Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Privacy in Optimization Algorithms based on Secure Multiparty Computation

Tjell, Katrine

PRIVACY IN OPTIMIZATION ALGORITHMS BASED ON SECURE MULTIPARTY COMPUTATION

BY
KATRINE TJELL

DISSERTATION SUBMITTED 2021

AALBORG UNIVERSITY
DENMARK

Privacy in Optimization Algorithms based on Secure Multiparty Computation

Ph.D. Dissertation
Katrine Tjell

Abstract

The privacy of individuals is an increasing concern in a time where a large amount of data is recorded and stored by different actors all around us. The collected data is for instance electricity and water consumption from individual households, the GPS location of smart phones, public transportation routines, and shopping from digital payment methods. Thus, it is not a wonder why people may start to feel monitored. Nonetheless, the recorded data is extremely valuable for instance in the efforts of reducing pollution and CO_2 emissions and increasing user experiences and living standards. It is therefore highly relevant to study how privacy sensitive data can be recorded and included in computations without violating the privacy of individuals.

This thesis is one such study. The angle of investigation is how cryptographic methods can be used to keep data private (or hidden) even when it is desired to perform computations on it. Said more straight forward, the thesis is concerned with how computations can be made on encrypted data, without needing to decrypt or in other ways reveal the data. The applied methods to protect the data are based on *secure multiparty computation* and *secret sharing*, while the objective is to apply optimization algorithms on the secret data.

The main contribution of the work is insights into the challenges of using cipher texts in algorithms and a novel *real number secret sharing scheme* that proposes a trade-off between privacy and usability of the scheme. The latter contribution challenges the generally accepted idea that the cryptographic techniques should ensure 100 percent privacy also when applied to real world problems.

Resumé

I en tid hvor data opsamles overalt i samfundet af forskellige aktører, er mange bekymret for den enkeltes privatliv. Det opsamlede data kan eksempelvis være elektricitet og vand forbrug, GPS lokationen af smartphones, offentlig transport rutiner og shopping ved brug af digitale betalings metoder. Det er derfor ikke mærkeligt, at mange kan føle sig overvåget. På den anden side er det opsamlede data yderst værdifuldt i udviklingen af teknologier, der reducerer forurening og udledning af CO2 og forbedrer bruger oplevelser og levevilkår. Det er derfor relevant at undersøge, hvordan data sikkerhed og privatliv kan forenes med opsamling og brug af sensitivt data.

Denne afhandling er netop sådan en undersøgelse. Udgangspunktet for studiet er brugen af kryptografiske metoder til at holde data hemmeligt, selv når beregninger skal laves på dataet. Sagt med andre ord undersøger afhandlingen, hvordan beregninger kan udføres på krypteret data uden at dekryptere eller på andre måder afsløre dataet. De anvendte metoder til at beskytte data er baseret på *secure multiparty computation* og *secret sharing*, mens formålet er at anvende optimerings algoritmer på det hemmelige data.

Resultatet af arbejdet er hovedsageligt en større forståelse af udfordringerne ved at bruge krypteret data i algoritmer og en ny secret sharing metode til reelle tal som giver en afvejning mellem privathed og praktisk brug af metoden. Det sidste resultat udfordrer den generelle accept af at kryptografiske metoder skal sikre privatliv 100 procent selv når de anvendes i praktiske problemstillinger.

Contents

Preface

This thesis is submitted as a collection of papers in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the *Department of Electronic Systems*, *Automation and Control, Aalborg University, Denmark*. The work has been carried out in the period from August 2018 to October 2021 under the Secure Estimation and Control Using Recursion and Estimation (SECURE) project funded by AAU.

The thesis is structured in two parts where the first part serves as an introduction and the second part consists of six published or submitted papers. The first part is divided into 4 chapters, where the first is a broad introduction to the subject followed by the background and specific methods used throughout the thesis. The following three chapters each presents an overall problem and suggestions for solutions. The solutions are presented as summaries of the papers. The summaries are meant as an overview of each paper, stating the specific problem of the paper and a high level introduction to the proposed solution.

I would like to thank Professor Rafael Wisniewski for not only delivering superb supervision throughout my studies, but also for being my mentor and trusted ally. It is for sure that I would not have managed all the challenges, both technical and personal, without your support and guidance.

I also want to thank Professor Moritz Schulze Darup, for opening the doors at TU Dortmund, Regelungstechnik und cyberphysische Systeme, for me even at the challenging times under the COVID-19 pandemic. I very much admire your ability and will to make me feel completely as a part of your excellent team from day one, and I greatly appreciate our talks about everything from technical challenges to career decisions.

Additionally, I would like to thank my colleagues in the SECURE project for broadening my knowledge about other fields than my own and for their great ideas for research and solutions.

Katrine Tjell
Aalborg University, October 29, 2021

Preface

Part I

Summary

Introduction

1   Motivation

In this digitized era of time, an enormous amount of data is recorded and stored by many different actors. For instance,

- credit card companies keep track of time and place of transactions,

- public transportation services record the number of passengers and where people are traveling to and from,

- smart phone providers monitor the GPS location of their smart phones,

- water, heat, and electricity consumption of households are monitored by providers,

- search engines (like Google) keep records of searches,

- and many more examples could be added to the list.

This recording is made possible by the fast developing sensor technology and also the extremely efficient storage technology enabling huge amounts of data to be stored at a low cost. In addition, the internet makes it very easy to collect and share data.

Beginning in the last decade or so, there has been an increasing concern about the privacy of individuals and how the collection of data might violate their privacy. This concern was manifested in 2018 by the "General Data Protection Regulation" law enacted by the European Union (EU) [1] which gives clear rules on how data is to be managed and the rights of EU-citizens to be in control of their own private data.

Some of the recorded data like time and place of credit card transactions and electricity consumption of a household might seem harmless. So why this concern about privacy and what are people afraid of? It turns out that even if a data set does not immediately pose a privacy threat (for instance if it is anonymous), combining the data with other available information may

be enough to identify individuals and thus constitute a huge violation of privacy. For instance, [2] uses publicly available data to de-anonymize disposable 10-ride tickets for public transportation. Buying such an anonymous 10-ride ticket, the user assumes her privacy stays intact, when in fact holding the information from the ticket (date and time of fares, geographical zone, and so on) together with other public data (such as phone-book records and university enrollment lists) identifies the user and the travel history of the individual.

Perhaps even more surprising, anonymous data from databases such as movie ratings or statistical facts about diseases may also lead to identification of the individuals in the database. For instance, in 2007 Netflix published their "Prize dataset", which contains anonymous movie ratings of 500,000 Netflix users, and those users of course expect their privacy to be preserved. However, [3] shows how to perform a de-anonymization attack on the dataset using information from the Internet Movie Database (IMDb), which is public. Thus, Netflix users were identified and there apparent political belief amongst other private information was revealed.

In conclusion, recorded data potentially hold a lot of sensitive information. Especially, if combined with other data sources. To this end, companies might collect seemingly non-senstive data, which individuals are not reluctant to hand over. Nonetheless, the data may allow sensitive information to be extracted when combined with other data sources.

The straightforward solution to the problem would be to stop recording data. However, in many cases such data are key to improve solutions to be greener, cheaper, and of better quality. For instance, Google uses the GPS location of smartphones to predict traffic congestion, which means that users of Google Maps gets directions that avoids tailbacks. In turn, this helps utilizing the already exiting roads and makes traffic more smooth for everyone. Transportation services can use recorded data about their passengers to make more efficient routes taking into account the needs of its costumers, which is beneficial both for the costumers in improved service and also helps make public transport greener and cheaper. Many more examples could be made, for instance if district water providers had detailed information about the consumption profiles of end-users, the pressure in the water grid could be much more adapted to the consumption. This would extend the lives of the pumps and reduce the electricity consumption. In conclusion, the data is indeed key to drive development of greener and better solutions forward and thus the answer is not to stop recording data. Yet, the privacy issue cannot be ignored either.

What this thesis explores is using methods to "hide" data using ciphertexts before the data leave the hands of its owner. Many applications already use encryption of data before transferring the data using a communication channel. Usually, the receiver then decrypts and obtains the plain text data

in order to perform calculations and processing. This thesis explores the possibilities of not allowing the receiver to decrypt, but instead allow him to do computations directly on the cipher texts, ensuring that he stays oblivious to the plain text data. Cryptographers have been working on these kinds of problems since the 1980's under the name of *secure multiparty computation* using techniques such as *homomorphic encryption* and *secret sharing*. Only recently the engineering communities have looked into using them in engineering applications. In the following section, the background and state-of-the-art of these methods are presented.

2 Background and State-of-the-Art

The underlining problem in this thesis is the one of secure multiparty computation (SMPC). Namely, assume n mutually distrustfully parties each has an input x_i, and they would like to compute $f(x_1, \ldots, x_n)$, where f is some function. As the parties do not trust each other, no party i wants to reveal their input x_i. Furthermore, the parties cannot agree on any trusted third party to whom everyone could hand over their input. To this end, the idea is that the parties run a secure protocol through communications with each other, which lets them compute the function f, while at all times remaining in control of their own data. This setup is referred to as SMPC, and is illustrated in Fig. 1.
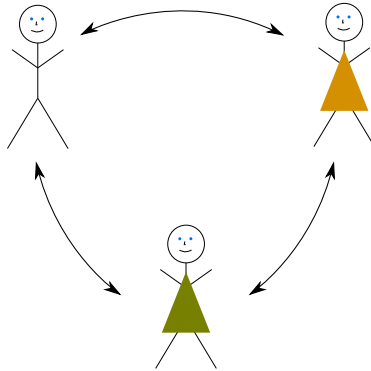
Fig. 1: Illustration of SMPC. Note that no trusted third party exists, hence computation is performed by the participants communicating with each other using a secure protocol.

The problem of SMPC is usually solved using cryptographic methods such as secret sharing, homomorphic encryption, and garbling. Yet, the notion of *differential privacy* is probably even more associated with privacy concerns, especially in the engineering communities. This thesis is mainly occupied with secret sharing based approaches; nonetheless, this section provides
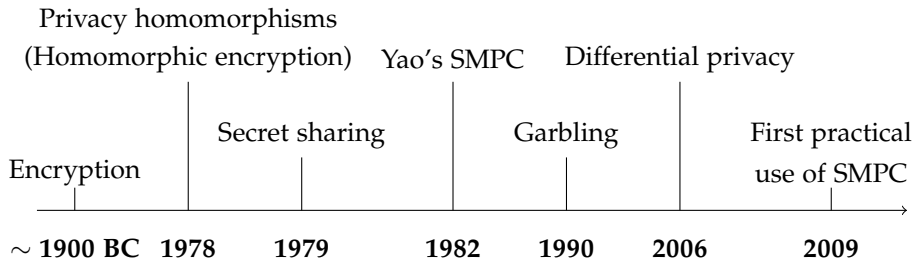
Fig. 2: Approximate timeline (starting from "encryption") showing when the different methods approximately appeared.

a bit of background to all four techniques to put them in relation to each other. As an overview, Fig. 2 depicts a timeline placing the approximate appearance of the methods.

The art of encryption has existed since even before Christ, and has been used in every thing from love letters to hiding orders from a war general to his soldiers, [4]. In the beginning, encryption was mainly carried out by replacing each letter with another according to some fixed table. These schemes were however relatively easy to break using the frequencies of the letters.

Only in the 1920s and 1930s the more complex schemes were invented using machines (notably the Hebern machine and the Enigma), probably due to the world wars, [4]. After that it was not until the introduction of the programmable computer that the development of cryptography again took a huge
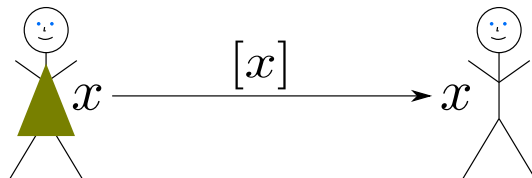
Fig. 3: Typical use of encryption. The sender encrypts the messages, x, sends the cipher text version, $[x]$, and the receiver decrypts to get the plain text.

step starting with Diffie and Hellman introducing public key cryptography in 1976, [5]. As evident, until around this point in time, cryptography (or encryption) was used to keep a message secret between sender and receiver, see Fig. 3. In other words, the sender encrypted the message and the receiver would decrypt to get the plain text.

In 1978, Rivest, Adleman, and Dertouzos proposed that the use of cryptography could be expanded. Namely, they considered an information system handling encrypted data and noticed that all it could do was storage and

6

retrieval of data. For any other operation, it was necessary to decrypt the data first.

In [6], they therefore addressed this limitation and suggested the existence of special encryption functions that allow cipher texts to be operated on, see Fig. 4.

They referred to these special functions as "privacy homomorphisms" and

Fig. 4: Expanded use of encryption. Computations can be done directly on the cipher texts $[a], [b], [c]$. This was first called privacy homomorphisms, but is today known as homomorphic encryption.

already then, they mentioned the limitations, that a large set of operations could not be performed on the encrypted data, in general. Today the concept is known as *homomorphic encryption*, and even though it has been about forty years since the first contribution, most homomorphic encryption schemes can handle either addition or multiplication (and only rarely both), so Rivest et al. were indeed right in their observation about the limitations.

Fig. 5: Secret sharing. A secret is divided into shares that are distributed among *n* participants. No single party has enough information to reconstruct the secret.

Around the same time, namely in 1979, Shamir gave an introduction to the concept of *secret sharing* with his work "How to Share a Secret", [7].

The idea behind secret sharing was (opposed to encryption) not about the transmission of secret data. In fact, Shamir assumed that private communication channels between parties existed. Rather, the idea was that one entity keeping a secret was vulnerable to single-point of failure or single-point attack. Thus, Shamir proposed to share a secret between *n* parties so that no single party (or coalition of *t* parties) would have enough information to reconstruct the data, see Fig. 5. Namely, at least $t + 1$ parties need to collaborate to learn the secret. The work of Shamir would later prove to be significant to the development of SMPC, [8].

The problem of SMPC has been studied by cryptographers since 1982, where Yao made the first contribution to solve it, [9]. This work mainly focuses on protocols for secure two party computation ($n = 2$), where Yao for instance solves the so-called "millionaires problem", where two millionaires seek to learn which of them are the richest without having to disclose the size of each fortune. This work was followed up by Goldreich, Micali and Wigderson in 1987 [10], where they present an algorithm for producing a secure protocol for playing any "mental game" for any number of players, see Fig. 6. By mental game, they refer to communications that does not take

Fig. 6: SMPC. The participants communicate with each other using a secure protocol to perform computations on the private data of each party.

place physically, for instance, coin tossing over the phone.

A year before that, in 1986, Benaloh noticed the homomorphism of many of the secret sharing schemes developed at the time. Particularly, he describes how Shamir's secret sharing scheme allows computations to be performed directly on the shares, [11]. After the work was published, several ideas on how to solve the problem of SMPC using secret sharing approaches appeared, [12–14].

Also, the so-called *garbling* is a method to solve the problem of SMPC, even though it is usually applied for 2-party secure computation. The method is based on boolean circuits, and the private evaluation of it comes from (loosely speaking) encrypting the wires in the circuit. To this end, the output of each gate is meaningless and only by applying the decrypt-information to the (encrypted) result of the last gate will the plain text result be revealed. The idea of representing the function as a boolean circuit was also considered by Goldreich et al. in [10], but it was not until the work of Beaver, Micali and Rogaway [15] in 1990 that the term "garbled circuit" was introduced.

The first contributions to the research of SMCP were primarily of theoretical nature and it was not until around the millennium that researchers started considering real world SMPC problems. In fact, in 1997 Goldwasser said that the field of multiparty computation is "an extremely powerful tool and rich theory whose real-life usage is at this time only beginning but will become in the future an integral part of our computing reality" [8]. This statement underlines the fact that this field is still quite new, and research on how to apply the methods to real world scenarios is far from done.

The first actual real world application of SMCP was in 2009, where it was used to trade contracts for sugar beet production, [16]. The SMPC scheme ensured that bids stayed completely hidden under the auction while still being able to compute the right prices for the trades. Since then, other real world usage of SMPC has not been reported. Nonetheless, there has been much research in applying the SMPC methods to various algorithms used in engineering communities. To name a few, privacy preserving graph filtering is presented in [17], privacy preserving distributed consensus based on additive secret sharing is presented in [18], in [19] privacy preserving distributed optimization is proposed, [20, 21] solves regression problems while preserving the privacy of the involved data, and in [22] deep learning is performed on secret shared data. The mentioned work appeared between 2013 and 2021.

As a kind of alternative to these cryptographic techniques, the notion of the so-called *differential privacy* was in 2006 introduced by Dwork in [23]. Dwork is occupied with the privacy of individuals participating in a database, see Fig. 7. The work starts by showing that it is impossible to guarantee that nothing about an individual can be learned from a database if it could not be learned without the database, which was the aim of previous works. On the contrary, Dwork suggests a privacy measure which describes the risk of participating in the database compared to not participating. The privacy measure is referred to as *differential privacy*. The underlining idea is very simple; private data is obfuscated by random noise. Obviously, the noise must be carefully designed to ensure that sufficient privacy is achieved and that sufficiently accurate information about the database can still be provided.

Fig. 7: Differential privacy (DP). A scientist designs a differential privacy mechanism and a set of individuals provides data to a database. Before the data enters the database it is obfuscated by the DP mechanism which guarantees with a high probability that the data cannot be traced back to the individual.

The main use of differential privacy was in the beginning about ensuring that publishing statistical results could not lead to the identification of individuals, [24, 25]. Since then, the differential privacy approach has found several applications like privacy in machine learning [26], signal processing [27], and control and optimization [28].

In conclusion, Table 1 gives a brief and general comparison of the cryptographic techniques mentioned above. Note that there are on-going research on how to improve the weaknesses of the techniques, thus the comparison

Scheme	Trade-off in privacy and utility	Parties	Comp. heavy	Comm. heavy
Homomorphic encryption	No	$n \geq 2$	Yes	No
Secret sharing	No	$n \geq 2$	No	Yes
Garbling	No	$n = 2$	Yes	No
Differential privacy	Yes	-	No	No

Table 1: Quick comparison of the cryptographic methods. Note that this is true for most schemes in each category and only for the time being since research on how to improve the weaknesses of the techniques is ongoing.

table holds in general and for the time being.

In this thesis, the use of secret sharing techniques in distributed algorithms are explored and the use of garbling is investigated in a minor degree. Therefore, the following provides a more elaborate introduction to these two techniques. To set the scene, finite field arithmetic is introduced first.

2.1 Finite fields and modular arithmetic

The *(mathematical) finite field* is impossible to ignore when discussing data privacy and computations on private data. Many encryption, homomorphic encryption and secret sharing schemes are based on finite field arithmetic. Here, the focus is on finite fields defined as integers modular q, where q is a prime, in this thesis denoted as $\mathbb{F}_q$. To this end, $\mathbb{F}_q$ is a finite set of elements on which addition, subtraction, multiplication, and division is defined. In this context, these operations are defined using modular arithmetic. To illustrate this, Fig. 8 shows each operator in $\mathbb{F}_3$. For a more elaborate introduction to finite fields and finite field arithmetic, refer to [29]. Throughout the thesis, notation is misused in they way that "equality" is used to denote the case, where two numbers are congruent modular q. That is, $x = y \mod q$ carries the same meaning as $x \equiv y \mod q$. Moreover, be aware that any modular equation is to be understood using modular arithmetic, unless otherwise stated.

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

-	0	1	2
0	0	2	1
1	1	0	2
2	2	1	0

·	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

/	0	1	2
0	0	0	0
1	-	1	2
2	-	2	1

(a) Addition in $\mathbb{F}_3$. **(b)** Subtraction in $\mathbb{F}_3$. **(c)** Multiplication in $\mathbb{F}_3$. **(d)** Division in $\mathbb{F}_3$.

Fig. 8: Illustration of the operators in $\mathbb{F}_q$, which effectively is modular q arithmetic.

2.2 Secret sharing

The method of secret sharing is about generating so-called shares of a secret which can be distributed among a set of n participants. Each participant has a unique label $p \in \mathcal{P}$, where $\mathcal{P} \subset \mathbb{F}_q$ with $|\mathcal{P}| = n$. The shares are usually constructed in such a way that any coalition of $t < n$ shares reveals nothing about the secret, and it takes a set of more than t participants to again reconstruct the secret.

A more formal definition of a secret sharing scheme is given in Definition 1, (from paper C, [30]).

> **Definition 1 (Secret Sharing Scheme)**
> A secret sharing scheme for n parties is defined by two algorithms `share` and `recon`. `share` takes a secret s and creates n values $s[1], ..., s[n]$, called shares. `recon` takes a set of at least $t+1$ shares and outputs the corresponding secret s. Moreover, any set of t or fewer shares reveals no information about the secret.

The p'th share of a secret s is denoted $s[p]$ and $s[]$ is used when referring to all n shares. Note that this notation is used regardless of the `share` algorithm. This is because for most cases in this thesis, any `share` algorithm can be used as long as it permits addition and multiplication to be perform directly on shares, see section 2.3.

Here, two `share` algorithms are introduced. The first is the *additive scheme* [31], where the shares are constructed such that

$$s = \left(\sum_p s[p] \right) \mod q, \tag{1}$$

and each share is uniformly distributed on $\mathbb{F}_q$. To reconstruct the secret, all n shares are needed.

The second considered scheme is Shamir's secret sharing scheme. In this scheme, the secret is hidden in the constant term of a random t degree polynomial and the shares of the secret are the polynomial evaluated in known points (usually the labels of the participants). That is, the random polynomial

$$f(x) = \left(s + r_1 x + r_2 x^2 + \cdots + r_t x^t \right) \mod q, \tag{2}$$

is generated by choosing $r_1, \ldots, r_t$ uniformly from $\mathbb{F}_q$, and the secret s is the constant term.

Fig. 9 depicts 50 shares of the secret $s = 5$ using a 10 degree random polynomial and as seen, the shares are uniformly random.
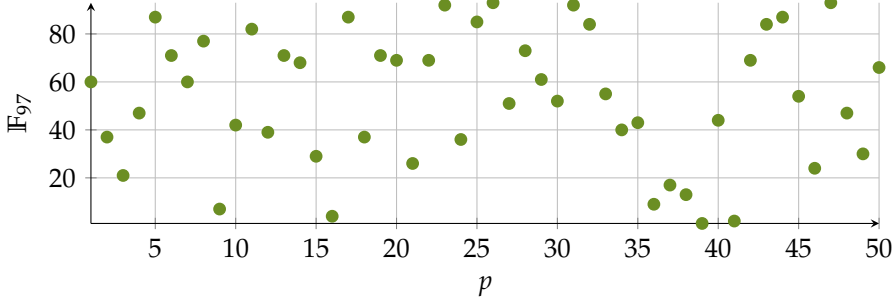
Fig. 9: 50 shares of a secret $s = 5$ created using Shamir's secret sharing scheme with $t = 10$ and $q = 97$.

Note that both in the additive scheme and in Shamir's secret sharing scheme, the generated shares are uniformly distributed on $\mathbb{F}_q$. Since the uniform distribution holds no information about the secret, no information is leaked. This is often referred to as *perfect privacy*, since the adversary gets no more information about the secret than he could by drawing uniformly random numbers himself.

2.3 SMPC based on secret sharing

Solving the problem of SMPC can be achieved using secret sharing. That is, computations can be done directly on shares without the need to reconstruct the secrets first. For the additive secret sharing scheme, addition (and subtraction) can easily be performed on shares of secrets. To see this, consider the addition of three secrets, s_1, s_2, s_3 computed on the shares of the secrets.

$$
\begin{aligned}
s_1 &= (s_1[1] + s_1[2] + s_1[3]) \mod q \\
s_2 &= (s_2[1] + s_2[2] + s_2[3]) \mod q \\
s_3 &= (s_3[1] + s_3[2] + s_3[3]) \mod q
\end{aligned}
$$

$$
\begin{aligned}
s_1 + s_2 + s_3 = ((s_1[1] + s_2[1] + s_3[1]) \\
+ \ (s_1[2] + s_2[2] + s_3[2]) \\
+ \ (s_1[3] + s_2[3] + s_3[3])) \mod q,
\end{aligned}
$$

(3)

where the idea is that the sum of the p'th shares are computed by the p'th participant. Similarly, it is easy to see that a secret multiplied with a public constant can be computed by each participant multiplying their respective share by the constant.

Also shares generated using Shamir's secret sharing scheme support addition (and subtraction) and multiplication with a public constant directly on

12
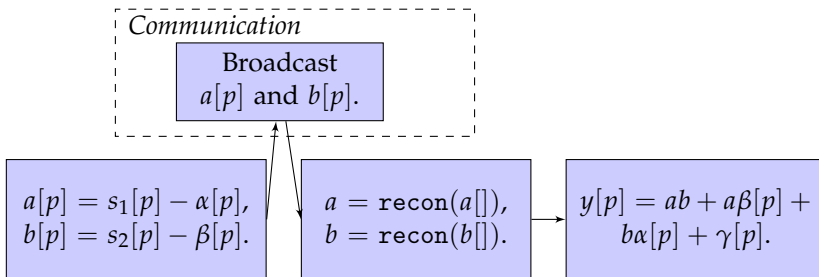
Fig. 10: Block diagram of multiplying two secrets, s_1 and s_2, using Beavers trick and a Beaver triplet, $[\alpha, \beta, \gamma]$, from the view of participant p. The product of the secrets is denoted $y = s_1 s_2$.

shares. The former follows from the fact that adding two t degree polynomials yields a new t degree polynomial with constant term being the sum of the constant terms of the summed polynomials. The latter holds since multiplying a polynomial by a constant simply scales the coefficients leading to the desired result.

When it comes to multiplication, things get more complicated. In case of Shamir shares, a participant can multiply its respective shares of two secrets, however, the resulting polynomial is then of degree $2t$. The growing degree of the resulting polynomial is a disadvantage since $t + 1$ shares are necessary to reconstruct the secret. To circumvent this situation, Beaver came up with a trick in the work [32] which is known as *Beavers trick*. It uses a so-called *Beavers triplet*, which is a set of three random numbers α, β, γ, where $\gamma = \alpha\beta$. The idea is that no party knows the actual value of the triplet, but each party has a share of each number. Generation of the triplets can be done by the participants themselves in a rather straight forward way when $t < \frac{n}{2}$. Beavers trick also works for additive shares, however, there is no obvious way the participants can generate a triplet based on additive shares. To this end, research in efficient ways for generating Beavers triplets for instance using homomorphic encryption, [33, 34], is ongoing.

The algorithm for multiplying two secrets, s_1 and s_2, directly on the shares using Beavers trick is sketched in the block diagram in Fig. 10 from the view of the p'th participant.

2.4 SMPC based on garbling

For private two party computations, functions that can be stated as a binary circuit can be quite efficiently computed using a garbled circuit. To construct and evaluate the garbled circuit, four algorithms are used; `Gb, En, Ev` and `De`. `Gb` is the garbling algorithm that takes as input the binary circuit and outputs the garbled circuit, the encoding, and decoding information. `En` and `De` are the encoding and decoding algorithms, respectively, where `En` takes a binary
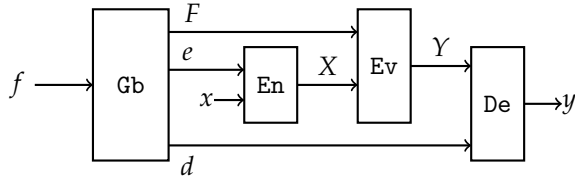
Fig. 11: The algorithms in a garbling scheme and their inputs and outputs. The figure is from [35].

string and the encoding information and outputs a garbled string, De takes a garbled string and the decoding information and outputs the binary output of the circuit. Ev is the evaluation algorithm that takes as input the garbled circuit and the garbled inputs and outputs the garbled output of the circuit. An overview of the algorithms is depicted in Fig. 11.

When applying the technique in the two party setup, one party is assigned to be the so-called *garbler*, and will be in charge of generating the garbled circuit from the binary circuit and encoding the binary inputs. The other party, the *evaluator*, evaluates the garbled circuit using Ev and the garbled inputs. To this end, there is one caveat; usually the input to the circuit is divided between the two parties, thus the evaluator needs to learn the garbled version of its input without the garbler learning the plain text version. To solve this, usually the so-called *oblivious transfer (OT)* is employed. In short, the garbler generates all possible garbled inputs (can only be 0 or 1 for each input wire) and the evaluator then "chooses" the garbled inputs corresponding to its inputs while staying oblivious to the other generated encodings, and the garbler stays oblivious to the plain text input of the evaluator. For the interested reader, details about OT can be found in [36] and its references.

Since the garbling technique is only sparsely used in this thesis, this high-level introduction suffices. For further details, see [35] and paper F, [37].

2.5   Adversary model

In the literature, the most common models for the adversary is; passive, active, adaptive, and eavesdropping, [38]. The models are almost self explanatory; the active adversary can instruct the corrupted participants to deviate from the protocol, while the passive adversary follows the protocol. The eavesdropper can listen to all communication while the adaptive adversary can corrupt new parties along the way.

In this thesis, the adversary is modeled as passive, such that the protocol is followed by both honest and corrupt participants. Thus, it is assumed that the adversary decides which parties to corrupt before protocol execution and that it cannot eavesdrop on the communication. Moreover, the adversary is assumed to be able to attack up to a maximum of t participants.

Choosing the more simple model for the adversary (the passive), makes sense since protocols developed in this thesis (if ever used in a real world scenario) are most likely to be used by companies that would completely loose their reputation if caught cheating. Nonetheless, we note that with only minor modifications, the presented protocols could be made secure against an active adversary by exchanging the secret sharing scheme, [39].

2.6 Privacy model

Privacy is viewed from two perspectives in this thesis. In the first part, privacy of the methods are viewed in terms of the so-called *simulation paradigm*, [38]. Following this paradigm, a scheme is privacy preserving if the data revealed to each party (or coalition of t parties) during protocol execution could be generated form its own input and the output of the protocol. The intuition is that in an ideal world, each party learns the output of the computation and any additional information that can be generated based on this data will therefore also be learned in the ideal world. Since the ideal world is by definition ideal, there is no aim to do better than what can be done here. Moreover, when the revealed data is uniformly distributed, the protocol is said to enjoy *perfect privacy*. This is due to the uniform distribution containing no information about the secret and that each element has an equal probability of occurring.

In the last part of the thesis, a more unconventional view of privacy is taken. Here, privacy is viewed in terms of how much (or little) information is leaked during a protocol execution. This means that perfect privacy is sacrificed, and a setup where privacy can be traded to gain for instance efficiency or accuracy is introduced.

3 Research questions

The aim of this thesis is to investigate whether privacy of data can be preserved when used in control and optimization algorithms. The challenge with algorithms used in the field of control is the usual strict requirement for the speed and accuracy of computations since a controller potentially becomes unstable otherwise. Consider for instance an inverted pendulum, desired to be balanced in the up-right position by a motor. If the controller for the motor is too slow or inaccurate, it is impossible to reach the steady state and instead the motor will alternate between turning to the right and to the left.

The initial goal is first of all to investigate, whether data privacy can be achieved in distributed algorithms (used in control) using cryptographic methods such as secret sharing. Thus, part of the thesis is a proof-of-concept.

The approach is to start out by crawling, and thus already the problem statement is reduced to focus on optimization (or estimation) methods which often is the main part of a control algorithm. Specifically, the focus is on privacy preserving optimization, which in the end can be used to preserve privacy in control.

The thesis builds on the following hypothesis;

- secret sharing approaches are applicable to distributed optimization algorithms used in control to achieve data privacy.

In continuation, a few concerns arise. Namely, SMPC methods based on secret sharing require that each participant can communicate privately with each of the other participants, i.e. a *fully connected* communication network is necessary. Moreover, the methods are known to be heavy in communication complexity and it is a requirement that the secret data can be stated as finite field elements. Since data used in control and optimization typically are real numbers, a conversion to integers is necessary, entailing quantization errors. To this end, privacy comes at a cost. To sum up, the following research questions are stated;

1. How can secret sharing approaches be used in networks that does not support fully-connected communication between nodes?

2. What is the cost of privacy preserving algorithms in distributed optimization compared to non-private solutions?

3. How could a trade-off be introduced, where perfect privacy is sacrificed in order to lower the cost of privacy preserving computations?

    (a) How can privacy be quantified?

The hypothesis is validated by developing privacy preserving distributed optimization algorithms using secret sharing techniques. The research questions are answered by evaluating the performance of the developed privacy preserving algorithms and investigating how improvements can be made.

4   Papers and Outline

The research questions are investigated in the following papers:

A **Tjell, K.**, & Wisniewski, R., "Privacy Preserving Distributed Summation in a Connected Graph", in *IFAC-PapersOnLine*, 53(2), pp. 3445-3450, 2020.

B **Tjell, K.**, & Wisniewski, R., "Private Aggregation with Application to Distributed Optimization", in *IEEE Control Systems Letters*, 5(5), 1591-1596, 2021.

C **Tjell, K.**, Cascudo, I., & Wisniewski, R., "Privacy Preserving Recursive Least Squares Solutions", in *Proc. 18th European Control Conference (ECC)*, pp. 3490-3495, 2019.

D **Tjell, K.**, & Wisniewski, R., "Privacy Preservation in Distributed Optimization via Dual Decomposition and ADMM", in *Proc. IEEE 58th Conference on Decision and Control (CDC)*, pp. 7203-7208, 2019.

E **Tjell, K.**, Schlüter, N., Binfet, P., & Darup, M. S., (Accepted/In press). "Secure learning-based MPC via garbled circuit", in *Proc. IEEE 60th Conference on Decision and Control (CDC)*, 2021.

F **Tjell, K.**, & Wisniewski, R., "Privacy in Distributed Computations based on Real Number Secret Sharing", submitted to *Information Sciences*, 2021.

The papers are found in full in the second part of the thesis. The remaining of the first part of the thesis is divided into 3 chapters that each is concerned with a subject and presents a number of the papers. Specifically,

- Chapter 2 deals with the problem of secure summation in node networks, where a fully connected communication network between the nodes is not assumed. Two privacy preserving solutions are presented as summaries of paper A and B.

- Chapter 3 investigates how to translate specific algorithms into privacy preserving versions. To this end, three privacy preserving algorithms are presented in the form of summaries of paper C, D, and E.

- Chapter 4 is concerned with how to introduce a trade-off between privacy and efficiency of privacy preserving algorithms. The outcome is a *real number secret sharing scheme* presented as a summary of paper F.

Secure Summation in Graphs

5 Aggregation of private data without fully connected communication

Computing shares of the sum of secrets using only shares of the secrets is easily achieved using either the additive or Shamir's secret sharing scheme with the limitation that each party must be able to privately communicate with each of the other n participants. In a real world scenario, setups where this assumption is not met appears; for instance, sensor networks scattered over a large area where each sensor can only communicate with its neighbors (e.i. the sensors that are within some physical distance from itself), see Fig. 12. This chapter addresses this issue and proposes how to extend SMPC based on secret sharing to setups, where a fully connected communication network is not assumed. Specifically, the research question addressed in this chapter is;

- how can secret sharing approaches be used in networks that does not support fully-connected communication between nodes?

Many distributed algorithms are built to communication networks that can be modeled as a connected (and not fully connected) graph, hence, these algorithms assume only neighbor to neighbor communication. Simultane-
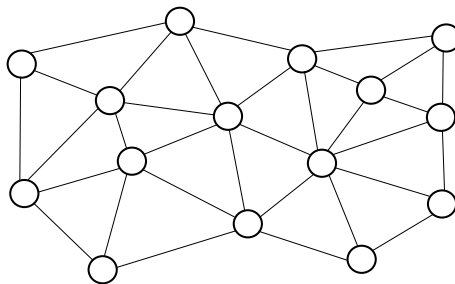
Fig. 12: Illustration of a network of sensors, where each sensor can only communicate with immediate neighbors. Thus, the communication network is not fully connected.

ously, a majority of these distributed algorithms are built on aggregation of data in each neighborhood. In other words, to perform local computations in the distributed algorithms, each node needs the sum of its neighbors data. Interestingly, each node does not need to know the individual data points of its neighbors, but rather the sum of all neighbors data. To this end, computing the aggregate of private data in each neighborhood without revealing individual data points, the whole distributed algorithm becomes privacy preserving. To distinguish this problem from the one where there is a fully connected communication network, the 'participants' are referred to as nodes in this case.

