition or multiplication. Partial homomorphic schemes are thus additive or multiplicative homomorphic, based on their supported homomorphic operation.

Additive Homomorphism

Additive Homomorphic Encryption as the name suggests, only facilitates addition and subtraction of plaintexts by executing preset operations on two or more encrypted values [33].

Multiplication of two ciphertexts results in the addition of the underlying texts given that they are generated using the same public key

$$D_{sk}(E_{pk}(m_1) * E_{pk}(m_2)) = m_1 + m_2. \tag{2.1}$$

.

Similarly, the multiplicative inverse of the second message, results in the subtraction of the two messages

$$D_{sk}(E_{pk}(m_1) * E_{pk}(m_2)^{-1}) = m_1 - m_2. \tag{2.2}$$

.

Say $m_1 = m_2$, it becomes clear that we can treat this in ciphertext space as multiplying m_1 twice, which means this is twice the message in plaintext. If we extend this further, it becomes clear that any ciphertext raised to the power of k results in the encryption of $k \cdot m$

$$D_{sk}(E_{pk}(m)^k) = k \cdot m. \tag{2.3}$$

2.5.1. Paillier Encryption

Paillier encryption [62] is a commonly used additively homomorphic encryption scheme in multi-party computation (MPC) algorithms [28, 61]. The security of the Paillier cryptosystem is based on the hardness of the composite decision residuosity problem, which is assumed to be hard [62]. Formally, it is algorithmically hard (NP) to decide if there exists a y, such that for a composite n and integer z:

$$z = y^n (\text{mod} n^2) \tag{2.1}$$

Suppose that we want to encrypt a message m. Define $n = pq$ to be the product of two large prime numbers p and q of equal length $\frac{k}{2}$ which gives $k-$bit security. The public key is defined as (n, g) where $g \leftarrow n + 1$. This public key can be used to encrypt m to produce a ciphertext. The encryption function is defined as:

$$E_g : \mathbb{Z}_n \text{ x } \mathbb{Z}_n^* \rightarrow \mathbb{Z}_{n^2}^* \tag{2.2}$$

Here the message space is $Z^* = \{z \mid z \in \mathbb{Z}, 0 < z < N, \text{gcd}(z, N) = 1\}$, marking the hardness of breaking the cryptographic system without having knowledge of p or q. Give

$c = g^m r^n \bmod n^2$ where r is a random factor uniformly selected from $(0, n)$. To ensure that every encryption of the same message is different, a different r is sampled for each encryption. For decryption, we create the secret key using parameters μ and λ where $\lambda \leftarrow (p-1)(q-1)$ while $\mu \leftarrow \lambda^{-1}(\text{mod} n)$. It is clear that this requires knowledge of p and q which is a hard problem from the knowledge of just n which is what is available publicly. We thus get:

$$sk = (\lambda, \mu) \text{ for } pk = (n, g) \tag{2.3}$$

For the decryption process, we define function L as:

$$L(x) \leftarrow \frac{x-1}{n}; x \in (0, n^2) \tag{2.4}$$

From this definition, we can find m [62, 76]:

$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \mod n \tag{2.5}$$

2.5.2. DGK Encryption

We use the DGK cryptosystem [20, 19] to provide a secure comparison protocol for the Partially Homomorphic approach. For generating the public and the private keys, there are three parameters: k, t, and l, where l < t < k. The process of key generation is as follows.

1. Choose two distinct t-bit prime numbers v_p, v_q.
2. Construct two distinct prime numbers p and q, such that $n = pq$ is a k-bit RSA modulus.
3. Choose u, the smallest possible prime number but greater than $\ell + 2$.
4. Choose a random r that is a $2.5t$-bit integer [19].
5. Choose g and h such that ord$(g) = u^{v_p v_q}$ and ord$(h) = v_p v_q$.

From this, we obtain public key (pk)= (n, g, h, u) and secret key (sk) = (p, q, vp, vq)

$$c = E_{pk}(m, r) = g^m \cdot h^r \bmod n \forall m \in Z_u. \tag{2.1}$$

A lookup table can be used for decryption, so knowledge of domain is helpful.

2.5.3. Secure Comparison Protocol

Secure Multiparty Computation (SMC) allows a number of mutually distrustful parties to carry out a joint computation of a function of their inputs while preserving the privacy of the inputs by keeping them hidden from others. An important building block of SMC protocols is comparison. Although any function can be securely computed using generic circuit-based protocols, these protocols require communication steps for a setup procedure. Even though the computation steps in these protocols are faster than other approaches, the communication overheads can potentially slow them down and make them inefficient for a majority of real-world use cases. With the importance of comparison for many protocols, research has begun to accelerate comparison protocols. There exist many variants of this problem, depending on whether the comparison result is public or not, and whether n_A, n_B are known to particular players, or unknown to everyone [20]. However, protocols can be modified with minimal changes to be applied to any variant in terms of privacy.

We employ optimizations presented in [59] and [53] over the Secure Comparison Protocol devised in 2007 by Damgard et al. [20]. The sub-protocol uses the DGK cryptosystem for efficiency reasons in the original version, however improvements suggest use of ECEG to be better in [53]. For simplicity, we assume Alice to be the user with the values a and b, while Bob is the server owning the secret key used to run the protocol. The protocol proceeds as follows to return 1 if a>b and 0 otherwise.

2.5.4. Exponential Elliptic-curve-ElGamal

The use of Exponential Elliptic-curve-ElGamal (ECEG) is motivated by the research of Zahedani et al. [52]. The authors show how a comparison protocol utilizing ECEG speeds up the process of secure comparison. The messages are encoded in the exponent rather like the standard ElGamal [27]. The key generation, encryption, decryption, and zero-check operations make use of mathematical operations involving point addition and scalar multiplication on the elliptic curve. This makes exponentiation faster due to smaller key sizes for a similar level of security. However, a message must be encoded using a generator. The security of the scheme relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is the assumption that given a point P and a scalar multiple kP, it is computationally infeasible to determine k. Without this, breaking the discrete logarithm problem is the only option. Decoding the message is thus a hard problem without domain knowledge to narrow down the search area.

2.6. Fully Homomorphic Encryption

The first FUlly Homomorphic Encryption (FHE) scheme in 2009 [37], proved that FHE is achievable in polynomial time. FHE schemes are able to evaluate arbitrary circuits of unbounded depth and can therefore be used for running any program by an untrusted party. Unlike PHE techniques, FHE protocols allow for an unbounded number of mathematical operations on the ciphertext and allow for allow multiplication and addition over the same encryption scheme. This is achieved by Fully Homomorphic Encryption by the process of bootstrapping.

Development being recent in Fully Homomorphic Encryption, there isn't a set golden standard for the scheme to apply in all cases. For example, to perform non-linear operations using FHE, arithmetic circuits are used, which work on a bitwise level. The operations are thus carried out in terms of logic gates. The process from here is straightforward, just convert the input into binary values and convert all operations into a series of logic gates as discussed in Section 2.4. This leads to an overhead while performing non-linear operations, which are taken care of by different schemes in different ways. We will now briefly discuss bootstrapping, various FHE schemes, and their strengths and weaknesses.

Bootstrapping

All common FHE schemes are based on noisy encryptions (the noise guarantees the security of fresh encryption). Evaluating homomorphic operations increases the noise magnitude and lowers the quality. Bootstrapping converts an exhausted ciphertext into an “equivalent” fresh ciphertext. A bootstrapable HE can homomorphically evaluate its own decryption procedure in addition to at least one additional operation. An example of how bootstrapping works is given in Figure 2.2 [3]

2.6.1. BGV/BFV

For protocols requiring Single Instruction Multiple Data (SIMD) operations [71], use of earlier protocols such as BFV [29] and BGV [7] may be sufficient. SIMD is a parallel computing technique that enables a single instruction to be applied to multiple data elements simultaneously.

[3]https://dualitytech.com/blog/bootstrapping-in-fully-homomorphic-encryption-fhe/

Train station ts		**ProRail** pr
$[a], [b]$		sk
$L =$ Length binary representation		
$r \leftarrow\!\$\ \mathbb{Z}_{\kappa+L}$		
$[d] \leftarrow [2^L + a - b + r] = [2^L] \cdot [a] \cdot [b]^{-1} \cdot [r]$		
	$\xrightarrow{[d]}$	
		$d' \leftarrow d \pmod{2^L}, \hat{d} \leftarrow d/2^L$
	$\xleftarrow{[d'], [\hat{d}]}$	
$r' \leftarrow r \pmod{2^L}, \hat{r} \leftarrow r/2^L$		
		$[[t_i]] \leftarrow [[d'_i + \Sigma_{j=i+1}^{L-1} d'_j \cdot 2^j]]$
	$\xleftarrow{[[t]] \equiv [[t_0, ..., t_{L-1}]]}$	
$s \leftarrow\!\$\ \{-1, 1\}$ $[[v_i]] \leftarrow [[s - r'_i - \Sigma_{j=i+1}^{L-1} r'_j \cdot 2^j]]$ $[[c_i]] \leftarrow [[v_i]] \cdot [[t_i]]$ $h_i \leftarrow\!\$\ \mathbb{Z}_u^*$ $[[e_i]] \leftarrow [[c_i \cdot h_i]] = [[c_i]]^{h_i}$		
Scramble order of $[[e_i]]$	$\xrightarrow{[[e_i]]}$	
		Zero-Check all $[[e_i]]$ If none are zero, $\delta' \leftarrow 0$ else, $\delta' \leftarrow 1$
	$\xleftarrow{[\delta']}$	
$[\delta] \leftarrow [\delta']$ if $s = 1$ $[\delta] \leftarrow [1] \cdot [\delta']^{-1}$ if $s = -1$		
$[z_l] \leftarrow [\hat{d} - \hat{r} - \delta] = [\hat{d}] \cdot [\hat{r}]^{-1} \cdot [\delta]^{-1}$		
	$\xrightarrow{[z_l]}$	
		Decrypt $[z_l]$ and release result

Protocol 2.1: Secure comparison protocol by Zahedani et al. [52]. Determines whether $a < b$

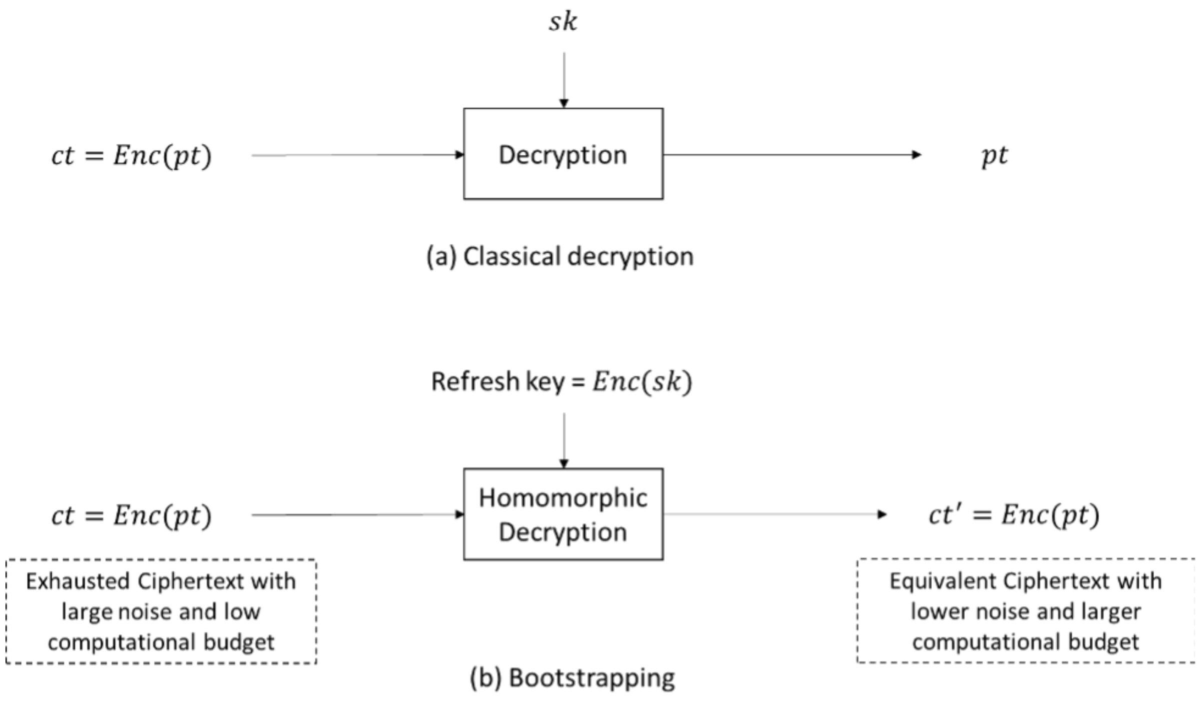


Figure 2.2: Bootstrapping

In FHE, SIMD operations allow for the efficient processing of multiple encrypted messages within a single ciphertext, leading to significant performance improvements in homomorphic computations. Packages multiple messages into a single ciphertext, enabling numerous operations to be performed in a single instruction. These protocols show good performance for the following usecases:

- Private information retrieval (PIR)
- Private set intersection (PSI)
- Integer computations

2.6.2. CKKS

CKKS is based on Approximate arithmetic on fixed-point numbers. It can thus support complex or real numbers, while allowing one to pack more bits into a ciphertext. CKKS, by definition is an approximate scheme and therefore cannot be used in applications which require high precision and is slower than alternatives. Based on these qualities, CKKS has good performance for the following usecases:

- Logistic regression training
- Statistical analysis

2.6.3. TFHE Encryption

TFHE [13] is an FHE scheme that performs binary operations. This speeds up its performance on non-linear operations with no such performance gain for addition or multiplication operations. The main selling point of TFHE is the bootstrapping operation, which allow for any bit to bit mapping operation while the bootstrapping operation continues, allowing for an overall larger throughput. TFHE supporting programmable bootstrapping is what makes it attractive for many different types of protocols that may involve inherently slower addition/multiplication steps as well.

The security of TFHE is affected by the underlying Learning With Error [68] problem which is hard to solve. This makes TFHE resistant against attacks from quantum computers. The letter 'T' in TFHE [68] refers to the real torus $T = R/Z$. Basically, T is the set $[0, 1)$ of real numbers modulo 1. Torus T is not a ring. If T were a ring, one would have $(a + b) * c = a * c + b * c$ and $a * (b + c) = a * b + a * c$, where $+$ and $*$ are defined over the torus (i.e., where $+$ and $*$ respectively stand for the addition and the multiplication over the real numbers modulo 1).