To formulate the problem precisely, consider n nodes having some communication network, for instance as depicted in Fig. 12. Assume that each node i has some private value x_i (for instance a measurement) and that it is preferred not to hand over raw data from node to node. Let $\mathcal{N}_i$ denote the neighborhood of node i, such that $j \in \mathcal{N}_i$ if node i can communicate with node j. We consider then the problem of computing the sum, y_i, of private values in each neighborhood $\mathcal{N}_i$, namely,

$$y_i = \sum_{j \in \mathcal{N}_i} x_j. \tag{4}$$

In the remaining part of this chapter, two methods to solve the above problem are presented. The methods are published in paper A and paper B, respectively. To this end, the following two sections can be read as a summary of the two papers.

6 Summary of paper A ("Privacy Preserving Distributed Summation in a Connected Graph")

The setup in paper A includes many details which has resulted in a somewhat heavy notation, entailing that a fairly simple method appears unnecessarily complicated. To circumvent this situation, the summarized solution presented here is an overview with much lighter notation than the detailed solution presented in paper A. To this end, all details are not included here, merely the most important ones are mentioned.

The particular setup of the problem addressed in the paper is illustrated in Fig. 13. Particularly, a network of nodes is considered where for each node i the aim is to compute y_i in (4). The communication network of the nodes is assumed to be such that each node has at least 2 neighbors. This is mainly because if a node has only one neighbor, then the node learns the private value of that neighbor as a consequence of learning (4). The method of the paper utilizes so-called cliques in the communication network. However, as

6. Summary of paper A ("Privacy Preserving Distributed Summation in a Connected Graph")
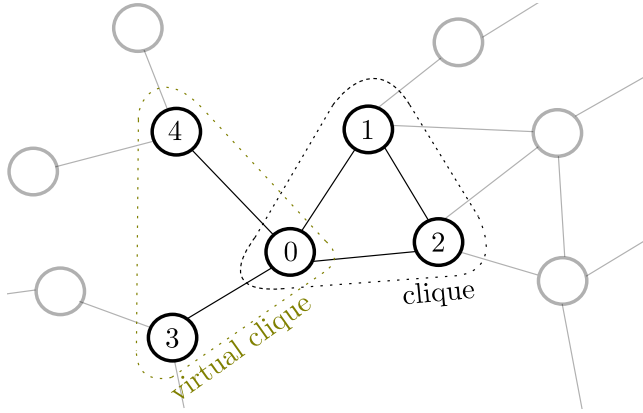
Fig. 13: Illustration of a clique and a virtual clique in a network.

evident from Fig. 13, not all nodes are necessarily part of a clique. Therefore, so-called virtual cliques (VCs) are introduced. The proposed method is, apart from cliques and VC's, based on additive secret sharing. In the following, a short introduction to cliques, VCs and the particular sharings, is given.

- **Clique.** A clique is a fully connected subgraph. In Fig. 13, a clique is illustrated between node 0,1, and 2. Cliques in the network are particularly useful for secret sharing since every node in the clique can communicate with each of the other nodes in the clique.

- **Virtual Clique.** In Fig. 13, node 0 has two neighbors, node 3 and 4, who cannot communicate with any of the other neighbors of node 0. To use the proposed privacy preserving method also for node 3 and 4, a virtual clique between them and their common neighbor (node 0) is defined. To this end, whenever node 3 and 4 communicate, they encrypt their messages and transmit the cipher texts to node 0 who forwards. For the encryption scheme, we use a public key crypto system, which could for instance be ElGamals scheme [40] where key distribution is easily obtained using Diffie-Hellmans key exchange [5]. Thus, in a VC, the nodes exchange keys which leads to a secret key that nodes 3 and 4 use for encrypting their messages, such that node 0 does not learn the communicated texts. Apart from that, the actions taking in a VC is (more or less) the same as those taking in a fully connected clique, thus, no distinction is made between the two types of cliques in the following. Assume there is a total of M cliques (both virtual and fully connected cliques) in the network and let each clique have a unique label $k \in \{1, \ldots, M\}$. Furthermore, define C_k for $k \in \{1, \ldots, M\}$ as a set containing the indices of the nodes in the clique. That is, $i \in C_k$ if node i is in the k'th clique.
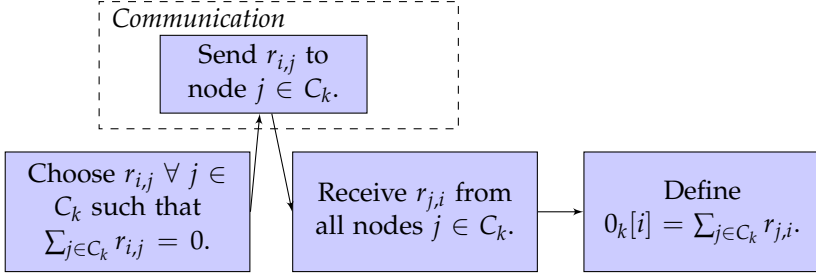
Fig. 14: Block diagram of the generation of a sharing of zero in the k'th clique, from the view of node i.

- **Sharings.** In the proposed solution, additive secret sharing (see section 2.2 and 2.3) is used. In the solution, sharings of a zero in the k'th clique and sharings of the sum s_k in each clique k are used. The i'th share in the k'th clique is denoted as $0_k[i]$ and $s_k[i]$, respectively. Moreover, the sharing of zero is created in each clique k by the nodes in clique k. The nodes do this using the method illustrated in the block diagram in Fig. 14, which is from the view of node $i \in C_k$.

It is now straight forward to explain the proposed method. Since the method is completely parallel, meaning each node performs the same actions, the method is presented from the view of node i. To this end, consider ζ_i to denote the set of cliques node i is part of. That is, $k \in \zeta_i$ if node i is part of clique k. The method is divided into 3 parts, namely a preprocessing, execution and postprocessing phase (and two communication rounds). The advantage is that the preprocessing phase can be run at any time prior to the execution phase since the private data is not involved in this step. Moreover, the communication phases are in this way limited to two rounds, where each party before hand make their messages ready for transmission, which should minimize the time each node needs to be online.

A block diagram of the method is given in Fig. 15. As seen, the nodes start by creating shares of zero in each clique, such that node i ends up with having its share $0_k[i]$ for each clique $k \in \zeta_i$. In the execution phase, node i creates a share of the sum of secret data, $s_k[i]$, in each clique k by adding its own secret data, x_i, to its share of zero, $0_k[i]$. Then one communication round is executed, where node i sends $s_k[i]$ to each node $j \in C_k$ for each clique $k \in \zeta_i$. In the postprocessing phase, node i reconstructs the sum, s_k, of each clique, $k \in \zeta_i$, from the received shares $s_k[j]$ for $j \in C_k$. Node i computes the sum of its neighbors values by $y_i = \sum_{k \in \zeta_i} s_k$.

The privacy of the method follows from each node adding their private value to their share of zero. Each share is essentially a uniformly random value and thus functions as a mask for the private data. Moreover, as long as each honest node has at least one honest neighbor in each clique, privacy is
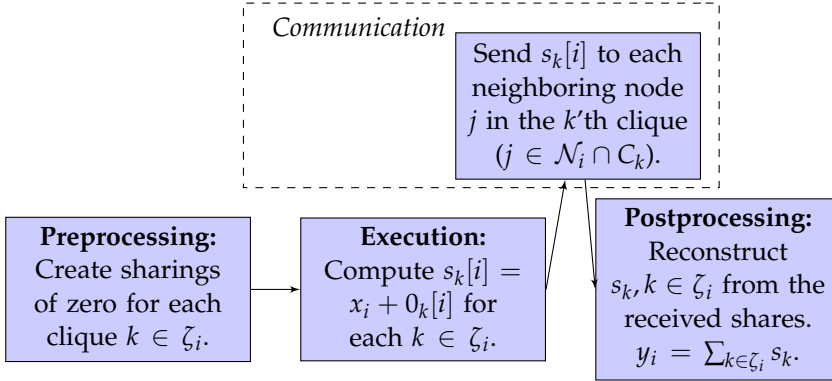
Fig. 15: Block diagram of the proposed method of paper A from the view of node i.

guaranteed. In other words, two honest nodes in each clique is necessary and sufficient to ensure the privacy of the honest nodes. Correctness follows from the fact that since a sharing of zero is used, the private values are added to zero, which obviously disappears in the final computation, leaving only the desired sum.

A detail that was skipped in this presentation of the method, is that if two nodes are part of multiple cliques, then each of them have to account for that. To see the problem, consider Fig. 16, where node 0 and 1 are both part of two cliques; one including node 2 and another including node 3. Thus, when node 0 calculates y_i by summing the results from the cliques, the calculation would yield $(x_0 + x_1 + x_2) + (x_0 + x_1 + x_3) \neq y_0$. Node 0 can of course subtract x_0, but not x_1 since it does not know that value. However, this is easily adjusted for by node 1 transmitting only half its value each time it communicates with node 0; specifically, $(x_0 + \frac{1}{2}x_1 + x_2) + (x_0 + \frac{1}{2}x_1 + x_3) - x_0 = y_0$.
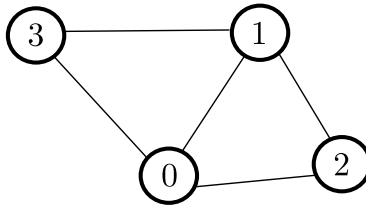
Fig. 16: Example of two nodes being part of multiple cliques.

6.1 Discussion

The disadvantage of the proposed method, is that it requires each node to have extensive knowledge of the communication network. The nodes both

need to know their neighbors and also which cliques they are part of, and if they are part of multiple cliques with any of their neighbors. Moreover, the virtual cliques make computations time consuming due to the extra encrypted communication link. However, we see the method as a first step in the direction of using SMPC based on secret sharing in networks which are not fully connected.

The paper also rise a new question. Namely, the problem in (4) is that each node should learn the sum of all their neighbors private data. Instead, the method presented here lets each neighbor learn the sum of private data in each clique it is part of, and afterwards the node sum these intermediate results to learn y_i in (4). The question is, whether these intermediate results leak information.

7   Summary of paper B ("Private Aggregation with Application to Distributed Optimization")

In this paper, some of the disadvantages of the method in paper A are addressed. In particular, both the cliques and the virtual cliques are disregarded with the purpose of limiting the knowledge each node must have of the communication network and also as an attempt to cut down on communication. Moreover, each node i learns nothing more than y_i in (4), hence, no temporary results leak information. Furthermore, to address the question risen by paper A, a privacy analysis is performed in paper B, where the leaked information about individual terms in a sum from the sum itself is quantified.

The setup in paper B is more or less the same as in paper A, with the distinction that no cliques are considered now. To present the proposed method with clarity, we consider a subset of a graph, specifically 5 nodes, where node 1-4 constitute the neighborhood of node 0, see Fig. 17. To improve readability, the method is presented from the view of node 0 and subsequently, the

Fig. 17: A subset of a graph.

view of the neighbors of node 0. To solve (4) for all nodes in the network, the method is run in parallel by each node. The method is a further development of the method from paper A, hence, secret sharing, encryption and key exchange are also used here.

- **Sharings.** Sharings of random numbers are used in the method. These sharings are made using Shamir's secret sharing scheme. This is mainly because it permits nodes to drop-out in the middle of the protocol, without the protocol having to be run over again. Regarding notation, $r_i[j]$ denotes the j'th share of the random number r_i.

- **Communication.** As illustrated in Fig. 17, there is no assumption that the neighbors of a node can communicate. In fact, the only assumption on the communication network is that each node has at least 2 neighbors. It is, however, necessary for the neighbors to communicate and this will be accommodated for by node 0. The term *distribution* (for instance key-distribution) is used to denote the actions when each node in $\mathcal{N}_0$ sends a message to node 0 and node 0 forwards these messages to each of its neighbors.

- **Encryption.** Obviously, privacy will be broken if node 0 were to learn all messages between its neighbors. Therefore, the neighbors use encryption to protect the messages. Any public key crypto system can be used, for instance ElGamal encryption and DiffieHellman key exchange system as in the previous paper. The public key system is important, since the neighbors need to generate the keys by sending un-protected messages to node 0 (which exactly is what a public key system permits). To denote the encryption algorithm, the notation $\mathtt{enc}(m, pk)$ is used, where m is the message and pk is the public encryption key. The decryption algorithm is denoted by $\mathtt{dec}(m, sk)$, where sk is the secret decryption key. After the key generation and key distribution, each node $i \in \mathcal{N}_0$ has a unique secret key, sk_i, and unique public keys, pk_j, for privately communicating with each of the other nodes $j \in \mathcal{N}_0$.

The idea in the method is simple; each node $i \in \mathcal{N}_0$ masks their private value x_i with a uniformly random number $r_i \in \mathbb{F}_q$, then

$$y_0 = \left((\sum_{i \in \mathcal{N}_0} x_i + r_i) - R \right) \mod q, \tag{5}$$

where $R = \left(\sum_{i \in \mathcal{N}_0} r_i \right) \mod q$.

The neighbors of node 0 computes (without node 0 learning anything) shares of R, such that each node $i \in \mathcal{N}_0$ learns only one share, $R[i]$. The shares of R are computed in a preprocessing phase, which does not depend
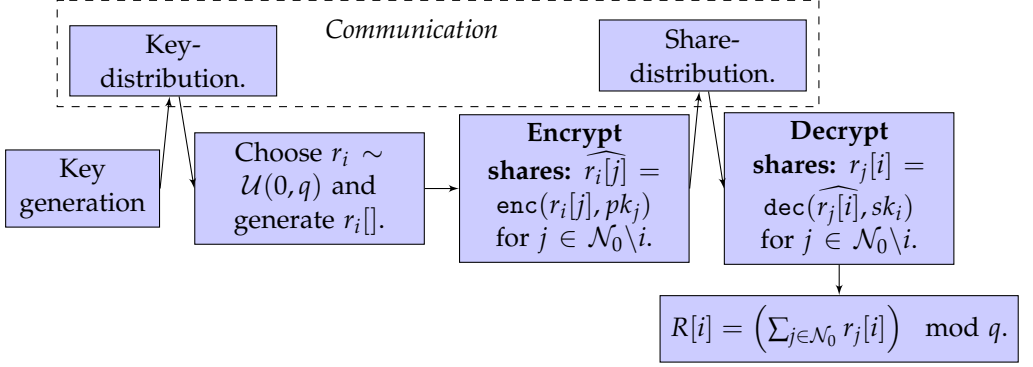
Fig. 18: Block diagram of the preprocessing phase from the view of node $i \in \mathcal{N}_0$. Partly reproduced from paper B, [41].
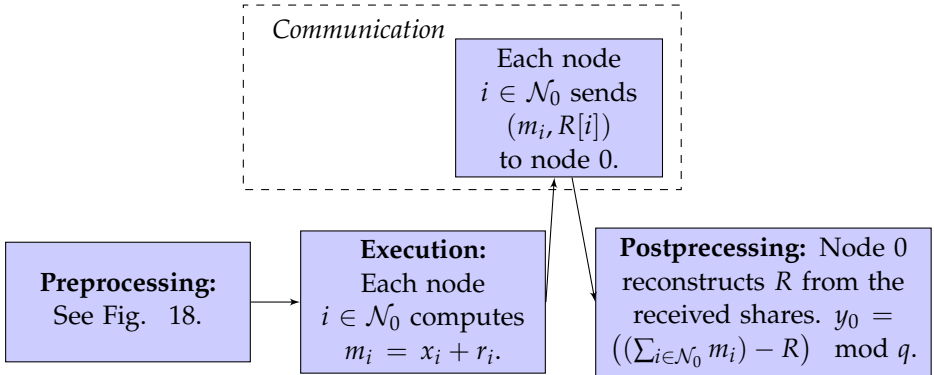
Fig. 19: Block diagram of the proposed method of paper B.

on any secret values and can thus take place at any time prior to the actual execution. The preprocessing phase is illustrated in the block diagram in Fig. 18.

In the execution phase, each node $i \in \mathcal{N}_0$ sends its secret value masked by the chosen random number along with its share $R[i]$. Node 0 then reconstructs R from the received shares and computes (5). The proposed method is seen in the block diagram in Fig. 19.

Privacy follows from the execution phase, where the private data of each node i is masked with the uniformly random number r_i chosen by i. Calculating R (the sum of the random numbers of each node in the neighborhood) is performed using only shares of the individual random numbers, which ensures that none other than node i learns r_i. Regarding corruptions, the protocol tolerates t passively colluding nodes in each neighborhood, where t is the degree of the polynomial used in Shamir's secret sharing scheme.

Correctness is easy to see from (5).

8   Discussion and Comparison

The solutions proposed in paper B was an attempt to address the disadvantages of paper A. The approach to do this, was mainly to avoid basing the solution on cliques or virtual cliques (VC). The hypothesis is that this has the following effects;

- the knowledge each node needs to have on the communication network is limited,

- the overhead in communication should be reduced since the expensive virtual cliques are avoided,

- and the intermediate results from the cliques are avoided.

The first point is true since in paper B, each node only needs to know its neighbors. Regarding the second point, it is not clear that the solution in paper B reduces communications compared to paper A. In fact, it turns out that the solution in paper B is basically based on one big VC linking the neighbors of each node. To count the number of communications in the solution proposed in paper A, it is necessary (for each node i) to distinguish between fully connected cliques, VCs where node i is a neighbor to each node in the VC and the VCs where node i is only neighbor to one other node in the VC. Denote the set of fully connected cliques node i is part of as ζ_i^C, the former VCs as ζ_i^V (also referred to as node i's "own" VCs) and the later VCs as $\zeta_i^{V_n}$ (also referred to as the VCs of the neighbors of node i). It is then possible to count the number of communications and communication rounds used

in the solutions proposed in paper A and paper B, respectively. The term communication round, covers multiple communications that can happen in parallel. That is, if a node sends some value (or different values) to each of its neighbors at the same time, this counts as one round.

$$\text{Paper A} \begin{cases} \text{Communications:} & 2\overbrace{\sum_{k\in\zeta_i^C}|c_k|}^{\text{Cliques}} + 2\overbrace{\sum_{k\in\zeta_i^{V_0}}|c_k|}^{\text{"Own" VC}} + \overbrace{3|\zeta_i^{V_n}|}^{\text{"Neighbors" VC}} \\ \\ \text{Rounds:} & \underbrace{2}_{} + \underbrace{2}_{} + \underbrace{3}_{} \end{cases}$$

$$\text{Paper B} \begin{cases} \text{Communications:} & \overbrace{2|\mathcal{N}_i|}^{\text{Key exchange}} + \overbrace{2|\mathcal{N}_i|}^{\text{Share exchange}} + \overbrace{|\mathcal{N}_i|}^{\text{Execution}} \\ \\ \text{Rounds:} & \underbrace{2}_{} + \underbrace{2}_{} + \underbrace{1}_{} \end{cases}$$

As seen, paper B reduces the number of communication rounds by two compared to the solution in paper A. However, if the communication network contains so many connections between nodes that it consists solely of cliques (and no VCs), then the solution in paper A can be executed in only 2 communication rounds, which will be much faster than the solution in paper B.

Finally, in paper B, the question raised by paper A concerning the leak of the intermediate sums, is addressed. Namely, in paper B, a privacy analysis concerning the leak about an individual value in a sum from the sum itself, is conducted. This analysis, shows that (not surprisingly) there is indeed a leak of information about the terms in a sum from the sum itself. Also, it is determined that increasing the terms in the sum, decreases the leak about the individual terms. To this end, it is clear that the solution proposed in paper A leaks more information about the private data than the solution in paper B.

In conclusion, paper B does address and improve on the weaknesses of paper A. Only the point about reducing communication overhead is still not significantly improved in paper B.

28

Secure Optimization Algorithms

9 Applying cryptographic methods in optimization algorithms

In the previous chapter, distributed algorithms based on sums in neighborhoods were considered, and two algorithms for computing such sums were proposed. Obviously, not all algorithms are based on the computation of only sums. Therefore, this chapter investigates how algorithms using also other operations than the sum can be made privacy preserving. Specifically, two optimization algorithms and one control method are considered; namely, the recursive least squares (RLS), the alternating direction method of multipliers (ADMM), and model predictive control.

The aim of this chapter is to investigate the research question;

- what is the cost of privacy preserving algorithms in distributed optimization compared to non-private solutions?

The approach is straight forward; a privacy preserving version of the considered algorithms are proposed, and afterwards simulation is used to evaluate the accuracy compared to the none-private versions.

10 Summary of paper C "Privacy Preserving Recursive Least Squares Solutions"

The least squares are usually used when input and output to an assumed linear system are observed, and it is desired to find a set of parameters, w, describing the relation between input and output. *The least squares solution* is the parameters, which minimize the sum of squared residuals between the observed output and the output computed from the inputs and the estimated

parameters. It is assumed that input, x_i, and output, y_i, are observed at distinct times i, and therefore, the *recursive least squares (RLS)* equations are used to update the previous estimate of the parameters. The RLS equations yields the parameter estimate at time i given as (reproduced from paper C, [30]),

$$P_i = P_{i-1} - \left(1 + x_i^\top P_{i-1} x_i\right)^{-1} P_{i-1} x_i x_i^\top P_{i-1}, \tag{6a}$$

$$g_i = P_i x_i, \tag{6b}$$

$$e_i = y_i - x_i^\top \hat{w}_{i-1}, \tag{6c}$$

$$\hat{w}_i = \hat{w}_{i-1} + g_i e_i, \tag{6d}$$

where P_0 is usually initialized as the identity matrix, and w_0 is initialized as the vector of zeros.

The purpose of a privacy preserving version of the RLS equations is to keep the observations of x_i and y_i secret at all times during computations. That is, the observations are secret shared and the computations in (6) are performed directly on the shares. The main difficulty is that the computations in (6) then take place in a finite field. The challenges with a privacy preserving version of the RLS equations are the following:

- **Division.** In the finite field, the division in (6a) has to be integer division for the result to be representable in the finite field. To compute integer division in the finite field, a solution is proposed in [42]. To denote integer division, \ is used and .\ denotes elementwise integer division. Clearly, if the nominator is smaller than the denominator, the result of integer division is zero entailing that each variable in (6) will maintain their initialized values and the algorithm will not converge. Another problem arising from the division is the "wrap-around-zero". Namely, if either of the nominator or denominator has "wrapped around zero", the result of the integer division will not be correct. To see this issue, consider the computation of $\frac{-2}{2} = -1$ in the finite field $\mathbb{F}_{23}$; $21\backslash 2 = 10 \neq -1 \mod 23 = 22$. To solve the two challenges, the following solutions are proposed.

  - **Scaling.** To solve the problem of the nominator being smaller than the denominator, the denominator is scaled by a public integer constant, 2^C, before the division. At some point, it will be necessary to rescale again, for which a protocol referred to as $\texttt{rescale}(x, 2^C)$ is used, where $x \in \mathbb{F}_q$ is the term to be rescaled.

  - **Comparison.** For solving the problem concerning wrap around zero of the nominator (the denominator is always positive due to P being positive semidefinite), the idea is to check every entry in

the nominator for being "smaller than zero". Smaller than zero is written in quotation marks, since in a finite field there is no order and as such every element (and no element) are smaller than zero. To account for that, the elements of the finite field are defined as follows

$$\{0, 1, \ldots, \underbrace{\left\lfloor \frac{p}{2} \right\rfloor}_{positive}, \underbrace{\left\lfloor \frac{p}{2} \right\rfloor + 1, \ldots, p - 1}_{negative}\}.$$

The result of $-a < \frac{p}{2}$ coincides with $a < 0$, and can be privately evaluated by the protocol proposed in [43]. This protocol is referred to as $\texttt{comp}()$.

Implementing scaling in (6) is done by multiplying 2^C on nearly all terms in the equations. To illustrate this, let the notation $\cdot^*$ denote $\cdot 2^C$ and consider the following rewrite of (6).

$$P_i^* = P_{i-1}^* - \left(\left(P_{i-1}^* x_i x_i^\top P_{i-1}^* \right) . \backslash \left(1^* + x_i^\top P_{i-1}^* x_i \right) \right) \tag{7a}$$

$$g_i^* = P_i^* x_i, \tag{7b}$$

$$e_i = y_i - x_i^\top \hat{w}_{i-1}^* \backslash 1^*, \tag{7c}$$

$$\hat{w}_i^* = \hat{w}_{i-1}^* + g_i^* e_i. \tag{7d}$$

The proposed privacy preserving implementation of (7) is sketched as a block diagram in Fig. 20. Since all variables in (7) must be treated as secrets, the notation of shares is disregarded in the block diagram in Fig. 20. Thus, the block diagram is to be read in the way that the participants receive shares of the observed x_i and y_i and of the initialized variables $P_0, \hat{w}_0$. The only operations used in the block diagram are addition, subtraction, multiplication, the integer division, the comparison and the rescale protocol. Thus, the parties can compute these operations directly on the shares.

The privacy of the privacy preserving RLS equations follows from the privacy of performing addition, subtraction, and multiplication directly on shares. The privacy of the protocol for integer division is proved in [42] and the privacy of $\texttt{comp}$ is proved in [43]. The protocol $\texttt{rescale}$ is build only on addition, subtraction, multiplication, and bit decomposition, which is proved privacy preserving in [31, p. 189].

$$d^* = 1^* + \boldsymbol{x}_i^\top \boldsymbol{P}_{i-1}^* \boldsymbol{x}_i$$

$$num^{**} = \left(\boldsymbol{P}_{i-1}^* \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{P}_{i-1}^*\right)$$

$$\texttt{comp}(num^{**})$$

$$|num^{**}|$$

$$|f^*| = |num^{**}| . \backslash d^*$$

$$f^*$$

$$i = i + 1$$

$$P_i^* = P_{i-1}^* - f^*$$

$$g_i^* = P_i^* x_i$$

$$e_i = y_i - \texttt{rescale}(x_i^\top \hat{w}_{i-1}^*, 1^*)$$

$$\hat{w}_i^* = \hat{w}_{i-1}^* + e_i g_i^*$$
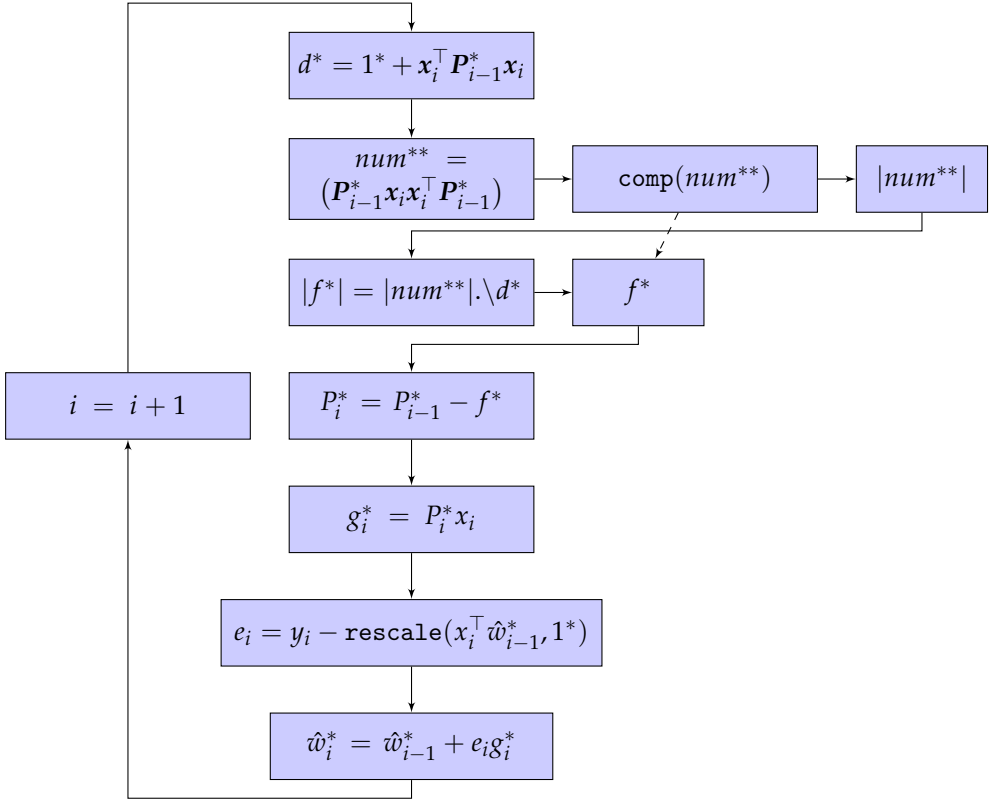
Fig. 20: Block diagram of the proposed method of paper C.

10.1 Results

The work proposes a privacy preserving RLS algorithm that performs each computation directly on the shares of the variables. All data used in the privacy preserving algorithm must be integers, and thus any real numbers must be scaled and truncated before being used. This effects the accuracy and in turn the convergence of the algorithm. To see this, simulations comparing the result of the RLS equations in (6) to the result of the privacy preserving protocol in Fig. 20, are conducted. To compare the results, the mean squared error between the true parameters w and the estimate $\hat{w}$ is computed at each time i;

$$e_{MSE_i} = \frac{1}{q} \sum_{j=1}^{q} (w_j - \hat{w}_j)^2, \quad \text{for } i = 1, 2, \ldots.$$

The simulation is reproduced from paper C, [30]. To recap, the test data is described by

$$y = 3x_1 + 2x_2 + 2x_3 + 4x_4 + 4x_5 + 1x_6, \tag{8}$$

and the observations of $\{x_i\}_{i=1,\ldots,6}$ are uniformly distributed numbers on the interval $[0, 5]$. The observations of y are given by (8) and are afterward corrupted by Gaussian noise with mean value zero and variance two. Fig. 21 depicts the MSE of the parameters estimated with the RLS equations, $\hat{w}$, and the MSE of the parameters estimated with the privacy preserving RLS equations, $\hat{w}_{priv}$.
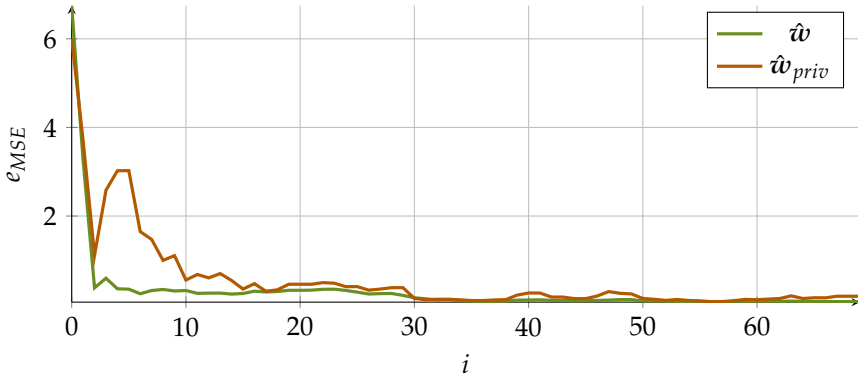
Fig. 21: The MSE of the parameters estimated with the RLS equations, $\hat{w}$, and the MSE of the parameters estimated with the privacy preserving RLS equations, $\hat{w}_{\text{priv}}$. Reproduced from paper C, [30].

11 Summary of paper D "Privacy Preservation in Distributed Optimization via Dual Decomposition and ADMM"

This work aims to solve a distributed optimization problem while keeping the privacy of the involved parties intact. The setup is illustrated in Fig. 22 and consists of N agents and n computing parties. The latter could for instance be independent cloud servers or other computing service providers and their purpose is to relieve the agents from computations and communication. The agents are those that have a problem they seek to solve. In this case, the problem is stated as

$$
\begin{aligned}
\underset{\boldsymbol{x}}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\
\text{subject to} \quad & \sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{x}_i - \boldsymbol{c} = \boldsymbol{0},
\end{aligned}
\tag{9}
$$

where $f_i(\boldsymbol{x}_i)$ is known solely to agent i and $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$. It is further assumed that the constraints in (9) are unknown to everyone except for a *superintendent*, who demand the solution to satisfy these certain conditions.

Solving (9) is done in [45] and is based on the ADMM algorithm [46].
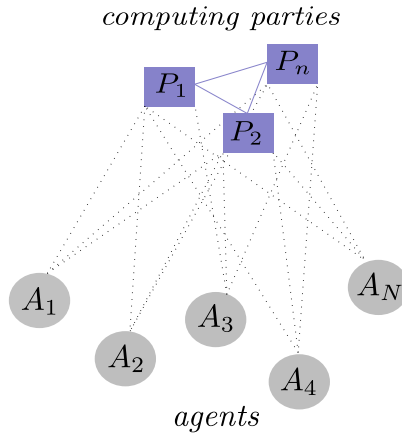
Fig. 22: N agents connected to n computing parties. Reproduced from paper D, [44].

11. Summary of paper D "Privacy Preservation in Distributed Optimization via Dual Decomposition and ADMM"

Their solution can be written the following way for each iteration k;

$$\tilde{x}_i^{k+1} = \min_{x_i} L_\rho(x_1^k, \ldots, x_i, \ldots, x_N^k, \lambda^k) \quad \text{for } i = 1, \ldots, N \tag{10a}$$

$$\tilde{\lambda}^{k+1} = \lambda^k + \rho \left(\sum_{i=1}^N B_i \tilde{x}_i^{k+1} - c \right). \tag{10b}$$

$$x_i^{k+1} = x_i^k - \frac{1}{N+1}(x^k - \tilde{x}_i^{k+1}) \tag{10c}$$

$$\lambda^{k+1} = \lambda^k - \frac{1}{N+1}(\lambda^k - \tilde{\lambda}^{k+1}), \tag{10d}$$

where L_ρ is the augmented Lagrangian which for (9) is

$$L_\rho(x, \lambda) = \sum_{i=1}^N (f_i(x_i)) + \lambda^\top (Bx - c) + \frac{1}{2}\rho \|Bx - c\|_2^2. \tag{11}$$

Note that the minimization over x_i (for a fixed i) of the Lagrangian in (11) can be written as

$$\min_{x_i} f_i(x_i) + \alpha_i x_i^2 + \beta_i x_i, \tag{12}$$

where α_i and β_i are two constants covering the terms to be multiplied with x_i^2 and x_i, respectively. For a derivation, see paper D, [44], and to easily see that (12) is true, consider the following example.

Example 1 (Minimizing the augmented Lagrangian)
Consider the following minimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \underbrace{(x_1 - 1)^2}_{\text{agent 1}} + \underbrace{(x_2 - 2)^2}_{\text{agent 2}} + \underbrace{(x_3 - 3)^2}_{\text{agent 3}} \\ \text{subject to} \quad & 2 = -5x_1 + 5x_2 + 3x_3, \\ & 5 = 2x_1 + x_2 + 5x_3. \end{aligned} \tag{13}$$

Let $i = 1$, then (10a) can be written as

$$\begin{aligned} \min_{x_1} L_\rho(x, \lambda) = \min_{x_1} \ & (f_1(x_1)) + \lambda^\top (Bx^k - c) + \frac{1}{2}\rho \|Bx^k - c\|_2^2 \\ = \min_{x_1} \ & f_1(x_1) \\ & -\lambda_1(-5x_1 + 5x_2^k + 3x_3^k - 2) + \lambda_2(2x_1 + x_2^k + 5x_3^k - 5) \\ & +\frac{\rho}{2}((-5x_1 + 5x_2^k + 3x_3^k - 2)^2 + (2x_1 + x_2^k + 5x_3^k - 5)^2) \end{aligned}$$

$$= \min_{x_1} f_1(x_1) + \underbrace{\frac{\rho}{2}(25+4) x_1^2}_{\alpha_1^k}$$

$$+ \underbrace{(-5\lambda_1 + 2\lambda_2 + \frac{\rho}{2}(-46x_2^k - 10x_3^k + 40)) x_1}_{\beta_1^k}$$

$$= \min_{x_1} f_1(x_1) + \alpha_1^k x_1^2 + \beta_1^k x_1 = \min_{x_1} L_\rho(\alpha_1^k, \beta_1^k)$$

To this end, the minimization of the Lagrangian with respect to x_i, can be written as a function of α_i and β_i. In continuation, each agent i minimizes the Lagrangian with respect to x_i (since agent i is the only one knowing $f_i(x_i)$) and in turn, the computing parties compute at time k, α_i^k and β_i^k (for each i) upon receiving shares of $\tilde{x}_i^{k+1}$ from each agent i. In Fig. 23, a block diagram of the proposed method of paper D is given.
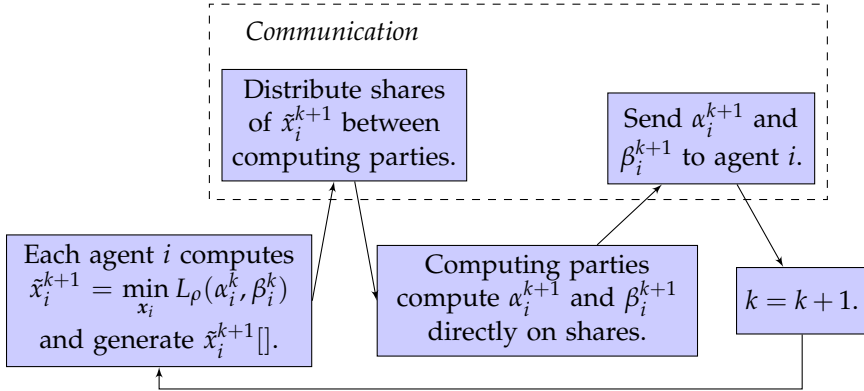
Fig. 23: Block diagram of the proposed method of paper D.

11.1 Results

To evaluate the performance of the proposed method, simulation results of it are compared to simulations of (10) which has no privacy properties. Both algorithms solve the problem in (9) where $N = 3$, $f_i(x_i) = (x_i - i)^2$ and B_i and c are chosen randomly. Refer to the estimate of x_i at each iteration k as $\hat{x}_i^k$.

As seen in Fig. 24, the privacy preserving estimates deviates slightly from the estimates of (10) when k is small. This is due to the rounding of reals to integers, which occurs after the agents has performed their local minimization.
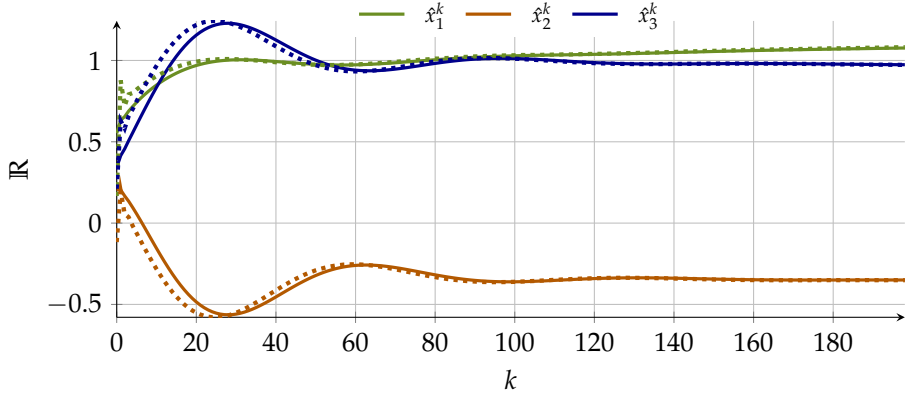
Fig. 24: Comparison between simulation results of the proposed method in Fig. 23 (marked with a solid line) and the non-private solution in (10) (marked with a dotted line).

Before the rounding, the value is scaled by 10^3 in this simulation. Clearly, the chosen scaling constant is sufficient to ensure that the privacy preserving solution eventually agrees with the solution from (10).

12 Summary of paper E "Secure learning-based MPC via garbled circuit"

The final algorithm considered in this chapter consists solely of addition, multiplication and maximum operators. Since it is not trivial to perform comparisons using secret sharing, the idea in the work is to investigate if secret sharing and garbling can be combined to create an efficient solution.

The setup of this work is similar to the setup of the previous papers, with the distinction that in this work, there is only one input provider and only 2 computing parties. Nonetheless, the proposed method is somewhat flexible, so there could in principle be multiple input providers.

The paper considers a control problem were an actuator aims to control a system and uses a concept known as *model predictive control* to calculate the control input. Here, this control theory will not be introduced since it is somewhat besides the topic of the thesis. Moreover, the control problem is rewritten into local computations and computations that are to be performed privately by the computing parties. Skipping the details of the control problem and the local computations, the problem to be solved privately is computing

$$y = \max\{Hx + b\} - \max\{Kx + c\} \tag{14}$$

where $\max\{x\}$ of the vector $x \in \mathbb{N}^p$ is $\max\{x_1, x_2, \ldots x_p\}$.

A privacy preserving solution to (14) is achieved with a combination of secret sharing and garbling. For this problem, the two techniques complement each other well since comparison is intractable to perform with secret sharing, while garbling does it efficiently. On the other hand, multiplication is a quite inefficient operation for a garbling scheme, while secret sharing combined with Beavers trick does it well. The big picture overview of the solution is as follows,

1. $v = Hx + b$ and $w = Kx + c$, are computed using additive secret sharing and Beavers trick.

2. Each computing party generates a garbled circuit for computing $\max\{v\}$ and $\max\{w\}$ respectively and afterwards they exchange circuits such that each computing party evaluates a garbled circuit.

In section 2.2 and 2.3, additive secret sharing and Beavers trick was introduced, thus this step hardly needs further explanation. The only thing worth mentioning is that in the finite field used for secret sharing, a field of 2^l elements is considered where l is a positive integer. The reason for choosing a field of size 2^l instead of a prime number, will become clear later.

For finding $\max\{v\}$ and $\max\{w\}$ it is necessary to define a boolean circuit for use in the garbled circuit. The computation of $\max\{v\}$ and $\max\{w\}$ will be identical, thus it is sufficient to focus on $\max\{v\}$. From the secret sharing part of the solution, each computing party has one additive share of v, thus computing party i holds $v[i]$. The boolean circuit starts by adding elementwise $v[1]$ and $v[2]$ to reconstruct v. To this end, it holds that

$$\tilde{v} = (v[1] + v[2]) \mod 2^l, \tag{15}$$

and consequently,

$$v = \begin{cases} \tilde{v} - 2^l & \text{if } v \geq \frac{2^l}{2} \\ \tilde{v} & \text{otherwise,} \end{cases} \tag{16}$$

assuming that $v \in [-\frac{2^l}{2}, \frac{2^l}{2} - 1]$.

Afterwards, the circuit computes in $\log_2(p)$ rounds the maximum element of v. Fig. 25 illustrates the circuit.

To consider the circuit as boolean, $v[1]$ and $v[2]$ are represented with l bits and the additions are standard carry-ripple full adders. The maximum operation is implemented by considering the following,

$$M = \max\{v_1, v_2\} = \begin{cases} v_1 & \text{if } v_1 \geq v_2, \\ v_2 & \text{otherwise,} \end{cases} = \begin{cases} v_1 & \text{if } v_1 - v_2 \geq 0, \\ v_2 & \text{otherwise.} \end{cases} \tag{17}$$

To this end, by defining σ as the sign bit of $v_1 - v_2$,

$$M = (1 - \sigma)v_1 + \sigma v_2. \tag{18}$$

12. Summary of paper E "Secure learning-based MPC via garbled circuit"
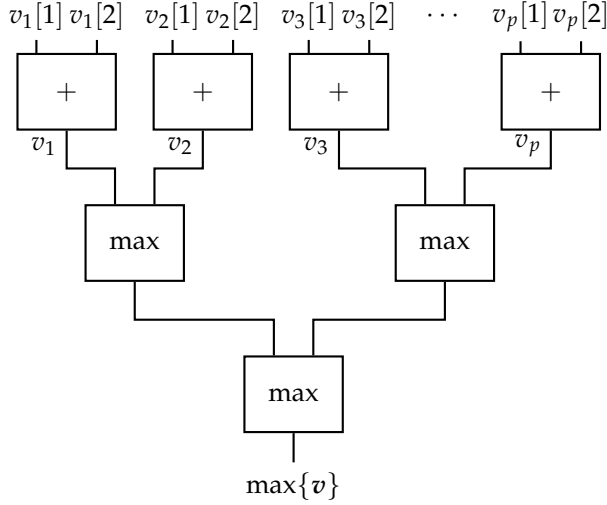
Fig. 25: Illustration of the max-out circuit for evaluating $\max\{v\}$ based on additive shares $v[1]$ and $v[2]$.

(18) can be efficiently evaluated in the boolean circuit by expressing the j'th bit of M as

$$M_j = \text{XOR}(\text{AND}(v_{1j}, \text{NOT}(\sigma)), \text{AND}(v_{2j}, \sigma)). \tag{19}$$

Immediately, several obstacles arise in the garbled circuit concerning the reconstruction of v in (15) and (16);

- The addition of the l bits using carry-ripple full adders may require one additional bit to represent the result (so the garbled circuit should be extended to represent $2^l + 1$ bit numbers).

- The addition of the l bits should be modular 2^l as seen in (15).

- After applying the addition and modular operators, $\tilde{v}$ in (15) is achieved. To reconstruct v, (16) must be computed.

However, it turns out that the additive secret sharing scheme and the binary circuit fits nicely together. Namely, the first two obstacles cancel each other out; it is not necessary to add an additional bit to the circuit as this action immediately would be cancelled by the modular 2^l operation. Or the other way around, the modular 2^l operation is carried out by simply considering only the 2^l least significant bits of $\tilde{v}$. Even more remarkably, the representation of $v \in \mathbb{F}_{2^l}$, (corresponding to $v \in [-\frac{2^l}{2}, \frac{2^l}{2} - 1]$) translates perfectly into the two's complement representation of 2^l bit numbers, where the most significant bit is the sign bit. More precisely, it is not necessary to carry out (16) after the addition, since if v is negative its most significant bit is set

corresponding to how a negative number is represented in the binary format. To this end, the garbled circuit can be constructed based on the circuit illustrated in Fig. 25 without further modifications. The output of computing party 1 is the garbled result of $\max\{v\}$ and the decode information to $\max\{w\}$ and from computing party 2, the garbled result of $\max\{w\}$ and the decode information to $\max\{v\}$. To this end, the input providing party can decode the garbled outputs and perform the subtraction to reach the solution to (14).

12.1 Results

As already mentioned, when the finite field consists of 2^l elements and the binary circuit takes l-bit numbers, there is a quite nice fit between additive secret sharing and garbling. For instance, performing modular 2^l in the binary circuit, is achieved simply by keeping the 2^l least significant bits and discarding the rest. Also, representing negative numbers in a finite field is usually done by defining the interval $[|\frac{q}{2}|, q-1]$ as the negative numbers. When the finite field consists of 2^l elements, a negative number is thus defined by the most significant bit being set, which is also how negative numbers are represented in the binary two's complement format. To this end, shares based on $\mathbb{F}_{2^l}$ translates effortlessly into binary inputs to a garbling scheme.

To evaluate the computation time, the solution is simulated and the execution time is measured. The data used in the simulation can be seen in paper E. For comparison, experiments are done for both $p = 8$ and $p = 16$ (where p is the number of entries in the vectors v and w) and for both cases, $l = 16$ and $l = 32$ is used. In table 2, the number of AND gates in the garbled circuit is seen as well as the average execution time for computing (14) with the privacy preserving solution.

p	l	#AND	t_{avg}
8	16	480	79 ms
	32	960	167 ms
16	16	992	170 ms
	32	1984	348 ms

Table 2: The number of AND gates and average computing time for computing (14) using the proposed privacy preserving method.

As seen from table 2, the execution times increase as the garbled circuit expands, which was expected. For reference, computing the control input in approximately 0.1 second should be sufficient to ensure the stability of the controller. Thus, sacrificing a bit of accuracy by choosing $p = 8$ and $l = 16$,

the proposed solution is a nice first step in the direction of privacy preserving model predictive control.

13 Discussion

The presented problem and solutions are all dependent on finite field arithmetic and consequently, are restricted to using only integers in all computations. Usually, the numbers involved in computations like least squares estimation, optimization and control calculations are real numbers and not integers. Mostly, this is not a crucial matter, since the real numbers can be scaled before rounding to an integer.

In paper E, it is shown that the error induced by scaling can be bounded and be made almost arbitrarily small. Yet, since computations take place in a finite field, the scaled numbers should be represented without "wrap around". Thus, the larger the scaling constant the larger the finite field has to be, which in the end might not be practical. Thus, there are limits to the precision of the scaled integers.

Another concern is that only finite field arithmetic can be used when operating in the finite field. This works nicely for addition, subtraction and multiplication, at least when discounting "wrap arounds". However, division in the finite field does not really translate well to real number division as shown in section 2.1 and 10. These 4 operations are the only operators defined on a finite field, thus any other operation (square root, exponentiation, and so on) cannot trivially be performed in the finite field.

In this thesis, the problems regarding the finite field is most clearly present in paper C, regarding privacy preserving recursive least squares (RLS). The proposed algorithm illustrates both problems; its convergence rate is irregular due to the rounding of reals to integers, and its computation and communication complexities are quite heavy due to the need for division. Moreover, our preliminary investigation suggests that some algorithms are very challenging to implement in this privacy preserving framework. For instance, the Kalman filter involves a fraction between zero and one, and representing this fraction as either zero or one does not provide convergence of the algorithm.

In conclusion, privacy preserving algorithms not based on finite field arithmetic would eliminate many of these problems. As nothing comes for free, it is interesting to investigate the costs of such a framework. This question is what the last part of this thesis is concerned with.

Real Number Secret Sharing

14   Secret sharing without finite field arithmetic

As already shown in this thesis, many privacy preserving works deal with rounding reals to integers before being able to perform privacy preserving calculations on the data, [47, 48]. The reason is that most of the established cryptographic techniques for privacy preserving computations rely on finite fields and finite field arithmetic, making it necessary to represent secrets as integers. However, there are not many examples of applying the privacy preserving methods in real world scenarios, perhaps due to their usual heavy overhead in computation and/or communication, or perhaps because it is difficult to restrict computations in real world scenarios to finite field computations. To this end, there is a need for considering trade-offs that potentially lead to more suitable algorithms for real world applications. In this chapter, the tradition that privacy preserving computations must rely on finite field arithmetic is challenged. The problem studied in this chapter is how to state a real number secret sharing scheme that is usable for SMPC and does not rely on finite fields. To achieve this, the approach is to sacrifice perfect privacy to hopefully gain a more efficient and practical scheme. Consequently, part of the question to be answered is how to quantify the privacy of such a scheme.

   This chapter aims to investigate the following research question.

   - How could a trade-off be introduced, where perfect privacy is sacrificed in order to lower the cost of privacy preserving computations?

      - How can privacy be quantified?

15 Summary of paper F "Privacy in Distributed Computations based on Real Number Secret Sharing"

This work proposes a real number secret sharing scheme, which does not rely on finite field arithmetic. The scheme is much related to Shamir's secret sharing scheme, with a few adaptions making it more suitable for real numbers. The reason for basing the real number scheme on the idea of Shamir's scheme, is due to its great properties for making calculations directly on the shares. To this end, with the real number secret sharing scheme follows straight forward implementations of addition, subtraction, multiplication and division of the shares. As already touched upon, the price to pay for all this is a lack of perfect privacy of the secret data. In the remaining part of this summary, the proposed scheme is presented, and the algorithms for the computations on the shares are sketched. Conclusively, the key points of the privacy analysis are given.

The proposed real number secret sharing scheme consists of a `share` algorithm that generates shares of the secret s, which can be distributed among n participants. The participants are labeled with distinct elements from some set of numbers $\mathcal{P}$, for instance $1, 2, \ldots, n$ or n values in the interval $[2, 4]$. The algorithm takes a secret s, the threshold t (which has the same role as in Shamirs scheme) and the labels of the participants, $\mathcal{P}$. The output of `share` is a share $s[i]$ for each participant i. The `share` algorithm is sketched as a block diagram in Fig. 26. As seen, the algorithm uses two parameters, μ_Y and σ_Y^2, which can be seen as privacy parameters of the scheme. μ_Y does not significantly affect the privacy assuming it is unknown to the adversary. The role of σ_Y^2 will become clear in the privacy analysis.

The reconstruct algorithm, `recon`, takes at least $t + 1$ shares and outputs the reconstructed secret. The algorithm is nothing more than Lagrange inter-
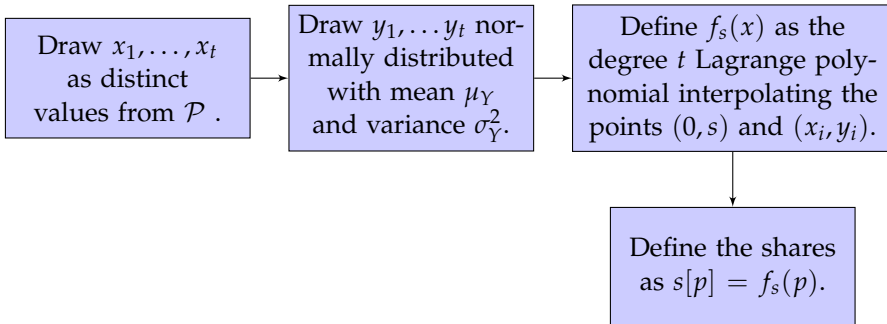
Fig. 26: Block diagram of the `share` algorithm of paper F.
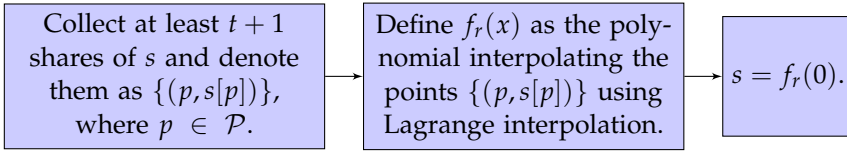
Fig. 27: Block diagram of the `recon` algorithm of paper F.

polation, and it is sketched in the block diagram in Fig. 27.

Performing computations on the shares is straight forward; addition, subtraction and multiplication are performed similarly as in Shamir's scheme, and division is performed using only three local computations and one broadcast. The block diagram for the division algorithm is in Fig. 28, seen from the view of participant $p \in \mathcal{P}$.
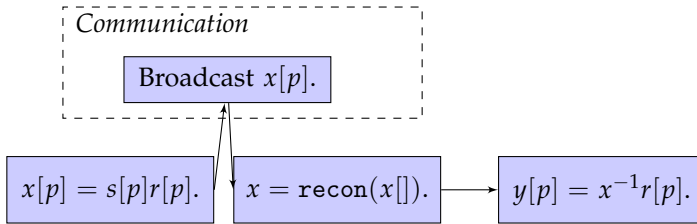
Fig. 28: Block diagram of the algorithm for the inversion of a secret performed on shares from the view of participant $p \in \mathcal{P}$.

Privacy analysis

In the privacy analysis, the view of the adversary is considered. That is, the secret, s, is considered as a random variable having some distribution that reflects the prior knowledge the adversary may have about the secret. To this end, s is here modeled as the outcome of a random variable S, and the uncertainty the adversary has about s can be expressed as the differential entropy of S, $h(S)$.

Assuming t participants are corrupted, the adversary has access to t shares of s. Each share $s[p]$, $p \in \mathcal{P}$, is also the outcome of a random variable $S[p]$.

Intuitively, the privacy of the scheme is expressed as the reduction in uncertainty, the adversary has about the secret after he learns the shares of the corrupted parties. This reduction is optimally zero (which is the case for Shamir's scheme and the additive scheme for instance), but as discussed previously, here we cannot do better than to make it as small as possible which will still be larger than zero. To state this result, consider first the following description of a share (reproduced from paper F, [49]).

$$s[p] = sL_0(p) + \underbrace{\sum_{j \in T} y_j \frac{p}{x_j} L_j(p)}_{b(p)} = sL_0(p) + b(p),$$ (20)

where $T = \{1, \ldots, t\}$ and $L_j(x)$ are Lagrange basis polynomials (see Paper F for the details).

Note that, (20) yields that

$$S[p] = SL_0(p) + B(p).$$ (21)

The reduction in uncertainty is quantified using the mutual information between S and a set of t shares, yielding;

$$I(S; S[1], \ldots, S[t]) = h(S[1], \ldots, S[t]) - h(B(1), \ldots, B(t))$$ (22)

To make the notation more concise, let X_S, X_B denote $(S[1], \ldots, S[t])$ and $(B(1), \ldots, B(t))$, respectively and remark the following:

1. the distribution of X_s is "unknown" in this analysis as it depends on the knowledge of the adversary,

2. increasing the value of $h(X_S)$ increases $I(S; X_S)$ in (22),

3. the distribution of X_B is t-variate Gaussian when considering the $L_j(p)$ (in (20)) as known constants.

Following the first two remarks, the mutual information $I(S, X_S)$ can be upper bounded by considering the maximum entropy distribution, (which on the real number line is the Gaussian distribution) as the distribution for X_S. In continuation, a N-variate Gaussian distributed variable X has the following entropy [50, p.249];

$$h(X) = \frac{1}{2} \log \left((2\pi e)^N \det(C_X) \right),$$ (23)

where C_X is the covariance matrix for X. Using this in (22) yields

$$\begin{aligned} I(S; X_S) &\leq \frac{1}{2} \log \left((2\pi e)^t \det(C_{X_S}) \right) - \frac{1}{2} \log \left((2\pi e)^t \det(C_{X_B}) \right) \\ &= \frac{1}{2} \log \frac{\det(C_{X_S})}{\det(C_{X_B})} \end{aligned}$$ (24)

To this end, the main result of the privacy analysis is an upper bound on the mutual information and equivalently the leaked information about the secret from a set of t shares. The upper bound is dependent on σ_Y^2 in Fig. 26, since the determinant of C_{X_B} is affected by σ_Y^2. Consequently, increasing σ_Y^2
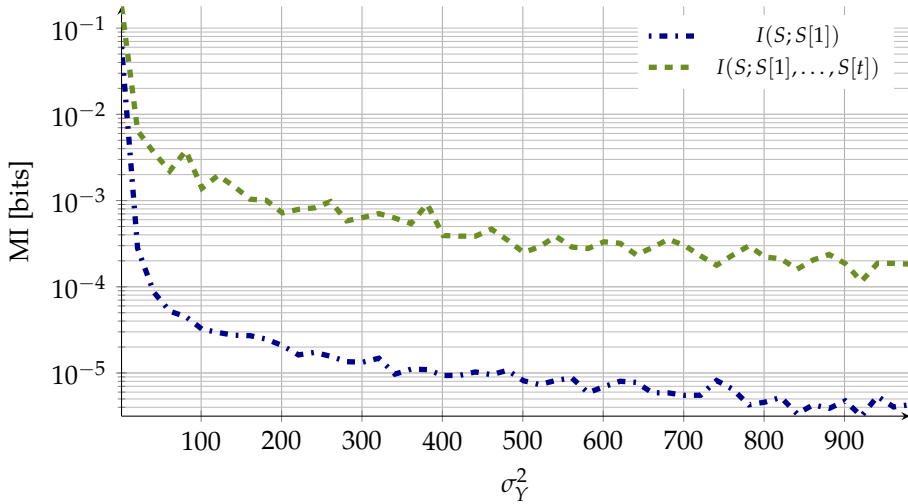
Fig. 29: Estimated mutual information (MI) between a standard normal distributed secret S and one share of s and t shares of s, respectively. Partly reproduced from paper F, [49].

decreases the mutual information. This result is validated numerically in Fig. 29 by estimating the privacy loss caused by one share, $S[1]$, and t shares, $S[1], \ldots, S[t]$. That is, a large sample size of each variable is simulated for each estimation and the mutual information is estimated using the python package NPEET [51]. For each estimation, the secret, S, follows a standard normal distribution, and the shares are computed based on the secret and a varying σ_Y^2. As seen, the mutual information (and hence the privacy loss) decreases as σ_Y^2 increases.

15.1 Results

The paper puts forward a real number secret sharing scheme, which can generate shares of decimal secrets and reconstruct them again. Also, addition, multiplication and division can be performed directly on the shares. All this is shown theoretically. To evaluate how the scheme performs practically, the reconstruct algorithm, and the protocols for addition, multiplication, and inversion is simulated. The chosen parameters are; 11 participants, the threshold $t = 5$, the secret $s_1 = 5.5$, and the second secret $s_2 = 34.7$ for the multiplication protocol. The share algorithm is used to generate shares of the secrets, where the value of σ_Y^2 is varied. That is, the accuracy of the algorithms is evaluated in terms of the variance of the y_j values in share in Fig. 26. The root square error (RSE) between the expected result, v, and the outcome of the algorithms $\hat{v}$ is depicted in Fig. 30 (reproduced from paper
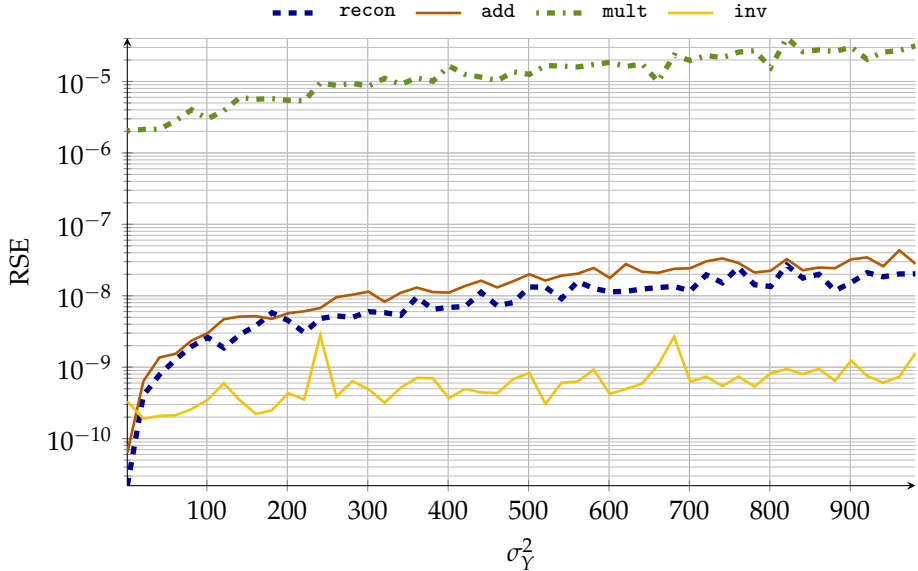
Fig. 30: RSE between the expected result and the reconstructed result for the algorithms `recon`, `add`, `mult`, and `inv` in terms of σ_Y^2. Reproduced from paper F, [49].

F, [49]), where the RSE is defined as

$$\text{RSE} = \sqrt{(v - \hat{v})^2}.$$

As seen in Fig. 30, the accuracy decreases as σ_Y^2 increases. Theoretically, there should be no issue with an arbitrarily large σ_Y^2. However, as σ_Y^2 grows, the numerical value of the shares grows, at least generally speaking. Consequently, precision is lost due to the floating point representation of reals used by computer systems. The reason why `mult` performs the worst is because in that protocol, shares are multiplied and becomes even larger, which decreases precision. On the other hand, in the inversion protocol, shares are scaled by a number between 0 and 1 as seen in Fig. 28. This makes the shares numerically small, which is why this protocol performs the best. To this end, when it comes to practical usage of the real number secret sharing scheme, there is in fact a trade off between output utility and privacy.

Finally, it is interesting to compare the performance of the real number secret sharing scheme in computing the recursive least squares equations to the performance of the solution in paper C based on Shamir's scheme. It is not even necessary to illustrate the solution based on the real number secret sharing scheme in a block diagram, since it is straight forward to perform all computations in the RLS equations (see (6)) using the scheme.

15. Summary of paper F "Privacy in Distributed Computations based on Real Number Secret Sharing"

To compare the two solutions, simulations of both methods are performed, when solving the same problem. To evaluate their performances, the mean squared error (MSE) between the estimate of the parameters and the true parameters is used. The MSE is at time k defined as

$$\text{MSE}_k = \frac{1}{m} \sum_{j=1}^{m} (w_j - \hat{w}_j)^2,$$

where m is the number of parameters to be estimated. The MSE of the solution in paper C is referred to as $\text{MSE}_k^{\text{Shamir}}$ and the MSE of the solution based on the real number secret sharing scheme is referred to as $\text{MSE}_k^{\text{RealNumber}}$.

For the simulation, the test data is described by

$$y = 3.5x_1 + 1.2x_2 + 2.8x_3 + 4.1x_4 + 2.9x_5 + 3.3x_6, \tag{25}$$

and the observations of $\{x_i\}_{i=1,\dots,6}$ are simulated as uniformly distributed on the interval $[0,5]$. The observations of y are given by (25) and afterwards corrupted by Gaussian noise with mean value zero and variance one.

Fig. 31 shows $\text{MSE}_k^{\text{Shamir}}$ and $\text{MSE}_k^{\text{RealNumber}}$. Note that the observations of y are real numbers, which can be directly used in the solution based on the real number secret sharing scheme, but they are scaled and truncated before used in the solution based on Shamir's scheme. As seen by the figure, the solution based on the real number secret sharing scheme delivers a much more smooth and fast convergence, thus outperforming its counterpart perfectly secure version.
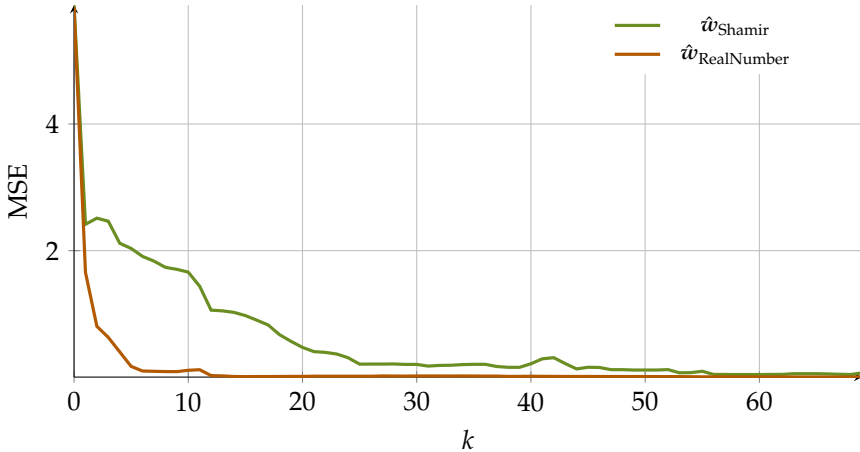
Fig. 31: MSE of the parameters estimated with the privacy preserving RLS equations based on the real number secret sharing scheme, $\hat{w}_{\text{RealNumber}}$ and the MSE of the parameters estimated with the privacy preserving RLS equations proposed in paper C, [30].

16 Discussion

The paper proposes a real number secret sharing scheme that handles decimal numbers and computes both addition, subtraction, multiplication, and division effectively directly on shares. The scheme is easy to implement and straight forward to use in algorithms that are based on addition, subtraction, multiplication and division. However, nonlinear operations are non-trivial to perform directly on shares using the scheme.

The disadvantages of the scheme is that in practice, a trade-off between accuracy and privacy emerges, which is due to the finite precision of computer systems. Additionally, the scheme is not perfectly secure, meaning that private information is leaked about the secrets from the shares. However, the leak can be upper bounded, which makes it possible to before hand calculate in average how many bits of information are leaked when sharing a secret using the scheme.

In conclusion, the real number secret sharing scheme as presented in paper F, is a good first step towards more practically useful schemes for solving the problem of SMPC in real world scenarios.

Conclusion and Outlook

This thesis is concerned with the privacy of data in distributed computations. That is, performing joint computations on a distributed dataset without revealing the individual datasets is investigated. To this end, multiple privacy preserving distributed algorithms with different network setups are proposed. Conclusively, a real number secret sharing scheme is introduced that addresses some of the disadvantages of the previously used methods. Based on the research, the following main conclusions are drawn.

- Using public key encryption together with secret sharing expands the use of SMPC techniques to networks that does not support fully connected communication between nodes. Exploiting the properties of the communication network can reduce execution time, assuming the nodes themselves have the knowledge about the network.

- The price to pay for privacy preservation in distributed computations using SMPC methods are generally an overhead in computation and communication complexity as well as a minor degradation in output accuracy. The latter comes from the (usual) need to round real numbers to integers as a part of the privacy preserving protocol. Additionally, using SMPC methods for other operations than addition, subtraction and multiplication is intractable. Thus, yet there is no all-purpose privacy preserving framework that allows all algorithms to be translated into an equivalent privacy preserving one and consequently, each algorithm desired to be privacy preserving (if even possible) must be translated one at time.

- Perfect privacy can be traded to lower the cost of privacy in algorithms, e.i. reducing computations and communication, improving output accuracy, and extending applicability. This trade-off can be introduced by the real number secret sharing scheme presented in paper F that bypasses the requirement to round real numbers to integers (and thus increases output accuracy) and computes addition, subtraction, multiplication, and division directly on cipher texts, thus expands on the

number of operations that can be done effectively while preserving privacy, at the cost of a degradation in privacy.

- Privacy can be quantified using information theoretic measures. Particularly, entropy and mutual information are candidates for measuring privacy in bits. That is, privacy can be measured by considering the prior uncertainty one has about the private data (the entropy of the data) and the posterior uncertainty one has after the execution of the privacy aware algorithm. The difference between the two is the on average privacy loss expressed in bits.

Even though this thesis deals with algorithms that are applied in real world settings, the work is of theoretical nature and is more a theoretical proof-of-concept. The question of how well the privacy preserving algorithms perform in practical settings is therefore unanswered. To this end, the actual computation time of the proposed methods and the scalability of them, is still to be determined.

To address, the most likely quite heavy increase in execute time (compared to a non-private solution), a trade-off between privacy and efficiency is introduced with the real number secret sharing scheme in paper F. However, there are still some open questions related to this work. Namely, the privacy of the real number secret sharing scheme is analyzed first of all by considering a set of shares and quantifying the average leak from shares. To produce those results, the information theoretic measure called entropy and mutual information is used. It is still an open question whether there exists (or can be made) a more suiting measure, since for some calculations the mutual information measure becomes intractable. Also, it is not clear how to interpret the mutual information measure and it is difficult to get intuition about what it means that 0.5 bits are leaked for instance.

The work in this thesis takes the approach of combining cryptographic methods into already existing optimization algorithms. The advantage is that privacy of the cryptographic methods has already been proved and convergence of the algorithms has also already been established. The disadvantage is that the privacy preserving techniques are not designed to be used in distributed algorithms. To this end, another interesting approach is to investigate the possibility of embedding privacy preservation directly into distributed algorithms. This work is in fact already begun by Li et al. in [17, 52] where careful noise insertion is used to achieve privacy in distributed convex optimization and in distributed graph filtering, respectively.

References

[1] "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," *Official Journal of the European Union*, vol. L 119, pp. 1–89, 2016.

[2] G. Avoine, L. Calderoni, J. Delvaux, D. Maio, and P. Palmieri, "Passengers information in public transport and privacy: Can anonymous tickets prevent tracking?" *International Journal of Information Management*, vol. 34, p. 682–688, 10 2014.

[3] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 2008, pp. 111–125.

[4] D. Davies, "A brief history of cryptography," *Information Security Technical Report*, vol. 2, no. 2, pp. 14–17, 1997.

[5] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[6] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[7] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[8] S. Goldwasser, "Multi party computations: past and present," in *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, 1997, pp. 1–6.