Encryption in TFHE required covering the plaintext data into an encoded unsigned integer. This process adds noise to the terms wherein the most significant bits represent the encoded value of an unsigned integer with noise added to the least significant bits for security. Like every Homomorphic Encryption technique, only a certain number of operations can be performed on TFHE ciphertext due to the noise introduced while encoding. The noise grows with every additive and multiplicative step. This error can potentially corrupt the underlying message, this is prevented by padding between the messages.

TFHE has a specific advantage of having programmable bootstrapping [12]. This process does not make the scheme faster, but allows a univariate function to be applied during the process of bootstrapping. Programmable bootstrapping allows for non-interactive secure comparison between integers [42]. We use this specific property for one of our improvements.

TFHE is a private-key encryption scheme. A private-key encryption scheme is symmetric: the same key is used for both encryption and decryption. This can be a bottleneck for multiparty computation protocols. However, any additively homomorphic private-key encryption scheme can be converted into a public-key encryption scheme. The public-key variant of private-key TGLWE encryption is obtained analogously. The TFHE library as developed in rust comes with primitives to extract a public key from the secret key; however, this impacts the performance of the protocol. Encryption becomes exponentially slower compared to using the private key. The use of a compact public key is able to speed up the process, however, the speeds at $\frac{1}{10}$ of using the client key (secret key), which, given the number of encryption steps and performance expectations, can become a bottleneck.

3 Related Work

In this chapter we discuss prior works in railway scheduling and demand forecasting. We further discuss various cryptographic techniques developed for homomorhphic addition and secure comparison protocol.

3.1. Railway Scheduling

Railway scheduling is a well-studied problem. With increasing network complexity and increasing infrastructure constraints, it is also an evolving problem [80]. There has been a great deal of systemization of knowledge papers written in the domain. Narayanaswami et al. [58] conducted one such analysis in 2011 where they analyzed paper scheduling techniques to schedule and reschedule railway timetables. Analyzing most of the techniques, the authors conclude that schedules take care of two possibly competing goals; Improving public utility and profits [58]. Strategic operations are handled by railway policy makers, including energy efficiency, capacity planning, pricing, type and size of service requirements, layout, and route planning. Scheduling is directed towards achieving specific objectives while maintaining the feasibility of safety and preferential constraints set by policy makers. Note some authors use timetabling for what we define as scheduling. These terms are used interchangeably in this section to abide by the terminologies the authors use in their respective work. The research started with methods like branch and bound and greedy algorithms, but, with the explosion in the number of constraints and the increase in the size of the problems, these techniques were quickly rendered insufficient. With increase in compute power and better understanding of the problem, many problems like line-solving, timetabling and crew assignment were combined into one for a more holistic approach. This leads to the use of contemporary methods like constraint optimization, linear programming, machine learning, and slot allocation.

3.1.1. Historical Approaches

Some earlier approaches to generate schedules were branch and bound algorithm [18] and greedy algorithm [9]. These techniques were used mainly for real-time conflict resolution problem, to find a conflict-free schedule compatible with the real-time status of the network. These approaches were used to design schedules so that secondary delays could be avoided. The authors do this by creating zones that can be occupied by one vehicle, including but not limited to platforms and signalled areas. Currently, a similar constraint is still in place when scheduling trains.

Cai et el. [9] aimed to solve the problem of trains sharing the same track while having the necessary overtaking to be achieved at certain points on parallel tracks. The authors try to solve this problem by making some simplification assumptions. This includes trains going at the same speed and most importantly trains going in only one direction, which means overtaking can be performed at any moment a parallel rail section is available. The authors acknowledged these assumptions and claim the problem being in NP means that their solutions will not be globally optimum.

Branch-and-bound and greedy methods were viable approaches due to the smaller size of the networks and fewer conflicts. In the greedy method, the idea was to minimize or maximize the time of current vehicle by allowing it to overtake everyone in their way and follow the same process for every subsequent vehicle or stop the train that is causing conflicts and let the rest of the network flow unstopped. The global solution can be a middle ground here, but finding that would be a hard problem as can be understood from its complexity Section 2.1.

3.1.2. Constraint Optimization Approach

Constraint optimization problems depend on the solution of multiple constraints to maximize or minimize an objective function or a set of objective functions. The best schedule serves the maximum number of passengers. Wang et al. [78] propose a travel demand space-decomposition algorithm to break these routes down into smaller sectors. The minimum amount of train service is described by:

$$S_k = \frac{Q_k}{C(1 - L_k)}$$

where Q_k is the number of passengers served by the station k in one day, and C is the capacity of a train. L_k is the average capacity utilization of the trains arriving at the station k. The total initial capacity being served by the trains is more than the total demand. Note that a train serves more than its capacity since some people de-board and are replaced by others during the journey.

After breaking down the problem into multiple parallel smaller problems, the authors aim to optimize the sub-timetables by **minimizing the total travel time**. It takes into account factors such as train frequency, passenger demand, and operational efficiency.

To combine all sub-problems, the overall passenger demand is maximized. The constraint is the chosen subroutes and their compatibility with other subroutes.

$$C_{\text{service}} : \sum_{i=1}^{m} p_i \cdot t_i \to \text{min}, \quad \text{where } p_i \text{ is passenger demand and } t_i \text{ is travel time}$$

- m: The total number of passenger groups or demand points.
- p_i: The passenger demand for group i.
- t_i: The travel time for passenger group i.

3.1.3. Mixed Integer Linear Programming

Mixed Integer Linear Programming (MILP) is a common train timetable scheduling technique. Given the nature of the problem, the reduction of constraints or improved solving methods is where current research in MILP is. Zhou et al. [83] use a MILP model with the aim of minimizing the total travel time of the trains on the network. The authors use multi-periodic train timetabling and combine it with routing or line planning. This is justified since distance can play an important role in the period of a train. The system can only be made periodic by taking a common multiple of these period, which may not be compatible with the demand that is expected from all the locations served by the schedule. They assume some predetermined trains of each period type on a high-speed railway network to simulate a train station that serves high-speed trains for different long distance destinations, e.g., Amsterdam Centraal.

The authors minimize the travel time of all trains by considering 22 constraints. These constraints can broadly be divided into subcategories of line planning and timetabling.

Timetabling Constraints

Definition 3.1.1 (Platform Availability) *Ensures that there is a platform available when a train arrives at the station.*

$$\text{PlatformAvailable}(tr, ti) = 1; \tag{3.1}$$

The 1 denotes that the platform is available for train tr at time ti

Definition 3.1.2 (Platform Conflict) *Prevents two trains from using the same platform at the same time.*

$$\text{PlatformConflict}(tr_1, tr_2, ti) = 0 \tag{3.2}$$

Platform conflict takes value 1 when for any pair of trains; tr_1 and tr_2, the time at the platform coincides. This should not be the case for any train pairs.

Definition 3.1.3 (Minimum Dwell Time) *Ensures that trains have enough time at a station for passengers to board and alight.*

$$\text{DepartureTime}(tr) - \text{ArrivalTime}(tr) \geq \text{MinimumDwellTime} \tag{3.3}$$

Definition 3.1.4 (Headway Constraint) *Maintains a minimum time gap between consecutive trains to ensure safety.*

$$\text{DepartureTime}(tr_{i+1}) - \text{DepartureTime}(tr_i) \geq \text{MinimumHeadway} \tag{3.4}$$

Definition 3.1.5 (Connection constraint) *Allows sufficient time for passengers to transfer between trains.*

$$\text{DepartureTime}(connecting\_tr) - \text{ArrivalTime}(tr) \geq \text{TransferTime} \tag{3.5}$$

Definition 3.1.6 (Capacity constraint) *Ensures that the number of trains scheduled does not exceed the station's capacity.*

$$\sum_{\text{all trains}} \text{TrainAtStation}(tr, ti) \leq \text{StationCapacity} \tag{3.6}$$

3.1.4. Merging Timetables

Lindner et al. [46] develop two MILP based strategies to merge two valid timetables into an overall better timetable. They do so under the realm of PERPlib, which is a popular library used to develop solutions to periodic event scheduling problems (PESPs). The authors study the heuristics of a created timetable to understand how to run a merging procedure without destroying the guarantees offered by either timetable. The two methods are the following.

- Utilization of Instance Structure: The authors look for patterns or regularities within the timetables that can be combined without violating the periodicity and connectivity requirements of the system.

- Minimum Weight Cycle Bases: This concept uses graph theory. A cycle base is a set of cycles from which any other cycle in the graph can be formed. The minimum weight cycle base is the one where the sum of the weights of its cycles is minimized. In PESP, this translates to finding a set of constraints that, when satisfied, ensure the feasibility of the timetable.

Both techniques rely on the underlying schedules to satisfy certain basic requirements, these underlying schedules can be created in anyway, but a MILP based technique is preferable since it gives guarantees of safety constraints being satisfied. The authors state some disadvantages of the technique being a very high computation cost. It is exponential and this makes it infeasible for real world implementations. If the underlying timetables are close to a local optima, the merged timetable is also likely to be stuck in a local optimum. The silver lining is that the merged timetable cannot be worse than the underlying timetables, hence by providing equal or strictly better results than before.

3.1.5. Reinforcement Learning

Another technique that is recently being applied to railway scheduling is Machine Learning, more specifically Reinforcement Learning. Yiwei Guo, in his paper using Reinforcement Learning to schedule railway timetables [39], considers factors that can be classified into 6 sub categories:

- Passenger demand related factors: train origin destination (OD) pair; the commencing and terminating times of a given train
- Rolling stock related factors: rolling stock circulation; rolling stock maintenance; etc.
- In-station tasks related factors: train routing and platforming; waste suction and water supply; crew changeovers; etc.
- Safety related factors: minimum running time; minimum headway; running time supplement; etc.
- Capacity related factors: efficient capacity allocation within busy hub area; etc.
- Other factors: reliability; punctuality; robustness; etc.

The paper defines a state as a possible time-space distribution of all the trains considered, which can be conceived of as an unfinished train path diagram. There are three categories of actions taken into account: parallel move, stop time adjustment, and swap.

The paper uses these action state pairs to try and solve feasibility (i.e. conflict-free) and evenness (i.e. an evenly-distributed train path diagram) in the train network. The reward function developed, simultaneously considers these two objectives functions.

Based on this, the RL formulates a route towards a better state and simulates a better route that improves the train schedule.

3.1.6. Slot Allocation

Slot allocation has been used for rail scheduling and for other transportation bottlenecks [45, 6]. The usecase of Airways is a direct parallel with the increase in the number of private railway services. Given the EU objective of equal access to infrastructure networks and the increasing scarcity of infrastructure capacity, the propagation of fairness and efficiency of slot allocation in the rail and aviation sectors has become a major issue. The allocation of airport slots as described in Section 2.1.6 is being criticized for discriminating against entrants and for not stimulating efficient use of airport slots. It gives priority to existing flight operators and routes, and hence hampers the introduction of new flights by new competitors. The author in [45] describe the role of all the actors in a railway scheduling system.

- **Operators**: The operator has the role of transport service provider and transport service producer, which means that operators are also responsible for balancing transport supply with traffic demand, they try to maximize the demand served. An allocation body is responsible for slot allocation, i.e. it has the role of traffic service provider in situations that slot allocation is applied.
- **ProRail**: A traffic controller has both the roles of traffic manager and infrastructure operator, implying that traffic controllers can directly implement the consequences of traffic management measures. In the Netherlands ProRail/Railned is responsible for slot allocation, they perform traffic management and operate the infrastructure.

The authors in [6] take an approach of slot allocation via auctions. Unlike the current process capacities are allocated to the most valuable uses, and not by inherited 'rights' or by simple priority rules. The authors criticize auctioning of slots independently of the rest of the network. They claim that allocating a slot of impacts on other slots as well, so just allocating a slot to an operator with maximum demand is not the ideal approach since this can harm locations that are already poorly served. They introduce the concept of combinatorial auction that allocates a multitude of interdependent slots simultaneously. For this, the authors propose how only a partial portion of the network should be auctioned at once. This is because optimization is a hard problem, and the problem size exponentially increases the complexity. With partial optimization, not only the problem becomes solvable, but incrementally making the schedule better over time would improve it throughout. The authors solve this using a graph which denotes all the arcs in the network and their capacities that denote the number of trains that can occupy an arc simultaneously.

The slot allocation is based on a bidding system. This bid is based on actual money that an operator is ready to pay for a slot. A slot request must specify (at least): a monetary base bid, a train type, a route, and time-value specifications. The monetary base bid is a sum of Euros the operator is willing to pay for the slot s, aside any additions or reduction. The actual choice of the selected operator is then made based on their other bids and other operators' bids. The combinatorial way of choosing the winning bids is based on how it would impact the rest of the network. This is seen by other trains serving a station and how that would impact the ridership of the train being scheduled. Similarly the operators can place bids related to each other, e.g., train from Leiden to Amsterdam, just after a train from Delft to Leiden, so passengers can make easy transfers to get to their final destinations. These complex bids are called 'Tours' and are treated as either entirely going through or not at all. So, either there is a train from Delft to Leiden to Amsterdam or neither of the two. This is implemented since the importance of a route in this case can inextricably linked with the other route as part of the tour. The problem of line management is solved using Linear Programming in this research.

3.2. Privacy Preserving Approaches

Based on literature review, there are limited protocols that schedule railway schedules for multiple parties [48, 46]. These approaches take a centralized approach to the problem which fails to preserve the privacy of sensitive information like 'demand'.

We describe the approaches present in literature that preserve privacy of sensitive information, either in a multi-party setting or a centralized approach. A centralized approach remains similar irrespective of the underlying algorithm. The idea is to send encrypted data to a central location. This central location utilizes homomorphic operation to optimize the schedule. Since the operations are conducted over encrypted data, the formed schedule is encrypted. The final schedule can only be revealed upon decryption. This decryption process can then be carried out in a way that continues hiding the sensitive information while revealing the underlying schedule. This can be done in multiple ways that offer different levels of trade-off between privacy, computation and communication.

We describe the approaches found in literature that can assist with privacy preserving scheduling. These approaches are generalized and apply to the underlying algorithms like Linear Programming, Constraint Optimization and Slot Allocation.