[9] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, pp. 160–164.

[10] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC '87. New York, NY, USA: Association for Computing Machinery, 1987, p. 218–229. [Online]. Available: https://doi.org/10.1145/28395.28420

[11] J. C. Benaloh, "Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract)," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 251–260.

[12] D. Chaum, C. Crépeau, and I. Damgard, "Multiparty unconditionally secure protocols," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 11–19. [Online]. Available: https://doi.org/10.1145/62212.62214

[13] T. Rabin and M. Ben-Or, "Verifiable secret sharing and multiparty protocols with honest majority," in *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, ser. STOC '89. New York, NY, USA: Association for Computing Machinery, 1989, p. 73–85. [Online]. Available: https://doi.org/10.1145/73007.73014

[14] D. Chaum, "The spymasters double-agent problem," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed. New York, NY: Springer New York, 1990, pp. 591–602.

[15] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," in *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, ser. STOC '90. New York, NY, USA: Association for Computing Machinery, 1990, p. 503–513. [Online]. Available: https://doi.org/10.1145/100216.100287

[16] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft, "Secure multiparty computation goes live," in *Financial Cryptography and Data Security*, R. Dingledine and P. Golle, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 325–343.

[17] Q. Li, M. Coutino, G. Leus, and M. G. Christensen, "Privacy-preserving distributed graph filtering," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2155–2159.

[18] Q. Li, I. Cascudo, and M. G. Christensen, "Privacy-preserving distributed average consensus based on additive secret sharing," in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[19] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, March 2019.

[20] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *PoPETs*, vol. 2017, pp. 345–364, 2017.

[21] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 334–348.

[22] T. Ryffel, P. Tholoniat, D. Pointcheval, and F. Bach, "Ariann: Low-interaction privacy-preserving deep learning via function secret sharing," 2021.

[23] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[24] C. Dwork and A. Smith, "Differential privacy for statistics: What we know and what we want to learn," *Journal of Privacy and Confidentiality*, vol. 1, pp. 135–154, 01 2009.

[25] J. Lee and C. Clifton, "How much is enough? choosing ϵ for differential privacy," in *Information Security*, X. Lai, J. Zhou, and H. Li, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 325–340.

[26] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 308–318. [Online]. Available: https://doi.org/10.1145/2976749.2978318

[27] A. D. Sarwate and K. Chaudhuri, "Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 86–94, 2013.

[28] J. Cortés, G. E. Dullerud, S. Han, J. Le Ny, S. Mitra, and G. J. Pappas, "Differential privacy in control and network systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 4252–4272.

[29] J. Justesen and T. Hoholdt, *A Course in Error-Correcting Codes (EMS Textbooks in Mathematics)*. European Mathematical Society, 2004.

[30] K. Tjell, I. Cascudo, and R. Wisniewski, "Privacy preserving recursive least squares solutions," in *2019 18th European Control Conference (ECC)*. United States: IEEE, Aug. 2019, pp. 3490–3495, null ; Conference date: 25-06-2019 Through 28-06-2019.

[31] R. Cramer, I. B. Damgrd, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, 1st ed. New York, NY, USA: Cambridge University Press, 2015.

[32] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, ser. Lecture Notes in Computer Science, vol. 576. Springer, 1991, pp. 420–432.

[33] K. Yang, X. Wang, and J. Zhang, "More efficient mpc from improved triple generation and authenticated garbling," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1627–1646. [Online]. Available: https://doi.org/10.1145/3372297.3417285

[34] D. Rathee, T. Schneider, and K. K. Shukla, "Improved multiplication triple generation over rings via rlwe-based ahe," in *Cryptology and Network Security*, Y. Mu, R. H. Deng, and X. Huang, Eds. Cham: Springer International Publishing, 2019, pp. 347–359.

[35] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 784–796. [Online]. Available: https://doi.org/10.1145/2382196.2382279

[36] B. Schoenmakers, *Oblivious Transfer*. Boston, MA: Springer US, 2011, pp. 884–885. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_9

[37] K. Tjell, N. Schlüter, P. Binfet, and M. Darup, "Secure learning-based mpc via garbled circuit," in *2021 IEEE 60th Conference on Decision and Control (CDC)*, Aug. 2021.

[38] C. Orlandi, "Is multiparty computation any good in practice?" in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5848–5851.

[39] I. Damgård, C. Orlandi, and M. Simkin, "Yet another compiler for active security or: Efficient mpc over arbitrary rings," in *Advances in Cryptology – CRYPTO 2018*, H. Shacham and A. Boldyreva, Eds. Cham: Springer International Publishing, 2018, pp. 799–829.

[40] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[41] K. Tjell and R. Wisniewski, "Private aggregation with application to distributed optimization," *IEEE Control Systems Letters*, vol. 5, pp. 1591 – 1596, 2021.

[42] M. Dahl, C. Ning, and T. Toft, "On secure two-party integer division," in *Financial Cryptography and Data Security*, A. D. Keromytis, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 164–178.

[43] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *Public Key Cryptography – PKC 2007*, T. Okamoto and X. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 343–360.

[44] K. Tjell and R. Wisniewski, "Privacy preservation in distributed optimization via dual decomposition and admm," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, ser. I E E E Conference on Decision and Control. Proceedings. United States: IEEE, Mar. 2020, pp. 7203–7208, 2019 IEEE 58th Conference on Decision and Control (CDC), CDC ; Conference date: 11-12-2019 Through 13-12-2019.

[45] B. He, L. Hou, and X. Yuan, "On full jacobian decomposition of the augmented lagrangian method for separable convex programming," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2274–2312, 2015. [Online]. Available: https://doi.org/10.1137/130922793

[46] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1561/2200000016

[47] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314 – 325, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109818303510

[48] Q. Li and M. G. Christensen, "A privacy-preserving asynchronous averaging algorithm based on shamir's secret sharing," in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[49] K. Tjell and R. Wisniewski, "Privacy in distributed computations based on real number secret sharing," 2021, submitted to Information Sciences.

[50] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. USA: Wiley-Interscience, 1991.

[51] G. V. Steeg, "Npeet," https://github.com/gregversteeg/NPEET, 2019.

[52] Q. Li, R. Heusdens, and M. Christensen, "Privacy-preserving distributed optimization via subspace perturbation: A general framework," *I E E E Transactions on Signal Processing*, vol. 68, pp. 5983 – 5996, 2020.

Part II

Papers

Paper A

Privacy Preserving Distributed Summation in a
Connected Graph

Katrine Tjell, Rafael Wisniewski

Abstract

Most decentralized algorithms for multi-agent systems used in control, signal processing and machine learning for example, are designed to fit the problem where agents can only communicate with immediate neighbors in the network. For instance, decentralized and distributed optimization algorithms are based on the fact that every agent in a network will be able to influence every other agent in the network even if each agent only communicates with its immediate neighbors (given that the network is connected). That is, a distributed optimization problem can be solved in a decentralized manner by letting the agents exchange messages with their neighbors iteratively. In many algorithms that solve this kind of problem, agents in the network does not need individual values from their neighbors, rather they need a function of the values from its neighbors. This observation makes it interesting to consider privacy preservation in such algorithms. By privacy preservation, we mean that raw data from individual agents will not be exposed at any time during calculations.

1   Introduction

Multi-agent systems with a decentralized graph topology appear in many areas such as; formation control, distributed resource allocation, workload balancing and energy optimization. Thus, many decentralized algorithms for solving different problems in a distributed and decentralized fashion has been proposed. For some of these algorithms, each agent are only required to communicate with immediate neighbors and moreover; each agent does not necessarily need to learn individual values from neighboring agents, rather they need to learn the sum of neighboring agents values. Such algorithms can for instance be seen in the work by [2], [3], and [4]. Other examples include the work by [5] that considers decentralized estimation of Laplacian eigenvalues in multi-agent systems and the work by [6] that proposes a communication efficient algorithm for resource-aware exact decentralized optimization.

This paper considers privacy preservation of agents in a multi-agent network where each agent needs to learn the sum of its neighboring agents values in order to carry out required local computations. Specifically, we are interested in the case where agents are unwilling to share raw data, perhaps because the data is sensitive from a business perspective or because it leaks personal information. Motivated by the many decentralized algorithms only requiring agents to learn the sum of neighboring agents values, we investigate the privacy preserving calculation of

$$\sum_{j \in \mathcal{N}_i} x_j, \tag{A.1}$$

where $\mathcal{N}_i$ is the set of indices of neighbors to agent a_i and x_j is a value known only to agent a_j.

To give thorough motivation for the problem, consider the minimization problem

$$\begin{aligned}\underset{x}{\text{minimize}} \quad &\sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \quad &\sum_{i=1}^{N} B_i x_i - c = 0,\end{aligned} \qquad (A.2)$$

where $x = [x_1, \ldots, x_N]$, x_i is a local variable, f_i is a local objective and B_i and c are constraints. For solving the problem distributed and decentralized, [3] proposes the following steps:

$$p_i^{(k)} = p_i^{(k-1)} + \rho \sum_{j \in \mathcal{N}_i} \left(v_i^{(k-1)} - v_j^{(k-1)} \right), \qquad (A.3)$$

$$\begin{aligned}x_i^{(k)} = \arg\min_{x_i} \Bigg\{ f_i(x_i) &+ \frac{\rho}{4|\mathcal{N}_i|} \left\| \frac{1}{\rho}(B_i x_i - \frac{1}{N}c) \right. \\ &- \frac{1}{\rho} p_i^{(k)} + \sum_{j \in \mathcal{N}_i} \left(v_i^{(k-1)} + v_j^{(k-1)} \right) \left\|_2^2 \right\} ,\end{aligned} \qquad (A.4)$$

$$\begin{aligned}v_i^{(k)} = \frac{1}{2|\mathcal{N}_i|} \Bigg(&\sum_{j \in \mathcal{N}_i} \left(v_i^{(k-1)} + v_j^{(k-1)} \right) \\ &- \frac{1}{\rho} p_i^{(k)} + \frac{1}{\rho} \left(B_i x_i - \frac{1}{N}c \right) \Bigg),\end{aligned} \qquad (A.5)$$

that are performed locally by each agent. As can be seen, values of neighboring agents appear in all three equations. In (A.3), the values appear as

$$\sum_{j \in \mathcal{N}_i} \left(v_i^{(k-1)} - v_j^{(k-1)} \right) = |\mathcal{N}_i| v_i^{(k-1)} - \sum_{j \in \mathcal{N}_i} v_j^{(k-1)}, \qquad (A.6)$$

and in (A.4) and (A.5) as

$$\sum_{j \in \mathcal{N}_i} \left(v_i^{(k-1)} + v_j^{(k-1)} \right) = |\mathcal{N}_i| v_i^{(k-1)} + \sum_{j \in \mathcal{N}_i} v_j^{(k-1)}. \qquad (A.7)$$

Evidently, if $\sum_{j \in \mathcal{N}_i} v_j^{(k-1)}$ can be computed without leaking individual v_j values, the algorithm is privacy preserving. Hence, the aim of this paper is to, for each agent in a multi-agent system, compute the sum of its neighboring agents values without individual terms of the sum being exposed.

Related work. Computing the sum of private values among a set of participants, without leaking the private values, is a well-known problem within the

field of cryptography. Most secret sharing schemes (see the work by [7]) such as Shamir's secret sharing scheme and the additive secret sharing scheme are able to compute *shares* of the sum of secret input values from *shares* of the input values. The same goes for more or less all the homomorphic encryption schemes, see [8]. All though our proposed solution is based both on secret sharing and encryption, the paper distinguishes it self by being applied to agents in a graph network. Thus, we do not assume that all agents can communicate or that there exist a central node which all agents can communicate with, which is the general assumption in this kind of work. Our work also distances it self from traditional secret sharing and homomorphic encryption based approaches, as it includes a preprocessing phase which speed up efficiency at the actual execution time.

Many works (also outside cryptography) are occupied with the privacy preserving summation of values, for instance [9], [10] and [11]. The work by [9] proposes a secure sum protocol for computing a sum of N private values among N participants. Their solution relies on an (untrusted) moderator to ensure privacy. Our proposed method does not require a moderator and for the setup considered in this paper, it is not a viable solution to adopt moderators in the network.

Structure. Section 2 states the problem of the paper in detail and gives the necessary preliminaries. The main content of the paper is in sections 3 and 4, where the problem is solved with two different assumptions on the graph topology. Section 5 gives an illustration of the scale-ability of the proposed method and finally, section 6 concludes the paper.

2 Problem Formulation

Consider a multi-agent network consisting of N agents, $a_1, \ldots, a_N$, with a certain communication network linking them. Fig. A.1 depicts such a multi-agent network, where the vertices are the agents and the edges illustrates the communication network. We define $\mathcal{N}_i$ as the neighbors to agent a_i; that is, $\mathcal{N}_i$ is the set of indices j of agents a_j where there is an edge between a_i and a_j. Note that we consider a_i to be a neighbor to itself.

Furthermore, each agent a_i has a private value x_i. The aim is to compute the sum

$$y_i = \sum_{j \in \mathcal{N}_i} x_j, \tag{A.8}$$

for each agent a_i in the multi-agent network. As each agent can communicate with all of their neighbors, the problem is trivially solved be letting all agents a_j for $j \in \mathcal{N}_i$ send their value x_j to agent a_i, who can then compute the sum. However, in this paper we are interested in the case where agents are unwilling to hand over raw data, for instance because the data is considered
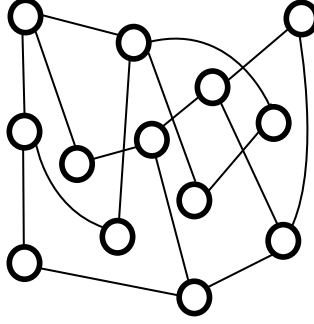
Fig. A.1: A multi-agent network, where agents are depicted as vertices and the communication channels are depicted as edges.

corporate secrets or simply because it leaks private information. Thus, the goal is that agent a_i learns only y_i and not the individual terms x_j in the sum. For this to be possible, we assume that all agents has at least two neighbors besides from itself, that is $|\mathcal{N}_i| \geq 3 \, \forall \, i$. Furthermore, we assume that agents will follow the protocol, however they may collude in attempting to disclose the secret values of other agents. To this end, we assume that a majority of an agents neighbors are not colluding, which will ensure the privacy of the honest agents.

We consider two different scenarios with respect to graph topology, we will refer to these as problem 1 (P1) and problem 2 (P2). The difference between P1 and P2 is that in P1, we assume that in each neighborhood $\mathcal{N}_i$, the agents $\{a_j\}_{j \in \mathcal{N}_i}$ form one or more *cliques*. A clique is a fully connected subset of a graph, thus in other words, we assume in P1, that each a_j, $j \in \mathcal{N}_i$ is connected to at least one other agent $a_{j'}$, $j' \in \mathcal{N}_i$, where $j \neq \{i, j'\}$. On the other hand, in P2, we make no assumptions on cliques in the neighborhood of a_i. In the following, P1 and P2 are more formally presented.

Problem A.1 (P1)

Let $a_1, \ldots, a_N$ be agents in a multi-agent network and assume that the agents in $\mathcal{N}_i$ form one or more cliques with cardinality at least three, with all of them including a_i. Then, the problem is to calculate (A.8) *for all agents a_i.*

Fig. A.2 is a simple example of the graph topology P1 is investigating. As seen, all agents has at least two neighbors (besides itself) and the neighbors of each agent forms at least one clique. For instance, for agent a_1, (a_1, a_2, a_3) forms a clique and so does (a_1, a_2, a_4).

For P2, we relax the constraint on the graph topology and accepts the case where not all neighbors to an agent is part of a clique.

Problem A.2 (P2)

For all agents, a_i, in an arbitrary connected multi-agent network, the problem is to calculate (A.8).
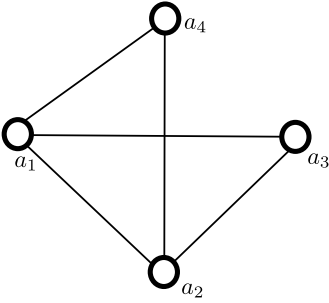
Fig. A.2: An example of a multi-agent network considered in P1.
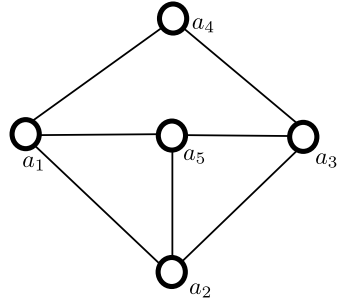
Fig. A.3: An example of a multi-agent network considered in P2.

Fig. A.3 is a simple example of the graph topology P2 is investigating. It can be seen that agent a_4 does not form a clique with other agents that are in the neighborhood of either a_1 or a_3.

The aim of the paper is to solve P1 and P2 without leaking the agents private values. For P1, we propose to use secret-sharing and two rounds of communication. This approach is evident under the assumption of cliques in the neighborhood. In the case of P2, where cliques in the graph are not assumed, we introduce a so-called virtual clique between agents in the same neighborhood, which is done with the use of encryption. In this way, we use the same method in P2 as in P1 but with the cost of additional communication rounds. This we see as a generalization of the proposed method, making it independent of the graph topology. Remark, P1 is a special case of P2, and is as such covered by the solution to P2. However, we believe that solving them one at a time will improve readability.

Notation

Let $\mathcal{N}_i$ denote the neighbors of agent a_i and $C_1, \ldots, C_m$ are the cliques in the multi-agent network, specifically, C_k is a set consisting of the indices j of agents a_j in the k'th clique. Moreover, Ci is the cliques agent a_i is part of; that is, $k \in Ci$ iff $a_i \in C_k$. Lastly, for each agent a_i, we will need to define the sets $C_{i,j}, j \in \mathcal{N}_i$, where for all $k \in Ci$, $k \in C_{i,j}$ iff $a_j \in C_k$. Note that $C_{i,j} = C_{j,i}$.

3 Privacy Preserving solution to P1

For solving P1 without leaking data, the idea is to compute the sum of values in each clique in the neighborhood of each agent. The sum is computed using secret sharing techniques, which will be key to preserve privacy in the protocol.

3.1 Secret Sharing

The aim of secret sharing is to share a secret among a set of participants such that none of the participants learn the secret and only by recombining the information of each participant, can the secret be reconstructed. In this way, an adversary attempting to learn the secret has to attack several entities instead of just one.

Dividing a secret into shares can be done in many ways. A simple scheme is called *additive secret sharing* and it has the property that the shares $s_1, \ldots, s_n$ of the secret s satisfies

$$s = s_1 + \cdots + s_n \mod p,$$

where p is a prime large enough that the probability of guessing s_i is negligible.

The modulo operator is used to make sure that each individual share does not leak information about s. Specifically, the shares are uniformly distributed on $0, \ldots, p-1$ and $p > s$. In our method, we will use a sharing of zero to achieve privacy.

3.2 Proposed method: Solution to P1

To present the idea in the method, consider a clique C_k, $k \in \{1, \ldots, m\}$. The approach is for each agent a_j for $j \in C_k$ to add a uniformly random number r_j to x_j before sending it to the other agents in the k'th clique. In this way, x_j is not revealed to the neighbors of a_j, since $x_j + r_j$ is a uniformly random number. Designing the random r_j values such that $\sum_{j \in C_k} r_j = 0$, ensures that the sum of the values in the cliques can still be learned by the agents in the clique.

To improve efficiency, we introduce a two-phase algorithm. The first phase is a *preprocessing phase* where the described uniformly random numbers are chosen; hence, this phase can be carried out in advance of the actual execution which boost efficiency at run-time. The second phase is the *execution phase*, where the private values are involved.

We explain the preprocessing phase with a numerical example, see Example A.1.

> **Example A.1**
> *Let a clique consist of the agents a_1, a_2 and a_3 and let $p = 23$. Table A.1 illustrates the steps in the preprocessing phase. The first step is that each agent chooses 3 random numbers which must sum to zero; this is shown in the first column. Then each agent sends one random number to each agent (including itself), the random number received by each agent is shown in the second column. The last column*

shows for each agent the sum of received numbers modulo 23, which is the result of the preprocessing phase. Note that the sum of the last column modulo 23 is zero.

	Choose uniformly random numbers	Recieved numbers	Sum of rec. numbers
Agent 1	0 = 15+5+3 mod 23	15,10,8	15+10+8 mod 23 = 10
Agent 2	0=10+6+7 mod 23	5,6,9	5+6+9 mod 23 = 20
Agent 3	0=8+9+6 mod 23	3,7,6	3+7+6 mod 23 = 16

Table A.1: Overview of the steps each agent take per clique in the preprocessing phase. In this example there is one clique consisting of three agents.

Example A.2 continues Example A.1, and shows the steps the agents take in the execution phase. As seen, all agents learn the sum of private values without exposing them.

Example A.2

This example is a continuation of Example A.1. Assume the agents have secret values, $x_1 = 5, x_2 = 2, x_3 = 10$. The goal is to find the sum of secret values in the clique, thus all agents should end up with learning the number 17, as $5 + 2 + 10 = 17$. Table A.2 illustrates the execution phase that would follow the preprocessing phase in Example A.1. The first step is for each agent to add their secret value to the random number generated in the preprocessing phase; this is shown in the first column. Then each agent sends this sum to all other agents; the second column shows the numbers each agent receives. The last colunmn shows the computed sum, which is the result of the execution phase.

	Add x_i to s_{i_1}	Received numbers	Sum of rec. numbers % 23
Agent 1 $x_1 = 5$	5+10 mod 23 = 15	15,22,3	17
Agent 2 $x_2 = 2$	2 +20 mod 23 = 22	15,22,3	17
Agent 3 $x_3 = 10$	10 + 16 mod 23 = 3	15,22,3	17

Table A.2: Overview of the steps each agent take per clique in the execution phase, which would follow the steps in the preprocessing phase shown in Example A.1.

There is still one thing that must be taken into account in the algorithm. If agent $a_j, j \in \mathcal{N}_i$ is part of more than one clique that includes a_i, then x_j would

---

Algorithm A.1: Privacy Preserving Solution to P1

$p > \max_i(\sum_{j \in \mathcal{N}_i} x_j)$ is a public prime and $D = \{0, \ldots, p - 1\}$.

1: For each a_j, $j \in \mathcal{N}_i$, a_i chooses $\lambda_{i,j,k}, k \in C_{i,j}$ such that $\left(\sum_{k \in C_{i,j}} \lambda_{i,j,k} \right)$ mod $p = 1$.

Preprocessing

2: For each $k \in Ci$, a_i draws from D a uniformly distributed number $r_{i,j,k}$ for each agent $j \in C_k$ (including itself), such that

$$\left(\sum_{j \in C_k} r_{i,j,k} \right) \mod p = 0, \quad k \in Ci.$$

3: a_i sends $r_{i,j,k}$ to agent a_j for $j \in C_k$ and $k \in Ci$.

4: Upon receiving $r_{j,i,k}$ from each agent a_j, $j \in C_k$, $k \in Ci$, agent a_i computes the following sum for each clique C_k, $k \in Ci$,

$$s_{i,k} = \left(\sum_{j \in C_k} r_{j,i,k} \right) \mod p, \quad k \in Ci.$$

Execution

5: For each $k \in Ci$, and each $j \in C_k$, a_i computes

$$p_{i,j,k} = \left(\lambda_{i,j,k} x_i + s_{i,k} \right) \mod p.$$

6: a_i sends $p_{i,j,k}$ to each agent $a_j, j \in C_k$ for each $k \in Ci$.

7: Upon receiving $p_{j,i,k}$ for each $j \in C_k, k \in Ci$, a_i computes

$$y'_k = \sum_{j \in C_k} p_{j,i,k} \mod p, \quad k \in Ci.$$

8: a_i computes

$$y_i = \left(\sum_{k \in Ci} y'_k \right) \mod p. \tag{A.9}$$

be added to the sum, (A.8), more than one time. To see this, consider the graph depicted in Fig. A.2. Attempting to compute $y_1 = x_1 + x_2 + x_3 + x_4$ by the described method would result in the computation of $(x_1 + x_2 + x_3) + (x_1 + x_2 + x_4) \neq y_1$. As seen, x_1 and x_2 are added twice. x_1 can be subtracted since a_1 knows this value, however, a_1 cannot subtract x_2. To circumvent this, a_2 must add only half of x_2 in each of the two cliques.

In Algorithm A.1, the privacy preserving solution to P1 is formally presented from the view of agent a_i, where we use the notation introduced in section 2.

We now provide a sketch of the proof of correctness and privacy of Algorithm A.1.

Sketch of Proof. Correctness. The protocol gives the correct result if a_i learns

$$\left(\sum_{j \in \mathcal{N}_i} x_j \right) \mod p, \tag{A.10}$$

which is equal to the sum in (A.8), since we choose $p > \sum_{j \in \mathcal{N}_i} x_j$. Consider the following rewrite of (A.9), where each equation is modular p even though we omit it to improve notation;

$$\begin{aligned}
y_i = \left(\sum_{k \in Ci} y'_k \right) &= \sum_{k \in Ci} \sum_{j \in C_k} p_{j,i,k} \\
&= \sum_{k \in Ci} \sum_{j \in C_k} \left(\lambda_{j,i,k} x_j + s_{j,k} \right) \\
&= \sum_{k \in Ci} \left(\sum_{j \in C_k} \lambda_{j,i,k} x_j + \sum_{j \in C_k} \sum_{i \in C_k} r_{i,j,k} \right) \\
&= \sum_{k \in Ci} \left(\sum_{j \in C_k} \lambda_{j,i,k} x_j + \sum_{i \in C_k} \underbrace{\sum_{j \in C_k} r_{i,j,k}}_{=0} \right) \tag{A.11} \\
&\overset{(*)}{=} \sum_{j \in \mathcal{N}_i} \sum_{k \in C_{i,j}} \lambda_{j,i,k} x_j \\
&= \sum_{j \in \mathcal{N}_i} x_j \underbrace{\sum_{k \in C_{i,j}} \lambda_{j,i,k}}_{=1} = \sum_{j \in \mathcal{N}_i} x_j,
\end{aligned}$$

where $(*)$ is because summing over the cliques a_i is in ($k \in Ci$) and all the agents in each of those cliques ($j \in C_k$) is equivalent to summing over all the neighbors a_j of a_i ($j \in \mathcal{N}_i$) and the cliques both a_i and a_j are in ($k \in C_{i,j}$) since we assume that all neighbors to a_i is part of a clique that includes a_i. This proves the correctness of the protocol.

Privacy. We consider the execution phase as the preprocessing phase does not involve any private values. For each $k \in Ci$, a_i sends $(\lambda_{i,j,k}x_i + s_{i,k}) \mod p$ to each a_j, $j \in C_k$. This communication is privacy preserving since $s_{i,k} = \sum_{j \in C_k} r_{j,i,k} \mod p$, is a uniformly random number, known only by a_i. For a_j receiving $v = (\lambda_{j,i,k}x_i + s_{i,k}) \mod p$, the information a_j gets is

$$\underbrace{\lambda_{j,i,k}}_{\text{known to } a_j} \ x_i = \underbrace{v}_{\text{known to } a_j} - \underbrace{r_{j,i,k}}_{\text{known to } a_j} - \underbrace{\sum_{j' \in C_k, j' \neq j} r_{j',i,k}}_{\text{unknown to } a_j}, \qquad (A.12)$$

$\square$

Since $|C_k| \geq 3$, $\sum_{j' \in C_k, j' \neq j} r_{j',i,k}$ will at least consist of two uniformly random numbers each of a probability of $\frac{1}{p}$. Hence, by choosing p large, the probability of a_j guessing the last term in (A.12) is negligible.

4 Privacy Preserving Solution to P2

The algorithm for solving P2 without leaking private data is based on Algorithm A.1. Actually, the only difference is for agents that are not part of a clique in a given neighborhood. To clarify, consider Fig. A.3, where a_4 is a neighbor to a_1 but a_4 does not form a clique with a_1 and another agent in $\mathcal{N}_1$. For a_1 to learn $x_2 + x_5 + x_4$ without learning individual values in the sum, we need to extend Algorithm A.1. Our propose is to create a so-called *virtual clique* between a_j, a_i and one other agent, $a_{j'}$, in $\mathcal{N}_i$. In Definition A.1, we define what is meant by a virtual clique.

> **Definition A.1 (Virtual Clique)**
> *Let a_1, a_2 and a_3 be agents in a multi-agent network and let there be a communication link between a_1 and a_2 and between a_1 and a_3. A virtual clique between a_1, a_2 and a_3 is made by letting a_2 encrypt its messages to a_3 such that only a_3 can decrypt them. a_2 sends the encrypted messages to a_1 who forwards to a_3 and vice versa for messages from a_3 to a_2.*

In continuation, let $V_m, \ldots, V_v$, be the virtual cliques in the multi-agent network, where m is the number of cliques, thus the indices of the virtual cliques starts from the last index of the regular cliques. Let Vi be defined similarly to Ci and define for each agent, a_i, the set $V_{i,j}, j \in \mathcal{N}_i$, such that for all $k \in Ci \cup Vi$, $k \in V_{i,j}$ iff $a_j \in C_k \cup V_k$. For encrypting messages, one can chose among many schemes, see for instance the study of encryption algorithms by [12].

Algorithm A.2: Privacy Preserving Solution to P2

---

p and D are as in Algorithm A.1.

1: a_i determines, for each $a_j, j \in \mathcal{N}_i$, the values $\{\lambda_{i,j,k}\}, k \in V_{i,j}$ such that

$$\sum_{k \in V_{i,j}} \lambda_{i,j,k} \mod p = 1, \quad j \in \mathcal{N}_i. \tag{A.13}$$

2: a_i executes Algorithm A.1 to compute the sum, $r1_i = \sum_{k \in Ci} \sum_{j \in C_k} x_j$, using the $\lambda_{i,j,k}$ values in (A.13).

Preprocessing

3: Through a secret key generation process, a_i determines the secret keys g_k for $k \in Vi$ for encrypted communication in the virtual cliques.

4: For each $k \in Vi$, a_i chooses two uniformly distributed random numbers $r_{i,k,0}$ and $r_{i,k,1}$ from D such that
$(r_{i,k,0} + r_{i,k,1}) \mod p = 0$.

5: For each $k \in Vi$, a_i encrypts $r_{i,k,1}$ using g_k and sends $enc(r_{i,k,1})_{g_k}$ to agent $a_j, j \in V_k \cap \mathcal{N}_i$.

6: Upon receiving $enc(r_{j,k,1})_{g_k}$, a_i decrypts and computes the sum

$$s_{i,k} = \left(r_{i,k,0} + r_{j,k,1} \right) \mod p.$$

Execution

7: For each $k \in Vi$, a_i computes $p_{i,j,k} = \lambda_{i,j,k} x_i + s_{i,k} \mod p, \quad j \in V_k \cap \mathcal{N}_i, j \neq i$.

8: a_i sends $p_{i,j,k}$ to $a_j, j \in V_k \cap \mathcal{N}_i, j \neq i, k \in Vi$.

9: Upon receiving $p_{j,i,k}$ a_i computes $y'_{i,k} = \sum_{j \in V_k, j \neq i} p_{j,i,k} \mod p, \quad k \in Vi$.

10: Finally, a_i adds the result from the cliques with the result from the virtual cliques,

$$y_i = r1_i + \sum_{k \in Vi} y'_{i,k} \mod p. \tag{A.14}$$

The protocol for solving *P*2 is formally written in Algorithm A.2 from the perspective of agent a_i.

The correctness of Algorithm A.2 follows from the correctness of Algorithm A.1. Considering only the execution phase, showing the privacy of Algorithm A.2 is equivalent to showing the privacy of Algorithm A.1. The distinction lies in the creation of the random numbers in the preprocessing phase since in Algorithm A.2 agents in a virtual clique with no comunication link between them need to send messages through their common neighbor. However, since we encrypt these messages, we make sure that the common neighbor cannot learn the messages which would otherwise break the privacy.

5   Scalability

To illustrate the scalability of Algorithm A.2, we have conducted simulations showing execution time as a function of a specific graph constellation. The simulations are carried out on a 2.70 GHz laptop, where one thread is created for each agent in the simulated network. For this reason, the absolute execution times may be misleading, as one would expect the execution time to be lower if each agent were given individual machines. However, the execution times are comparable to each other, thus providing an illustration of scalability.

Fig. A.4 shows how the execution time is affected by the number of edges
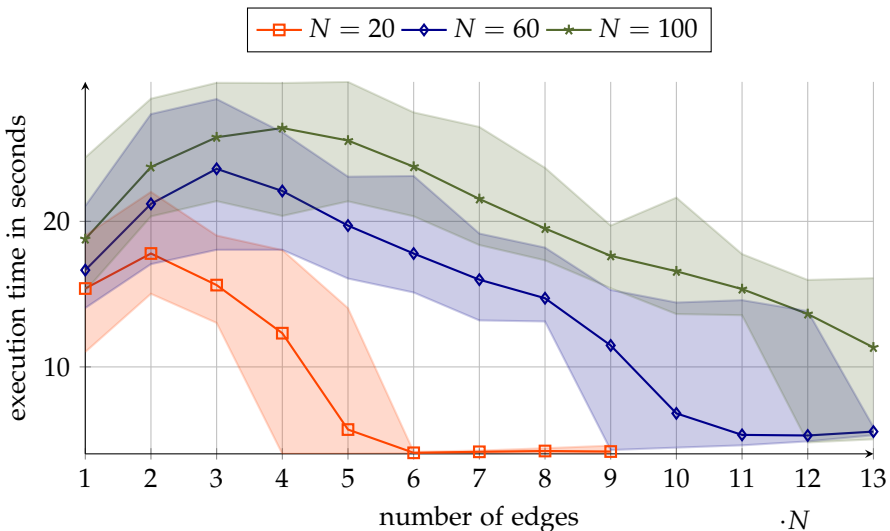
Fig. A.4: Execution time versus number of edges in the graph. Note that the *x*-axis is scaled by *N*.

in the graph. The orange line shows an average simulation of the protocol, where the number of agents $N = 20$ and the execution time is measured with the number of edges in the graph being equal to $k \cdot N$ for $k = 1, \ldots, 13$. It may be counter-intuitive, that the execution time decreases as edges are added to the network. However, this is explained by the fact that as edges are added, the number of virtual cliques in the graph is decreasing, resulting in a faster computation time.

The same goes for the blue line, where the number of agents is 60 and the green line where the number of agents is 100.

6   Conclusion

The paper presents privacy preserving protocols for calculating a sum function among neighbors in a connected graph where each agent can only communicate with their immediate neighbors. The result of the paper can be directly applied in existing decentralized protocols (where agents need the sum of its neighbors values) for achieving privacy. The protocols has a constant number of communication rounds and simulations show great scalability. For future work, it will be interesting to consider other functions of neighbors values than the sum function. To this end, the simple additive secret sharing protocol can be substituted by Shamir's secret sharing scheme.

References

[1] K. Tjell and R. Wisniewski, "Privacy preserving distributed summation in a connected graph," vol. 53, no. 2. Elsevier, 2020, pp. 3445–3450, 21th IFAC World Congress ; Conference date: 12-07-2020 Through 17-07-2020.

[2] G. Banjac, F. Rey, P. Goulart, and J. Lygeros, "Decentralized resource allocation via dual consensus admm," *2019 American Control Conference (ACC)*, Jul 2019.

[3] T. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus admm," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, Jan 2015.

[4] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis, "Hybrid admm: a unifying and fast approach to decentralized optimization," *EURASIP Journal on Advances in Signal Processing*, vol. 2018, no. 1, p. 73, Dec 2018.

[5] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized laplacian eigenvalues estimation for networked multi-agent systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Dec 2009, pp. 2717–2722.

References

[6] C. Liu, H. Li, and Y. Shi, "Resource-aware exact decentralized optimization using event-triggered broadcasting," 2019, in press. [Online]. Available: https://arxiv.org/abs/1907.10179v2

[7] R. Cramer, I. B. Damgaard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, 1st ed. Cambridge University Press, 2015, iSBN: 978-1-107-04305-3.

[8] M. A. Will and R. K. Ko, "Chapter 5 - a guide to homomorphic encryption," in *The Cloud Security Ecosystem*, R. Ko and K.-K. R. Choo, Eds. Boston: Syngress, 2015, pp. 101 – 127. [Online]. Available: https://doi.org/10.1016/B978-0-12-801595-7.00005-7

[9] S. Mehnaz, G. Bellala, and E. Bertino, "A secure sum protocol and its application to privacy-preserving multi-party analytics," in *Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies*, ser. SACMAT '17 Abstracts. New York, NY, USA: ACM, 2017, pp. 219–230. [Online]. Available: http://doi.acm.org/10.1145/3078861.3078869

[10] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu, "Tools for privacy preserving distributed data mining," *ACM SIGKDD Explorations Newsletter*, vol. 4, 12 2002.

[11] R. Sheikh, K. Beerendra, and D. Mishra, "Privacy-preserving k-secure sum protocol," *International Journal of Computer Science and Information Security*, vol. 6, 12 2009.

[12] G. Singh and Supriya, "Article: A study of encryption algorithms (rsa, des, 3des and aes) for information security," *International Journal of Computer Applications*, vol. 67, no. 19, pp. 33–38, April 2013.

Paper B

Private Aggregation with Application to Distributed Optimization

Katrine Tjell, Rafael Wisniewski

Abstract

The paper presents a fully distributed private aggregation protocol that can be employed in dynamical networks where communication is only assumed on a neighbor-to-neighbor basis. The novelty of the scheme is its low overhead in communication and computation due to a pre-processing phase that can be executed even before the participants know their input to aggregation. Moreover, the scheme is resilient to node drop-outs, and it is defined without introducing any trusted or untrusted third parties. We prove the privacy of the scheme itself and subsequently, we discuss the privacy leakage caused by the output of the scheme. Finally, we discuss implementation of the proposed protocol to solve distributed optimization problems using two versions of the alternating direction method of multipliers (ADMM).

1   Introduction

Distributed computing is emerging everywhere in fields such as signal processing, control, and machine learning. Concerns about privacy in such distributed systems are arising since typically lots of data are collected and sensitive information is held at the network's nodes. Several works have shown how collected data can be used to identify individuals, and how private information can be inferred from it. For instance, [2] discusses how private information can be derived even though data is randomized.

In this work, we seek to circumvent these privacy issues by proposing a computation framework where data is only used indirectly and will not be exposed. Typically, the price to pay for privacy is an increase in computational complexity and a communication overhead. However, we introduce a pre-processing phase that only involves none-private data and can be executed before or in-between the actual processing phases, resulting in only a minimal increase in computations and no communication overhead.

Considering already existing distributed algorithms, commonly, the nodes exchange certain values and in preceding computations, the sum of these values is used. That is, the algorithms rely on the sum of communicated values and not the individual values, see for instance [3–5].

Our proposed method solves the problem of privately computing the sum of values belonging to individual users while not revealing the values. We prove that the protocol itself does not leak information and moreover; we study the information leakage caused by the output of the protocol (the sum). We note that differential privacy techniques could be added on top of our method, to avoid the leakage caused by the output, at the cost of loosing precision of the solution.

We explore the use of our proposed method to achieve privacy in distributed optimization. That is, for nodes $i \in \mathcal{N} = \{0, 1, \ldots, N\}$ we assume

that node i has a private convex cost function $f_i(x_i)$, and we consider the following minimization problem:

$$\underset{x_1,\dots,x_N}{\text{minimize}} \quad \sum_{i\in\mathcal{N}} f_i(x_i)$$

$$\text{subject to} \quad \sum_{i\in\mathcal{N}} B_i x_i - c_i = 0, \tag{B.1}$$

$$x_i \in X_i, \quad i \in \mathcal{N}$$

where $x_i \in \mathbb{R}^q$, $B_i \in \mathbb{R}^{M\times q}$, and $c_i \in \mathbb{R}^{M\times 1}$ are assumed to be known to node i, and $X_i \subset \mathbb{R}^q$ is a convex and compact set. This problem is often seen in resource allocation or load balancing problems. In section 6, we use our proposed protocol to achieve privacy in two already existing ADMM-like algorithms which solve (B.1) in two different scenarios: 1) each node i can communicate only with its neighboring nodes, and 2) each node i can communicate with a (untrusted) central unit.

1.1 Related Work

In the literature, there are three main approaches for privacy preservation in distributed computation tasks; secret sharing based secure multiparty computation (SMPC) [6], homomorphic encryption techniques [7], and differential privacy [8]. For the problem in this paper, namely secure aggregation of private data, both SMPC and homomorphic encryption based approaches are evident methods as both can compute a cipher text version of a sum based on cipher text versions of the terms in the sum, [9–12]. The drawback in SMPC is that the schemes typically require all participating parties to be connected by private channels. Regarding homomorphic encryption, the disadvantage is that usually a trusted third party needs to generate and distribute encryption keys. [13] is closely related to our work as they also consider private aggregation in a peer-to-peer network. In their solution, each node needs to communicate with the neighbors of its neighbors, which is in contrast to the assumption in this paper where each node can communicate with its immediate neighbors only. Also, the solution in [13] requires the presence of a trusted third party, which our solution does not.

1.2 Contribution

The paper puts forth a novel private aggregation scheme which bypasses the strict communication requirements in SMPC and the engagement of a trusted third party in homomorphic encryption approaches. The main contribution of the paper can be summarized as:

- The proposed scheme comes with an efficient pre-processing phase requiring only 2 communication rounds. This phase can be executed

prior to the actual computations, even before the nodes have access to their individual inputs. This partitioning makes the scheme extremely light weight at computation time, compared with state-of-the-art SMPC methods.

- In contrast to SMPC and homomorphic encryption based approaches the proposed scheme can be employed in distributed settings where a fully connected communication network cannot be assumed and where an (un)trusted third party does not exist.

- Unlike most of the existing private aggregation schemes, the proposed method in this paper is resilient to node-dropouts.

1.3 Outline

In section 2, we formally state the problem of the paper. A few cryptographic primitives are presented in section 3, while the main contribution is in section 4. Section 5 gives the privacy analysis of the proposed method and section 6 applies the method to distributed optimization.

1.4 Notation

We model the network of nodes as an undirected graph with the nodes $\mathcal{N} = \{1, \ldots, N\}$ and the set of edges $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ where $(i, j) \in \mathcal{E}$ iff node i can communicate with node j. The notation $\mathcal{N}_i$ is used to denote the neighborhood of i, that is $j \in \mathcal{N}_i$ iff $(i, j) \in \mathcal{E}$. Note that we do not consider node i to be a neighbor to itself, i.e. $i \notin \mathcal{N}_i$. Moreover, we use $\mathbb{F}_q$ to denote the finite field of p elements, where p is a prime.

2 Problem Statement

At the outset, we state the problem formally.

Problem B.1
Let $i \in \mathcal{N}$ denote the index of nodes in a network and assume that each node has a private value $s_i \in \mathbb{F}_q$ which it would like to keep secret. Moreover, each node has a set of neighbors, $\mathcal{N}_i$, and each node are interested in learning the sum of its neighbors secret value:

$$y_i = \sum_{j \in \mathcal{N}_i} s_j. \tag{B.2}$$

The problem is to compute (B.2) *without exposing any secret value to any node in the network with the assumption that node i can only communicate with its neighbors $\mathcal{N}_i$.*

Defining the attacker model, we consider the case where up to $t - 1 < n$ nodes may collude in attempting to learn private information of the remaining nodes. However, we assume that all nodes follow the protocol, which is often referred to as the honest-but-curious adversary [1].

3 Cryptographic Tools

To put forth a privacy preserving solution to Problem 1, we use a few cryptographic primitives which are introduced in this section.

3.1 Secret Sharing Scheme

In this section, we give a very brief introduction to secret sharing and we refer to [6] for more elaborate explanation. Suppose a node i has a secret s_i which it would like to share with n other nodes (hereafter called participants) such that at least $t \leq n$ of the nodes need to cooperate in order to learn the value of s_i. The scheme is defined over a finite field $\mathbb{F}_q$, where p is a large prime and it uses a set $\mathcal{P}$ of distinct elements in $\mathbb{F}_q$ to identify the participants, e.g. $\mathcal{P} = \{1, 2, \ldots, n\}$ and $|\mathcal{P}| = n$. The scheme is comprised of two algorithms, and the first is $\texttt{share}(s_i, t, \mathcal{P}) = \{s_i(j)\}_{j \in \mathcal{P}}$ which outputs a share $s_i(j)$ for each participant $j \in \mathcal{P}$ upon receiving a secret s_i, the threshold $t \leq |\mathcal{P}|$, and $\mathcal{P}$. Note that we use $s_i(j)$ to denote the j'th share of the secret s_i. The second algorithm is denoted by $\texttt{reconstruct}(\{s_i(j)\}_{j \in \mathcal{P}'}, t) = s_i$, and produces the secret s_i based on the threshold t and the shares from a set $\mathcal{P}' \subseteq \mathcal{P}$ of participants, where $|\mathcal{P}'| \geq t$. Note that $\mathcal{P}'$ can be any combination of at least t elements from $\mathcal{P}$.

We have the following requirements for the secret sharing scheme. Let $\{s_i(j)\}_{j \in \mathcal{P}} = \texttt{share}(s_i, t, \mathcal{P})$, $\{\bar{s}_i(j)\}_{j \in \mathcal{P}} = \texttt{share}(\bar{s}_i, t, \mathcal{P})$ for arbitrary $s_i, \bar{s}_i \in \mathbb{F}_q$, $t \leq |\mathcal{P}|$, and $\mathcal{P} \subseteq \mathbb{F}_q$.

1. For all $\mathcal{P}' \subseteq \mathcal{P}$, with $|\mathcal{P}'| \geq t$,

$$\texttt{reconstruct}\left(\{s_i(j)\}_{j \in \mathcal{P}'}, t\right) = s_i.$$

2. For all $\mathcal{P}' \subseteq \mathcal{P}$ with $|\mathcal{P}'| < t$:

$$\{s_i(j)\}_{j \in \mathcal{P}'} \sim \{\bar{s}_i(j)\}_{j \in \mathcal{P}'}, \tag{B.3}$$

where $\sim$ means identically distributed.

---

[1]An honest-but-curious adversary is also sometimes referred to as *a passive adversary*, see [6]

3. Finally, we require that

$$
\begin{aligned}
&\texttt{reconstruct}(\{s_i(j)\}_{j\in\mathcal{P}'} + \{\bar{s}_i(j)\}_{j\in\mathcal{P}'}, t) \\
&= s_i + \bar{s}_i,
\end{aligned}
\tag{B.4}
$$

for all $\mathcal{P}' \subseteq \mathcal{P}$ with $|\mathcal{P}'| \geq t$.

3.2 Encryption

For the privacy preserving computations, it will be necessary to encrypt certain values (messages), such that these values cannot be learned by unauthorized nodes. The encryption scheme encompasses three algorithms, where the first is denoted $\texttt{keys} = (sk_i, pk_i)$. It generates a secret- and public key pair, (sk_i, pk_i) for a participant $i \in \mathcal{P}$. The third algorithm is the encryption algorithm $\texttt{enc}(x, pk_i) = [x]$. It takes a message x and a public key and outputs a cipher-text version $[x]$ of the message. The decryption algorithm takes a cipher-text message $[x]$ and a secret key and outputs the plain text version of the message. This algorithm is denoted by $\texttt{dec}([x], sk_i) = x$. For all $x, \bar{x} \in \mathbb{F}_q$, we have the following requirements for the scheme, where (sk_i, pk_i) are any secret- and public key pair:

1.
$$
\texttt{dec}(\texttt{enc}(x, pk_i), sk_i) = x \tag{B.5}
$$

2.
$$
\texttt{enc}(x, pk_i) \sim \texttt{enc}(\bar{x}, pk_i), \tag{B.6}
$$

see details in [11].

4   Proposed Method

Note that both the secret sharing scheme and the encryption scheme is defined over a finite field $\mathbb{F}_q$. Consequently, modular arithmetic is used in the proposed method. Choosing $p > y_i$ and assuming that each secret s_i is an element of $\mathbb{F}_q$, the modular arithmetic will not affect the precision. In case $s_i \in \mathbb{R}$, s_i can be scaled before rounding to the nearest element in $\mathbb{F}_q$, under which circumstance the precision of the method will depend on the scaling factor.

To introduce the proposed method, consider a central node C and k nodes $i \in \mathcal{N}_C$ that are all neighbors to C but not necessarily neighbors to each other.

We present our method by focusing only on node C and the nodes $i \in \mathcal{N}_C$. Solving Problem 1 can be achieved by executing the method in parallel for all nodes $j \in \mathcal{N}$.
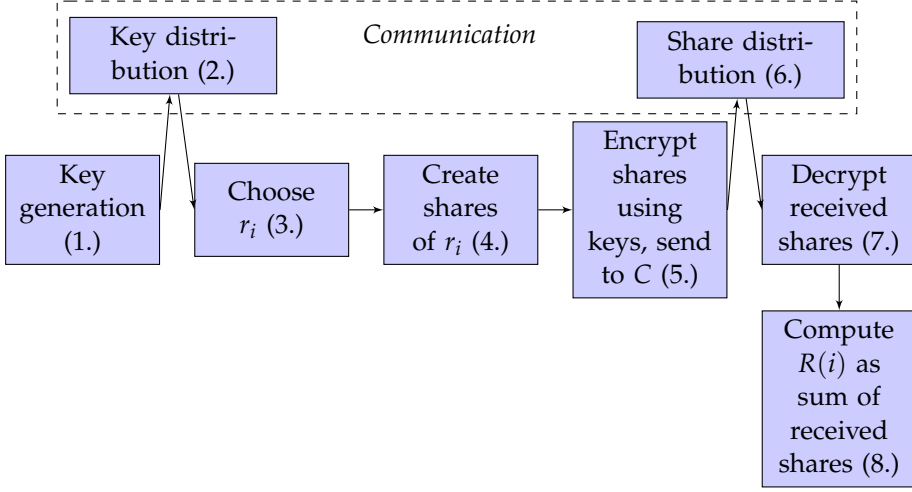
Fig. B.1: Block diagram illustrating the pre-processing phase of the proposed method from the view of node $i \in \mathcal{N}_C$. The numbers in parenthesis refer to the corresponding step in Algorithm B.1.

The idea is to solve (B.2) in Problem B.1 by computing

$$\left(\sum_{i \in \mathcal{N}_C} s_i + r_i \right) \mod p = \left(\sum_{i \in \mathcal{N}_C} s_i + R \right) \mod p, \qquad (B.7)$$

where $R = \left(\sum_{i \in \mathcal{N}_C} r_i \right) \mod p$ and $\{ r_i \in \mathbb{F}_q \}_{i \in \mathcal{N}_C}$ are uniformly chosen by node i. In this way, s_i is masked by the random r_i. We propose a pre-processing phase where each node $i \in \mathcal{N}_C$ computes a share, $R(i)$, of R. At the execution time, node i sends $(s_i + r_i) \mod p$ and $R(i)$ to node C, that can use $R = \texttt{reconstruct}(\{R(i)\}_{i \in \mathcal{N}_{C,t}})$ to compute R and (B.7) to compute $y_C = \sum_{i \in \mathcal{N}_C} s_i$.

In Fig. B.1, we give an overview of the pre-processing phase, where each communication block covers the steps where each node $i \in \mathcal{N}_C$ sends a vector of values to C who forwards the vectors to all nodes $j \in \mathcal{N}_C$. As seen, each node $i \in \mathcal{N}_C$ starts by generating the keys $(sk_i, pk_i) = \texttt{keys}$, and distributes the public key pk_i. Then each node $i \in \mathcal{N}_C$ chooses r_i uniformly from $\mathbb{F}_q$ and creates shares, $\{r_i(j)\}_{j \in \mathcal{N}_C}$, of r_i. The shares are encrypted using the public key of the corresponding node $j \in \mathcal{N}_C$ and the encrypted shares are distributed. Upon receiving encrypted shares from the other nodes, each node $i \in \mathcal{N}_C$ decrypts the shares using its own secret key, sk_i. Node i then computes its share of R by $R(i) = \sum_{j \in \mathcal{N}_C} r_j(i)$, which holds under the third requirement for the secret sharing scheme. The outcome of the pre-processing phase is that node $i \in \mathcal{N}_C$ learns $R(i)$.

We state the proposed method formally in Algorithm B.1 from the view of node $i \in \mathcal{N}_C$.

Algorithm B.1: Private Sum in Graphs

Input: $\mathbb{F}_q$ with $p > \sum_{j \in \mathcal{N}_C} s_j$, and the threshold $t < |\mathcal{N}_C|$ are publicly available.

Output: node C learns $y_C = \sum_{j \in \mathcal{N}_C} s_j$, where s_j is the secret known only to node j.

Pre-processing:

1: $(sk_i, pk_i) = \texttt{keys}$
2: Send public key pk_i to C who forwards to $j \in \mathcal{N}_C \backslash \{i\}$.
3: Draw $r_i \in \mathbb{F}_q$ uniformly.
4: $\{r_i(j)\}_{j \in \mathcal{N}_C} = \texttt{share}(r_i, t, \mathcal{N}_c)$.
5: $[r_i(j)] = \texttt{enc}(r_i(j), pk_j)$ for $j \in \mathcal{N}_C \backslash \{i\}$.
6: Send $[r_i(j)]$ to node C who forwards to node $j \in \mathcal{N}_C \backslash \{i\}$.
7: $\{r_j(i)\}_{j \in \mathcal{N}_C \backslash \{i\}} = \{\texttt{dec}([r_j(i)], sk_i)\}_{j \in \mathcal{N}_C \backslash \{i\}}$.
8: $R(i) = \sum_{j \in \mathcal{N}_C} r_j(i)$.

Execution:

9: $m_i = (s_i + r_i) \mod p$.   (B.8)
10: Send $\{m_i, R(i)\}$ to node C.

Node C does:

1. $R = \texttt{reconstruct}(\{R(i)\}_{i \in \mathcal{N}_C}, t)$.

2. $y_C = ((\sum_{i \in \mathcal{P}} m_i) - R) \mod p$.   (B.9)

4.1 Handling dropped nodes

The proposed protocol is inherently able to handle nodes dropping out as long as there is at least t remaining nodes. To elaborate, we use $\mathcal{P}$ to denote the nodes participating from the beginning and $\mathcal{P}'$ to denote the nodes remaining after some nodes have dropped out. Specifically, $\mathcal{P}' \subset \mathcal{P}$ and $|\mathcal{P}'| \geq t$. If nodes drop out in the pre-processing phase, the remaining nodes can carry on without any modification. If nodes drop out in the execution phase and fail to perform step 10, each node $i \in \mathcal{P}'$ computes $R_{new}(i) = \sum_{j \in \mathcal{P}'} r_j(i)$ and sends R_{new} to node C. Node C can then compute $\sum_{i \in \mathcal{P}'} s_i$. The advantage is that the pre-processing does not have to be run again.

5 Privacy Analysis

We use the standard simulation-based proof to prove that in executing Algorithm B.1, any set of fewer than t colluding nodes will not be able to infer the private values of the honest nodes. To this end, we introduce the term *view* of a node, which is the information known to it during protocol execution.

> **Definition B.1 (View)**
> The view of a node $i \in \mathcal{N}_C$ is a vector, $\texttt{VIEW}_i$, consisting of the values i knows and receives. For a subset $\mathcal{A} \subset \mathcal{N}_C$ of nodes, $\texttt{VIEW}_{\mathcal{A}}$ denotes the vector containing the view of each node $i \in \mathcal{A}$.

Note that, $\texttt{VIEW}_i$ is a random variable since it is based on random choices made by the nodes. What will be shown, is the existence of a simulator which essentially is an algorithm with the ability to simulate a view that is indistinguishable from the view of a set of nodes.

We use $\mathcal{S}_{\mathcal{A}} = \{s_i\}_{i \in \mathcal{A}}$ to denote the set of private values of the nodes $i \in \mathcal{A}$.

Theorem B.1 (Honest-but-curious privacy)
Consider a set $\mathcal{N}_C \subseteq \mathbb{F}_q$. For all integers $t \leq |\mathcal{N}_C|$ and any set $\mathcal{A} \subset \mathcal{N}_C$ with $|\mathcal{A}| < t$ and the central node $C \in \mathcal{A}$, there exist a probabilistic polynomial-time (PPT) simulator $\texttt{SIM}$ which upon the inputs; $\mathcal{S}_{\mathcal{A}}, \mathbb{F}_q$, the threshold t, and the output of Algorithm B.1, y_C, outputs a vector perfectly indistinguishable from $\texttt{VIEW}_{\mathcal{A}}$, namely

$$\texttt{VIEW}_{\mathcal{A}} \sim \texttt{SIM}_{\mathcal{A}}(\mathcal{S}_{\mathcal{A}}, \mathbb{F}_q, t, y_C). \tag{B.10}$$

Proof. The simulator must simulate each element in the view. We list the elements and discuss how the simulated equivalent to each element is chosen. Each simulated equivalent is marked by a $\{\cdot\}^s$. $\texttt{VIEW}_{\mathcal{A}}$ consists of the following values:

$$
\begin{aligned}
&s_i, r_i, sk_i, \{r_j(i)\}_{j \in \mathcal{N}_C}, &&i \in \mathcal{A} \\
&\{m_i, R(i)\}, pk_i, \{[r_j(i)]\}_{j \in \mathcal{N}_C}, &&i \in \mathcal{N}_C \\
&R, y_C = \sum_{i \in \mathcal{N}_C} s_i
\end{aligned} \tag{B.11}
$$

The simulator starts by choosing R^s uniformly from $\mathbb{F}_q$ since R (from (B.7)) is distributed in this way. Then it uses $\{R^s(i)\}_{i \in \mathcal{N}_C} = \texttt{share}(R^s, t, \mathcal{N}_C)$. $\{m_i\}_{i \in \mathcal{N}_C}$ (from (B.8)) are uniformly distributed on $\mathbb{F}_q$ with the condition that $\sum_{i \in \mathcal{N}_C} m_i = y_C + R$, thus $\{m_i^s\}_{i \in \mathcal{N}_C}$ are simulated according to this. $\{r_i^s\}_{i \in \mathcal{N}_C}$

are simulated by drawing uniformly random values from $\mathbb{F}_q$, with the condition that $\sum_{i \in \mathcal{N}_C} r_i^s = R$, see (B.7). Based on $\{r_i^s\}_{i \in \mathcal{N}_C}$, the simulator can use the steps in the pre-processing phase of Algorithm B.1 to simulate the remaining elements of $\mathtt{SIM}_{\mathcal{A}}(\mathcal{S}_{\mathcal{A}}, \mathbb{F}_q, t, y_C)$. $\qquad\square$

5.1 Leakage of Information from Output

As noted earlier, information can be gained from the output of the protocol. To study this in detail, we consider the sum,

$$z_N = \sum_{i=1}^{N} s_i, \tag{B.12}$$

where $s_i \in [0, K]$ for $K \in \mathbb{F}_q$ corresponds to the secret value of node i, and $N > 1$ is the number of terms in the sum. Particularly, we will investigate the amount of information z_N leaks about a particular s_i, say s_1. To do this, z_N and $\{s_i\}_{i \in \mathcal{N}}$ are viewed as outcomes of the random variable Z_N and uniformly distributed variables $\{S_i\}_{i \in \mathcal{N}}$, respectively. We use the uniform distribution for each S_i since this will be true from the view of the adversary given that he has no prior knowledge.

We start the discussion by considering the mutual information between S_1 and Z_N, given as

$$I(S_1, Z_N) = \sum_{s_1, z_N} p(s_1, z_N) \log_2 \left(\frac{p(s_1, z_N)}{p(s_1) p(z_N)} \right), \tag{B.13}$$

where $p(x)$ is the probability mass function of the random variable X, and $p(x, y)$ is the joint probability mass function of the random variables X and Y. Intuitively, $I(S_1, Z_N)$ is the reduction in uncertainty about S_1 gained from Z_N. By the data processing inequality (see for instance [14]),

$$I(S_1, Z_{N-1}) \geq I(S_1, Z_N). \tag{B.14}$$

Hence, the mutual information is non-increasing as N is increased. To explore this result further, consider the conditional entropy of S_1 conditioned on Z_N, which is given as

$$H(S_1 | Z_N) = H(S_1) - I(S_1, Z_N). \tag{B.15}$$

This is a measure of the uncertainty about S_1 after Z_N is given. The uncertainty is measured in bits and the more bits, the more uncertainty there is about the variable.

Finding a closed form expression for $H(S_1 | Z_N)$ is still an open question. Hence, to illustrate it, we calculate $H(S_1 | Z_N)$ numerically for small values of N and K since the combinatorics starts to be intractable for larger values. Fig. B.2 depicts $H(S_1)$ and $H(S_1 | Z_N)$ for $N = 2, \ldots, 13$ and $K = 4$. The figure
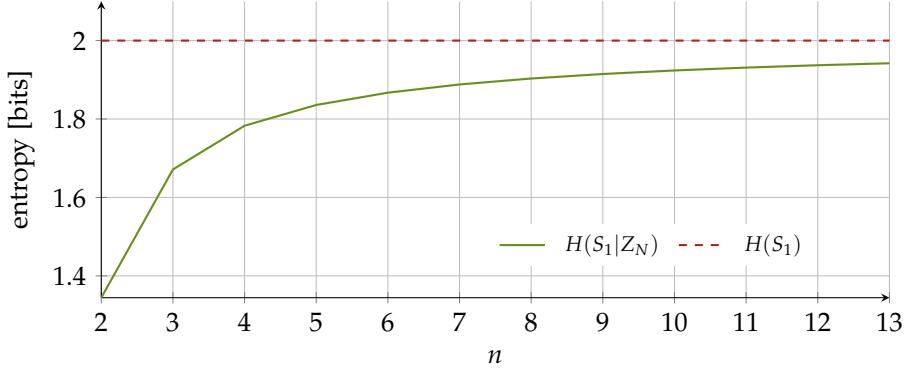
Fig. B.2: Comparison between the entropy of S_1 (red dashed line) with the entropy of S_1 conditioned on Z_N (green line).

compares the uncertainty of S_1 to the uncertainty of S_1 conditioned on Z_N. As seen, increasing N decreases the leakage about S_1. This means that the more neighbors a node has, the less information it will gain on the private values of its neighbors.

We conclude this section, by studying the probability of the adversary guessing $S_1 = s_1$ after learning $Z_N = z_N$ or in other words; the probability of leaking the secret. To establish a closed form expression for this probability, we assume that each S_i is uniformly distributed on $[0, z_N]$.

Proposition B.1

Let $S_1, \dots, S_n$ be independent uniformly distributed on $[0, K]$ and let $Z_N = \sum_{i=1}^{N} S_i$. Then the conditional probability of S_1 conditioned on Z_N is given by

$$P(S_1|Z_N) = \frac{(z_N - s_1 + N - 2)! z_N! (N-1)!}{((z_N - s_1)!(N-2)!(z_N + (N-1))!} \tag{B.16}$$

for $z_N \in \{0, 1, \dots, K\}$.

Proof. The conditional probability can be written as

$$P(S_1|Z_N) = \frac{P(S_1, Z_N)}{P(Z_N)}. \tag{B.17}$$

By counting the number of combinations of $s_1, \dots, s_N$ that satisfies (B.12) with given z_N, where each s_j can take values in the interval $[0, z_N]$, the probability mass function of Z_N yields

$$P(Z_N) = \frac{(z_N + N - 1)!}{z_N!(N-1)!} \frac{1}{T}, \tag{B.18}$$

where T is the total number of outcomes of Z_N.

Similarly, by counting the number of combinations of $s_2, \ldots, s_N$ that satisfies (B.12) with given z_N and s_1, where each s_j can take values in the interval $[0, z_N]$, the joint probability mass function between S_1 and Z_N, yields

$$P(S_1, Z_N) = \frac{(z_N - s_1 + N - 2)!}{(z_N - s_1)!(N - 2)!} \frac{1}{T},$$ (B.19)

which concludes the proof. □

6 Application to Dist. Optimization

In the following, we consider the minimization problem in equation (B.1) assuming: 1) a centralized setting, and 2) a decentralized setting. We study two already existing distributed optimization algorithms and use Algorithm B.1 to achieve privacy in each of them.

6.1 Centralized Optimization

Solving (B.1) in the scenario where each node communicates with an untrusted central unit, can be achieved by using a modified version of the ADMM algorithm. Such an algorithm is presented in [15] by the following steps in each iteration $k \geq 0$:

$$d^{k+1} = \frac{1}{N} \sum_{j=1}^{N} B_j x_j^k - c_j$$ (B.20a)

$$x_i^{k+1} \in \underset{x_i \in X_i}{\operatorname{argmin}} \Big\{ f_i(x_i) + \lambda^{k\top} B_i\, x_i$$

$$+ \frac{\rho}{2} ||B_i x_i - B_i x_i^k + d^{k+1}||^2 \Big\}$$ (B.20b)

$$\lambda^{k+1} = \lambda^k + \rho d^{k+1},$$ (B.20c)

using the initial values $x_i^0 \in X_i$, and $\lambda^0 \in \mathbb{R}^m$. Since the nodes cannot communicate with each other, (B.20a) will be computed by the central unit. The algorithm in (B.20) is related to the traditional ADMM algorithm presented in [16], where the steps (B.20b) and (B.20c) can be identified as the primal and dual updates, respectively. However, there is an important distinction which accounts for the differences; in the traditional ADMM, the primal update is separated into two parts that are updated sequentially, while the primal update here is separated into N parts that are updated simultaneously. [17] proves convergence of (B.20) given that the solution set to the problem in (B.1) is nonempty. To preserve privacy, we propose to compute $\sum_{i=1}^{N} B_i x_i^k - c_i$ using Protocol B.1. We refer to the following two steps as privacy preserving (PP) parallel ADMM:

1. The nodes uses Protocol B.1 to compute $d^{k+1} = \sum_{i=1}^{N} B_i x_i^k - c_i$, where the nodes $j \in \mathcal{N}$ takes the roles of nodes $i \in \mathcal{N}_C$ and the central unit takes the role of node C. The central unit returns d^{k+1} to the nodes $i \in \mathcal{N}$.

2. Each node $i \in \mathcal{N}$ computes (B.20b) and (B.20c) in parallel.

6.2 Decentralized Optimization

For solving (B.1) under the assumption that each node i can only communicate with its neighboring nodes $j \in \mathcal{N}_i$, [18] presents a fully decentralized variant of the ADMM algorithm referred to as tracking-ADMM. It uses a consensus matrix, $w \in \mathbb{R}^{n \times n}$, which is a semidefinite doubly stochastic and symmetric matrix, see [18] for details. Given, $x_i^0 \in X_i, \lambda_i^0 \in \mathbb{R}^M$ and $d_i^0 = B_i x_i^0 - c_i$, the following steps computed by each node $i \in \mathcal{N}$ in parallel solves (B.1);

$$\delta_i^k = w_{i,i} d_i^k + \sum_{j \in \mathcal{N}_i} w_{i,j} d_j^k \tag{B.21a}$$

$$l_i^k = w_{i,i} \lambda_i^k + \sum_{j \in \mathcal{N}_i} w_{i,j} \lambda_j^k \tag{B.21b}$$

$$x_i^{k+1} \in \underset{x_i \in X_i}{\arg\min}\{f_i(x_i) + l_i^{k^\top} B_i x_i + \frac{\rho}{2}||B_i x_i - B_i x_i^k + \delta_i^k||^2\} \tag{B.21c}$$

$$d_i^{k+1} = \delta_i^k + B_i x_i^{k+1} - B_i x_i^k \tag{B.21d}$$

$$\lambda_i^{k+1} = l_i^k + \rho d_i^{k+1}, \tag{B.21e}$$

where $\rho > 0$ is a penalty parameter. The algorithm in (B.21) is quite different from the standard ADMM due to the fully decentralized setting. The information $\sum_{i=1}^{N} B_i x_i^{k+1} - c_i$ is not available to the nodes, since each node can only communicate with its neighbors. Thus, as explained in [18], steps (B.21a), (B.21b) and (B.21d) roughly acts as a dynamic average consensus mechanism for estimating this term. [18] proves that this algorithm converges given that each $f_i(x_i)$ is convex and that (B.1) and the dual problem of (B.1) admits optimal solutions. To preserve privacy, we propose to use Protocol B.1 to compute δ_i^k and l_i^k for each node i. That is, (B.21a) and (B.21b) is substituted with the following steps;

1. Protocol B.1 is utilized to compute δ_i^k, where node i takes the role of the central node C and the nodes $j \in \mathcal{N}_i$ takes the role of the nodes $j \in \mathcal{N}_C$. The nodes $j \in \mathcal{N}_i$ inputs $w_{i,j} d_j^k$ to the protocol and node i learns $\bar{\delta}_i^k = \sum_{j \in \mathcal{N}_i} w_{i,j} d_j^k$ and computes $\delta_i^k = w_{i,i} d_i^k + \bar{\delta}_i^k$.
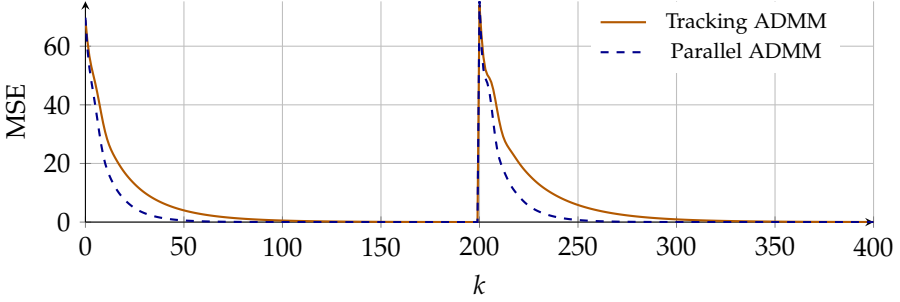
Fig. B.3: Convergence of PP tracking ADMM and PP parallel ADMM with $N = 30$ nodes. After $k = 200$ iterations, 10 randomly selected nodes drop out.

2. Protocol B.1 is utilized to compute l_i^k. The nodes $j \in \mathcal{N}_i$ inputs $w_{i,j}\lambda_j^k$ to the protocol and node i learns $\bar{l}_i^k = \sum_{j\in\mathcal{N}_i} w_{i,j}\lambda_j^k$ and computes $l_i^k = w_{i,i}\lambda_i^k + \bar{l}_i^k$.

3. Each node $i \in \mathcal{N}$ compute in parallel the remaining steps; (B.21c), (B.21d), (B.21e).

We refer to these three steps as PP tracking ADMM.