3.2.1. Linear Programming

In his paper titled 'A Privacy-Aware Distributed Approach for Loosely Coupled Mixed Integer Linear Programming Problems' [31] author, Mohammad Javad Feizollahi proposes two exact distributed algorithms to solve MILP problems with multiple agents where data privacy is important for the agents. Both approaches, rely on the MILP problem being loosely coupled. This means that the problem can be broken down into subsystems which can be solved independently with only a few constraints joining the sub-problems together. This means that not all constraints need to be solved centrally or by the same party. e.g., consider the following optimization problem;

$$max(x+y) \tag{3.1}$$

Say there were only 2 constraints to solve this objective function;

$$x \leq 10, x \in \mathbb{N} \tag{3.2}$$

$$y \geq y^2 - 10, y \in \mathbb{N} \tag{3.3}$$

If x and y were owned by different entities, entity_x could solve 3.2 and entity_y could solve 3.3 independently without revealing the domain of their values. They could just reveal the final values of x and y to solve for 3.1. This can be an iterative process if the objective function is unknown or private.

To distribute the MILP into these sub-parts, the author uses the concept of primal cuts to achieve a loosely coupled MILP. Primal cuts are constraints added to the MILP problem to exclude certain solutions that are not optimal. By using primal cuts, each agent can solve a part of the MILP problem without needing to share its private data with others. The other method proposed by the author is to apply Lagrangian Relaxation which is the process of associating a penalty cost to simplify constraints. The Lagrangian Relaxation is applied to constraints that require private information, the idea is to solve easier constraints first in order to narrow down the domain of sensitive information and be able to replicate the central problem with these constrained values. Both algorithms are designed to ensure finite convergence for MILPs that include only binary and continuous variables. This means that they are guaranteed to find an optimal solution in a finite number of steps. This approach is interactive and requires the parties to be online and in constant communication to reduce the size of the problem and realize the objective function jointly.

To apply this technique to the state-of-the-art MILP algorithm as introduced in [83], we need to find out if the algorithm is loosely coupled and if the constraints can be made continuous or binary. Based on the constraints defined in Section 3.1.3, other than Equation 3.6, every other constraint is binary or continuous.

$$\sum_{\text{all trains}} \text{TrainAtStation}(tr, ti) \leq \text{StationCapacity}$$

Since the aim of the research in [83] is to minimize travel time, we still need to introduce the concept of demand which we want to maximize. Since demand for all participants is private to them, the objective function cannot be independently evaluated. The demand of all operators determine the final schedule, which means that this needs to be done at a centralized location. Relaxation and primal cuts are expensive operations. Primal cuts can be achieved in the following ways:

- **Creating Cuts**: To get rid of these non-optimal solutions, we create additional constraints, called primal cuts. These cuts are like precise lines drawn to exclude certain areas from consideration.
- **Iterative Process**: We might need to add multiple primal cuts iteratively. Each time we add a cut, we solve the problem again to see if we can find a better solution.
- **Mathematical Operations**: The creation of primal cuts involves various mathematical operations not limited addition or multiplication; it's about analyzing the problem, understanding the geometry of the feasible region, and then strategically excluding parts of it.

Lagrangian relaxation requires the following steps:

- **Penalty Multiplication**: In Lagrangian relaxation, penalties (multipliers) are multiplied with the constraints that are being relaxed to incorporate them into the objective function.
- **Addition/Subtraction**: The relaxed constraints are added to or subtracted from the objective function, altering the problem's landscape.
- **Iterative Adjustments**: The process often involves iterative methods that adjust the multipliers and the solution, which can include a range of operations depending on the specific algorithm used.

3.2.2. Constraint Optimization

The authors in [38] introduce P Sync-BB. Sync-BB is an ascynchronous branch-and-bound algorithm. Wherein the P in this stands for privacy preserving. When considering a version of SyncBB that preserves constraint privacy, one must pay special attention to the upper bound. In SyncBB, the upper bound is the most fundamental piece of information during the problem solving process, and it is publicly known to all agents. The effectiveness of the algorithm lies in the continuous comparisons of the costs of partial assignments with the current upper bound, in order to prune the search space.

The authors introduce the concept of **secure comparison protocol**s to hide the upper bound and to check the assignment cost against this value in a way that preserves privacy. This is a major source of trouble from the perspective of constraint privacy. This separation is achieved by preventing any agent at any stage from knowing both the upper bound and the current partial assignment (CPA).

The process begins with the first agent, A_1, who sets the initial limit, called the upper bound, to a very high value, which is infinity. This is to ensure that no potential solution is overlooked from the start. A_1 also activates a status indicator, known as ComputedCPA, to signify that it has calculated the cost of the partial solution it has received. This step is about setting up the correct status for future stages.

Next, A_1 proceeds to the task of assigning values to its part of the problem, which is referred to as the assign CPA procedure. Given that there's no need to reconsider previous choices at this initial stage (a process known as backtracking), A_1 selects a value for its variable from the available options and updates the partial solution with this value. Then, A_1 sends this updated partial solution, now containing A_1's contribution, to the next agent, A_2.

There's a checkpoint in the algorithm where an agent assesses if the cost associated with the current partial solution is approaching the best current solution, which would mean that exploring this path further is unnecessary. However, since A_1 does not have a preceding agent to receive such a cost from, it skips this evaluation. For confidentiality purposes, it might be decided that the following agents, A_2 and A_3, should also bypass this step.

3.2.3. Slot Allocation

In the study "Privacy-Preserving Implementation of an Auction Mechanism for ATFM Slot Swapping" by Feichtenschlager et al. [30], the authors introduce a genetic algorithm-based method to allocate Air Traffic Flow Management (ATFM) slots to aircraft. This method is designed to address the assignment problem by pairing airplanes with their departure and arrival slots according to the bids submitted.

The bidding process involves airlines providing encrypted bids that include their preferred departure and arrival times, along with the maximum delay or the earliest they can tolerate. To determine the value of each slot, airlines also submit an 'importance value', which reflects the flight's duration and significance. For instance, larger operators often have flights with numerous connecting passengers, and the importance value is adjusted to account for the higher costs associated with delays for these flights. This value of importance, crucial for competitive reasons, is kept confidential between competing airlines and scheduling authorities to ensure the integrity of the system, even in the presence of a potentially malicious authority.

The system's goal is to minimize the overall cost resulting from deviations from the preferred slots. This is achieved by establishing a continuous importance value, where costs linearly increase as the actual schedule diverges from the preferred timing.

To maintain privacy and security, the authors employ a secret-sharing-based system. This protocol is particularly effective in scenarios where the scheduling authority might be compromised, ensuring that the airlines' data remains secure. The paper, however, does not provide explicit details on the workings of the system nor does it offer a comprehensive security and privacy analysis. The authors refer the reader to a different paper for details of the cryptographic solution used [48].

The authors here claim to use some form of secret sharing technique but do not explicitly state how the technique is applied, who has ownership of what information. They claim that the system leaks 'some' [48] information but this is not defined. Overall, the results show that the method is applicable and can be used for slot allocation in a privacy preserving way, however, analysis of the code provided on their GitHub [1] provided in the paper does not reveal much information other than the fact that they use secret sharing which supports secure $m-$party computation tolerating a dishonest minority of up to t passively corrupt parties, where $m \geq 1$ and $0 \leq t < m/2$.

[1]https://github.com/lschoe/mpyc

4 Privacy Preserving Railway Scheduling

In this chapter, we present the design of our protocol for Privacy Preserving Railway Scheduling. The goal of the protocol is to facilitate an ecosystem in which railway demand data can be shared in a privacy-preserving way to create optimized Railway Schedules at International Train Stations without leaking the demand to any participant of the protocol. We aim to solve this problem for International Stations, since they, by definition, have multiple operators serving the passengers and are not owned by a single operator. It is important to have multiple independent operators with possibly competing goals to enter the protocol, which is a requirement satisfied by international stations which are served by atleast two nationalized or private rail operators e.g., Amsterdam Centraal. We do not need a trusted party, as long as none of the participants is malicious. We will describe the stakeholders, assumptions, and sub-protocols that compose our protocol.

4.1. Roles

The participants in the protocol are entities present in the real world. Since railway scheduling is not a multiparty problem currently, the entities described in Section 2.1 are lacking. Existence of just one party apart from the operators means that the system of checks and balances required in a multi-party approach is missing. Apart from the parties discussed (Operators and ProRail), we introduce roles for the party 'Railway Station', this is done in order to distribute duties and thereby strengthen the protocol against collusion.

Operators

Operators are stakeholders who want their trains to be scheduled on a track at a train station. Our scheme assumes that less than half the operators are malicious, a class of adversaries discussed in Section 2.2. In our design, a malicious operator can record every message they receive. They can also deviate from the protocol run and incorrectly compute the values they are responsible for. This scenario with $< 50\%$ malicious operators should not give information to any operator to determine anyone else's private 'demands' or manipulate the protocol to select certain slots. In some specific cases discussed in Chapter 5, the protocol can leak information about the ordering of specific slot 'demands' between specific operators, however, this information is not sufficient to infer anything else about another operator's demands. The operators are the owners of sensitive data 'demand'. The 'demand' denotes the number of passengers they predict would want to take a train from the current location to the destination the operator is trying to serve in the current protocol run. The operators should submit the correct demands as predicted by their estimation models. There is no way to verify if the demands they submit are correct, since this information is sensitive and only available to the operators themselves. There is a way to check the plaintext value of their demand by decrypting it using the secret key, however, it is still not possible to reveal if this decrypted value is correct.

Network Operator

A Network Operator is responsible for scheduling a rail network, this includes arrivals, departures, and times spent at different segments of the rail network. This party is the owner of the system and create the keys used during the protocol run. They are the sole owner of the secret key and publish the public key and server keys (in case of FHE) to the relavent parties. This task is undertaken by ProRail in the Netherlands. For the purposes of this protocol, we will refer to this party as ProRail for the protocol description, since this gives an on-ground picture for the rail system in the Netherlands. We assume the network operator to be semi-honest. In essence, it does not make much sense for network operator to deviate from the protocol since it is in their interest to have the best train schedule for their network.

Railway Station

The railway station which hosts the trains to be scheduled is responsible for serving the maximum passengers it can within any period of time. The railway stations in the Netherlands are owned by the NS, however, the trainlines are not, so, it is ideal for a train station to have maximum passengers visit irrespective of the operator. We aim to solve the problem of multi-operator scheduling for international train stations in particular, since the arrival and departure slots here are of higher value to companies and thus warrant expensive operations for accurate scheduling. We assume the railway station to be semi-honest. The railway station can also abort the protocol if they detect that any operator acted maliciously while creating schedules. The railway station helps the network operator to find the best schedule by entering the secure comparison protocol.

4.2. Design

The purpose of this protocol is to allow multiple parties to schedule their trains on a shared infrastructure of the rail platform. The process followed is inspired by various scheduling protocol discussed in Section 3.1. The novelty of this solution comes from being able to make this schedule without leaking values private to various parties. Various demand prediction models are used by the participating parties to find an approximate value which is used for scheduling. The protocol can be divided into the following independent steps. The initial setup phase requires the operators to predict demands for the route they plan on scheduling. This is an internal step since the operators have their own private methods to approximate this value. The rest of the protocol is carried out in a multi-party setting. The description of all the steps in the protocol are introduced in this section. We mention the TFHE specific improvement where it differs from the PHE and other FHE based approaches that do not allow non-interactive secure comparison designs.

1. Demand forecasting
2. Sharing encrypted demand
3. Creating schedules
4. Computing the best schedule
5. Publishing the schedule

4.2.1. Demand Forecasting

Demand forecasting is important for the correct planning of the supply of the service according to passenger demand. Expert system models are built up of rules for demand forecasting based on the knowledge of a human expert. It is extremely difficult to transform the knowledge of an expert to mathematical rules [25]. Nar et al. [57] in their paper hold a two stage plan to tackle the problem of accurate demand forecasting.

In the first stage of the study, passenger demand forecasting is made with statistical techniques such as regression analysis and simple average. In the second stage of the study, the forecasting of passenger demand is performed with the artificial neural network and the machine learning (ML) algorithms technique on the station basis. They finally compare the results of these techniques by comparing the predicted values to actual passenger demands and conducting other common statistical tests. The results obtained using machine learning techniques were found to be more accurate. Specifically, one uses a Decision Tree. The study forecasts the demand for 2020 using the 2019 travel data for the Yenikap M1-Kirazl M1 line. ML Techniques are applied over station data from 2019, this includes the number of check-ins and check-outs at the station. Information provided to the authors as open data.

It is noteworthy that this data is seen based on trains already plying on routes and doesn't consider any new schedules. Nederlandse Spoorwegen (NS) which is the biggest train operator in the Netherlands published the way they forecast demand [77]. To estimate these values, the authors and, by proxy, NS use the following:

- The Exogenous Scenario Module uses external input for demographic, economic and car-related variables to create a year-by-year dataset on the municipal level.
- A Station Assignment Module uses the municipality data set to create a station-level dataset of these same variables.
- An Elasticity Module uses growth factors for scenario and timetable developments to forecast demand volumes at OD-level, separately for each of six travel purposes.
- A final module forecasts demand volumes for newly opened stations: the New Stations Module

The plaintext data for these datasets is not made available, and is private information available only to regulated authorities. Another operator, Eurostar, which serves international train stations such as Rotterdam, Amsterdam and Schiphol Airport in the Netherlands, uses the Passenger Demand Forecasting Handbook (PDFH) [66] for scheduling purposes. PDFH is a source of evidence and guidance in rail demand forecasting and is provided to members of the Rail Delivery Group. This source is not publicly accessible. This handbook summarizes over 20 years of research on rail demand forecasting and provides guidance on various aspects such as the effects of service quality, fares, and external factors on rail demand. This private source is used by railway operators to make strategic decisions like:

- Investment appraisal
- Pricing decisions
- Timetabling and operating decisions
- Business planning and budgeting

The forecasted demand is used by operators to estimate the number of trains they need to schedule on a line and find out the optimum arrival and departure times at train stations. The number of passengers not only determines the number of trains that should serve a station but also the duration for which the train should be stationary at a platform for passengers to deboard and alight the train. This is an important safety concern, since the passengers should feel safe while waiting to board a train. Schedule creation allows only for the demands values that satisfy the underlying security requirements.

4.2.2. Sharing Encrypted Demand

The private demands are encrypted but can still only be shared amongst different operators for creation of schedules. The railway station should not have access to these demands because this can lead to a leakage in demand orders. e.g., in a two slot setting, if the railway station through the comparison protocol finds out that $[a_1 + b_2] > [a_1 + a_2]$, then the railway station knows that $b_2 > a_2$. This can be prevented by not revealing the encrypted demands to the railway station since they don't need these. They are only concerned with the formed schedules that do not leak the underlying individual demands. The schedules need to be formed by all operators to ensure an equal distribution of power. This is the first multi-party step of the protocol. The operators need to enrol into the system by running a sub-protocol where they can share their encrypted demands with each other. There is a one-time setup process in which the operators come up with a shared symmetric key. The process is described in Protocol 4.1. The symbols used in the protocol and used later for the analysis are explained in Table 4.1. Note that the aggregated demands ζ_i that are broadcasted by the operators at the end of the secure exchange sub-protocol, must be identical for all operators. It is possible that they are not in the same order, but all the values must be the same. Since these schedules are published, anyone can spot differences. The train station aborts the protocol if all schedules do not match up. This implies that atleast one operator acted maliciously.

Table 4.1: Table of symbols and explanations

Symbol	Explanation
Sl	No. of slots
Pa	No. of parties (Operators)
pk, sk	Public & Secret key pair for the protocol run
T_i	No. of trains for operator i
k	Total No. of slots
$seed_i$	PRNG seed of operator i
D_i	Demand vector of operator i
$d_{i_{sl}}$	Demand of operator i for slot sl
$[d_{i_{sl}}]$	Paillier encryption of $d_{i_{sl}}$
ζ_i	Schedule vector created by operator i

f maps the schedule to the underlying encrypted demand The operators have their private demands that they encrypt to protect privacy. The operators enter a setup phase. During this phase, the operators acquire a public key from ProRail. The operators also enter a key generation protocol amongst themselves. This step is used to create a shared key for the operators to enter subsequent protocol run to schedule trains on their other routes. The operators choose a computationally secure Pseudorandom Number Generator (PRNG) like the 'rand' crate in Rust. We use this crate in our implementation for all randomness. Every operator independently comes up with a seed $seed_i$ for every operator i and share this seed with every other operator. All the operators now form a shared symmetric key. In a two operator setting; $pa_{key} = seed_a \oplus seed_b$. Only the operators have access to this shared symmetric key. When an

Operator pa_a	PRNG function	**Operator** pa_b
$(T_A, seed_a)$		$(T_B, seed_b)$
(pk)	$T_A + T_B \geq k$	(pk)
$D_A = \{d_{a_1}, ..., d_{a_k}\}$, where $d_i \in \mathbb{N}$		$D_B = \{d_{b_1}, ..., d_{b_k}\}$, where $d_i \in \mathbb{N}$
$(E_{pk}(d_{a_1}), ..., E_{pk}(d_{a_k})) =$		$(E_{pk}(d_{b_1}), ..., E_{pk}(d_{b_k})) =$
$[d_{a_1}], [d_{a_2}], ..., [d_{a_k}]$		$[d_{b_1}], [d_{b_2}], ..., [d_{b_k}]$
	$\xrightarrow{T_A, seed_a}$	
	$\xleftarrow{T_B, seed_b}$	
$\alpha_{key} \leftarrow PRNG(seed_a) \oplus PRNG(seed_b)$		$\alpha_{key} \leftarrow PRNG(seed_a) \oplus PRNG(seed_b)$
Broadcast$(D_A \oplus \alpha_{key})$		**Broadcast**$(D_B \oplus \alpha_{key})$
$\zeta_a \leftarrow \{(x_1 + \ldots + x_k) \mid x_i \in \{d_{a_i}, d_{b_i}\}$		$\zeta_b \leftarrow \{(x_1 + \ldots + x_k) \mid x_i \in \{d_{a_i}, d_{b_i}\},$
$x_i \neq x_j$ for $i \neq j,$		$x_i \neq x_j$ for $i \neq j,$
$s.t. \lvert\{x_i : x_i \in D_A\}\rvert \leq T_a,$		$s.t. \lvert\{x_i : x_i \in D_A\}\rvert \leq T_a,$
$\lvert\{x_i : x_i \in D_A\}\rvert \leq T_b\}$		$\lvert\{x_i : x_i \in D_A\}\rvert \leq T_b\}$
$\implies \lvert\zeta_a\rvert \leq 2^k$		$\implies \lvert\zeta_b\rvert \leq 2^k$
$f : \{(x_1, \ldots, x_k) \mid x_i \in \{a_i, b_i\}, x_i \neq x_j$		$f : \{(x_1, \ldots, x_k) \mid x_i \in \{a_i, b_i\}, x_i \neq x_j$
for $i \neq j\} \rightarrow \zeta_a$		for $i \neq j\} \rightarrow \zeta_b$
Broadcast ζ_a		**Broadcast** ζ_b

Protocol 4.1: Secure exchange of Demand in a two operator setting

operator XORs their demand with this key, the demand is encrypted and can only by accessed by other operators. These double encrypted demands are broadcasted. The operators gain access to the public key encrypted demand of every other operator. This combined with the information on the number of trains each operator has, all possible schedules are created by every operator. These schedules are broadcasted for the train station to verify their validity and enter a secure comparison protocol with ProRail.

4.2.3. Creating Schedules

The schedule can be created using any of the methods discussed in Section 3.1, however, for our usecase, some approaches are simply more robust than others. The train schedule must follow some safety guidelines that can be neatly written as constraints. The constrains we use for this inspired from [83]. Based on the literature review of all major algorithms that are used to schedule railway timetables, discussed in Section 3.1, we can create a list of pros and cons that come with every technique. Research in multi operator scheduling is limited, however, majority agree on the existence of multiple operators with competing interests. The methods that can take advantage of existence of multiple operators are limited. Options like constraint optimization and linear programming are state of the art to creates schedules and routes that can service a route optimally. However, these are centralized approaches; this explains why they are able to deal with demand which is considered sensitive information by the operators. Slot allocation is one of the well researched methods that model the existence of multiple operators and scheduling as a problem for authorities separate from the operators. This can be seen in the duty distribution put forth by [45].

Given the importance of the formed schedule, the algorithm should be able to explain it. The decisions made in terms of scheduling should be transparent and agreeable among all parties. A system using Machine Learning might produce a schedule that is able to achieve the optimum schedule, but, this might not be transparent enough. The Q-value function obfuscates a lot of the decision making process, henceforth making it non-ideal for a process which has parties with competing interests. Linear programming is an easy-to-explain approach and the constraints used here are straightforward for any party experienced with scheduling. In a similar vein, constraint optimization is a useful approach and it has the added benefit of being easy to solve if a new constraint is added. The issue with both algorithms comes with ownership. Since the scheduling process needs to be multiparty, any participating party should be able to run the protocol without revealing information they deem sensitive. This can be achieved by formulating a privacy preserving version of the algorithms.

Another approach that can be used for multiple operator scheduling is Slot Allocation. Unlike MILP and Constraint Optimization, Slot Allocation is an inherently multiparty approach. The approach is explainable and the process of reaching the result of a bid is transparent for all the parties to observe. Unlike other approaches, this method is scalable and easy to adapt to the use-case. Moreover, unlike Linear programming and constraint optimization, slot bidding is an active algorithm which requires the parties to be online for atleast some part of the process. Privacy of sensitive data still need to be preserved since, the bids might reveal information that the operators deem sensitive. We analyse the algorithms based on the core requirements of our protocol presented in Table 4.3. The variables and symbols used in Table 4.3 are explained in Table 4.2.

Table 4.2: Table of symbols and explanations

Symbol	Explanation
N	No. of trains
K	No. of loops
St	No. of states for 3 actions at every state; St > N
Pe	No. of distinct Periods
Pa	Np. of operators
Sl	No. of slots
●	The algorithm guarantees this specific property
◐	The algorithm does not guarantee this specific property however, some of the properties can be achieved
○	The algorithm does not guarantee this specific property

Table 4.3: Scheduling algorithms present in the literature and their analysis; * = Complexity calculated by us, not stated in the papers.

Technique	Explanability	Mult. Party	Optimality	Partial Sol.	Complexity
Greedy [9]	●	○	○	◐	$\mathcal{O}(NK)$
Branch-Bound [18]	●	◐	◐	●	$o(NlogN)$
RL [39]	○	○	●	○	$\mathcal{O}(3^{St})^*$
Constraint Opt [78]	●	○	●	●	-
MILP [83]	◐	○	●	○	$\mathcal{O}(N^P)^*$
Schedule Merging [46]	◐	◐	◐	●	Exponential
Slot Allocation [6, 5, 45]	●	●	◐	○	$\mathcal{O}(N.Sl^{Pa})^*$

Based on the literature analysis, we divide the train scheduling problem into two steps. In the first step, a partial schedule is created, this schedule is based on safety requirements of railway lines; based on headways, platform infrastructure, and station capacity. Calculation for these are independent of operators and can be globally calculated. We refer to the timetable constraints introduced in Equation 3.4 to Equation 3.6. Based on these, we can calculate the average throughput on a platform. This can then be used to estimate the number of a trains that can serve a station in a period which can then be extended throughout the schedule. In the Netherlands, the NS has the maximum throughput of 15 trains an hour, but this is based on both directions. This leads to a throughput of ≈ 7 trains an hour.

We use a Slot Allocation algorithm to finally create the schedules. The operators have a maximum number of trains they can schedule for a particular route which has varying demands across time. Based on this maximum number of trainsets and demands for every slot, we create all possible schedules. A schedule is denoted by the demand it serves, which is done by adding the demands of all the slots. Since not every operator has an infinite supply of trains, this is not a simple process of comparing demands served at a single time slot and scheduling the train that serves the maximum demand. The operators use combinatorics to create all possible schedules. The schedules consist of all slots filled by exactly one operator and every operator can schedule a maximum of the number of train available to them.

This is an exponential step owing to the implementation of TFHE on Rust creating large sized ciphertexts owing to slower runs when using dynamic programming for schedule creation. The exponential way of creating the schedule is simply filling each slot in all possible way. Given pa participants and sl trains for each participant, each slot can be filled in pa ways. In our implementation, we are able to schedule four operators simultaneously. This is an upperlimit we have kept for a 7 slot setting. This is in line with the maximum number of operators serving any railway station in the Netherlands. Amsterdam Centraal serves connections with NS, Eurostar, ICE, and SNCB.

4.2.4. Computing the best schedule

The best schedule is the one that serves the maximum demand. This step essentially finds out the maximum demand from all available schedules. We present two protocols for this step. We present an online protocol based on partial homomorphic encryption. We also present an offline TFHE based fully homomorphic encryption protocol.

Partially Homomorphic Encryption

The train station has the aggregated demand of possible schedules. They enter a secure comparison protocol with ProRail to determine the maximum aggregated demand. The protocol presented in Protocol 2.1 is iteratively used to find out the maximum value and its index in the array. The train station keeps track of the current maximum demand and, when it reaches the end of the array, knows the maximum aggregated demand.

TFHE

FHE (specifically TFHE) allows one to find the maximum value in an array of encrypted values without the use of a secret key. Using non-deterministic encryption, the maximum value revealed is re-encrypted. A secret key is required to reveal the plaintext value of the maximum. Since aggregated demand is created by the operators, finding the maximum is not sufficient. We need to know the index of this maximum value. There is no direct method to find the index of any value in TFHE, so we create a function based on the provided operations which can find out the encrypted index of the maximum value in an array.

This function is based on the homomorphic multiplication between plaintext and ciphertext.

Comparing two encrypted values to return the maximum can be done using the ‘if_then_else’ comparison operation in TFHE. This operation outputs an encrypted ‘0’ if the first value is greater than the second value and ‘1’ otherwise. By initializing the maximum index as 1, we check if any index value is greater than the current best value. If this is the case, the comparison operation output is an encrypted ‘1’. We homomorphically multiply this with the index of the current comparison value. The homomorphic multiplication leads to the encrypted index value. This value however is encrypted, which means the train station can find out the best schedule but is unable to know the plaintext value without the help of the secret key.

This is an overhead required to run the TFHE implementation to find out the index without requiring any communication with the owner of the secret key. Since the TFHE implementation can be run without interaction with the owner of the secret key, this protocol allows for an offline approach to finding the best schedule.

4.2.5. Publishing the Schedule

At the end of the comparison protocols, an index is revealed to the train station in plaintext. This index corresponds to the schedule that serves the maximum number of passengers. The train station broadcasts the encrypted value that corresponds to this index. The operators know the underlying encrypted demands for the revealed schedule and the overall demand that will be served is not revealed, since this is not necessary for the final schedule. All operators must reveal the same final schedule at the end of this step. The train station reveals the final slot winners as revealed by the operators on the basis of the plaintext index the train station reveals.

TFHE Specific Improvement

The ring-based TFHE deisng allows allows for non-interactive secure comparison. This allows the train station to create the schedules instead of the operators. The train station gets access to the encrypted index of the best schedule, this means that it cannot get any partial orderings among schedules. Since the train station cannot get partial orderings, it can be allowed to create the schedules. In an interactive approach, the schedule orderings can reveal ‘slot demand’ ordering, this process was entrusted with the operators, however, this is not necessary due to non-interactive secure comparison. This means that the train station can create all the possible schedules on their own.

The train station is sent encrypted ‘demands’. It then uses the same technique as the operators to create all possible schedules. Since the train station is assumed to be semi-honest, the schedules formed are all accurate. This improvement reduces the number of communication and computation steps thereby affecting the overall throughput.

4.3. Interactions and Assumptions

We assume that all communication will happen through authenticated secure channels [56]. We assume each entity to have the public key that corresponds to the secret key created by the network operator (ProRail) for a run of the protocol. The operators do not interact directly with the network operator. For the theoretical protocol run, we expect all operators to submit their real forecasted demands. Overall it is possible for the train station to determine if the forecasted demands are close to the real demand seen at the train station, this system of checks and balances should prevent even malicious operators from submitting incorrect forecasted demands. The railway station acts as an intermediary and assumes the role of a second server for the partial homomorphic approach. For the fully homomorphic approach, the train station does not act as a server since the protocol is created for offline use and thus works with one server.

Input Size

Based on publicly available information on train sets used in the Netherlands [60], we assume that demand for a slot cannot exceed the maximum capacity of a train. Unless there is no prior slot for a route, it would mean that demand outweighs supply. Since we are dealing with periodic scheduling, this should not be the case. This would imply that every subsequent slot will have a demand that exceeds the supply. This is unrealistic and means that the only train to this one destination should be scheduled at all times.

Assuming that the number of passengers a train can serve is twice the seating capacity. The longest train can be up to 700 m [79]. With every train car being 25 m on average and having a maximum capacity of 200 in the Netherlands, one train can at maximum serve 5600 passengers. Based on our assumptions, this is a safe upper bound for the maximum demand per slot. Since the overall demand is made up of all the slots scheduled for up to 10 slots in an hour, we can safely set the upper bound of encrypted demand to be 56000 which can be represented by 16 bits. We thus optimize the protocol for up to 16 bits of data to perform homomorphic operations. We also assume that the total number of trains available for scheduling is at least equal to the number of time slots available. This assumption is based on the busy nature of an international train station, thus not allowing for any empty slot.