6.3 Numerical Experiments

We simulate privacy preserving (PP) parallel ADMM and PP tracking ADMM solving the same optimization problem of the form of (B.1) with $q = 1$, $M = 2$, $f_i(x_i) = (x_i - i)^2$, and randomly generated B and c matrices. This problem is solved with $N = 30$ nodes and in the case of PP tracking ADMM, each node has on average 20 neighbors. After 200 iterations, we simulate the dropout of 10 randomly selected nodes. To compare the performance of PP parallel ADMM and PP tracking ADMM, Fig. B.3 shows the mean squared error (MSE) of the estimate from both methods at each iteration k. As seen in Fig. B.3, PP tracking ADMM has a slower convergence rate compared to PP parallel ADMM which is due to information being distributed in the network much slower. In fact, the convergence rate of PP tracking ADMM is dependent on the number of neighbors of each node. This can be observed in Fig. B.4 that shows the convergence rate of PP tracking ADMM when the numerical problem is solved and each node has $n = 5, 10, 15, 20, 29$ neighbors, respectively. Note that in the case $n = 29$, the network is fully connected and the convergence rate matches with the rate of the PP parallel ADMM. That PP tracking ADMM converges faster the more neighbors each node has matches nicely with the result from section 5.1 stating that the more neighbors a node has, the less information is revealed by the output of the method.
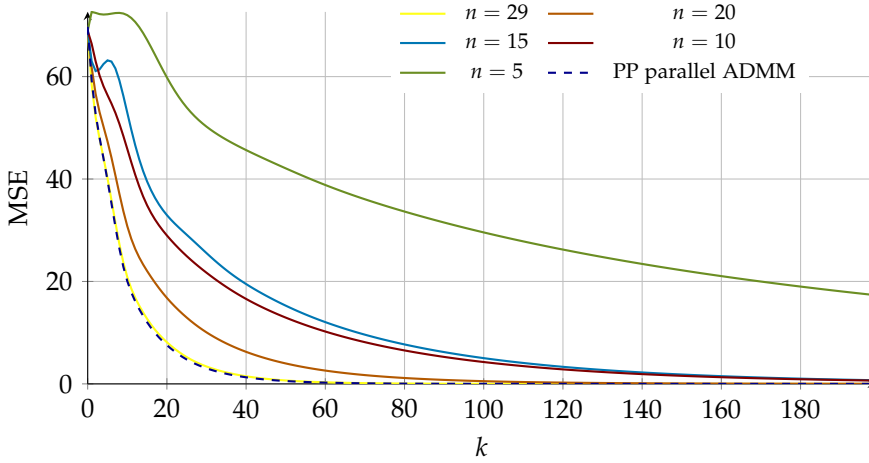
Fig. B.4: Comparison of the convergence of PP tracking ADMM where each node has respectively $29, 20, 15, 10$, and 5 neighbors.

7 Conclusion

The paper presents a privacy preserving fully distributed and parallel aggregation scheme for computing the sum of private values held by individual nodes. Two straight forward applications of the proposed protocol are given in the paper, namely privacy preserving distributed optimization. We note that the protocol can be applied in many other distributed control algorithms to preserve privacy, see for instance [19].

References

[1] K. Tjell and R. Wisniewski, "Private aggregation with application to distributed optimization," *IEEE Control Systems Letters*, vol. 5, pp. 1591 – 1596, 2021.

[2] Z. Huang, W. Du, and B. Chen, "Deriving private information from randomized data," in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '05. NY, USA: ACM, 2005, pp. 37–48.

[3] D. Yuan, A. Proutiere, and G. Shi, "Distributed online linear regression," 2019.

[4] G. Chen and J. Li, "A fully distributed admm-based dispatch approach for virtual power plant problems," *Applied Mathematical Modelling*, vol. 58, pp. 300 – 312, 2018.

[5] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized laplacian eigenvalues estimation for networked multi-agent systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Dec 2009, pp. 2717–2722.

References

[6] R. Cramer, I. B. Damgaard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, 1st ed. Cambridge University Press, 2015.

[7] M. A. Will and R. K. Ko, "Chapter 5 - a guide to homomorphic encryption," in *The Cloud Security Ecosystem*, R. Ko and K.-K. R. Choo, Eds. Boston: Syngress, 2015, pp. 101 – 127.

[8] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12.

[9] E. Shi, T.-H. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," vol. 2, 01 2011.

[10] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin, "Smart meter aggregation via secret-sharing," in *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, ser. SEGS '13. NY, USA: Association for Computing Machinery, 2013, p. 75–80.

[11] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. NY, USA: Association for Computing Machinery, 2017, p. 1175–1191.

[12] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 111–125.

[13] D. Bickson, T. Reinman, D. Dolev, and B. Pinkas, "Peer-to-peer secure multiparty numerical computation facing malicious adversaries," *Peer-to-Peer Networking and Applications*, vol. 3, 01 2009.

[14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. USA: Wiley-Interscience, 1991.

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. USA: Prentice-Hall, Inc., 1989.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[17] B. He, L. Hou, and X. Yuan, "On full jacobian decomposition of the augmented lagrangian method for separable convex programming," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2274–2312, 2015.

[18] A. Falsone, I. Notarnicola, G. Notarstefano, and M. Prandini, "Tracking-admm for distributed constraint-coupled optimization," 2019, submitted to Automatica, eprint: arXiv:1907.10860.

References

[19] Y. Wang, Z. Huang, S. Mitra, and G. E. Dullerud, "Differential privacy in linear distributed control systems: Entropy minimizing mechanisms and performance tradeoffs," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 118–130, March 2017.

Paper C

Privacy Preserving Recursive Least Squares
Solutions

Katrine Tjell, Ignacio Cascudo and Rafael Wisniewski

Abstract

Individual privacy is becoming a more prioritized issue in the modern world, because the world is becoming increasingly more digitized and citizens are starting to feel monitored. Private information could furthermore be misused in the wrong hands.

Many control systems rely on data that often contain privacy sensitive information. These are systems such as the power grid, water network, and smart house where data contain individual consumption profiles and daily schedules. The systems use the data to compute optimized solutions; hence, the data is valuable but it contains private information. To this end, it is desirable to achieve algorithms able to calculate optimized solutions while keeping the data secret.

As a step towards this goal, we propose a privacy preserving recursive least squares protocol that computes a least squares estimate of the parameters of a linear system based on observations of input and outputs. This estimate is calculated while ensuring no leakage of information about observations.

1 Introduction

The tendency in modern and upcoming infrastructures such as smart-grids and smart-transportation systems is that control and monitoring depend on exchange of data. This data is used to calculate estimations and optimized solutions that adapt to individual users and subsequently lead to decisions made by the system. More often than not, the data contains privacy sensitive information that can lead to inferences about individuals. Consider for instance a smart meter that collects fine-grained measurements of the electricity consumption of a household. Using data mining methods with other publicly available data, these measurements reveal information about the activities of the occupants of the household, [2]; moreover [3] show that even an anonymous 10-ride bus ticket reveals information that can lead to identification of a specific bus passenger.

These privacy issues are resolved by the application of secure communication channels or differential privacy in most of the literature, [4], [5], [6]. While these methods may be useful, ultimately they rely on trust since data owners are required to handover raw data. As there is no guarantee that the receiver is not corrupt, data owners may be reluctant to share their data.

This paper proposes a method for calculating estimations based on private data that does not rely on trusted parties. The aim is that raw data will not leave the data owner, and any computations from the data will not leak information. To achieve this goal, both distributed computing and recent results from the research field of *secure multiparty computation (SMPC)* are employed.

To provide concrete results, the paper has its focus on a privacy preserving recursive least squares (RLS) protocol. The particular choice of the least squares algorithm is due to its general applicability and because it is the foundation of many other estimation techniques. We choose to investigate its recursive form, since it is used in many control systems. We see this privacy preserving RLS protocol as a first step toward secure control computations.

Related work. As mentioned, much of the privacy involved literature is occupied with differential security. To name a few: [7] is concerned with differentially private and robust statistics, hereunder differentially private linear regression and [8] presents differentially private Kalman filtering. Another noticeable work is [9], that proposes an optimal distributed estimation algorithm, that uses added noise to preserve privacy.

Within the field of SMPC there are a number of works concerned with solving regression problems, some of which are in the context of machine learning, such as [10], [11] and [12]. Moreover, it should be noted that recently practical secure computation frameworks have been proposed, which are specifically tailored to situations where the computation is delegated on a small number of computing parties. Usually such frameworks only tolerate one corrupted party, i.e., there are no collusions. In particular, of interest for our problem are the general framework Sharemind[13], upon which the suit of statistical algorithms Rmind[14] has been built; and the frameworks ABY3 and SecureML for efficient machine-learning related secure multiparty computation protocols introduced by Mohassel and Rindal in [15] and Mohassel and Zhang in [16] respectively, based on the ABY framework for practical secure computation introduced in [17].

In this work, we focus on a privacy preserving recursive least squares algorithm and adopt the more general approach where we can base our protocol on any linear secret sharing scheme that assumes there is a sub-protocol for secure multiplication of shared secrets. This is a flexible approach that in particular allows for an arbitrary number of computing servers (which in particular could be the input-providing parties themselves).

The structure of this paper. Section 2 gives a brief introduction to secure multiparty computation while section 3 presents the main content of the paper, i.e., a privacy preserving protocol for solving the RLS equations. Thereafter, section 4 evaluates the performance of the proposed protocol by providing simulation results that compare the privacy preserving protocol to the traditional (non-secure) computation of the RLS equations. Finally, section 5 gives a conclusion on the findings.

2 Secure Multiparty Computation

SMPC is concerned with the development of protocols that allow a set of parties to jointly and privately compute a function taking an input from each of the parties. In particular, the goal is that a set of parties, $P_1, \ldots, P_m$, each having a value, respectively $x_1, \ldots, x_m$, compute some function $y = f(x_1, \ldots, x_m)$ without any party obtaining more information about other parties' inputs than is implied by its own input and the output of the function. This should hold even if up to a certain number t of parties are corrupted (meaning that they collude and pool together all information they have seen during the execution of the protocol). SMPC assumes that communication between each pair of parties is already realised over secure authenticated channels (for which encryption and digital signatures can be used) and focuses on the information that may be leaked to other parties participating in the computation (by the interaction with them) rather than being concerned about external eavesdroppers. In this article we only consider passive corruption, meaning that corrupted parties do not deviate from the established protocol. Security of SMPC protocols is formally defined via a simulation paradigm, where roughly speaking it is shown that the views of the parties during the protocol could in fact be simulated in a world where parties only have access to their inputs and the output of the function. Based on this paradigm, the notion of universal composability ensures that SMPC protocols can be composed securely. For the interested reader, [18] gives a formal presentation of this topic, while [19] gives a more concise introduction to SMPC.

The SMPC protocols proposed in this paper are based on *secret sharing*. Secret sharing allows a party (dealer) to distribute a secret among a set of parties without revealing the secret to any individual party or small coalition.

Definition C.1 provides a definition of a secret sharing scheme.

> **Definition C.1 (Secret Sharing Scheme)**
> A t-threshold secret sharing scheme for m parties is a pair of algorithms Share and Reconstruct. Share takes a secret s and creates m values $s_1, \ldots, s_m$, called shares. Reconstruct is an algorithm that given any set of more than t shares outputs the corresponding secret s. In addition it is satisfied that any set of t shares or less gives no information about the secret.

We represent the secret sharing scheme with $[\cdot]$. That is, given a secret s, $[s]$ represents a set of shares of s, $(s_1, \ldots, s_m)$.

The idea of secret-sharing based SMPC is that the inputs of the computation are shared among the computing parties in the first phase, and subsequently these parties can compute on the shared values until the last step,

where only the sharing corresponding to the output of the computation is reconstructed.

Most secure secret-sharing based SMPC protocols are built upon a secret sharing scheme where both the secret and each of the shares are elements of a sufficiently large finite field $\mathbb{F}_q = \{0, 1, \ldots, p-1\}$ where p is a prime number, and operations are modulo p. Moreover, the secret sharing scheme satisfies that given $[a], [b]$, the set of parties can compute shares of the sum $[a + b]$ and of the product $[ab]$ modulo p. The first of these properties (called linearity) is achieved by most secret-sharing schemes, and it does not require any communication among the parties: each party just needs to sum the shares they have received. An example of such linear secret sharing is additive sharing, where the dealer chooses the $m - 1$ first shares $s_1, \ldots, s_{m-1}$ uniformly at random in $\mathbb{F}_q$, and chooses s_m such that $s = s_1 + \ldots + s_m$. In this case $t = m - 1$. When we write $[c] = [a] + [b]$ we mean that each party P_i does the local operation $c_i = a_i + b_i \mod p$, and therefore the secrets satisfy $c = a + b$.

Multiplication is more involved and requires communication between the parties. A common strategy to alleviate this is to push the communication intensive part to a preprocessing phase (which can occur before even the inputs of the computation are known). This is done by means of the so-called Beaver's circuit randomization technique, where sharings of correlated random data are created in that preprocessing phase. Given that preprocessed data, the computation of a multiplication in the actual computing phase consists only on local operations and the opening of two sharings, which can be done by broadcasting shares. There are several ways of creating the preprocessed data, using tools such as semi-homomorphic encryption or oblivious transfer, but this is out of the scope here. For a recent comparison of these alternatives see for example [20].

We will assume that parties have access to a multiplication protocol. We write $[c] = [a][b]$ when parties execute this protocol, where $c = ab \mod p$.

In addition, some of the protocols stated in this paper take as input $[r]$, where r is a uniform random number in $\mathbb{F}_q$ unknown to all parties. This can be also obtained in a pre-processing phase. The interested reader is referred to [18], to learn about pre-processing phases and random number generation in SMPC.

Secret sharing based SMPC also allows to delegate the computation on a small set of computing parties (or servers) that do not need to be the same as the input-providers. Indeed the input-providers can share the inputs among the servers, and these can carry the rest of the computation, operating on the secret shared values. In the last step, the servers reconstruct the output of the computation.

We will need some additional notation: we will write $[a]$ and $[A]$ to denote the fact that each party has one share per entry in the vector a, respectively

the matrix A. In addition, we write $[a]^B$ to denote the situation where each party has one share per bit in a, i.e., the parties hold $[a_0], [a_1], \ldots, [a_{l-1}]$, where $a_0, \ldots, a_{l-1}$ are the bits of a with a_0 being the least significant bit.

3 A Privacy Preserving Recursive Least Squares Protocol

3.1 The Recursive Least Squares Equations

To motivate the recursive least squares algorithm, consider the problem of recursive linear regression.

Suppose that an unknown linear system takes a vector of inputs and outputs a linear combination of these. To be precise, the output, $y_i \in \mathbb{R}$, at time i is modeled as

$$y_i = x_i^\top w_i, \tag{C.1}$$

where $x_i \in \mathbb{R}^{q \times 1}$ is a vector of input observations at time i and $w_i \in \mathbb{R}^{q \times 1}$ is the vector of unknown parameters at time i. Assuming the observations arrive one at a time, the idea is to recursively estimate the parameter vector w_i, such that at each time i, it minimizes the sum of squared residuals,

$$\hat{w}_i = \min_{w_i} ||y_i - x_i^\top w_i||^2. \tag{C.2}$$

The solution, at time i, to this minimization problem is known as *the recursive least squares equations*, and can be formulated as

$$P_i = P_{i-1} - \left(1 + x_i^\top P_{i-1} x_i\right)^{-1} P_{i-1} x_i x_i^\top P_{i-1}, \tag{C.3}$$

$$g_i = P_i x_i, \tag{C.4}$$

$$e_i = y_i - x_i^\top \hat{w}_{i-1}, \tag{C.5}$$

$$\hat{w}_i = \hat{w}_{i-1} + g_i e_i, \tag{C.6}$$

where P_0 is usually initialized as the $q \times q$ identity matrix, I_q, and w_0 is initialized as the $q \times 1$ vector of zeros, 0_q. For a more elaborate introduction to the recursive least squares equations refer to [21, p. 454].

3.2 Scenario and challenges

Our aim is to develop a protocol for the privacy preserving recursive least squares algorithm where we think of the following scenario: the computation will be carried out by a (small) number of computing servers $P_1, ..., P_m$. Every sensor making an observation x_i or y_i will secret share that value among

$P_1, ..., P_m$. By using additive secret sharing we can assume that up to $m - 1$ of these servers are passively corrupted, by doing the preprocessing in the way described in the previous section.

Developing a privacy preserving multiparty protocol for computing (C.3), (C.4), (C.5), and (C.6), entails a few challenges. The first one is the division in (C.3). Since the protocol can apply solely finite field arithmetic, division must be replaced with integer division. Remark, that integer division is not the same as division in $\mathbb{F}_q$, since this would lead to an entirely different result. In this regard, a definition of integer division is given in Definition C.2.

Definition C.2 (Integer Division)
The term *integer division* refers to the operation of dividing two integers and afterwards truncating the result, such that the output is also an integer. The operation is denoted by the symbol \, such that

$$a \backslash z = \left\lfloor \frac{a}{z} \right\rfloor.$$

Furthermore, let .\ denote the element-wise integer division operation such that

$$
\begin{bmatrix}
a_{1,1} & \cdots & a_{1,n} \\
\vdots & \ddots & \vdots \\
a_{n,1} & \cdots & a_{n,n}
\end{bmatrix} . \backslash z =
\begin{bmatrix}
\left\lfloor \frac{a_{1,1}}{z} \right\rfloor & \cdots & \left\lfloor \frac{a_{1,n}}{z} \right\rfloor \\
\vdots & \ddots & \vdots \\
\left\lfloor \frac{a_{n,1}}{z} \right\rfloor & \cdots & \left\lfloor \frac{a_{n,n}}{z} \right\rfloor
\end{bmatrix} .
$$

To carry out secure integer division of two secrets, we use a sub protocol, $\mathsf{divSec}([d], [n])$, that takes as input two shared l-bit integers, $[d]$ and $[n]$, and outputs $[n \backslash d]$.

Since we use integer division, scaling is introduced to minimize the truncation error. This is achieved simply by multiplying the nominator by 2^C before the division, where C is an appropriately chosen integer.

To rescale, we will use a sub protocol, $\mathsf{reScale}([a], 2^C)$, that takes as input a shared integer $[a]$ and a public constant 2^C and outputs $[a \backslash 2^C]$.

The final issue to be aware of is that, because we map all integers to $\{0, ..., p - 1\}$ in order to use that finite field for secure computation, the division protocol will not produce the correct result if either or both of the nominator and denominator are negative. To see the problem, consider the quotient $-2 \backslash 2 = -1$. Abusing notation, we define ($a \mod p$) to be the integer in $\{0, ..., p - 1\}$ which equals a modulo p. Let for example $p = 23$. Computing the above integer division naively yields

$$(-2 \mod 23) \backslash (2 \mod 23) = 21 \backslash 2 = 10$$

On the other hand $(-1 \mod 23) = 22$, which is different than 10. To get a correct result, the sign of the nominator and denominator must be checked before performing the division. For this we will use a sub protocol for comparison, $\text{comp}([a])$, that takes as input the shared integer, $[a]$, and outputs a shared integer $[c]$, where $c = 1$ if $a < 0$ and $c = 0$ otherwise.

To sum up the modifications of (C.3), (C.4), (C.5), and (C.6) necessary to turn them into a SMPC protocol, they are rewritten into the following equations, which could be referred to as the *integer recursive least squares equations*.

$$P_i = P_{i-1} - \left(\left(P_{i-1} x_i x_i^\top P_{i-1} \right) .\backslash \left(2^C + x_i^\top P_{i-1} x_i \right) \right) \tag{C.7}$$

$$g_i = P_i x_i, \tag{C.8}$$

$$e_i = y_i - x_i^\top \hat{w}_{i-1} \backslash 2^C, \tag{C.9}$$

$$\hat{w}_i = \hat{w}_{i-1} + g_i e_i. \tag{C.10}$$

3.3 SMPC Protocol for Solving the RLS

Now the initial problem has been reduced to creating a SMPC protocol for computing (C.7), (C.8), (C.9), and (C.10). This protocol is referred to as $\text{RLS}(([y_1], [y_2], \ldots), ([x_1], [x_2], \ldots), 2^C)$ and is stated in Protocol C.1.

3.4 Sub-protocols

In this section, the sub protocols used in Protocol C.1 are formally presented.

comp($[a]$)

The first sub protocol is $\text{comp}([a])$, which evaluates the truth of the relation $a < 0$, for $a \in \mathbb{F}_q$. To see how this can be performed, define the field elements as

$$\{0, \underbrace{1, \ldots, \lfloor \tfrac{p}{2} \rfloor}_{positive}, \underbrace{\lfloor \tfrac{p}{2} \rfloor + 1, \ldots, p - 1}_{negative}\}.$$

Following this definition, if a secret number is smaller than $\frac{p}{2}$ it is positive and if the opposite is true it is negative. Thus, the desired result can be obtained from evaluating the truth of $[-a] < \frac{p}{2}$. In [22], a protocol for this exact computation is proposed, so this will be used as the $\text{comp}([a])$ protocol in this paper. It should be noted that the comparison protocol uses bit-decomposition and is thus expensive in communication.

Protocol C.1:

$([2^C \hat{w}_N]) \leftarrow \mathsf{RLS}(([y_1], [y_2], \ldots), ([x_1], [x_2], \ldots), 2^C)$

Input: $([y_1], [y_2], \ldots)$, where $y_1, y_2, \ldots \in \mathbb{F}_q$, $([x_1], [x_2], \ldots)$, where $x_1, x_2, \ldots \in \mathbb{F}_q^{q \times 1}$, and $2^C \in \mathbb{F}_q$.

Output: $[2^C \hat{w}_N]$ where $w_n \in \mathbb{F}_q^{q \times 1}$.

1: $[P_0] = 2^C [I_q]$
2: $[\hat{w}_0] = [0_q]$
3: **for all** $N = 1, \ldots$ **do**
4: $\quad [d] = 2^C + [x_N]^\top [P_{N-1}][x_N]$.
5: $\quad [K] = [P_{N-1}][x_N][x_N]^\top [P_{N-1}]$.
6: $\quad$ **for all** entries $\{i, j\}$ in $[K]$ **do**
7: $\quad\quad [H_{i,j}] = \mathsf{comp}([K_{i,j}])$.
8: $\quad$ **end for**
9: $\quad$ **for all** entries $\{i, j\}$ in $[H]$ **do**
10: $\quad\quad [G_{i,j}] = [K_{i,j}] - 2[H_{i,j}][K_{i,j}]$
11: $\quad$ **end for**
12: $\quad$ **for all** entries $\{i, j\}$ in $[G]$ **do**
13: $\quad\quad [Q_{i,j}] = \mathsf{divSec}([G_{i,j}], [d])$.
14: $\quad$ **end for**
15: $\quad$ **for all** entries $\{i, j\}$ in $[Q]$ **do**
16: $\quad\quad [Z_{i,j}] = [Q_{i,j}] - 2[H_{i,j}][Q_{i,j}]$
17: $\quad$ **end for**
18: $\quad [P_N] = [P_{N-1}] - [Z]$
19: $\quad [g_N] = [P_N][x_N]$
20: $\quad [e_N] = [y_N] - \mathsf{reScale}([x_N]^\top [\hat{w}_{N-1}], 2^C)$
21: $\quad [\hat{w}_N] = ([\hat{w}_{N-1}] + [e][g])$
22: **end for**

Correctness. In steps 4 and 5, the nominator and denominator in (C.7) are calculated. Since P is a positive semidefinite matrix, the denominator cannot be negative; thus, we only need to check which entries in the matrix nominator are negative. This is done in steps 6 and 7. Then, steps 9 and 10 multiply the negative entries in the nominator matrix with -1. Steps 12 and 13 integer divides the nominator matrix with the denominator, and afterwards, steps 15 and 16 adds the sign to the appropriate entries again. Steps 18 to 21 calculate (C.8) to (C.10).

divSec$([n], [d])$

The next sub protocol is divSec$([n], [d])$, that outputs the integer quotient, $n \backslash d$ for two l- bit integers $n, d \in \mathbb{F}_q$. This protocol itself consists of several sub protocols which will be described in the following.

The idea in divSec$([n], [d])$ is borrowed from the work [23], but the proposed protocols are for secure two-party computation, so to use them for SMPC they must be adjusted.

The idea presented in [23] is to compute the integer division, $n \backslash d$ by first computing $a = 2^k \backslash d$ and then computing $c = n \cdot a \backslash 2^k$, where k is a public constant. The first division, $a = 2^k \backslash d$, can be calculated using the Taylor series;

$$a = 2^{k-l_d} \sum_{i=0}^{w} \left((2^{l_d} - d) 2^{-l_d} \right)^i, \qquad (C.11)$$

where l_d is the bit length of d, i.e. $l_d = \lfloor \log_2(d) + 1 \rfloor$, $k = l^2 + l$ and $w = l$. The derivation of (C.11) can be found in [23].

Thus, the sub protocols needed are those to calculate $[2^{l_d}]$ and $[2^{-l_d}]$.

Calculating $[2^{l_d}]$, is done in [23] by decomposing d into bits, even though they note that it can also be done without bit decomposition. For the simulations in 4 we implement the protocol that does not use bit decomposition, but here the other choice is presented, since it is more intuitive and readable.

Referring to $d_0, \ldots, d_{l-1}$ as the bits of d, where d_0 is the least significant bit, and letting $u_i = \vee_{j=i}^{l-1} [d_j]$ for $i = 0, \ldots, l-1$, 2^{l_d} can be written as

$$[2^{l_d}] = 1 + \sum_{i=0}^{l-1} u_i 2^i. \qquad (C.12)$$

To use this, a protocol for bit decomposing a secret is needed, this protocol is referred to as bitDec$([a])$ and returns $[a]^B$. Here, we use the one presented in [18, p. 189], which we do not introduce further here. Now, bitLen$([x])$ that returns information about the bit length of the input is stated in Protocol C.2.

After $[2^{l_d}]$ is calculated, 2^{-l_d} can be calculated by using a protocol, inverse$([x])$, that returns the inverse field element $[x^{-1}]$. This protocol uses the Reconstruct algorithm defined in **Definition C.1** that, using communication, allows the parties to recreate a secret from its shares. inverse$([x])$ is stated in Protocol C.3.

Now, divSec$([n], [d])$ is stated in Protocol C.4.

reScale$([x], 2^k)$

The final sub protocol, reScale$([x], 2^k)$, securely computes $[2^{-k}x]$, with $x, 2^k \in \mathbb{F}_q$. The protocol is a combination of a public integer division protocol proposed by [24] and a public modulo reduction protocol proposed by [25]. The idea

---

Protocol C.2: $([2^c]) \leftarrow \mathsf{bitLen}([x])$

Input: $[x]$, where $x \in \mathbb{F}_q$.
Output: $[2^c]$, where $c = \lfloor \log_2(x) + 1 \rfloor$.
 1: $[x]^B = \mathsf{bitDec}([x])$.
 2: **for all** $i = 0, \ldots, l - 1$ **do**
 3:     $[u_i] = 1 - \prod_{j=i}^{l-1}(1 - [x_i])$.
 4: **end for**
 5: $[2^c] = 1 + \sum_{i=0}^{l-1}[u_i]2^i$.

Correctness. Step 1 extracts the bitwise representation of $[x]$. Then, steps 3 and 5, calculates the so-called *prefix-OR* of $[x]$, which means that $u_i = \vee_{j=i}^{l-1} x_j$. Finally, step 5 calculates the integer representation of u, which added with one, constitutes the result.

Protocol C.3: $([c]) \leftarrow \mathsf{inverse}([x], [r])$

Input: $[x]$, where $x \in \mathbb{F}_q$, and $[r]$, where r is a random number.
Output: $[c]$, where $c = x^{-1} \in \mathbb{F}_q$.
 1: $[w] = [x][r]$
 2: $w = \mathsf{Reconstruct}([w])$.
 3: $[c] = w^{-1}[r]$.

Correctness. Step 1 securely multiplies $[x]$ with $[r]$, and opens it to all parties. Since r is a uniformly random integer unknown to all parties, xr is uniformly random and thus does not leak information about x. $(rx)^{-1}$ can then be computed in the open and in step 3 it is securely multiplied with $[r]$, which gives the desired result.

Protocol C.4: $([c]) \leftarrow \mathsf{divSec}([n], [d])$

Input: $[n]$ and $[d]$, where $n, d \in \mathbb{F}_q$ are l-bit integers.
Output: $[c]$, where $c = n \backslash d$.
 1: $[2^{l_d}] = \mathsf{bitLen}([d])$
 2: $[2^{-l_d}] = \mathsf{inverse}([2^{l_d}])$
 3: $[d'] = ([2^{l_d}] - [d])[2^{-l_d}]$
 4: $[a] = 2^{l^2 + l}[2^{l_d}] \sum_{i=0}^{l}[d']^i$.
 5: $[q] = [n][a]$
 6: $[c] = \mathsf{reScale}([q], 2^{l^2 + l})$

Correctness. The protocol performs the necessary steps to securely compute (C.11).

in the protocol is to first subtract $x \mod 2^k$ from the nominator and then multiply this with 2^{-k}. The protocol is stated in Protocol C.5.

Protocol C.5: $([c]) \leftarrow \mathsf{reScale}([x], [r_\top], 2^k)$

Input: $[x]$, where $x \in \mathbb{F}_q$, $[r]$, where $r_\top \in \mathbb{F}_q$ is random, and a public constant $2^k \in \mathbb{F}_q$.
Output: $[c]$, where $c = x \backslash 2^k$.
1: $[x]_B \leftarrow \mathsf{bitDec}([x])$.
2: $[\bar{x}] = \sum_{i=0}^{k-1} [x_i] 2^i$.
3: $[\hat{x}] = [x] - [\bar{x}]$.
4: $k_{inv} = 2^{-k} \mod p$
5: $[c] = k_{inv}[\hat{x}]$.

Correctness. Step 1 extracts the bit decomposition of $[x]$, which is used in step 2 to compute $[\bar{x}] = [x \mod 2^k]$. $[\hat{x}]$ is obtained in step 3 by subtracting $[\bar{x}]$ from $[x]$, which ensures that 2^k divides $[\hat{x}]$. $[c]$ is achieved in step 5 by multiplying the inverse field element of 2^k with $[\hat{x}]$.

4 Simulations and Results

In this section, we compare simulation results of Protocol C.1, with the corresponding results of solving the RLS equations in a traditional (non-secure) manner. This serves both as an evaluation of the accuracy of the proposed protocol, but also as an illustration of the developed concept. To compare results from both methods, the mean squared error (MSE) between the estimations and true parameters is used. The MSE of the parameter estimate at time i is defined as

$$e_{MSE_i} = \frac{1}{q} \sum_{j=1}^{q} (w_j - \hat{w}_j)^2, \quad \text{for } i = 1, 2, \ldots.$$

The simulation results are obtained from the implementations we have published in [26].

In the sequel, we have estimated parameters of a six dimensional system, and similar results were obtained independent of the dimension of the system.

To estimate the parameters of a six dimensional linear system, we use the test data described by

$$y = 3x_1 + 2x_2 + 2x_3 + 4x_4 + 4x_5 + 1x_6, \tag{C.13}$$
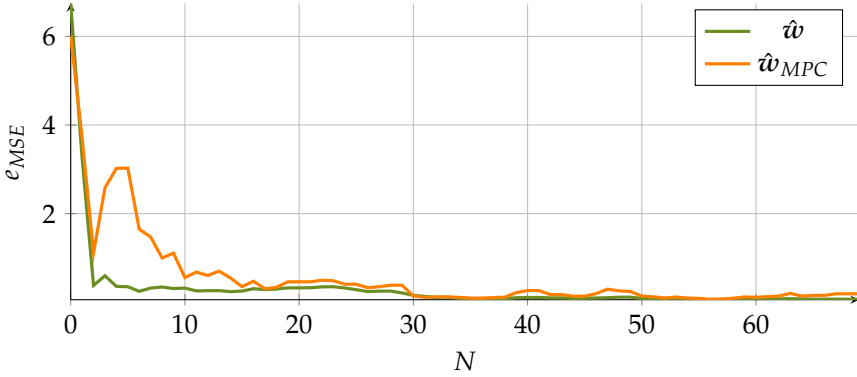
Fig. C.1: Estimating $w = [3, 2, 2, 4, 4, 1]$, using the described observations. $\hat{w}$ is the estimate obtained with the RLS equations and $\hat{w}_{MSE}$ is the estimate obtained with Protocol C.1.

thus the true parameter vector is $w = [3, 2, 2, 4, 4, 1]$.

The observations of $\{x_i\}_{i=1,\dots,6}$ are uniformly distributed on the interval $[0, 5]$ and the observations of y are given by (C.13). To each observation of y we have added a Gaussian distributed noise term with mean value zero and variance two.

The estimates of w obtained with the RLS equations ($\hat{w}$) and Protocol C.1 ($\hat{w}_{MPC}$) gives the e_{MSE_N} for both estimates respectively, as seen in Fig. C.1. It should be pointed out that the observations of y are real numbers, but to use them in Protocol C.1 they are truncated. This, together with the integer division used in Protocol C.1 explains the differences between the two curves in Fig. C.1.

4.1 Communication Complexity

Protocol C.1 consist of $\mathcal{O}(lq^2)$ multiplications. When the preprocessing phase is not counted, each multiplication results in each party broadcasting shares. While the communication complexity therefore is heavy, many local operations can be done in parallel, meaning that we can talk about *rounds of communication*, in which a number of values are broadcasted at once. For instance, a matrix-vector product can be computed in parallel, requiring only one round of communication. The round complexity of Protocol C.1 is $\mathcal{O}(l)$.

5 Conclusions

The paper proposes a secure multiparty protocol for a privacy preserving recursive least squares solution. As simulation shows, results of the secure protocol is similar to results of a non-secure evaluation of the recursive least

squares equations. The downside is that the proposed protocol is heavy in terms of communication, perhaps even so heavy that practical usage is infeasible. In some cases, we believe that further research can lead to efficient protocols, that are useful in practice.

Acknowledgement

References

[1] K. Tjell, I. Cascudo, and R. Wisniewski, "Privacy preserving recursive least squares solutions," in *2019 18th European Control Conference (ECC)*, (United States), pp. 3490–3495, IEEE, Aug. 2019. null ; Conference date: 25-06-2019 Through 28-06-2019.

[2] M. A. Lisovich, D. K. Mulligan, and S. B. Wicker, "Inferring personal information from demand-response systems," *IEEE Security Privacy*, vol. 8, pp. 11–20, Jan 2010.

[3] G. Avoine, L. Calderoni, J. Delvaux, D. Maio, and P. Palmieri, "Passengers information in public transport and privacy: Can anonymous tickets prevent tracking?," *International Journal of Information Management*, vol. 34, pp. 682–688, oct 2014.

[4] S. Han and G. J. Pappas, "Privacy in control and dynamical systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 309–332, 2018.

[5] Y. Wang, S. Mitra, and G. E. Dullerud, "Differential privacy and minimum-variance unbiased estimation in multi-agent control systems," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9521 – 9526, 2017. 20th IFAC World Congress.

[6] Y. Wang, Z. Huang, S. Mitra, and G. E. Dullerud, "Differential privacy in linear distributed control systems: Entropy minimizing mechanisms and performance tradeoffs," *IEEE Transactions on Control of Network Systems*, vol. 4, pp. 118–130, March 2017.

[7] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, (New York, NY, USA), pp. 371–380, ACM, 2009.

[8] K. H. Degue and J. L. Ny, "On differentially private kalman filtering," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 487–491, Nov 2017.

[9] J. He, L. Cai, and X. Guan, "Preserving data-privacy with added noises: Optimal estimation and privacy analysis," *IEEE Transactions on Information Theory*, vol. 64, pp. 5677–5690, Aug 2018.

[10] I. Giacomelli, S. Jha, M. Joye, C. D. Page, and K. Yoon, "Privacy-preserving ridge regression with only linearly-homomorphic encryption," in *Applied Cryptography and Network Security* (B. Preneel and F. Vercauteren, eds.), (Cham), pp. 243–261, Springer International Publishing, 2018.

[11] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *PoPETs*, vol. 2017, pp. 345–364, 2017.

[12] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *2013 IEEE Symposium on Security and Privacy*, pp. 334–348, May 2013.

[13] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, (Berlin, Heidelberg), pp. 192–206, Springer-Verlag, 2008.

[14] D. Bogdanov, L. Kamm, S. Laur, and V. Sokk, "Rmind: A tool for cryptographically secure statistical analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, pp. 481–495, May 2018.

[15] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," *IACR Cryptology ePrint Archive*, vol. 2018, p. 403, 2018.

[16] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, May 2017.

[17] D. Demmler, T. Schneider, and M. Zohner, "Aby - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[18] R. Cramer, I. B. Damgaard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 1 ed., 2015.