5
Evaluation

We divide this chapter into two sections. In the first section, we conduct a theoretical analyses of the protocol, we conduct a security and privacy analysis of the design. Later we evaluate the protocols' complexity and runtime. Overall the aim of this chapter is to provide concrete proof of the security and privacy of the protocols and discuss their specific real life use-cases.

5.1. Theoretical Analysis

Cryptographic assumption

Hardness of Learning with Errors (LWE): The hardness assumption of the LWE assumes that solving the LWE problem is computationally hard for a sufficiently large n and q, and an appropriately chosen noise distribution, even if given a polynomial number of samples. This assumption is supported by reductions from worst-case lattice problems, which are believed to be hard to solve. This connection to worst-case hardness is what gives the LWE problem its cryptographic strength and makes it a candidate for post-quantum cryptography. The robustness of the LWE assumption is also significant because it implies security even if the secret is taken from an arbitrary distribution with sufficient entropy, and even in the presence of hard-to-invert auxiliary inputs. We use this assumption for the security of our fully homomorphic encryption protocol in particular.

We assume the secure of our PRNG 'rand' crate in Rust. We assume that it operates over a wide enough unknown n-bit key making its output computationally indistinguishable from uniformly random bits. Further, we assume the hardness of the factoring problem, the decisional composite residuosity problem, discrete logarithm problem, and the Diffie-Hellman decisional problem. We act in the random oracle model, assuming the existence of cryptographic hash functions to act as random oracles.

We first provide sketches on how the properties mentioned in Section 1.2.2 are met. We identify the most important privacy properties [65] from this list and design the protocol to satisfy the following properties:

1. Confidentiality
2. Collusion-Resistance
3. Traceability
4. Non-Repudiation

We will now describe how the protocol achieves the mentioned properties.

5.1.1. Confidentiality

The demands of all operators should remain private throughout the execution of the protocol. The protocol should not reveal any ordering between the submitted demands. Operators predict their expected demands and encrypt these using the public key created by ProRail. Additionally, the operators form a shared symmetric key to transfer demands to other operators for schedule creation. Demands are first encrypted with the public key and then re-encrypted using the shared symmetric key. Under the assumption of the composite residuosity problem, operators cannot determine the plaintext values of the demands. The schedules created by the operators are sent to the train station, which then enters a comparison protocol with ProRail.

Similar to the operators, the train station does not possess any information about the secret key. This means that the confidentiality properties that apply to the operators also apply to the train station, thereby keeping the demands secret. For the secure comparison step, the definition of the secure comparison protocol (see Section 2.5.3) ensures that the network operator does not learn the values of the schedules or their ordering. This is done by hiding the difference between a and b by using randomness only known to the train station. This randomness prevents ProRail from learning anything besides whether $a > b$. The secure comparison protocol ensures that ProRail does not learn the underlying values of a and b. If an operator wins all but one ‘slot’ without having all their trains scheduled, they infer that the winning operator’s demand exceeds their own; otherwise, they would have won the slot.

Although we assume a semi-honest train station during the protocol run, we argue that the protocol preserves confidentiality even if the train station is malicious. Since the train station does not have access to encrypted slot demands, it cannot decompose a schedule into individual slot demands. A malicious train station may attempt to use secure comparisons to leak information by presenting an arbitrary value a to learn more about the schedule. By varying a until the comparison result changes, the train station can deduce the exact value of a schedule. However, ProRail limits the number of comparisons to pa^{sl}, where pa is the number of operators and sl is the number of slots. This is the number required to deterministically determine the best schedule. Hence, while the train station can learn comparison results and present arbitrary values, this can only be done for a limited number of schedules.

Using domain knowledge, the train station might guess the approximate region of the correct schedule based on previous results, reducing the number of required comparisons. Since the underlying demands for these schedules are unknown to the train station, knowing the sum of random demands does not leak information about the individual demands. For example, knowing that the overall schedule demand is 3000 for 6 slots only reveals the average demand of 500. It does not disclose the demand of any particular slot, nor does the train station know whose demands are part of the schedule.

TFHE Specific Improvement

Since TFHE allows for non-interactive secure comparison, the train station can create the schedules. Malicious operators no longer have the ability to create incorrect schedules and since they are no longer sharing encrypted demands with each other, they do not get to see any other operator’s demands thereby ensuring confidentiality. We assume a semi-honest train station, which means that they always create valid schedules from the encrypted demands received from the operators. The only interaction they have with the secret key owner is to decrypt the index of the best schedule. The train station knows the exact owners of the demands of this schedule, but not the underlying plaintext demands, ensuring confidentiality. ProRail only gets access to the encrypted index and not the encrypted schedule. Since ProRail never sees any demand, confidentiality of demands is maintained.

5.1.2. Collusion-Resistance

Collusion between operators involves sharing plaintext or ciphertext data privately. A collusion-resistant protocol ensures that operators who share data cannot gain insights into the private demands of non-colluding operators. Assuming fewer than 50% of operators are malicious as described in Section 4.1, all colluding operators know the ‘slots’ demands submitted by other colluders.

- A colluding operator can deduce the value of any schedule that does not involve non-colluding operators. Since operators have to create and send seeds to other operators, this is a potential step where operators can act maliciously. By strategically sending incorrect seeds to non-colluding operators, colluders prevent valid schedules. The train station, unaware of malicious operators, may mistakenly consider honest operators as malicious. Stopping the protocol prevents leakage of honest operators’ demands to malicious parties. Since the seeds are sent between the operators through an authenticated secure channel, it is possible to determine that some operators sent different seeds to different operators, thus proving their malicious behavior. This problem does not arise in the TFHE based improvement since operators do not need to form any seeds and directly send their demands to the train station.
- Another point of disruption occurs during the revelation of winners. When the train station reveals the encrypted schedule corresponding to winning bids, operators must all reveal the winners. Colluding operators can now reveal the same incorrect schedule. Since the number of malicious operators is less than half, the train station can select the slot results revealed by the majority of operators. This ensures that the protocol result remains undisturbed, and no information about non-colluding operators’ demand leaks. This problem does not arise in the TFHE based improvement since the train station being semi-honest will reveal the correct winning schedule and the right slot winners.

5.1.3. Traceability

The scheme relies on traceability to identify the real identity of malicious operators and the operators who win any slots during the protocol run, along with the slots they have won. Malicious behavior can manifest in two ways:

Incorrect Seed Usage

An operator might use an incorrect seed to encrypt their created schedules. Alternatively, they could use the correct seed but intentionally create incorrect schedules by adding random encrypted demands instead of legitimate demands from other operators. To address this, the train station enters the secure comparison protocol only if all schedules created by the operators match. Detection via Authenticated Channels: Communication flows through authenticated secure channels. By verifying whether an operator sends the same seed to other operators, the train station can detect discrepancies. If an operator uses the correct seed but produces incorrect schedules, their created schedule will not match those of other operators. Since fewer than half the operators are malicious, a majority of operators will reveal the same schedule. Similar to collusion-resistance, there is no seed used in the TFHE based improvement, thereby making this behavior impossible.

Revealing Incorrect Winners
After the secure comparison protocol reveals the selected schedule, operators may reveal incorrect winners of slots. However, since more than half the operators are honest, the winners for every slot can be chosen by majority consensus. Anyone revealing a different winner than the majority is acting maliciously. Similar to collusion resistance, the winners are revealed by the train station, making this behaviour impossible.

5.1.4. Non-Repudiation
Operators should not be able to deny participation after submitting bids. Here's how non-repudiation is ensured:

- Operators join the protocol run by encrypting their demands using the public key provided by ProRail. They then share a seed with other operators along with the number of trains they have. At this stage, a symmetric key is formed using the seeds of all the operators. Operators broadcast their encrypted demand by re-encrypting it using the formed symmetric key.
- The authenticated secure communication channel used for all communication Section 4.3 prevents denial of participation. Once a message is sent through such a channel, it can be assumed that it was indeed sent by the operator. The authentication process ensures the identities of the sender and receiver. Since all operators have used this channel to share the number of trains, their seeds, and double-encrypted demands, no operator can deny being part of the protocol run.

5.2. Complexity Analysis
We provide a computational and communicational complexity analysis of the protocols introduced. We also provide a throughput analysis wherein we also compare the two protocols to the TFHE specific improvement utilizing design specific primitives that preserve privacy of demands even when the train station has direct access to encrypted demand.

To define the complexity of our protocol, we will define the parameters over which the complexity depends. First, the computation complexity is discussed; expressed in the number of operators participating in the protocol, pa and the number of slots available for scheduling, sl.

Table 5.1: The communication and computation complexity analysis of the two approaches.
$pa \rightarrow$ No. of operators
$sl \rightarrow$ No. of slots.

Complexity $\rightarrow$	Computation		Communication	
Protocol Step $\downarrow$	PHE	FHE	PHE	FHE
Demand Forecasting	Internal		0	
Encrypting Demand	$\mathcal{O}(sl)$		0	
Sharing Encrypted Demand	$\mathcal{O}(sl)$		$\mathcal{O}(pa)$	
Creating Schedules	$\mathcal{O}(pa^{sl})$		$\mathcal{O}(1)$	
Computing best schedules	$\mathcal{O}(pa^{sl})$		$\mathcal{O}(pa^{sl})$	0
Publishing the schedule	$\mathcal{O}(1)$		$\mathcal{O}(1)$	

Demand forecasting
The complexity of this step is operator dependent. The process can be as complex as the operator desires.

Encrypting demand

For encrypting demands, all operators obtain the same public key that corresponds to the secret key (client key in TFHE) of the network operator. The encryption step converts plaintext into ciphertext. This is a constant time step that goes over every plaintext and encrypts it. This step is performed by all operators. Since the number of forecast demand values for each operator is sl, the complexity of this step is $\mathcal{O}(sl)$.

PHE specific computation

Even though the computational complexity under the two encryption techniques remains the same, the PHE protocol primitives dissociate the public key from the ciphertext. This entails an extra association step which reassociates the ciphertext with the public key. This is an important step, since disassociated ciphertext cannot be used in homomorphic encryption operations necessitating the reassociation step.

Sharing encrypted demand

The encrypted demand is operator specific. In order to make schedules, an operator requires the demand of every other operator. All operators enter a symmetric key establishing protocol. For this, every operator independently comes up with a seed. A combination of every seed is used to create a symmetric key. This operation uses a PRNG function. An operator shares their seed with every other operator. Upon receiving all the seeds, the symmetric key is established. This is used to encrypt the already encrypted slot demands. This double encrypted value is broadcasted and only the operators can recover the encrypted demands using the symmetric shared key. The computational complexity for encryption is $\mathcal{O}(sl)$.

Creating schedules

The schedules are created by making all possible combinations of demands for each slot submitted by all the operators. The operators make all the schedules by combining the demands of every operator. Since the encrypted demand was encrypted by another symmetric key, only the operators have access to encrypted demand and can make the schedule.

It is the responsibility of the train station to ensure that all created schedules are identical. The creation of schedules itself does not require any communication; however, the schedule must be shared with the train station. This is a constant operation since the schedules are parallely broadcasted by all operators. In terms of computation, every slot can have as many as the number of 'operators' options, and these can be individually scheduled across 'slots' number of slots, leading to the overall complexity of $\mathcal{O}(pa^{sl})$. The total number of schedules created is thus an upper bound of this value. The realistic case will not have as many schedules, as the number of trains available to operators will decrease the number of possible schedules. The final created schedules can only have as many slots allocated to an operator as the number of trains this operator has.

Computing best schedules

Algorithmically, both protocols compare the schedules pairwise while keeping a track of the best schedule. This makes the computational complexity equal to the size of the array of schedules. This value is in the upper bound by (pa^{sl}). The difference in the communication complexities is an important property in fully homomorphic encryption. FHE allows for the train station to independently calculate the best schedule from an array of schedules. This does not reveal any information about the chosen schedule, any other schedule, or any ordering. The DGK-based secure comparison protocol, on the other hand, requires six communication steps to compare two ciphertexts. The upper bound on the number of communication steps is thus $6 \cdot (pa^{sl})$.

Publishing the schedule

After computing the best schedule, the train station knows the index of the best schedule but not the underlying encrypted demands. These encrypted demands dictate the winner of a slot. The train station contacts the operators who can reveal the owner of the encrypted demands, in this way the owner of every slot of revealed by the operators. Since all the operators created the same schedules, they all reveal the underling slot winners. All of these should be the same. If not all the revealed slot owners match, the train station can call off the protocol on account of an operator behaving maliciously. This is a constant communication step, the train station enquires the slot owners pertaining to a specific slot and the operators respond with the list of slot owners.

TFHE specific improvement

In the TFHE specific improvement, the train station does not need to contact the operators to reveal the best schedule or to reveal the slot winners. The sharing encrypted demand step is also simplified. The encrpyted demand is now shared directly with the train station, this means that a PRNG does not need to be decided and no seeds need to be shared amongst operators. Instead of the double encrypted demand being broadcasted, it needs to be sent over a secure channel to the train station. This is so ProRail cannot access these values since they can now decrypt them using their secret key. Since the train station created the schedules based on submitted encrypted demands, it can reveal the winning schedule and the slot winners without any communication.

5.3. Experimental Analysis

We implemented our protocols in Rust and in this section we analyse their throughputs and runtimes.

5.3.1. Throughput

The two protocols introduced in this thesis are largely identical. The only difference between these protocols is the comparison protocol. PHE protocol relies on communication to securely compute the best schedule. This is not the case with the FHE protocol, wherein the train station can compute the best schedule without consulting the network operator. We analyse the throughputs of the two protocols in a 3 operator 7 slot setting, with every operator having 7 trains each. We compare this to the TFHE specific improvement under the same setting. We note that a lower throughput is preferable, since the protocols achieve the same solutions with different throughputs. A lower throughput means that the protocol is able to achieve this while utilizing less bandwidth on the netwoek, making other tasks quicker. The sizes of some common primitives like keys and ciphertexts differ amongst the two approaches. Sizes of these primitives are given in Table 5.2. We also use primitives defined in the secure comparison protocol presented in Protocol 2.1. The size of these are $[d], [d'], [\hat{d}], [z_l], [\delta']$ equal to 140 Bytes and $[[t]], [[e_i]]$ equal to 4104 Bytes. These primitives are used for the throughput analysis presented in Table 5.3.

ProRail creates the public key corresponding to their secret key (client key in TFHE). This public key is compressed and then sent to the operators and the train station (server key in TFHE). The operators share their seed for the PRNG and the number of trains with other operators. We take a 256 bits seed to ensure security. The operators encrypt their values with the public key and then again with the symmetric key. The process of seed sharing costs bandwidth equal to the size of the seed. This is represented in the sharing of encrypted demand step. In TFHE, the compressed ciphertext is used for quick transfer. These ciphertexts are then broadcast for use by other operators.

Table 5.2: Size of primitives used in Bytes

Primitive	Size (in Bytes)
Public Key PHE	216
Public Key FHE	2151679656
Compact & Compressed Public Key (FHE)	16568
Ciphertext PHE	150
Ciphertext FHE	131692
Compressed Ciphertext (FHE)	748
Seed	32

Table 5.3: The throughput analysis of the two approaches and the TFHE specific improvement. In this analysis, we consider three operators with seven trains each, scheduling for seven slots

Throughput (in Bytes)	PHE	FHE	TFHE
Public Key Exchange	216	16568	16568
Sharing Encrypted Demand	1082	5240	5240
Sharing Schedules	2296350	288010404	-
Computing best schedules	19473048	131692	131692
Finding the slot winners	150	131692	131692
Publishing the schedule	2	2	2
Total	21770848	288295598	285194

The operators create all possible schedules after acquiring every operator's encrypted demands. This is a homomorphic addition step. In TFHE, the operators have to decompress the ciphertext, since compressed ciphertexts do not allow for homomorphic operations. The operators share their created schedules with the train station. This is where the FHE protocol utilizes a lot of storage. These ciphertexts cannot be compressed since compression is only possible during the encryption stage making this a one time expensive step.

The train station now enters the comparison protocol with ProRail to find which schedule serves the maximum demand. In the PHE protocol, the train station computes and sends $[d], [[e_i]], [z_l]$ to ProRail, while ProRail computes and sends $[d'], [\hat{d}], [[t]], [\delta']$ to the train station per comparison. In the FHE design, the train station can compute the best schedule over encrypted data. The train station sends a ciphertext to ProRail. This ciphertext represents the encrypted index of the best schedule. ProRail decrypts this to sends the plaintext index of the best schedule.

The train station sends the corresponding ciphertext to the operators. Based on this ciphertext, the operators reveal the underlying encrypted demands and the corresponding slot winners. These are then sent to the train station, which finally publishes the schedule. The overall throughput of this example run is $\approx$ 22MB for PHE and $\approx$ 285MB for FHE.

TFHE specific improvement

Since the TFHE design allows for the train station to create the schedules, a system constrained by throughput bandwidth can utilize the TFHE specific design. This design guarantees the same privacy-preserving properties. The major reduction in throughput is seen at the step of sharing schedules. This protocol does not require transfer of ciphertexts after homomorphic operations. Transfers are limited to compressed ciphertexts and keys, ensuring optimal use of limited infrastructure. This optimization allows for a much fewer bytes being transferred on the network, leading to $\approx$ 285KB

5.3.2. Runtime

Understanding the computation complexity, we can already begin seeing where the protocols can be improved. The total runtime of both approaches for the largest problem size of three operators upto ten slots is presented in Table 5.4. It is important to note that the runtime of the TFHE specific improvement is identical to the FHE based approach. This is because we assume 0 latency, thereby ensuring that communication steps do not change runtimes which essentially is the improvement we get over the FHE design. The PHE protocol is a lot quicker

Table 5.4: Total running time in a three operator setting. TLE = Time Limit Exceeded

No. of slots	PHE	FHE
	Time in Seconds	
2	1.0	19.1
3	4.6	46.0
4	15.3	134.0
5	43.8	426.2
6	124.1	1365.5
7	379.4	4414.2
8	1196.7	14384.6
9	3651.4	46428.7
10	11162.6	TLE

than the FHE counterpart, making it the de-facto default protocol. PHE achieves the best schedule (serving the maximum demand) in ≈ 3hours while the FHE approach exceeds the 24 hour time limit we set.

This result, gives an incomplete picture, to get a clearer idea of where improvements can be made, we divide the protocol into three broad steps. Note that these are not the same steps as the complexity analysis. This is because the steps in the complexity analysis can be combined since we do not consider latency. Communication steps do not cost us runtime.

1. Encryption
2. Creation of Schedules
3. Schedule Maximization (Comparison Protocol)

We show the encryption times in a two operator setting since this stays the same irrespective of the number of operators. The creation of schedules and schedule maximization times are given in a four operator setting to get an idea of the real world scenario. The time for encryption with the two protocols is given in Table 5.5. The time taken to create the schedules is given in Table 5.6. The time taken to find the best schedule is given in Table 5.7.

According to Table 5.6 and Table 5.7, it is clear that the comparison step takes the longest time. In the implementation of PHE, this is an area for improvement. The FHE approach goes over every schedule exactly once and has no communication overhead, which means improving upon this step within the bounds of the cryptographic assumptions (of LWE Section 5.1) is not feasible. As the problem size grows larger, the time to encryption becomes negligible (Table 5.5), this is owing to the increase in the number of encryption steps being linear in the number of slots while the number of homomorphic addition steps required for creating schedules grow exponentially.

In Figure 5.1, we can see that the creation of combinations is an exponential step for FHE. This can be improved using dynamic programming, however, due to the size of the keys that

Table 5.5: Encryption time in a two operator setting

No. of slots	PHE	FHE
	Time in Seconds	
2	0	4.1
3	0	6.0
4	0	8.3
5	0	11.2
6	0	13.5
7	0	16.3
8	0	18.6
9	0	21.4
10	0	24.1

Table 5.6: Time taken to create the schedules in a four operator setting

No. of slots	PHE	FHE
	Time in Seconds	
2	0	2.9
3	0	21.4
4	0	131.5
5	0.002	697.8
6	0.009	3349.7
7	0.048	13801.1

reside with encrypted values, this step does not work as expected. The overhead of sharing these values in memory defeats the speedups we get by doing fewer operations. The experiments showed that the dynamic programming way to make combinations was actually slower than using the naive approach here. The PHE protocol takes negligible time to create these combinations.

Finding the best schedule is an exponential step as visualized in Figure 5.2 and Figure 5.3. The time taken by both the protocols increases exponentially in the number of slots and operators. The increase in the number of operators changes the base. The increase in the number of slots changes the power. Either of these lead to an exponential increase in runtime. This is in line with our complexity analysis shown in Table 5.1, thereby validating our complexity claims and verifying our runtimes.

According to Figure 5.4, there is a factor of 10 between the speeds of the PHE and FHE protocols. This makes the PHE protocol a better approach for finding the schedules in the quickest time possible.

5.4. Conclusion

The experimental results highlight the differences between the two homomorphic encryption approaches. With both approaches providing the same privacy properties, the differences are practical. In terms of complexity, both the protocols have similar computational complexities which should ideally lead to similar runtimes since we do not consider latency in our runs. This however is not the case since the design of the homomorphic encryption techniques is fundamentally different. Similar steps like encryption and homomorphic addition take significantly different times. e.g., encryption with a PHE public key is instant, while using FHE public key,

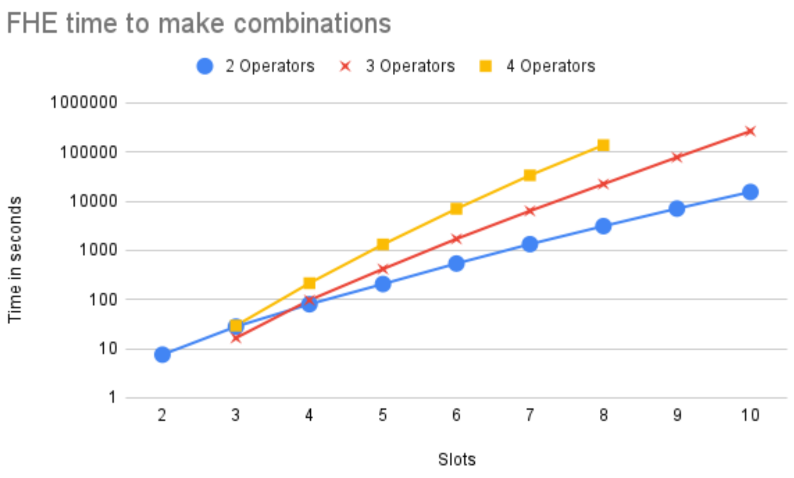


Figure 5.1: Time (in seconds) taken by FHE to create schedules

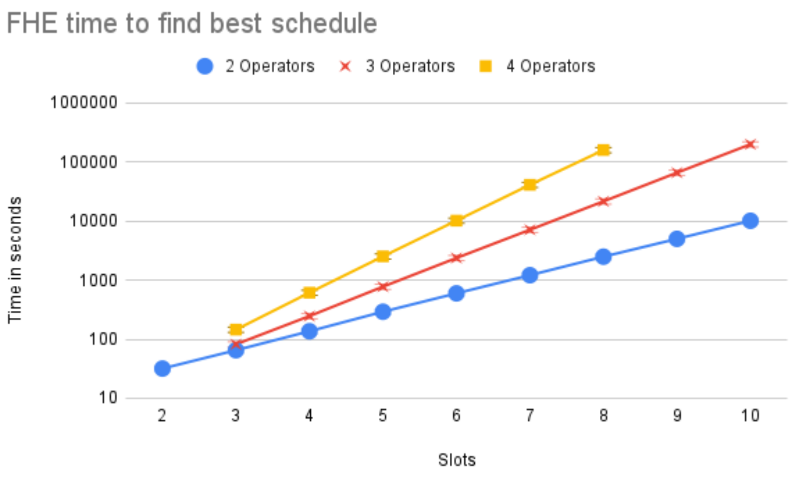


Figure 5.2: Time (in seconds) taken by FHE to find the best schedule

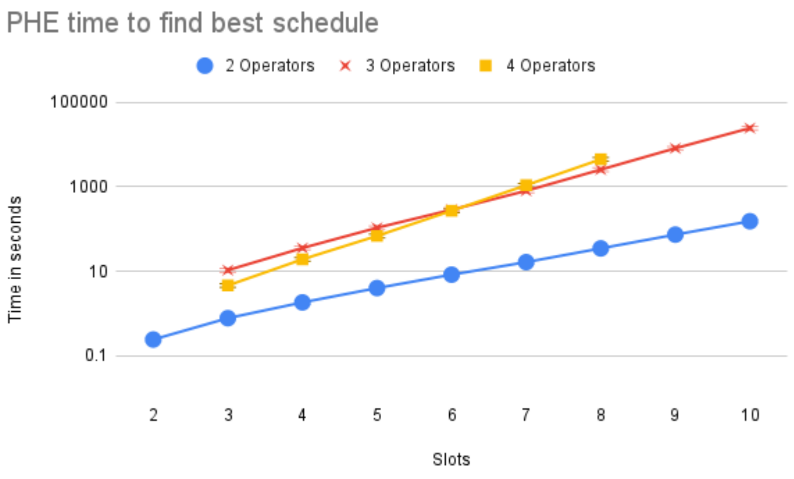


Figure 5.3: Time (in seconds) taken by PHE to find the best schedule

Table 5.7: Time taken to find the best schedule in a four operator setting

No. of slots	PHE	FHE
	Time in Seconds	
2	0.4	14.4
3	1.9	61.3
4	6.8	252.1
5	26.9	1016.9
6	109.0	4108.3
7	447.7	15899.3

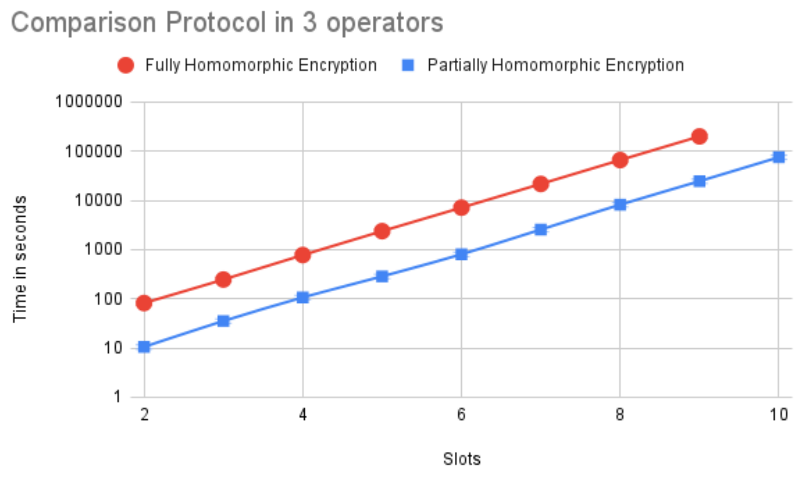


Figure 5.4: Time to find the best schedule PHE (Secure Comparison) vs FHE

the step takes 4 seconds owing to the nature of the public key. Adding two encrypted 16 bit values takes 0.0019ms in the PHE design, while in the FHE design, the same step takes 189ms. Homomorphic addition is a frequent process in our protocol design which can be seen when analysing the two protocols wherein PHE is 10x faster than FHE while having the same algorithmic complexity at every step.

While comparing the communication complexities, we can see that the FHE protocol requires a linear number of communication steps equal to the number of operators to share seeds and schedules with while the PHE protocol requires an exponential number of steps with the number of slots also having an impact on the total number of communication steps. Upon using the TFHE specific approach, the number of communication steps can be made constant for the FHE approach since the demands only need to be shared with the train station and this is a one step process. Overall, if communication is a bottleneck for creating schedules, which can be the case when dealing with a high number of operators, FHE is a better approach since the limited number of communication steps ensures fewer fault points in the protocol run.

The throughput analysis is an extension to the communication complexities. Even though the size of primitives used by FHE is exponentially larger, the constant number of communication steps means that the amount of data being transferred is less. Especially with the TFHE specific improvement, we see the FHE approach taking significantly less bandwidth on the network.

Overall, both approaches have their own advantages, PHE always produces quicker results even though it might put more strain on the infrastructure. If time is the only constraint, according to our experiments; for upto four operators scheduling upto seven slots, PHE is the way to go. However, as seen from overall times, it might be infeasible for ProRail and the train station to be in constant communication for upto 3 hours to jointly compute the best schedule as seen in Table 5.4. Even though the FHE protocol will take around 24 hours to solve the same schedules, the network operator is only involved for a few seconds. This solution can useful, especially when a network operator is dealing with a large network of a lot of independent train stations. Bandwidth can thus be bottleneck for large network utilizing PHE protocols simultaneously on multiple sections.