[19] C. Orlandi, "Is multiparty computation any good in practice?," *I E E E International Conference on Acoustics, Speech and Signal Processing. Proceedings*, pp. 5848–5851, 2011. Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference.

[20] M. Keller, V. Pastro, and D. Rotaru, *Overdrive: Making SPDZ Great Again*, pp. 158–189. 01 2018.

[21] S. Haykin, *Adaptive Filter Theory*. Pearson, 5 ed., 2014.

References

[22] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *Public Key Cryptography – PKC 2007* (T. Okamoto and X. Wang, eds.), (Berlin, Heidelberg), pp. 343–360, Springer Berlin Heidelberg, 2007.

[23] M. Dahl, C. Ning, and T. Toft, "On secure two-party integer division," in *Financial Cryptography and Data Security* (A. D. Keromytis, ed.), (Berlin, Heidelberg), pp. 164–178, Springer Berlin Heidelberg, 2012.

[24] C. Ning and Q. Xu, "Multiparty computation for modulo reduction without bit-decomposition and a generalization to bit-decomposition," in *Advances in Cryptology - ASIACRYPT 2010* (M. Abe, ed.), (Berlin, Heidelberg), pp. 483–500, Springer Berlin Heidelberg, 2010.

[25] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft, "Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation," in *Theory of Cryptography* (S. Halevi and T. Rabin, eds.), (Berlin, Heidelberg), pp. 285–304, Springer Berlin Heidelberg, 2006.

[26] K. Tjell, "Protocol and algorithm implementations." `goo.gl/9ddxL4`.

References

Paper D

Privacy Preservation in Distributed Optimization via Dual Decomposition and ADMM

Katrine Tjell and Rafael Wisniewski

Abstract

In this work, we explore distributed optimization problems, as they are often stated in energy and resource optimization. More precisely, we consider systems consisting of a number of subsystems that are solely connected through linear constraints on the optimized solutions. The focus is put on two approaches; namely dual decomposition and alternating direction method of multipliers (ADMM), and we are interested in the case where it is desired to keep information about subsystems secret. To this end, we propose a privacy preserving algorithm based on secure multiparty computation (SMPC) and secret sharing that ensures privacy of the subsystems while converging to the optimal solution. To gain efficiency in our method, we modify the traditional ADMM algorithm.

1 Introduction

Developments in technology have enabled efficient collection of data and ensured powerful signal transmissions. This is beneficial in many cases since collected data has the potential of aiding accurate estimations and predictions as well as the calculation of solutions customized to individuals. Take for instance electricity production and pressure control in water networks as examples of control systems with a potential of reducing resource losses by accurately predicting the consumption through fine grained measurements. Another example could be the future smart house that will adapt more and more to the occupants in pace with the more data it collects. To this end, collected data is valuable, however, more often than not it contains privacy sensitive information. For instance, the activities of occupants in a household can be inferred from electricity consumption measurements as showed in [2]. Most people are cautious to share this kind of information; hence, they will also be reluctant to share their consumption data. The idea, thus, is to develop methods that can exploit the benefits of the data without risking the privacy of individuals.

The focus in this paper is on distributed optimization where several agents each hold a part of the optimization problem. This kind of problem appears in many applications, take for instance resource allocation between subsystems, formation control of autonomous vehicles and distributed resource generation.

The problem in focus can be described as a system containing N agents that each have a private objective. Moreover, the overall system describes a set of linear constraints that couples the N individual optimization problems. To exemplify, the agents could be power production units and a constraint could be that the sum of produced power meet a certain requirement or that the sum of resources used may not exceed a given limit. Each agent is unwill-

ing to share information concerning their system, so the individual objectives as well as optimized solutions must be kept private. Furthermore, the constraints must also remain secret as they may reveal information about the individual systems. Specifically, we assume that the constraints are designed by a *superintendent* and will only be used in encrypted form.

We study distributed optimization via dual decomposition and Alternating Direction Method of Multipliers (ADMM) and propose privacy preserving solutions. In particular, we put forward a method based on Secure Multiparty Computation (SMPC) and cloud computing, that minimizes the objective while leaking no information about the system.

Related work. The number of papers focusing on privacy preserving control and optimization is increasing. Zhang et al.'s work [3] is closely related to ours as they also consider an ADMM based privacy preserving distributed optimization solution. However, they allow every agent to communicate with their neighbour using homomorphic encryption, which is in contrast to our solution, where the agents communicate only with a number of so-called computing parties using secure multiparty computation. We believe that this solution scales better with the number of agents and is computationally less demanding. Many recent papers within this subject are based on homomorphic encryption, for instance [4] and [5]. There is also a substantial amount of work, basing the privacy preservation on differential privacy, for instance [6], [7], and [8]. This approach requires at least some trust in the system, which our solution does not exhibit.

Also of interest to our work is [9] and [10] that investigates how to preserve privacy and integrity in cloud computations.

Structure. In Section 2, the specific problem studied throughout the paper is presented. Subsequently, Section 3 gives an introduction to the two building blocks of the paper, namely SMPC and secret sharing. In the Sections 4 and 5, we introduce the main contribution, which is privacy preserving distributed optimization. In Section 4, our ideas are presented using a dual decomposition approach. The main purpose of this section is to provide intuition about the methods, since the ADMM based approach presented in Section 5 is more likely to have a practical relevance. Finally, a discussion is provided in Section 6.

2   Problem formulation

In the paper, we consider the setup consisting of N agents, $A_1, \ldots, A_N$, each having a local objective function $f_1(\boldsymbol{x}_1), \ldots, f_N(\boldsymbol{x}_N)$ and a *superintendent* having requirements for the agents. Remark, we assume that $f_i(\boldsymbol{x}_i)$ is known only to agent i and that the constraints are known only to the superintendent. The agents want to minimize their individual objective function, while

the superintendent wants the result achieved by the agents to fulfill certain requirements. These requirements are formulated as M linear constraints on the form

$$\sum_{i=1}^{N} B_i x_i = c, \tag{D.1}$$

where $x_i \in \mathbb{R}^q$, $B_i \in \mathbb{R}^{M \times q}$, and $c \in \mathbb{R}^{M \times 1}$. The overall problem can thus be stated as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{N} B_i x_i - c = 0, \end{aligned} \tag{D.2}$$

where $x = [x_1, \ldots, x_N]$.

Concerning the approach to solving (D.2), we are interested in the case where each agent are not willing to share information about their system with the other agents or the public in general. A reason for this could for instance be that the system information is considered corporate secrets. At the same time, we assume that the agents are honest in the sense that they will follow computational instructions and not tamper with any intermediate results. Nonetheless, they may attempt to disclose the secret information of other participants and our methods must prevent this.

The goal in the paper is to solve (D.2), without leaking any private information. We base the methods on SMPC and secret sharing. However, one could exchange SMPC with homomorphic encryption in our protocols. This would entail less communication but more computation, thus the choice is application dependent. A short introduction to SMPC and secret sharing is provided in the following section, and the reader is referred to [11] for a more elaborate explanation.

3 Secure Multiparty Computation based on Secret Sharing

Consider the evaluation of a function $f(x_1, \ldots, x_n)$, where each of the inputs are provided by a distinct party, say $P_1, \ldots, P_n$. The parties want to learn the output of the function, nonetheless, they are not willing to reveal their individual inputs.

This scenario is the central problem in SMPC, that has the aim of developing protocols that allow the parties to evaluate the function without disclosing their private information. This is achieved by applying encryption techniques to hide the secret data. In particular, secret sharing methods are

popular to use as the encryption technique in SMPC, since this usually entails protocols with a low computational complexity.

Secret sharing aims to avoid the situation where one entity holds a secret, since in case of an attack, the adversary then has to attack multiple entities to learn the secret. The secret is "chunked" into pieces that can be distributed among several parties. One piece of information is referred to as a *share* and by itself it reveals nothing about the secret. Specifically, it takes some or all shares to recreate the secret. In Definition D.1, we give a definition of a secret sharing scheme.

> **Definition D.1 (Secret Sharing Scheme)**
> A secret sharing scheme consists of two algorithms namely Share and Reconstruct. The first one takes a secret s and creates the *shares*, which are values $s_1, \ldots, s_n$, where n is the number of parties. The latter one outputs s upon given any set of at least t shares, where t is the threshold for the scheme. Additionally, it holds that no information about the secret can be gained from a set of fewer than t shares.

Since a set of fewer than t shares reveals nothing about the secret, the threshold can be used to adopt the protocol to the expectation of so-called *corrupted* parties. If a set of parties collude in inferring information about private values of other parties they are referred to as corrupted parties. Moreover, we distinguish between corrupted parties that follow the protocol and ones that do not. The former is referred to passive corruption, while the latter is referred to as active.

We use $[\cdot]$ to denote the secret sharing scheme, i.e., $[s]$ is the shares $(s_1, \ldots, s_n)$ of the secret s.

The particular choice of the algorithm Share in the SMPC protocol, determines the computations that can be done on the secrets. In this paper, one can use any linear secret sharing scheme that takes an integer secret modulo some prime p and outputs integer shares also modulo p. Moreover, the scheme should satisfy that a shared version of the sum $[x + y]$ and product $[xy]$ modulo p, can be computed given $[x]$ and $[y]$. One can for instance use *Shamir's secret sharing scheme*, for which the above mentioned properties hold when the threshold $t < \frac{n}{2}$.

To give an overview, the privacy preserving protocols proposed in this paper works by hiding secret values using the Share algorithm. A caveat is that, as mentioned, secret values are integers modulo a prime p. We can choose p large enough, so that we do not have wrap-arounds, but eventually the values x, B and c needs to be truncated before Share can be used to create shares of them. Scaling can be used to minimize the truncation error, however, it

is unavoidable not to induce quatization errors. We will not touch upon this issue any further.

The desired output is computed as each party alternately distributes the shares among the parties and performs local operations on the shares. At the end, all parties can use the Reconstruct algorithm to learn the output.

In our case, the aforementioned parties can be the agents themselves or to reduce communication, independent computing parties (for instance cloud servers), can be applied to perform the calculations. For the latter choice, efficient frameworks such as ABY3 and SecureML introduced by Mohassel and Rindal in [12] and Mohassel and Zhang in [13] respectively, can be employed. The number of computing parties are three in the case of ABY3 and two in the case of SecureML, and security is achieved against one actively corrupted server.

For this work, we consider the case where n computing parties perform the SMPC, but one can adapt the protocols to the other case as well. That is, only the computing parties will do computations on shares, and the agents and superintendent will only provide inputs to the protocol. Fig. D.1 gives an illustration of how the agents are connected to the computing parties while the computing parties are all connected to each other.
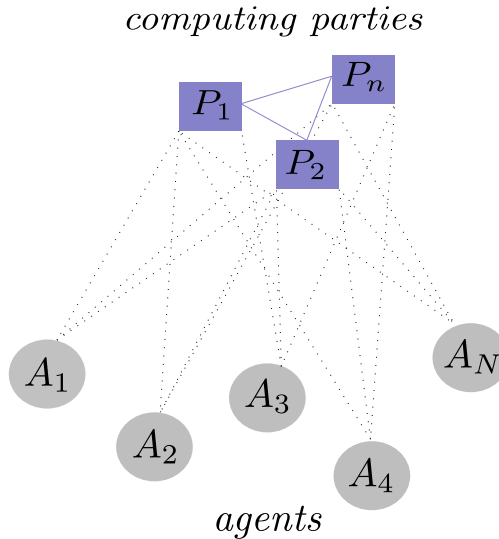

Fig. D.1: The N agents connected to the n computing parties.

3.1 Preliminaries

As already mentioned, we will use $[x]$ to denote the shares of the value x. When we say that a number of parties *hold* $[x]$, we mean that each party has one share of x. In continuation, when a number of parties hold $[x]$ and $[X]$, it means that each party has a share of each of the entries in the vector x, respectively the matrix X. Furthermore, the following is a recap of the assumptions we make.

- The agents are honest but curious. That is, they will follow instructions but may attempt to disclose information. However, we assume the agents do not collude.

- A majority of the computing parties are honest and not colluding. If active attacks on the computing parties are expected, one can for instance use the techniques presented in [14] to prevent this, otherwise one can use for instance Shamir's Secret sharing scheme to ensure privacy against a passive adversary.

- The superintendent is honest and not colluding with any other entity.

- Only A_i knows f_i.

- Only the superintendent knows the constraints.

4 Dual Decomposition based Privacy Preserving Optimization

In this section, we use the idea in dual decomposition to solve (D.2). In the following, we merely present the standard dual decomposition algorithm, thus the interested reader is referred to [15] for a more thorough introduction to the subject.

For (D.2), the Lagrangian is

$$L(x, \lambda) = \sum_{i=1}^{N} (f_i(x_i)) + \lambda^\top (Bx - c), \tag{D.3}$$

where $B = [B_1, \ldots, B_N]$. However, it can equivalently be written as

$$L(x, \lambda) = \sum_{i=1}^{N} \left(f_i(x_i) + \lambda^\top B_i x_i \right) - \lambda c. \tag{D.4}$$

The idea in dual decomposition is then to minimize the Lagrangian by solving N sub problems, one for each x_i. This is done in iterations, where subsequently the dual variables are updated. The algorithm can be described by

the following to steps, where the first is performed in parallel by each agent i:

$$\begin{aligned}
x_i^{k+1} &= \min_{x_i} f_i(x_i) + \lambda^{k\top} B_i x_i \\
\lambda^{k+1} &= \lambda^k + \alpha(Bx^{k+1} - c),
\end{aligned} \tag{D.5}$$

where $\alpha > 0$ is a step size and $x^k = [x_1^{k\top}, \ldots, x_N^{k\top}]^\top$. In the following section, SMPC and cloud computing are employed to introduce a privacy preserving version of dual decomposition, where x^k, B and c remain secret during protocol execution.

4.1 Privacy Preserving Optimization using Dual Decomposition

We start by make a couple of remarks. The first one is that by sampling a uniform random matrix T and applying it on the constraints, i.e.,

$$TBx - Tc = 0,$$

yields the same constraints, but they are now masked by a random matrix. This will only work if T is non-singular. A singular T can almost always be avoided by constructing it in a special way. This result is stated in Lemma D.1, which follows from application of Theorem 2 in [16].

Lemma D.1
Let K be a finite field with cardinality $|K|$. Suppose $T = UL$, where

$$U = \begin{bmatrix} 1 & u_2 & u_3 & \ldots & u_n \\ 0 & u_1 & u_2 & \ldots & u_{n-1} \\ \ldots & & & & \\ 0 & 0 & 0 & \ldots & u_2 \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}, \quad L = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ v_2 & 1 & 0 & \ldots & 0 \\ v_3 & v_2 & 1 & \ldots & 0 \\ \ldots & & & & \\ v_n & v_{n-1} & v_{n-2} & \ldots & 1 \end{bmatrix}$$

are $K^{n \times n}$ Toeplitz matrices with independent uniformly distributed random entries. Then

$$P[\det(T) \neq 0] \geq 1 - \frac{n(n+1)}{|K|}.$$

We propose to create T using Lemma D.1 and a preprocessing phase that the computing parties will run before protocol execution, i.e., each party will obtain a share of each entry in T and no party will know the actual T. For details about how T is created in a preprocessing phase, we refer to [11]. After T is created, the computing parties must securely check the determinant of T, which can be done directly on the shares as shown in [17]. If the determinant

is zero they will start over, thus ensuring a non-singular T at the beginning of protocol execution.

The second remark is that the minimization of agent i can be written as

$$x_i^{k+1} = \min_{x_i} f_i(x_i) + v_i^{k\top} x_i, \tag{D.6}$$

where

$$v_i^{k\top} = \lambda^{k\top} T B_i. \tag{D.7}$$

The privacy preserving algorithm will then alternate between the agents solving their individual minimization problem in parallel and the computing parties doing SMPC. When each agent has solved its problem, it will create shares of the result and send one share to each of the computing parties. Each computing party will also receive a share of B and c from the superintendent, meaning that the parties can compute v_i^k, by performing computations directly on shares. In order to avoid having the computing parties working with α, which is not an integer, we introduce an integer $d > 1$ and let $\alpha = \frac{1}{d}$. The computing parties will then compute $\tilde{v}_i^k = dv_i^k$.

Each agent i will receive $\tilde{v}_i^k$ from the computing parties where after it can compute $v_i^k = \frac{1}{d}\tilde{v}_i^k$. It is of course vital that v_i^k does not leak information allowing agent i to learn either B, c, or x_j for $j \neq i$. We provide a proof that v_i^k does not leak information in the following.

Lemma D.2
Disclosing v_i^k to agent i at time k does not leak information.

Proof. Because T is applied on B, each entry in v_i^k is a sum of uniformly distributed random variables. Thus, agent i does not learn more from v_i^k than he could from drawing numbers from a uniform distribution. □

The protocol is written formally in Algorithm D.1.

We now expand the solution presented in this section, to a solution based on the ADMM algorithm.

5 Alternating Direction Method of Multipliers

The ADMM method is based on the *augmented* Lagrangian, which entails that the ADMM has convergence advantages over dual decomposition. We refer the reader to [15] for an introduction to ADMM; here, we merely present the algorithm. In regards to this, we note that the standard ADMM problem is on the form

$$\begin{aligned}
\underset{x}{\text{minimize}} \quad & f(x) + g(z) \\
\text{subject to} \quad & A_1 x + A_2 z = b,
\end{aligned} \tag{D.11}$$

Algorithm D.1: Privacy Preserving Dual Decomposition Method

1: The computing parties hold $[T]$, where $[T]$ is a uniformly random $M \times M$ matrix obtained in a preprocessing phase.
2: The computing parties hold $[B]$ and $[c]$, received from the super intended.
3: $d > 1$ is an integer.
4: $\alpha = \frac{1}{d}$, $v^0 = \mathbf{0}$, $\lambda^0 = \mathbf{0}$.
5: **for all** $k = 0, \ldots$ **do**
6:   Each agent i computes

$$v_i^k = \frac{1}{d} \tilde{v}_i^k. \tag{D.8}$$

  after receiving all shares of $\tilde{v}_i^k$ from the computing parties.
7:   Each agent i computes x_i^{k+1} by solving (D.6).
8:   Each agent i creates shares of x_i^{k+1} and send one share to each of the computing parties.
9:   The computing parties compute jointly:

$$[\lambda^{k+1}] = [\lambda^k] + ([T][B] - [T][c])[x^{k+1}], \tag{D.9}$$

  and

$$[\tilde{v}_i^{k+1^\top}] = [\lambda^{k+1^\top}][T][B_i]. \tag{D.10}$$

  for $i = 1, \ldots, N$.
10:   The computing parties send all shares of $[\tilde{v}_i]$ to agent i for $i = 1, \ldots, N$.
11: **end for**

---

and the augmented Lagrangian for this problem is

$$
\begin{aligned}
L_\rho(x, z, \lambda) = & f(x) + g(z) + \lambda^\top (A_1 x + A_2 z - b) \\
& + \frac{1}{2}\rho||A_1 x + A_2 z - b||_2^2.
\end{aligned}
\tag{D.12}
$$

Consequently, the ADMM algorithm consists of the following steps

$$
\begin{aligned}
x^{k+1} &= \min_x L_\rho(x, z^k, \lambda^k) \\
z^{k+1} &= \min_z L_\rho(x^{k+1}, z, \lambda^k) \\
\lambda^{k+1} &= \lambda^k + \rho(A_1 x^{k+1} + A_2 z^{k+1} - b),
\end{aligned}
\tag{D.13}
$$

where $\rho > 0$.

The problem in (D.2) is almost on the standard ADMM form, the main difference is that instead of two blocks in (D.11), there are N blocks in (D.2). This modification is important to avoid communication among agents. The augmented Lagrangian for (D.2) is

$$
L_\rho(x, \lambda) = \sum_{i=1}^{N} (f_i(x_i)) + \lambda^\top (Bx - c) + \frac{1}{2}\rho||Bx - c||_2^2.
\tag{D.14}
$$

Depending on the size of N, performing the sub optimizations sequentially may take a long time. Therefore, we propose that each agent solve their individual minimization in parallel, i.e., we propose the modified ADMM algorithm

$$
\begin{aligned}
x_i^{k+1} &= \min_{x_i} L_\rho(x_1^k, \ldots, x_i, \ldots, x_N^k, \lambda^k) \quad \text{for } i = 1, \ldots, N \\
\lambda^{k+1} &= \lambda^k + \rho\left(\sum_{i=1}^{N} B_i x_i^{k+1} - c \right).
\end{aligned}
\tag{D.15}
$$

Unfortunately, this approximation of the ADMM algorithm is likely to diverge unless precautions are taken. One approach is proposed (and proved) by [18] and involves adding an underrelaxation step, such that the algorithm becomes the following;

$$
\begin{aligned}
\tilde{x}_i^{k+1} &= \min_{x_i} L_\rho(x_1^k, \ldots, x_i, \ldots, x_N^k, \lambda^k) \quad \text{for } i = 1, \ldots, N \\
\tilde{\lambda}^{k+1} &= \lambda^k + \rho\left(\sum_{i=1}^{N} B_i \tilde{x}_i^{k+1} - c \right). \\
x_i^{k+1} &= x_i^k - \frac{1}{N+1}(x^k - \tilde{x}_i^{k+1}) \\
\lambda^{k+1} &= \lambda^k - \frac{1}{N+1}(\lambda^k - \tilde{\lambda}^{k+1}).
\end{aligned}
\tag{D.16}
$$

5.1 Privacy Preserving ADMM

For obtaining a privacy preserving ADMM protocol, we use the same principles as we did in the privacy preserving dual decomposition method. Namely, we mask B and c with a uniformly random matrix T, such that the masked constraints yields

$$TBx = Tc. \tag{D.17}$$

Next, we rewrite the minimization problem of each agent. To do this, consider the minimization of (D.14), where we focus on $\lambda^\top(Bx - c) + \frac{\rho}{2}||Bx - c||_2^2$ since these terms cannot be computed locally. We consider the minimization with respect to only one element, x_e, in the vector x.

$$
\begin{aligned}
&\min_{x_e} \lambda Bx + \frac{\rho}{2}\left(\sum_{j=1}^{M}(b_j x - c_j)^2\right) \\
=&\min_{x_e} \sum_{j=1}^{M}\lambda_j b_{j,e} x_e + \frac{\rho}{2}\left(\sum_{j=1}^{M} b_j x x^\top b_j^\top - 2b_j x c_j\right) \\
=&\min_{x_e} \sum_{j=1}^{M}\lambda_j b_{j,e} x_e + \frac{\rho}{2}\sum_{j=1}^{M} b_{j,e}^2 x_e^2 \\
&+ \frac{\rho}{2}x_e \sum_{j=1}^{M}\sum_{k=1,k\neq e}^{N} 2b_{j,e}b_{j,k}x_k - \frac{\rho}{2}x_e\sum_{j=1}^{M} 2b_{j,e}c_j \\
=&\min_{x_e} x_e^2 \frac{\rho}{2}\sum_{j=1}^{M} b_{j,e}^2 + \\
&x_e \sum_{j=1}^{M} b_{j,e}\left(\lambda_j - \rho c_j + \rho\sum_{k=1,k\neq e}^{N} b_{j,k}x_k\right) \\
=&\min_{x_e} \beta_{1,e}x_e^2 + \beta_{2,e}x_e,
\end{aligned}
\tag{D.18}
$$

where b_j is the j'th row of B, $b_{j,e}$ is the j, e'th element of B and

$$
\begin{aligned}
\beta_{1,e} &= \frac{\rho}{2}\sum_{j=1}^{M} b_{j,e}^2 \\
\beta_{2,e} &= \sum_{j=1}^{M} b_{j,e}\left(\lambda_j - \rho c_j + \rho\sum_{k=1,k\neq e}^{N} b_{j,k}x_k\right).
\end{aligned}
\tag{D.19}
$$

Most of the reductions in (D.18) are because terms that are constant with respect to x_e does not affect the minimization and can be disregarded.

Define $s_{t,j} = [\beta_{t,(j-1)q+1}, \beta_{t,(j-1)q+2}, \dots, \beta_{t,(j-1)q+q}]$ for $j = 1,\dots,N$ and $t = 1, 2$. To improve readability, $s_{1,j}$ and $s_{2,j}$ are illustrated in the following.

$$
\begin{aligned}
\left.\begin{array}{c} \beta_{1,1} \\ \vdots \\ \beta_{1,q} \end{array}\right\} s_{1,1} \qquad\qquad \left.\begin{array}{c} \beta_{2,1} \\ \vdots \\ \beta_{2,q} \end{array}\right\} s_{2,1} \\
\left.\begin{array}{c} \beta_{1_{q+1}} \\ \vdots \\ \beta_{1_{2q}} \end{array}\right\} s_{1_2} \qquad\qquad \left.\begin{array}{c} \beta_{2_{q+1}} \\ \vdots \\ \beta_{2_{2q}} \end{array}\right\} s_{2_2} \\
\vdots \qquad\qquad\qquad \vdots \\
\left.\begin{array}{c} \beta_{1,(N-1)q+1} \\ \vdots \\ \beta_{1,Nq} \end{array}\right\} s_{1,N} \qquad \left.\begin{array}{c} \beta_{2,(N-1)q+1} \\ \vdots \\ \beta_{2,Nq} \end{array}\right\} s_{2,N}
\end{aligned}
\tag{D.20}
$$

The optimization problem of node i is now formulated as;

$$
\tilde{x}_i^k = \min_{x_i} f_i(x_i) + \operatorname{diag}(\{s_{1,i}\})x_i^2 + \operatorname{diag}(\{s_{2,i}\})x_i, \tag{D.21}
$$

where $\operatorname{diag}(x)$ is a matrix with x on the diagonal and 0 everywhere else.

The privacy preserving algorithm will then operate by at each time k, each agent i will calculate $\tilde{x}_i^k$ by solving (D.21) upon receiving $s_{1,i}^k$ and $s_{2,i}^k$ from the computing parties. The agents then create shares of $\tilde{x}_i^k$ and sends one share to each computing party. The computing parties will calculate $s_{1_i}^{k+1}$ and $s_{2,i}^{k+1}$ for $i = 1, \ldots, n$ by operating directly on the shares. The algorithm will obviously only be privacy preserving if $s_{1,i}^{k+1}$ and $s_{2,i}^{k+1}$ does not leak information. We provide a proof of this immediately.

Lemma D.3
Disclosing $s_{1,i}^k$ and $s_{2,i}^k$ to agent i does not disclose information.

Proof. Since T is a uniformly random matrix, and the sum and product of two uniform random variables are uniformly distributed, [11] page 254, it can be verified that all terms in $s_{1,i}^k$ and $s_{2,i}^k$ are uniformly distributed random variables. Thus, that agent i learns these two values does not reveal information. □

In Algorithm D.2, the privacy preserving protocol for the modified ADMM algorithm is stated formally.

Remark, in (D.23) we have assumed that there exist SMCP protocols for dividing a secret with a public integer, see [19] for more on this topic.

Algorithm D.2: Privacy Preserving ADMM

1: The computing parties hold $[T]$, where $[T]$ is a uniformly random $M \times M$ matrix obtained in a preprocessing phase.
2: The computing parties hold $[B]$ and $[c]$, received from the super intended.
3: $\rho > 1$ is an integer.
4: $\hat{B} = TB$, $\hat{c} = Tc$, $\lambda^0 = \tilde{\lambda}^0 = 0$, $x = \tilde{x} = 0$.
5: $s_{1,i}^0 = 0$ and $s_{2,i}^0 = 0$ for $i = 1, \ldots, N$.
6: **for all** $k = 0, \ldots$ **do**
7:     Each agent i receives $s_{1_i}^k$ and $\tilde{s}_{2,i}^k$.
8:     Each agent i computes $s_{2,i}^k = \frac{1}{d}\tilde{s}_{2,i}^k$ and $\tilde{x}_i^{k+1}$ by solving (D.21).
9:     Each agent i creates shares of $\tilde{x}_i^{k+1}$ and send one share to each computing party.
10:     The computing parties compute jointly:

$$[\tilde{\lambda}^{k+1}] = [\lambda^k] + \rho([\hat{B}][\tilde{x}^{k+1}] - [\hat{c}]). \tag{D.22}$$

11:     The computing parties compute

$$
\begin{aligned}
[x_i^{k+1}] &= [x_i^k] - \frac{([x^k] - [\tilde{x}_i^{k+1}])}{N+1} \\
[\lambda^{k+1}] &= [\lambda^k] - \frac{([\lambda^k] - [\tilde{\lambda}^{k+1}])}{N+1}.
\end{aligned}
\tag{D.23}
$$

    and $[s_{1,i}^{k+1}]$ and $[s_{2,i}^{k+1}]$ by using (D.20) and

$$
\begin{aligned}
[\beta_{1,e}] &= \left\lfloor \frac{\rho}{2} \right\rfloor \sum_{j=1}^{M} [\hat{b}_{j,e}][\hat{b}_{j,e}] \\
[\beta_{2,e}] &= \sum_{j=1}^{M} [\hat{b}_{j,e}] \\
&\left([\lambda_j^{k+1}] - \rho[\hat{c}_j] + \rho \sum_{t=1,t\neq e}^{N} [\hat{b}_{j,t}][x_t^{k+1}] \right),
\end{aligned}
\tag{D.24}
$$

    for $e = 1, \ldots, Nq$.
12:     Each of the computing parties sends their share of $[s_{1_i}^{k+1}]$ and $[s_{2_i}^{k+1}]$ to agent i for $i = 1, \ldots, N$.
13: **end for**

---

6 Discussion

The paper presents a privacy preserving dual decomposition protocol as well as a privacy preserving ADMM protocol. The former serves mostly as a more gentle introduction to the ideas in the paper, while the main contribution is the latter protocol.

Our proposed methods enjoy a decentralized setting, all though a number of so-called *computing parties* are employed to avoid communication between sub-systems. The computing parties are feed exclusively with encrypted values and only by attacking all parties (which we assume is infeasible) can information be learned.

7 Acknowledgment

References

[1] K. Tjell and R. Wisniewski, "Privacy preservation in distributed optimization via dual decomposition and admm," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, I E E E Conference on Decision and Control. Proceedings, (United States), pp. 7203–7208, IEEE, Mar. 2020. 2019 IEEE 58th Conference on Decision and Control (CDC), CDC ; Conference date: 11-12-2019 Through 13-12-2019.

[2] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin, "Private memoirs of a smart meter," in *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, (New York, NY, USA), pp. 61–66, ACM, 2010.

[3] C. Zhang, M. Ahmad, and Y. Wang, "Admm based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, pp. 565–580, March 2019.

[4] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. 314 – 325, 2018.

[5] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada, "Privacy-aware quadratic optimization using partially homomorphic encryption," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 5053–5058, Dec 2016.

[6] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, ICDCN '15, (New York, NY, USA), pp. 4:1–4:10, ACM, 2015.

[7] V. Rostampour, R. Ferrari, A. M. Teixeira, and T. Keviczky, "Differentially-private distributed fault diagnosis for large-scale nonlinear uncertain systems," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 975 – 982, 2018. 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018.

[8] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via objective perturbation," in *2016 American Control Conference (ACC)*, pp. 2061–2066, July 2016.

[9] Z. Xu and Q. Zhu, "Secure and resilient control design for cloud enabled networked control systems," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, CPS-SPC '15, (New York, NY, USA), pp. 31–42, ACM, 2015.

[10] N. Drucker, S. Gueron, and B. Pinkas, "Faster secure cloud computations with a trusted proxy," *IEEE Security Privacy*, vol. 15, pp. 61–67, November 2017.

[11] R. Cramer, I. B. Damgaard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 1 ed., 2015.

[12] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," *IACR Cryptology ePrint Archive*, vol. 2018, p. 403, 2018.

[13] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, May 2017.

[14] M. Pettai and P. Laud, "Automatic proofs of privacy of secure multi-party computation protocols against active adversaries," in *2015 IEEE 28th Computer Security Foundations Symposium*, pp. 75–89, July 2015.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, pp. 1–122, Jan. 2011.

[16] E. Kaltofen and B. D. Saunders, "On wiedemann's method of solving sparse linear systems," in *Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, AAECC-9, (London, UK, UK), pp. 29–38, Springer-Verlag, 1991.

[17] R. Cramer and I. Damgård, "Secure distributed linear algebra in a constant number of rounds," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, (London, UK, UK), pp. 119–136, Springer-Verlag, 2001.

[18] B. He, L. Hou, and X. Yuan, "On full jacobian decomposition of the augmented lagrangian method for separable convex programming," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2274–2312, 2015.

References

[19] O. Catrina and S. de Hoogh, "Improved primitives for secure multiparty integer computation," in *Security and Cryptography for Networks* (J. A. Garay and R. De Prisco, eds.), (Berlin, Heidelberg), pp. 182–199, Springer Berlin Heidelberg, 2010.

Paper E

Secure learning-based MPC via garbled circuit

Katrine Tjell, Nils Schlüter, Philipp Binfet, and Moritz Schulze Darup

Abstract

Encrypted control seeks confidential controller evaluation in cloud-based or networked systems. Many existing approaches build on homomorphic encryption (HE) that allow simple mathematical operations to be carried out on encrypted data. Unfortunately, HE is computationally demanding and many control laws (in particular non-polynomial ones) cannot be efficiently implemented with this technology.

We show in this paper that secure two-party computation using garbled circuits provides a powerful alternative to HE for encrypted control. More precisely, we present a novel scheme that allows to efficiently implement (non-polynomial) max-out neural networks with one hidden layer in a secure fashion. These networks are of special interest for control since they allow, in principle, to exactly describe piecewise affine control laws resulting from, e.g., linear model predictive control (MPC). However, exact fits require high-dimensional preactivations of the neurons. Fortunately, we illustrate that even low-dimensional learning-based approximations are sufficiently accurate for linear MPC. In addition, these approximations can be securely evaluated using garbled circuit in less than 100 ms for our numerical example. Hence, our approach opens new opportunities for applying encrypted control.

1 Introduction

Cloud computing and distributed computing are omnipresent in modern control systems such as smart grids, robot swarms, or intelligent transportation systems. While networked control offers exciting features, it also involves privacy and security concerns. Cybersecurity thus becomes an important part of the controller design [11]. Encrypted control addresses this challenge by providing modified controllers that are able to run on encrypted data [20].

Encrypted controllers have already been realized for various control schemes. For instance, encrypted implementations of (static or dynamic) linear feedback have been proposed in [5, 7, 12]. Encrypted distributed controllers can, e.g., be found in [7, 21]. Finally, encrypted MPC schemes have been presented in, e.g., [1, 18, 22].

For practical applications, secure MPC is of particular interest since networked control systems are often subject to input or state constraints and since optimization-based control can benefit from cloud-based implementations. Hence, encrypted MPC is also in the focus of this paper. More specifically, we propose a substantial extension to the scheme in [18] that enhances security, performance, and applicability.

In order to specify our contributions, we briefly summarize the underlying scheme next and point out its weaknesses. The approach in [18] builds on the observation that linear MPC can be exactly represented by tailored max-

out neural networks. These networks then build the basis for a (partially) secure two-party implementation. The main weaknesses of the approach are twofold: First, the exact representation of the MPC control law requires high-dimensional preactivations that significantly limit performance and applicability. Second, the scheme requires decryption of some data to evaluate the max-out neurons, which results in a security flaw.

We overcome the weaknesses in [18] with two central improvements. Instead of a computationally demanding exact MPC representations, we consider learning-based approximations that allow us to significantly reduce the size of the considered neural networks. Furthermore, we use garbled circuits [3] in combination with secret sharing [6] and show that the resulting architecture enables secure and efficient evaluations of max-out networks.

The remaining paper is organized as follows. Section 2 provides background on MPC, secret sharing, and garbled circuits. Our main result, i.e., a novel secure implementation of max-out neural networks, is presented in Section 3. Finally, Section 4 illustrates our method with a numerical example and Section 5 states conclusions and an outlook.