6 Discussion and Future Work

Privacy as a topic of discussion has gained recent popularity. This discussion, however has not lead to any real world improvements in the way we create new technologies. Privacy is still seen as an add-on feature, not a part of the core design. This is because privacy is considered to be difficult to do right and it is often believed that privacy makes systems slow [2]. With a justified yet surprising increase towards sustainability goals in travel, especially due to the improvements in air quality seen during early COVID times [70], the development in train transportation has been rampant in Europe [49].

This thesis proposes a privacy-preserving train scheduling protocol that tackles a novel problem of cooperatively scheduling multiple operator trains at train stations without leaking sensitive information such as forecast demand. We present two solutions which can be applied to every instance of the problem but have their salient features and shortcomings. In this chapter, we reevaluate our research questions and critically look at the presented solutions. This leads us to realize where the protocol can be further improved as future research directions.

6.1. Discussion

The primary research question this thesis tackles is:

How to design a privacy-preserving schedule for a train timetable at International Stations?

We tackle the problem of multi-operator train scheduling. We focus on international stations. We define international stations as train stations where operators are not related and are reluctant to share sensitive information with other parties. We design a train scheduling scheme that preserves the confidentiality of the forecasted demand using homomorphic encryption. We introduced the role of the train station in this process, as they are a stakeholder of the formed schedule profiting from the passengers using the services.

We divided the train scheduling problem into demand estimation and multiparty slot allocation via auctions that allocate slots at the train station to operators by maximizing the overall demand served through all slots. We present a novel multi-party computation solution to train scheduling. For this, we redefine the roles and interactions between the various stakeholders.

We utilize the properties of homomorphic encryption to maximize demands. We show that our protocol works in-case of $< 50\%$ malicious operators and semi-honest train stations and network operators. We experimented with upto four operators and seven slots. This was done in line with the Dutch raillines, where the maximum number of operators serving a train station is four and the maximum number of trains that serve one track in a direction per hour is seven. Based on the results, we found the runtimes to be acceptable where the largest problem can be solved within 3 hours.

How to achieve similar privacy guarantees utilizing a constant number of communication steps between the participating parties?

The PHE based solution requires the train station and the network operator (ProRail) to be in constant communication. Real-world constraints such as latency and network errors can slow down or even disrupt the protocol. The PHE version overall takes less bandwidth compared to the identical FHE solution. Our security analysis shows that the PHE and FHE designs offer the same privacy guarantees. Realizing the potential of FHE, specifically the TFHE implementation, we presented an alternative FHE design to reduce the number of communication steps and to reduce the role of the prospective malicious parties. This solution requires exponentially less bandwidth which can be attributed to the reduction in communication steps. Since the core comparison protocol remains the same, the timetable generated is the same as the other designs.

6.2. Limitations

We note that this work has a few limitations. One major limitation is the inability to check whether the operators submitted valid demands. If the actual forecasted demand is x but an operator submits a bid for say $2 \cdot x$, they might win the slot they would not have otherwise won. The only way to inspect the winning slots and the demands they actually serve is to find passenger data after the train schedule is created. This limitation requires procedure to overcome. The operators are required to agree that the train station can conduct an investigation if they find that the actual passengers traffic is ‘sufficiently’ below the predicted demands. The train station then has the right to acquire the demands from the operators. Since operators have all the demands and majority of them are honest, getting encrypted demand even without the help of malicious operators is possible. Then the train station has the right to decrypt these values with the help of ProRail to determined who submitted incorrect demands. We believe that this system of checks and balances will ensure operators submit correctly formed demands. The advantages the system brings in form of induced demand [10] should encourage operators to submit demands even if they believe they will not win a slot.

Another limitation is the inability to schedule multiple routes for the same operator. Even though it is possible to schedule multiple routes for an operator, this will entail an operator having multiple demands for the same slot. Since an operator has a specific number of train sets available for multiple routes. If a train is scheduled on Route A, the number of trains available for Route B is reduced by 1. The number of trains submitted is only used to create possible schedules and it is not until the end of the protocol run that it is revealed how many trains of an operator are scheduled on a route. This means that the operator cannot know the number of trains they have available for Route B, until the end of the protocol run for Route A. This significantly increases the complexity of the schedule creation algorithm. This is a runtime based limitation and a possible solution would be to separate routes based on tracks at a train station.

Since we assume that the number of colluding operators is less than $< 50\%$, we run into a problem if the number of operators is 2. It becomes hard to know which operator is malicious in case the created schedules do not match. The system of checks and balances that allows the train station to conduct investigations in case of demand discrepancies can be applied here.

In case of discrepancies, the train station will just break the protocol run, but finding out who the corrupt operator is can be tricky. This problem, however disappears in our TFHE specific improvement. The operators now just submit their demands with the train station creating the schedule, thereby the schedule will be optimum.

We safeguard the demands of all the operators using the public key corresponding to the secret key created by the network operator. The protocol design entails potentially malicious operators creating all possible schedules utilizing every operator's encrypted demands. If any of the operators collude with the network operator, they can leak all the private 'demand' values. Even though this is remedied in the TFHE specific implementation, the problem still persists since now the train station has all the demands and they can still collude with the network operator. The assumption that the network operator is incorruptible is important for the security of the protocol. At the core of the problem is the assumption that the network operator is semi-honest. One way to tackle this would be to use a secret sharing (introduced in Section 2.4) technique where one corrupt party cannot not leak information. We do not use this approach to keep the system similar to the real world train scheduling design. We argue that since the network operator is a government body, assuming them to be semi-honest is sufficient.

The train station cannot be malicious in our schemes. The train station being malicious in the PHE scheme would entail the train station revealing the wrong index as the optimum schedule.

6.3. Future Work

During the duration of this research, a new version of the TFHE library, v0.6, has been released, which includes new features which can speed up the computation of our protocol. As an experiment, this new version was run for a smaller problem size. The encryption times using compact public keys improve significantly here, from 4 seconds to 0.5 seconds. Although the authors claim faster operations across the board, further research is necessary to determine the impact on our work.

Line Planning

We introduce the concept of train scheduling in Chapter 1 and look at the research scope in the field based on Figure 1.1. We focus on the train timetabling problem here. The next obvious step for applying multi-party privacy preserving techniques to scheduling would be to tackle the problem of line planning. This is a harder problem since it deals with more constraints which makes this a more expensive step. In Section 3.1 we discuss the use of MILP for train scheduling. Here we briefly mention the line planning approach used by the authors for a holistic timetable which goes beyond allotting slots at the train station. Work on this problem would make our solution more complete in producing end to end timetables for trains.

Parallel Implementation

The current implementation requires the least number of comparisons to determine the maximum schedule. We utilize an array structure to implement this. Parallelization can be implemented by maintaining multiple (say 4) maximum index terms. This can be done by dividing the array into sub-arrays and finding out the maximum index in each. The overall maximum can be found by comparing the results of the sub-arrays. Since knowing some orderings between overall schedules does not leak any information about the underlying demands, a parallel approach should be viable, however, more research needs to go into understanding at what point partial orderings can leak private information about the specific demands. The FHE specific implementation is based on the train station creating the schedules. However, there is no communication involved, which means the train station can use as many threads as it has to complete the comparison process quicker. The FHE implementation overall has more parallelization potential for computing the best schedule since the process is centralized.

6.4. Concluding Remarks

In conclusion, our work shows that it is possible for multiple operators to schedule their trains at an international station without the need of a trusted third party. This can be done in a privacy preserving way, wherein their forecasted demands which is sensitive information remains confidential. Our work shows that confidentiality can be maintained even if some ($< 50\%$) operators are malicious and colluding. We utilize homomorphic encryption to create candidate schedules. We implement our scheduling scheme in Rust and evaluate the performance of the individual steps. Our results show that the scheme is viable in runtime on real life problem sizes. We proposed two protocol based on different levels of homomorphism to create schedules. These protocols both run in acceptable duration. Our TFHE specific improvement is able to make the entire process bandwidth friendly by utilizing space in Kilobytes, making the entire system light. Since we create a design using Fully Homomorphic Encryption, it is possible to make the schedule making process more complicated if required for a particular usecase e.g., Amsterdam Centraal might value NS trains more than Eurostar trains. Unfortunately, with the current speeds seen with FHE, a larger problem size seems infeasible, which means that larger problems still require communication, and these are the problems which have more communication steps.

Bibliography

[1] Wirawan Agahari. “Multi-Party Computation as a Privacy-Enhancing Technology: Implications for Data Sharing by Businesses and Consumers”. English. Dissertation (TU Delft). Delft University of Technology, 2023. isbn: 978-94-6384-480-2. doi: `10.4233/uuid:83e0a9bc-f429-479a-ac2b-8a42e6da63f1`.

[2] Anonymous. “Privacy is Hard and Seven Other Myths: Achieving Privacy Through Careful Design”. In: *Blog* (2021). url: `https://blog.xot.nl/2021/10/12/privacy-is-hard-and-seven-other-myths-achieving-privacy-through-careful-design/index.html`.

[3] Yonatan Aumann and Yehuda Lindell. “Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries”. In: *Theory of Cryptography*. Ed. by Salil P. Vadhan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 137–156. isbn: 978-3-540-70936-7.

[4] Grzegorz Bocewicz et al. “A cyclic scheduling approach to maintaining production flow robustness”. In: *Advances in Mechanical Engineering* 10.1 (2018), p. 1687814017746245. doi: `10.1177/1687814017746245`. eprint: `https://doi.org/10.1177/1687814017746245`. url: `https://doi.org/10.1177/1687814017746245`.

[5] Ralf Borndörfer, Thomas Schlechte, and Elmar Swarat. “Railway Track Allocation - Simulation, Aggregation, and Optimization”. In: *Proceedings of the 1st International Workshop on High-Speed and Intercity Railways*. Ed. by Yi-Qing Ni and Xiao-Wei Ye. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–69. isbn: 978-3-642-27963-8.

[6] Ralf Borndörfer et al. “An Auctioning Approach to Railway Slot Allocation”. MA thesis. Delft University of Technology, Oct. 2005. url: `https://www.zib.de/members/borndoerfer`.

[7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. “(Leveled) fully homomorphic encryption without bootstrapping”. In: *ACM Transactions on Computation Theory (TOCT)* 6.3 (2014), pp. 1–36.

[8] Michael R. Bussieck, Thomas Winter, and Uwe T. Zimmermann. “Discrete optimization in public rail transport”. In: *Math. Program.* 79 (1997), pp. 415–444. doi: `10.1007/BF02614327`. url: `https://doi.org/10.1007/BF02614327`.

[9] Xiaoqiang Cai and C.J. Goh. “A fast heuristic for the train scheduling problem”. In: *Computers Operations Research* 21.5 (1994), pp. 499–510. issn: 0305-0548. doi: `https://doi.org/10.1016/0305-0548(94)90099-X`. url: `https://www.sciencedirect.com/science/article/pii/030505489490099X`.

[10] Ennio Cascetta and Pierluigi Coppola. “High Speed Rail (HSR) Induced Demand Models”. In: *Procedia - Social and Behavioral Sciences* 111 (2014). Transportation: Can we do more with less resources? – 16th Meeting of the Euro Working Group on Transportation – Porto 2013, pp. 147–156. issn: 1877-0428. doi: `https://doi.org/10.1016/j.sbspro.2014.01.047`. url: `https://www.sciencedirect.com/science/article/pii/S1877042814000482`.

[11] David Chaum, Claude Crépeau, and Ivan Damgard. “Multiparty unconditionally secure protocols”. In: *Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC ’88* (1988). doi: `10.1145/62212.62214`.

[12] Ilaria Chillotti, Marc Joye, and Pascal Paillier. “Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks”. In: *Cyber Security Cryptography and Machine Learning: 5th International Symposium, CSCML 2021, Be’er Sheva, Israel, July 8–9, 2021, Proceedings*. Vol. 5. Springer. 2021, pp. 1–19.

[13] Ilaria Chillotti et al. “TFHE: Fast Fully Homomorphic Encryption Over the Torus”. In: *Journal of Cryptology* 33 (2019), pp. 34–91. url: `https://api.semanticscholar.org/CorpusID:44099955`.

[14] Joseph Choi and Kevin Butler. “Secure Multiparty Computation and Trusted Hardware: Examining Adoption Challenges and Opportunities”. In: *Security and Communication Networks* 2019 (Apr. 2019), pp. 1–28. doi: `10.1155/2019/1368905`.

[15] Jens Clausen. *Branch and Bound Algorithms - Principles and Examples*. Springer, 1999.

[16] Georgiana Crihan, Marian Crăciun, and Lumini□a Dumitriu. “A Comparative Assessment of Homomorphic Encryption Algorithms Applied to Biometric Information”. In: *Inventions* 8.4 (2023). issn: 2411-5134. doi: `10.3390/inventions8040102`. url: `https://www.mdpi.com/2411-5134/8/4/102`.

[17] Sharon Curtis. “The classification of greedy algorithms”. In: *Science of Computer Programming* 49.1 (2003), pp. 125–157. issn: 0167-6423. doi: `https://doi.org/10.1016/j.scico.2003.09.001`. url: `https://www.sciencedirect.com/science/article/pii/S0167642303000340`.

[18] Andrea D’Ariano, Dario Pacciarelli, and Marco Pranzo. “A branch and bound algorithm for scheduling trains in a railway network”. en. In: *European Journal of Operational Research* 183.2 (Dec. 2007), pp. 643–657. issn: 03772217. doi: `10.1016/j.ejor.2006.10.034`.

[19] Ivan Damgard, Morten Geisler, and Mikkel Kroigard. “A correction to ’efficient and secure comparison for on-line auctions’”. In: *International Journal of Applied Cryptography* 1.4 (2009), pp. 323–324.

[20] Ivan Damgård, Morten Geisler, and Mikkel Krøigaard. “Efficient and secure comparison for on-line auctions”. In: *Australasian Conference on Information Security and Privacy (ACISP)*. Vol. 4586. Lecture Notes in Computer Science. Springer. 2007, pp. 416–430.

[21] Ivan Damgård and Jesper Buus Nielsen. *Secure Multi-party Computation*. 2010. url: `https://www.cambridge.org/core/books/secure-multiparty-computation-and-secret-sharing/1D5C8A3B3A2A76EE8C49C27EA307881E`.

[22] George Dantzig. “Linear Programming under Uncertainty”. In: *Management Science* 1.3-4 (1955), pp. 197–206.

[23] D. de Werra. “An introduction to timetabling”. In: *European Journal of Operational Research* 19.2 (1985), pp. 151–162. issn: 0377-2217. doi: `https://doi.org/10.1016/0377-2217(85)90167-5`. url: `https://www.sciencedirect.com/science/article/pii/0377221785901675`.

[24] The Rust Project Developers. *rand 0.8.4 Documentation*. 2021. url: `https://docs.rs/rand/0.8.4/rand/`.

[25] Bersam Bolat Dilay Celebi and Demet Bayraktar. “Light rail passenger demand forecasting by artificial neural networks”. In: *2009 International Conference on Computers & Industrial Engineering*. Troyes, France: IEEE, 2009, pp. 239–243.

[26] Cynthia Dwork. “Differential Privacy”. In: *Automata, Languages and Programming*. Ed. by Michele Bugliesi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. isbn: 978-3-540-35908-1.

[27] Taher ElGamal. “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms”. In: *Advances in Cryptology*. Ed. by George R. Blakley and David Chaum. Springer Berlin Heidelberg, 1985, pp. 10–18. isbn: 978-3-540-39568-3.

[28] Zekeriya Erkin et al. “Privacy-preserving face recognition”. In: *Privacy Enhancing Technologies*. Ed. by Ian Goldberg and Mikhail J. Atallah. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 235–253.

[29] Junfeng Fan and Frederik Vercauteren. “Somewhat practical fully homomorphic encryption”. In: *Cryptology ePrint Archive* (2012).

[30] Paul Feichtenschlager et al. “Privacy-Preserving Implementation of an Auction Mechanism for ATFM Slot Swapping”. In: *2023 Integrated Communication, Navigation and Surveillance Conference (ICNS)*. 2023, pp. 1–12. doi: `10.1109/ICNS58246.2023.10124262`.

[31] Mohammad Javad Feizollahi. “A Privacy-Aware Distributed Approach for Loosely Coupled Mixed Integer Linear Programming Problems”. en. In: arXiv:2205.00356 (Apr. 2022). arXiv:2205.00356 [math]. url: `http://arxiv.org/abs/2205.00356`.

[32] M.K. Franklin and M.K. Reiter. “The design and implementation of a secure auction service”. In: *IEEE Transactions on Software Engineering* 22.5 (1996), pp. 302–312. doi: `10.1109/32.502223`.

[33] Keith B Frikken. “Privacy-Preserving Set Union”. In: *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*. Springer. 2007, pp. 237–252.

[34] Hisham S. Galal and Amr M. Youssef. “Succinctly Verifiable Sealed-Bid Auction Smart Contract”. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Ed. by Joaquin Garcia-Alfaro et al. Cham: Springer International Publishing, 2018, pp. 3–19. isbn: 978-3-030-00305-0.

[35] Craig Gentry. “Fully Homomorphic Encryption Using Ideal Lattices”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. isbn: 9781605585062. doi: `10.1145/1536414.1536440`. url: `https://doi.org/10.1145/1536414.1536440`.

[36] Craig Gentry. “Fully homomorphic encryption using ideal lattices”. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC ’09. Bethesda, MD, USA: Association for Computing Machinery, 2009, pp. 169–178. isbn: 9781605585062. doi: `10.1145/1536414.1536440`. url: `https://doi.org/10.1145/1536414.1536440`.

[37] Daniel Gibert, Carles Mateu, and Jordi Planes. “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges”. In: *Journal of Network and Computer Applications* 153 (2020), p. 102526. issn: 10958592. doi: `10.1016/j.jnca.2019.102526`.

[38] Tal Grinshpoun and Tamir Tassa. “A Privacy-Preserving Algorithm for Distributed Constraint Optimization”. en. In: ().

[39] Yiwei Guo. “A Reinforcement Learning Approach to Train Timetabling for Inter-City High Speed Railway Lines”. In: *Journal of Railway Research* (2021).

[40] Rafik Hamza et al. “Towards Secure Big Data Analysis via Fully Homomorphic Encryption Algorithms”. In: *Entropy (Basel)* 24.4 (2022), p. 519. doi: `10.3390/e24040519`. url: `https://www.mdpi.com/1099-4300/24/4/519`.

[41] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. 10th ed. McGraw-Hill, 2014.

[42] Yu Ishimaki and Hayato Yamana. “Non-interactive and fully output expressive private comparison”. English. In: *Progress in Cryptology – INDOCRYPT 2018 - 19th International Conference on Cryptology in India, Proceedings*. Ed. by Debrup Chakraborty and Tetsu Iwata. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Publisher Copyright: © 2018, Springer Nature Switzerland AG.; 19th International Conference on Cryptology in India,

INDOCRYPT 2018 ; Conference date: 09-12-2018 Through 12-12-2018. Springer Verlag, 2018, pp. 355–374. isbn: 9783030053772. doi: 10.1007/978-3-030-05378-9_19.

[43] Weizhao Jin et al. *FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System*. 2023. arXiv: 2303.10837 [cs.LG].

[44] Thorsten Koch et al. "Progress in mathematical programming solvers from 2001 to 2020". In: *EURO Journal on Computational Optimization* 10 (2022), p. 100031. issn: 2192-4406. doi: https://doi.org/10.1016/j.ejco.2022.100031. url: https://www.sciencedirect.com/science/article/pii/S2192440622000077.

[45] Kaspar Koolstra. "Transport Infrastructure Slot Allocation". MA thesis. Delft University of Technology, June 2005. url: https://repository.tudelft.nl/islandora/object/uuid%3Acd8cd3b6-358c-4c6f-a26a-829ce3800f05.

[46] Niels Lindner and Christian Liebchen. "Timetable merging for the Periodic Event Scheduling Problem". en. In: *EURO Journal on Transportation and Logistics* 11 (2022), p. 100081. issn: 21924376. doi: 10.1016/j.ejtl.2022.100081.

[47] Marin Litoiu, Traian C. Ionescu, and Jesus Labarta. "Dynamic task scheduling in distributed real time systems using fuzzy rules". en. In: *Microprocessors and Microsystems* 21.5 (Feb. 1998), pp. 299–311. issn: 01419331. doi: 10.1016/S0141-9331(97)00049-5.

[48] Thomas Lorünser, Florian Wohner, and Stephan Krenn. "A Verifiable Multiparty Computation Solver for the Assignment Problem and Applications to Air Traffic Management". In: *Journal of Air Traffic Management* (2017).

[49] Alessio D Marra, Linghang Sun, and Francesco Corman. "The impact of COVID-19 pandemic on public transport usage and route choice: Evidences from a long-term tracking study in urban area". In: *Transport Policy* 116 (2022), pp. 258–268. doi: 10.1016/j.tranpol.2021.12.009.

[50] Christian Matt and Ueli Maurer. "The one-time pad revisited". In: *2013 IEEE International Symposium on Information Theory*. IEEE. 2013, pp. 2706–2710.

[51] L Meier. *Explaining yao's garbled circuits*. url: https://cronokirby.com/posts/2022/05/explaining-yaos-garbled-circuits/.

[52] Armin Memar Zahedani, Jelle Vos, and Zekeriya Erkin. "Practical Verifiable & Privacy-Preserving Double Auctions". In: *Proceedings of the 18th International Conference on Availability, Reliability and Security*. New York, NY, USA: Association for Computing Machinery, 2023. isbn: 9798400707728. doi: 10.1145/3600160.3600190. url: https://doi.org/10.1145/3600160.3600190.

[53] Armin Memar Zahedani, Jelle Vos, and Zekeriya Erkin. "Practical Verifiable Privacy-Preserving Double Auctions". English. In: *ARES 2023 - 18th International Conference on Availability, Reliability and Security, Proceedings*. ACM International Conference Proceeding Series. United States: Association for Computing Machinery (ACM), 2023. doi: 10.1145/3600160.3600190.

[54] Kundan Munjal and Rekha Bhatia. "A systematic review of homomorphic encryption and its contributions in healthcare industry". In: *Complex Intell. Syst.* 9 (2023), pp. 3759–3786. doi: 10.1007/s40747-022-00756-z. url: https://doi.org/10.1007/s40747-022-00756-z.

[55] Walter Murray et al. "Recent developments in constrained optimization". In: *Journal of Computational and Applied Mathematics* 22.2 (1988), pp. 257–270. issn: 0377-0427. doi: https://doi.org/10.1016/0377-0427(88)90405-0. url: https://www.sciencedirect.com/science/article/pii/0377042788904050.

[56] Chanathip Namprempre. "Secure Channels Based on Authenticated Encryption Schemes: A Simple Characterization". In: *Advances in Cryptology — ASIACRYPT 2002*. Ed. by

Yuliang Zheng. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 515–532. isbn: 978-3-540-36178-7.

[57] Melek Nar and Seher Arslankaya. "Passenger demand forecasting for railway systems". In: *Chemistry International* (2022). Accessed: 2024-05-10. doi: `10.1515/chem-2022-0124`.

[58] Sundaravalli Narayanaswami and Narayan Rangaraj. "Scheduling and Rescheduling of Railway Operations: A Review and Expository Analysis". en. In: *Technology Operation Management* 2.2 (Dec. 2011), pp. 102–122. issn: 0974-8091, 2249-2364. doi: `10.1007/s13727-012-0006-x`.

[59] Majid Nateghizad, Zekeriya Erkin, and Reginald L Lagendijk. "An efficient privacy-preserving comparison protocol in smart metering systems". In: *Journal of Ambient Intelligence and Humanized Computing* 8.5 (2017), pp. 699–708.

[60] NS. *Our trains - NS Annual Report 2022*. Accessed: 2024-05-15. 2022. url: `https://2022.nsannualreport.nl/annual-report-2021/about-ns/the-profile-of-ns/our-trains`.

[61] Rafail Ostrovsky and William E. Skeith. "Private searching on streaming data". In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 223–240.

[62] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques, EUROCRYPT'99*. Berlin, Heidelberg: Springer-Verlag, 1999, pp. 223–238.

[63] F.F. Pashchenko et al. "Implementation of Train Scheduling System in Rail Transport using Assignment Problem Solution". en. In: *Procedia Computer Science* 63 (2015), pp. 154–158. issn: 18770509. doi: `10.1016/j.procs.2015.08.326`.

[64] Kinjal Patel. "Secure multiparty computation using secret sharing". In: *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. 2016, pp. 863–866. doi: `10.1109/SCOPES.2016.7955564`.

[65] Kun Peng et al. "Robust, Privacy Protecting and Publicly Verifiable Sealed-Bid Auction". In: *Information and Communications Security*. Ed. by R. Deng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 147–159. isbn: 978-3-540-36159-6. doi: `10.1007/3-540-36159-6_13`.

[66] Rail Delivery Group. *About the Passenger Demand Forecasting Handbook*. `https://www.raildeliverygroup.com/pdfc/about-the-pdfh.html`. Accessed: 2024-05-10.

[67] Anjana Rajan et al. "Callisto: A Cryptographic Approach to Detecting Serial Perpetrators of Sexual Misconduct". In: June 2018, pp. 1–4. doi: `10.1145/3209811.3212699`.

[68] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40.

[69] Francesca. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier Science, 2006.

[70] Ana Catarina T Silva, Pedro T. B. S. Branco, and Sofia I. V. Sousa. "Impact of COVID-19 Pandemic on Air Quality: A Systematic Review". In: *International Journal of Environmental Research and Public Health* 19.4 (2022), p. 1950. doi: `10.3390/ijerph19041950`.

[71] Nigel P Smart and Frederik Vercauteren. "Fully homomorphic SIMD operations". In: *Designs, codes and cryptography* 71 (2014), pp. 57–81.

[72] National Institute of Standards and Technology. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. 2012. url: `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf`.

[73] Victor Toporkov, Dmitry Yemelyanov, and Alexey Tselishchev. “Effective Slot Selection and Co-allocation Algorithms for Economic Scheduling in Distributed Computing”. In: *Procedia Computer Science* 18 (2013). 2013 International Conference on Computational Science, pp. 2424–2427. issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2013.05.415. url: https://www.sciencedirect.com/science/article/pii/S1877050913005589.

[74] A. Vince. “Scheduling periodic events”. In: *Discrete Applied Mathematics* 25.3 (1989), pp. 299–310. issn: 0166-218X. doi: https://doi.org/10.1016/0166-218X(89)90008-5. url: https://www.sciencedirect.com/science/article/pii/0166218X89900085.

[75] Nikolaj Volgushev et al. “Conclave: secure multi-party computation on big data”. In: *Proceedings of the Fourteenth EuroSys Conference 2019*. EuroSys ’19. Dresden, Germany: Association for Computing Machinery, 2019. isbn: 9781450362818. doi: 10.1145/3302424.3303982. url: https://doi.org/10.1145/3302424.3303982.

[76] T. Volkhausen. *The Paillier Cryptosystem: A Mathematical Introduction*. 2006. url: http://www2.cs.uni-paderborn.de/cs/ag-bloemer/lehre/proseminar_WS2005/material/Volkhausen_Ausarbeitung.pdf.

[77] Bert de Vries and Jasper Willigers. “On the State-of-the-Art Demand Forecasting Model Developed by Netherlands Railways”. In: *European Transport Conference*. Paper presented on 10-12. Glasgow, Scotland, UK, Oct. 2011.

[78] Zeyu Wang et al. “An Efficient Hybrid Approach for Scheduling the Train Timetable for the Longer Distance High-Speed Railway”. en. In: *Sustainability* 13.5 (Feb. 2021), p. 2538. issn: 2071-1050. doi: 10.3390/su13052538.

[79] Wikipedia. *Sittard railway station - Wikipedia*. Accessed: 2024-05-15. 2024. url: https://en.wikipedia.org/wiki/Sittard_railway_station.

[80] Fei Yan and Rob M.P. Goverde. “Combined line planning and train timetabling for strongly heterogeneous railway lines with direct connections”. In: *Transportation Research Part B: Methodological* 127 (2019), pp. 20–46. issn: 0191-2615. doi: https://doi.org/10.1016/j.trb.2019.06.010. url: https://www.sciencedirect.com/science/article/pii/S0191261517311797.

[81] Andrew Chi-Chih Yao. “Protocols for Secure Computations”. In: *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS ’82)* (1982), pp. 160–164.

[82] Chuan Zhao et al. “Secure Multi-Party Computation: Theory, practice and applications”. In: *Information Sciences* 476 (2019), pp. 357–372. issn: 0020-0255. doi: https://doi.org/10.1016/j.ins.2018.10.024. url: https://www.sciencedirect.com/science/article/pii/S0020025518308338.

[83] Wenliang Zhou, Xiaorong You, and Wenzhuang Fan. “A Mixed Integer Linear Programming Method for Simultaneous Multi-Periodic Train Timetabling and Routing on a High-Speed Rail Network”. en. In: *Sustainability* 12.3 (Feb. 2020), p. 1131. issn: 2071-1050. doi: 10.3390/su12031131.