Notation. We denote the sets of real, integer, and natural numbers (including 0) by $\mathbb{R}$, $\mathbb{Z}_p$, and $\mathcal{N}$, respectively. We further denote positive integers by $\mathcal{N}_+$ and define $\mathcal{N}_q$ as $\{0, \ldots, q-1\}$ for some $q \in \mathcal{N}_+$. Moreover, we frequently use the modulo operation $z \bmod q := z - q\lfloor z/q \rfloor$ and we state congruence modulo q as $z_1 \equiv z_2 \bmod q$. In this context, $\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$, and $\lfloor \cdot \rceil$ refer to the floor function, the ceiling function, and rounding to the nearest integer, respectively. Next, for a vector $v \in \mathbb{R}^q$, $\max\{v\}$ stands for $\max\{v_1, \ldots, v_q\}$. Finally, let $n \in \mathcal{N}_+$ and consider the matrices $K \in \mathbb{R}^{q \times n}$ and $P \in \mathbb{R}^{q \times q}$. Then, $\|K\|_{\max} := \max_{i,j} |K_{ij}|$ and $\|v\|_P^2 := v^\top P v$.

2 Preliminaries and background

2.1 Model predictive control via max-out networks

MPC. Classical MPC builds on solving an optimal control problem of the form

$$V(x) := \min_{\substack{\hat{x}(0),\ldots,\hat{x}(N) \\ \hat{u}(0),\ldots,\hat{u}(N-1)}} \|\hat{x}(N)\|_P^2 + \sum_{\kappa=0}^{N-1} \|\hat{x}(\kappa)\|_Q^2 + \|\hat{u}(\kappa)\|_R^2 \qquad \text{(E.1)}$$

$$\text{s.t.} \quad \hat{x}(0) = x,$$

$$\hat{x}(\kappa+1) = A\,\hat{x}(\kappa) + B\hat{u}(\kappa), \qquad \forall \kappa \in \mathcal{N}_N,$$

$$(\hat{x}(\kappa), \hat{u}(\kappa)) \in \mathcal{X} \times \mathcal{U}, \qquad \forall \kappa \in \mathcal{N}_N,$$

$$\hat{x}(N) \in \mathcal{T}$$

in every time step $k \in \mathcal{N}$ for the current state $x = x(k)$. Here, $N \in \mathcal{N}_+$ refers to the prediction horizon and P, Q, and R are positive (semi-) definite weighting matrices. The dynamics of the linear prediction model are described by $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. State and input constraints can be incorporated via the polyhedral sets $\mathcal{X}$ and $\mathcal{U}$. Finally, the terminal set $\mathcal{T}$ allows to enforce closed-loop stability (see [15] for details). The resulting control law $g : \mathcal{F} \rightarrow \mathcal{U}$ is defined as

$$g(x) := \hat{u}^*(0), \tag{E.2}$$

where $\mathcal{F}$ denotes the feasible set of (E.1) and where $\hat{u}^*(0)$ refers to the first element of the optimal input sequence. We briefly note that one could replace (E.1) by robust tube-based MPC (as done in [18, 22]) in order to account for occurring approximation errors (see Prop. E.1) and to ensure robust constraint satisfaction. We here omit this step for ease of presentation.

Max-out networks. In this paper, we exploit learning-based approximations of $g(x)$ in order to realize secure cloud-based MPC. More specifically, we approximate $g(x)$ based on max-out neural networks [9]. For ease of presentation, we will focus on scalar inputs, i.e., $m = 1$, throughout the paper although our approach is not limited to this special case. Hence, we only consider scalar-valued $g(x)$ that we denote by $g(x)$. Moreover, we restrict ourselves to max-out networks of the form

$$\hat{g}(x) := \max\{Kx + b\} - \max\{Lx + c\} \tag{E.3}$$

with $K, L \in \mathbb{R}^{p \times n}$ and $b, c \in \mathbb{R}^p$, i.e., to a single hidden layer with two neurons that each pool $p \in \mathcal{N}_+$ affine preactivations.

The restriction to networks of the form (E.3) is due to three observations. First, according to [9, Thm. 4.3], (E.3) allows to approximate any continuous function $\tilde{g} : \mathbb{R}^n \rightarrow \mathbb{R}$ arbitrarily well and (E.2) is well-known to be continuous. Second, by exploiting the piecewise affine structure of linear MPC [4], (E.3) even allows to exactly describe (E.2) as specified in [19]. Third, as recently noted in [18], the form (E.3) is beneficial for encrypted implementations.

2.2 Secret sharing and secure two-party computation

As mentioned in the introduction, we will use secret sharing [6] in combination with garbled circuits [3] to overcome the security flaws in [18] and to derive a fully encrypted implementation of (E.3). In the following, we briefly summarize these cryptographic tools.

Secret Sharing. The central idea of secret sharing is quite simple. Assume we intend to perform cloud-based computations on a secret number $z \in \mathbb{Z}_p$. Then, we can simply divide z into two (or more) random shares and perform

the upcoming computations via two (or more) non-colluding clouds. Here, we focus on the special case of two clouds, i.e., two-party computation, that has also been considered in [18]. Slightly more formally, the essentials of secret sharing and two-party computation can be summarized as follows. Two shares of z are constructed by randomly choosing $z^{(1)}$ from $\mathcal{N}_q$ and subsequently selecting $z^{(2)} \in \mathcal{N}_q$ such that

$$z \equiv z^{(1)} + z^{(2)} \mod q, \tag{E.4}$$

where $q \in \mathcal{N}_+$ defines the size of the message space. It can easily be verified that neither $z^{(1)}$ nor $z^{(2)}$ leak information about z and that z can only be reconstructed by combining both shares (as detailed below).

Two-party computations. Remarkably, the shares allow to carry out computations. To see this, we introduce the shorthand notation $[z] := [z^{(1)}, z^{(2)}]$ for shared values. Now, consider two secret numbers $z_1, z_2 \in \mathbb{Z}_p$ and the corresponding shares $[z_1]$ and $[z_2]$. Then, the number z_3 associated with

$$[z_3] = [z_1] + [z_2] := \left[z_1^{(1)} + z_2^{(1)}, z_1^{(2)} + z_2^{(2)}\right] \mod q \tag{E.5}$$

satisfies $z_3 \equiv z_1 + z_2 \mod q$. In other words, the shares allow to carry out secure additions. Analogously, secure multiplications and additions with public constants $a, b \in \mathbb{Z}_p$ can be performed. In fact,

$$[z_4] = a[z_1] + b := \left[az_1^{(1)} + b, az_1^{(2)}\right] \mod q \tag{E.6}$$

is such that $z_4 \equiv az_1 + b \mod q$. Finally, even secure multiplications of two secret numbers can be realized. However, this requires the utilization of so-called Beaver triples [2]. A Beaver triple consists of shares $[\alpha]$, $[\beta]$, and $[\gamma]$ from $\mathcal{N}_q$ that satisfy $\alpha\beta \equiv \gamma \mod q$. Moreover, the shares are generated by randomly choosing $[\alpha]$ and $[\beta]$ without revealing both shares of α, β, or γ to any of the computing parties. Without giving details, we briefly note that various protocols for two-party generation of Beaver triples exist [8, 17]. Now, given a Beaver triple, secure multiplications of the form $z_1 z_2$ can be carried out as follows. First, the shares $[\delta] := [z_1] - [\alpha]$ and $[\epsilon] := [z_2] - [\beta]$ are computed, where subtractions are defined analogously to additions. Since $[\alpha]$ and $[\beta]$ have been randomly chosen, $[\delta]$ and $[\epsilon]$ contain no information on $[z_1]$ or $[z_2]$. Hence, the two parties can exchange their shares and reveal δ and ϵ. Afterwards, the shares

$$[z_5] := [\gamma] + \delta[\beta] + \epsilon[\alpha] + \delta\epsilon \tag{E.7}$$

can be computed in a distributed and secure fashion. One can easily verify that these shares satisfy the desired multiplicative relation $z_5 \equiv z_1 z_2 \mod q$.

Reconstruction. Obviously, being able to perform secure computations on shared values is useful for encrypted control. However, we eventually

Table E.1: Garbled AND-gate with labelled data.

labelled inputs		outputs	encrypted outputs
ℓ_0^v	ℓ_0^w	ℓ_0^y	$\texttt{Enc}_{\{\ell_0^v,\ell_0^w\}}(\ell_0^y)$
ℓ_0^v	ℓ_1^w	ℓ_0^y	$\texttt{Enc}_{\{\ell_0^v,\ell_1^w\}}(\ell_0^y)$
ℓ_1^v	ℓ_0^w	ℓ_0^y	$\texttt{Enc}_{\{\ell_1^v,\ell_0^w\}}(\ell_0^y)$
ℓ_1^v	ℓ_1^w	ℓ_1^y	$\texttt{Enc}_{\{\ell_1^v,\ell_1^w\}}(\ell_1^y)$

need to reconstruct the actual control input and not only a congruent value modulo q. To this end, we exploit that the function $\mu : \mathbb{Z}_p \to \mathbb{Z}_p$

$$\mu(z) := \begin{cases} z - q & \text{if } z \geq q/2, \\ z & \text{otherwise} \end{cases} \tag{E.8}$$

is a partial inverse of the modulo operation in the sense that the relation $z = \mu(z \bmod q)$ holds for every

$$z \in \mathbb{Z}_{p_q} := \{-\lfloor q/2 \rfloor, \ldots, \lceil q/2 \rceil - 1\}.$$

2.3 Secure circuit evaluations using garbled circuits

As specified in Section 3.3 further below, relations analogous to (E.5)–(E.7) allow to securely evaluate the arguments of the max-operations in (E.3). However, we do not have a procedure to securely evaluate the max-operations yet.

To overcome this issue, we consider so-called garbled circuits [3], which enable secure two-party evaluation of arbitrary Boolean circuits. In the following, we give a brief but intuitive introduction to garbling by considering a Boolean circuit consisting of one AND-gate only. This circuit takes the input bits v and w and returns the output bit $y = \text{AND}(v, w)$. Now, assume v belongs to the first cloud and w to the second one. Then, a garbled circuit allows to compute y without revealing any information on v or w to the other cloud (apart from what can be inferred from y). The corresponding procedure can be summarized as follows.

Garbling. The first cloud is the so-called *Garbler*. The Garbler randomly picks two labels (that are associated with 0 and 1, respectively) for every input and output in the circuit. Thus, for our simplistic example, the Garbler picks the labels $\ell_0^v, \ell_1^v, \ell_0^w, \ell_1^w, \ell_0^y$, and ℓ_1^y, where the indices are purely for presentation. The Garbler then creates a truth table for the AND-gate based on the generated labels (see Tab. E.1). Next, the Garbler encrypts the output column using the corresponding input labels as secret keys leading to, e.g., the ciphertext $\texttt{Enc}_{\{\ell_0^v,\ell_0^w\}}(\ell_0^y)$. The garbled circuit is finally generated by randomly

ordering the rows of the encrypted truth table (or by using sophisticated ordering techniques such as point-and-permute [14]).

Communication. The Garbler sends the garbled circuit to the second party who acts as the *Evaluator*. Clearly, in order to evaluate the circuit, the Evaluator also needs the garbled inputs in terms of the corresponding labels. Since the labels do not reveal any information on the actual inputs, the Garbler simply sends its labelled input along with the circuit.

For instance, in our example, the Garbler sends ℓ_0^x if its input is 0. Since only the Garbler holds the labels and since the Evaluator intends to keep its input secret, it is slightly more complicated to provide the label corresponding to the Evaluator's input. Fortunately, there exist efficient cryptographic protocols that allow to solve this issue. In fact, the Evaluator can infer ℓ_w^w without revealing w to the Garbler by using a so-called oblivious transfer (OT, [16]).

Evaluation. Based on the labels ℓ_v^v and ℓ_w^w and the garbled circuit, the Evaluator can decrypt the related output label (and none of the others).

For instance, under the assumption that $v = 0$ and $w = 1$, the Evaluator obtains ℓ_0^v and ℓ_1^w and is hence able to decrypt ℓ_0^y. Importantly, the Evaluator does not yet know y since the label is a random number.

Revealing. After evaluating the garbled circuit, the Evaluator holds the resulting output label ℓ_y^y but, so far, only the Garbler would be able to interpret it. Depending on the application, these pieces of information can be (re)combined in three different ways. In fact, the plaintext y can either be revealed to the Evaluator, the Garbler, or a third party. We later exploit the first case and therefore comment on this one only. In this scenario, the Garbler simply sends the output labels with the corresponding plaintexts to the Evaluator (which can be done together with the garbled circuit) and the Evaluator uses this information to obtain y.

3 Secure evaluation of max-out networks

Our main contribution is a secure and efficient two-party implementation of max-out networks as in (E.3) based on a tailored combination of secret sharing and garbled circuits. Before presenting our implementation, we specify the goals of the approach and outline the main concept.

3.1 Goals and concept

We initially assume that a suitable controller (E.3) has been identified by the system operator in terms of K, b, L, and c. Our goal then is to realize a cloud-based evaluation of $\hat{g}$ without revealing the system's state $x(k)$, the control actions $\hat{g}(x(k))$, or the controller parameters K, b, L, and c to any

of the two clouds. To this end, we will use both secret sharing and garbling and try to combine their strengths while avoiding their weaknesses. Namely, as apparent from Section 2.2, secret sharing is efficient in securely evaluating additions and multiplications while comparisons (as required for max-operations) are intractable. In contrast, garbled circuits can efficiently handle comparisons and additions while being very inefficient for multiplications. As a consequence, we aim for secret sharing-based evaluations of the affine operations in (E.3) while we will exploit garbled circuits for the max-operations.

3.2 Integer-based reformulation of max-out networks

In order to apply secret sharing, we need to reformulate (E.3) based on integers. To this end, we simply choose some positive scaling factors $s_1, s_2 \in \mathbb{R}_+$, define $s_3 := s_1 s_2$, and construct the integers $\boldsymbol{\xi} := \lfloor s_1 \boldsymbol{x} \rceil$, $\mathcal{K} := \lfloor s_2 \boldsymbol{K} \rceil$, $\boldsymbol{\beta} := \lfloor s_3 \boldsymbol{b} \rceil$, $\mathcal{L} := \lfloor s_2 \boldsymbol{L} \rceil$, and $\boldsymbol{\gamma} := \lfloor s_3 \boldsymbol{c} \rceil$. Then, the integer-based approximation

$$\max\{\mathcal{K}\boldsymbol{\xi} + \boldsymbol{\beta}\} - \max\{\mathcal{L}\boldsymbol{\xi} + \boldsymbol{\gamma}\} \approx s_3 \hat{g}(\boldsymbol{x}) \tag{E.9}$$

holds for sufficiently large scaling factors. More precisely, the following proposition provides an upper bound for possible approximation errors. As a preparation, we introduce the vectors

$$\boldsymbol{v} := \mathcal{K}\boldsymbol{\xi} + \boldsymbol{\beta} \qquad \text{and} \qquad \boldsymbol{w} := \mathcal{L}\boldsymbol{\xi} + \boldsymbol{\gamma} \tag{E.10}$$

that express the results of the affine preactivations in terms of integers.

Proposition E.1
Let $\hat{g}(\boldsymbol{x})$, the scaling factors s_i, and $\boldsymbol{v}$ and $\boldsymbol{w}$ be defined as above. Further, let $\eta \in \mathbb{R}_+$ and assume that

$$\|\boldsymbol{x}\|_\infty \leq \frac{\eta}{2s_1}, \quad \|\boldsymbol{K}\|_{\max} \leq \frac{\eta}{2s_2}, \quad \text{and} \quad \|\boldsymbol{L}\|_{\max} \leq \frac{\eta}{2s_2}. \tag{E.11}$$

Then, deviations between $\hat{g}$ and its integer-based reformulation are bounded above by

$$\left| \hat{g}(\boldsymbol{x}) - \frac{1}{s_3} (\max\{\boldsymbol{v}\} - \max\{\boldsymbol{w}\}) \right| \leq \frac{1}{s_3} \left(n\eta + \frac{n}{2} + 1 \right). \tag{E.12}$$

Proof. We first note that the derivation of the upper bound in (E.12) is similar to the one in [18, Prop. 2]. Hence, we can shorten this proof to its essentials. For instance, the corresponding proof in [18] shows that the max-operations in (E.12) are irrelevant for the desired error bound. As a consequence, the left-hand side of (E.12) is upper bounded by

$$\left\| \boldsymbol{K}\boldsymbol{x} + \boldsymbol{b} - \frac{1}{s_3}\boldsymbol{v} \right\|_\infty + \left\| \boldsymbol{L}\boldsymbol{x} + \boldsymbol{c} - \frac{1}{s_3}\boldsymbol{w} \right\|_\infty. \tag{E.13}$$

We will use the same strategy to evaluate upper bounds for the two terms in (E.13).

Therefore, we only consider the first term in the following and multiply the resulting bound by 2 to obtain (E.12). To this end, we define the deviations

$$\Delta x := x - \frac{1}{s_1}\xi, \ \ \Delta K := K - \frac{1}{s_2}\mathcal{K}, \ \text{ and } \ \Delta b := b - \frac{1}{s_3}\beta,$$

and note that these are bounded by

$$\|\Delta x\|_\infty \leq \frac{1}{2s_1}, \ \ \|\Delta K\|_{\max} \leq \frac{1}{2s_2}, \ \text{ and } \ \|\Delta b\|_\infty \leq \frac{1}{2s_3} \tag{E.14}$$

due to the nearest integer rounding. Based on these deviations, we can rewrite the first expression in (E.13) as

$$\left\| Kx + b - \frac{1}{s_3}v \right\|_\infty = \|\Delta Kx + K\Delta x - \Delta K\Delta x + \Delta b\|_\infty.$$

Using subadditivity, we can overestimate the right-hand side by upper bounding the individual terms according to

$$\|\Delta Kx\|_\infty \leq \|\Delta K\|_\infty \|x\|_\infty \leq n\|\Delta K\|_{\max}\|x\|_\infty \leq \frac{n\eta}{4s_1 s_2},$$

where the two latter relations hold due to (E.11) and (E.14). Proceeding analogously with the three remaining terms, summing up the individual bounds, and multiplying the result by 2 finally leads to the bound in (E.12). □

Remarkably, the error bound (E.12) can be made arbitrarily small by increasing s_1 and s_2 and hence s_3.

3.3 Secret sharing for affine preactivations

Next, we exploit secret sharing and two-party computation to securely evaluate the integer-based preactivations (E.10). To this end, we assume that each of the two clouds holds one share of the matrices $\mathcal{K}$ and $\mathcal{L}$ as well as the vectors β and γ that have been uploaded before runtime of the controller. As above, we summarize these shares with $[\mathcal{K}]$, $[\mathcal{L}]$, $[\beta]$, and $[\gamma]$. Additionally, we assume that the sensor computes ξ and related shares $[\xi]$ in every time-step based on the current state x. With these quantities at hand, we intend to securely compute shares for v and w as in (E.10). In the following, our focus is on the computation of suitable shares $[v]$ since $[w]$ is computed analogously.

Clearly, computing v requires only scalar-valued multiplications and additions. Therefore, a secure computation based on shares can be carried out

according to the procedures in Section 2.2. In the interest of a compact notation, we extend some of these procedures to matrices next. For instance, we assume that each cloud holds a share of a matrix-valued Beaver triplet $[\mathcal{A}]$, $[\mathcal{B}]$, and $[\mathcal{C}]$, where $\mathcal{A}$ and $\mathcal{B}$ have been randomly picked from $\mathcal{N}_q^{p \times n}$ and where $\mathcal{C}$ is such that

$$\mathcal{A} \circ \mathcal{B} \equiv \mathcal{C} \mod q$$

with "$\circ$" denoting the Hadamard (or elementwise) product. Then, the shares $[v]$ can be computed as follows.

Proposition E.2
Let the shares $[\mathcal{K}]$, $[\beta]$, and $[\xi]$ and the Beaver triplet $[\mathcal{A}]$, $[\mathcal{B}]$, and $[\mathcal{C}]$ be defined as above and let

$$[\Xi] := \begin{pmatrix} [\xi^\top] \\ \vdots \\ [\xi^\top] \end{pmatrix}, \quad [\mathcal{D}] := [\mathcal{K}] - [\mathcal{A}], \quad \text{and} \quad [\mathcal{E}] := [\Xi] - [\mathcal{B}].$$

Then, shares for v as in (E.10) can be computed via

$$[v] := ([\mathcal{C}] + \mathcal{D} \circ [\mathcal{B}] + \mathcal{E} \circ [\mathcal{A}] + \mathcal{D} \circ \mathcal{E}) \mathbf{1} + [\beta]. \tag{E.15}$$

Proof. We first note that the Hadamard product allows to separate multiplications and additions in the computation of v. In fact, we obviously have $v \equiv (\mathcal{K} \circ \Xi) \mathbf{1} + \beta \mod q$, where Ξ refers to the recombination of $[\Xi]$. Now, a shared computation of $\mathcal{K} \circ \Xi$ can be carried out analogously to (E.7) using Beaver's procedure. Clearly, this leads to the expression in the round brackets in (E.15). The remaining operations in (E.15) exploit the procedures in (E.5) and (E.6). $\qquad\square$

3.4 Boolean circuit for max-out

After evaluating the affine preactivations using secret sharing, the first cloud holds $v^{(1)}$ and $w^{(1)}$ and the second cloud holds $v^{(2)}$ and $w^{(2)}$. All shares are, by construction, contained in $\mathcal{N}_q$ and, hence, non-negative. However, the vectors v and w may also contain negative numbers. Thus, before evaluating the max-operations, we need to reconstruct v and w. Because $[v]$ (and $[w]$) represent proper shares of v (and w) according to Proposition E.2, we can easily derive the following statement.

Corollary E.1
Assume $\mathcal{K}\xi + \beta \in \mathbb{Z}_{p_q^p}^p$. Then,

$$v_i = \mu \left(v_i^{(1)} + v_i^{(2)} \mod q \right) \tag{E.16}$$
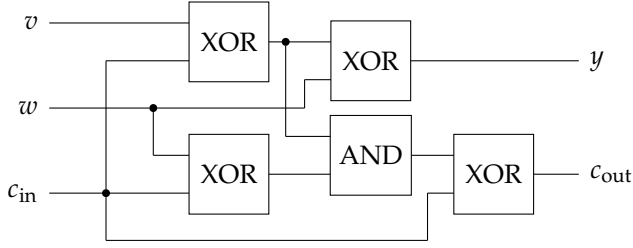
for every $i \in \{1, \ldots, p\}$.

Fig. E.1: A Boolean circuit realizing a full adder (FA).

Clearly, the reconstructed vectors v and w should not be revealed to any of the two clouds. Therefore, the reconstruction (E.16) has to be included in the Boolean circuit, which is later garbled. More precisely, we will design two garbled circuits for $\nu := \max\{v\} + r_1 \mod q$ and $\omega := \max\{w\} + r_2 \mod q$, where the role of the random numbers $r_1, r_2 \in \mathcal{N}_q$ will be clarified further below in Section 3.6. Since the functionality of both circuits is equivalent, we only discuss the design of the ν-circuit. Obviously, a Boolean circuit requires Boolean input data and we thus need bit-wise representations of $v^{(1)}$, $v^{(2)}$, and r_1 here. In order to provide this data, the following assumption is made.

Assumption E.1
The modulus q is of the form $q = 2^l$ for some $l \in \mathcal{N}_+$.

Bit-wise representation. Due to this assumption, the vectors $v^{(1)}, v^{(2)} \in \mathcal{N}_q^p$ and the scalar $r_1 \in \mathcal{N}_q$ can be represented in terms of l-bit unsigned numbers. In order to formalize this observation, we note that there exist unique matrices $V^{(1)}, V^{(2)} \in \{0,1\}^{p \times l}$ such that the relations

$$v_i^{(1)} = \sum_{j=1}^{l} 2^{l-j} V_{ij}^{(1)} \quad \text{and} \quad v_i^{(2)} = \sum_{j=1}^{l} 2^{l-j} V_{ij}^{(2)}$$

hold for every $i \in \{1, \ldots, p\}$. Since $V^{(1)}$ and $V^{(2)}$ are Boolean and since they uniquely represent $v^{(1)}$ and $v^{(2)}$, we use them as inputs for our Boolean circuit. Analogous observations hold for r_1 but are not required for the following discussion. Now, the first task of the circuit is to compute $v_i^{(1)} + v_i^{(2)}$ based on the corresponding entries in $V^{(1)}$ and $V^{(2)}$ and, subsequently, to evaluate (E.16).

Bit-wise addition and reconstruction. In principle, the addition can easily be carried out using standard ripple-carry adders based on l full adders (FA) as illustrated in Figures E.1 and E.2. However, one has to take into account that properly representing $v_i^{(1)} + v_i^{(2)}$ might require an additional bit. In fact, this case arises if the last carry bit ($c_{i,l+1}$ in Fig. E.2) evaluates to 1. At this point, one might be tempted to extend the bit-wise representation of the
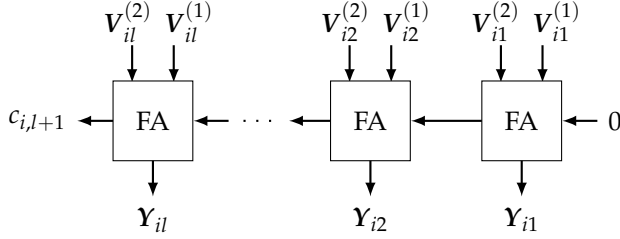
Fig. E.2: A ripple-carry adder for adding l-bit numbers.

computed sum by this carry bit. Yet, as apparent from (E.16), the addition is followed by a modulo operation with $q = 2^l$. Clearly, whenever we enter this operation with an unsigned number with at least l bits, it will simply return the l least significant bits and cut off all remaining bits.

Hence, a possible extension by the mentioned carry bit would immediately be undone by the modulo operation and is thus meaningless. More precisely, let the i-th row of the matrix $Y \in \{0,1\}^{p \times l}$ reflect the l least significant bits of the sum $v_i^{(1)} + v_i^{(2)}$. Then,

$$\sum_{j=1}^{l} 2^{l-j} Y_{ij} = v_i^{(1)} + v_i^{(2)} \mod q.$$

Now, in order to reconstruct v_i according to (E.16), it remains to apply μ as in (E.8). Interestingly, for $q = 2^l$ and $z \in \mathcal{N}_q$, the bit-wise interpretation of (E.8) simply reflects the conversion from unsigned to signed numbers. Therefore, we obtain

$$-2^{l-1} Y_{i1} + \sum_{j=2}^{l} 2^{l-j} Y_{ij} = \mu \left(\sum_{j=1}^{l} 2^{l-j} Y_{ij} \right)$$

using the two's complement convention. In other words, evaluating (E.8) within the circuit does not require any computations but just a reinterpretation of the bit-wise number format. We study two trivial numerical examples in Table E.2 to illustrate the computations up to this point.

Max-operations. Having reconstructed all v_i, it remains to compute $\max\{v\}$. This can be done by successively evaluating max-of-two-operations in a tournament-like fashion with $\lceil \log_2(p) \rceil$ rounds (i.e., quarterfinals, semifinals, etc.). We illustrate the bit-wise evaluation of these operations by exemplarily investigating $\max\{v_1, v_2\}$. Obviously,

$$\max\{v_1, v_2\} = \begin{cases} v_1 & \text{if } v_1 \geq v_2, \\ v_2 & \text{otherwise.} \end{cases} \tag{E.17}$$

Now, the two cases in (E.17) can also be identified based on the sign-bit σ of $v_1 - v_2$. In fact, the first case refers to $\sigma = 0$ and the second to $\sigma = 1$. Thus,

Table E.2: Bit-wise evaluation of addition, modulo q, and μ for two numerical examples with $l = 3$ and $c_{i,l+1} \in \{0,1\}$.

var./op.	val.	bit-wise	var./op.	val.	bit-wise
$v_1^{(1)}$	3	0 1 1	$v_2^{(1)}$	5	1 0 1
$+v_1^{(2)}$	6	1 1 0	$+v_2^{(2)}$	2	0 1 0
$=$	9	(1) 0 0 1	$=$	7	(0) 1 1 1
mod q	1	0 0 1	mod q	7	1 1 1
$\mu(\cdot)$	1	0 0 1	$\mu(\cdot)$	-1	1 1 1

the maximum can be expressed as $(1 - \sigma)v_1 + \sigma v_2$. Fortunately, this sum can be computed without relying on full-adders. In fact, the j-th bit of this sum simply results from

$$\text{XOR}\left(\text{AND}\left(Y_{1j}, \text{NOT}(\sigma)\right), \text{AND}\left(Y_{2j}, \sigma\right)\right). \tag{E.18}$$

Repeating the previous steps for the remaining entries of v and the remaining rounds of the "tournament" completes the circuit-based evaluation of $\max\{v\}$.

Randomization. It only remains to add the random number r_1 modulo q. To this end, we note that

$$\nu = (\max\{v\} \bmod q) + r_1 \bmod q.$$

As above, the bit-wise evaluation of $\max\{v\} \bmod q$ simply reflects a conversion from signed to unsigned numbers. Adding r_1 can finally be realized analogously to Figure E.2, where we can (again) ignore the $(l + 1)$-th carry bit since the modulo q operation follows.

3.5 Garbling and reconstruction of control actions

Garbled circuits. In order to securely evaluate the designed circuit, we use garbling as briefly introduced in Section 2.3. We note, in this context, that the presented circuit only contains AND-, XOR-, and NOT-gates. Now, without giving details, securely evaluating XOR- and NOT-gates does not require any garbling [13]. Hence, garbling as in Table E.1 is only required for one AND-gate per FA and two AND-gates per operation (E.18). Zooming out, we require l FAs per addition (and per subtraction). Further, each max-of-two-operation requires one subtraction and l operations of the form (E.18). Finally, by assuming that p is likewise a power of 2, carrying out the "tournament" requires $p - 1$ max-of-two-operations. Thus, the whole circuit contains

$$pl + 3(p - 1)l + l = (4p - 2)l \tag{E.19}$$

AND-gates, where the three summands on the left-hand side reflect the computations of (E.16), the evaluation of $\max\{v\}$, and the addition of r_1, respectively. Remarkably, the size of the garbled circuit solely depends on the dimension p of the max-out network and the chosen bit-length l.

The clouds use the garbled circuits to securely compute $v := \max\{v\} + r_1 \bmod q$ and $\omega := \max\{w\} + r_2 \bmod q$, respectively. More precisely, the first cloud acts as a Garbler for the v-circuit, which is subsequently evaluated by the second cloud. The roles are inverted for the ω-circuit. As a result, the overall computations are symmetrically allocated between both clouds.

Reconstructing control actions. After evaluating the garbled circuits, the first cloud holds ω and the second one v. We show next how to use this data to reconstruct (an approximation of) $\hat{g}(x)$ at the actuator. To this end, the first cloud sends $\Delta\omega := \omega + r_1 \bmod q$ and the second cloud transmits $\Delta v := v + r_2 \bmod q$. The actuator then computes and applies

$$u(k) = \frac{1}{s_3}\mu\left(\Delta v - \Delta\omega \bmod q\right), \tag{E.20}$$

where we note that both random values r_i cancel out. Formally, we require $\max\{v\} - \max\{w\} \in \mathbb{Z}_{p_q}$ for a correct evaluation of this final step. However, this condition is typically significantly less restrictive than the one in Corollary E.1 since the resulting control inputs are restricted to $\mathcal{U}$ anyway (apart from approximation errors).

3.6 Overall architecture and security guarantees

After having introduced all building blocks of our scheme, we briefly comment on their interplay and resulting security guarantees. In this context, the secure evaluation of the controller can be subdivided in an offline and online phase.

Offline. In the offline phase, the system operator first selects p and then obtains $K, L, b,$ and c by training the max-out network (E.3). Afterwards, $\mathcal{K}, \mathcal{L}, \beta,$ and γ are formed by choosing suitable scaling factors s_1 and s_2. Finally, $q = 2^l$ is specified and the shares $\mathcal{K}^{(i)}, \mathcal{L}^{(i)}, \beta^{(i)},$ and $\gamma^{(i)}$ are generated and transmitted to the i-th cloud.

Online. The online procedure is illustrated in Figure E.3. In every time step, the sensor measures $x(k)$ and computes ξ. It then generates the shares $\xi^{(i)}$ and sends them to cloud i. Now, the two clouds first exploit (E.15) (and the analogue for $[w]$) to securely compute shares of v and w. To this end, they generate two sets of matrix-valued Beaver triples $[\mathcal{A}], [\mathcal{B}],$ and $[\mathcal{C}]$ using, e.g., the procedure in [8]. Next, each cloud generates a random number r_i, garbles the circuit for one max-out neuron and labels its own inputs to the circuit (i.e., $v^{(1)}$ and r_1 for the first cloud). After that, it transmits the
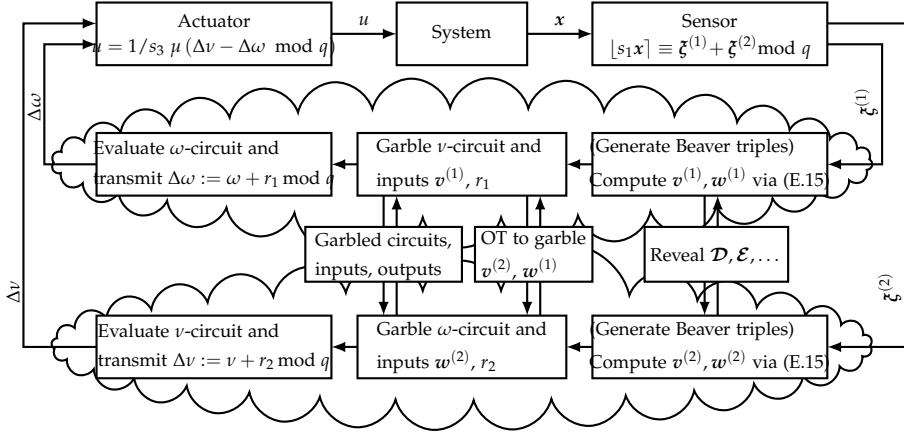
Fig. E.3: Overall architecture of the proposed scheme based on secret sharing and garbled circuits.

circuit together with the output table to the other cloud. Simultaneously, the remaining input labels (i.e., for $v^{(2)}$ in the v-circuit) are obtained through OT. Afterwards, each cloud evaluates the received garbled circuit, derives its output, and transmits the result masked by the random r_i to the actuator. Finally, the actuator reconstructs $u(k)$ according to (E.20) and applies it.

Security. To specify security guarantees, we make the (realistic) assumption that the two clouds are non-colluding and honest but curious. In other words, the two clouds will strictly follow the specified protocols without interchanging further data but they may try to infer information from the given data. Clearly, our security goal is to prevent the latter. In this context, we first note that the evaluation of the affine preactivations is perfectly secure (i.e, information theoretically secure) by the use of secret sharing [6]. Next, garbled circuits are proven to be secure against semi-honest adversary [10] which fits our assumption about the clouds. Finally, we mask the computed outputs of the max-out neurons by the random numbers r_i, which is equivalent to secret sharing and, therefore, enjoys perfect security. In summary, the clouds cannot obtain any information about ξ, $\mathcal{K}$, $\mathcal{L}$, β, γ, v, w, $\max\{v\}$, $\max\{w\}$, or the resulting u. We note, however, that generating new Beaver triples and garbled circuits for each evaluation of $u(k)$ is essential for security.

4   Numerical benchmark

We illustrate our approach for the standard double integrator example with the system matrices

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0.5 \\ 1 \end{pmatrix}$$

and the constraints $\mathcal{X} = \{x \in \mathbb{R}^2 \,|\, |x_1| \leq 25, |x_2| \leq 5\}$ and $\mathcal{U} = \{u \in \mathbb{R} \,|\, |u| \leq 1\}$. The predictive controller associated with (E.1) is specified as follows. We set $N = 15$, $Q = I$, and $R = 0.01$ and choose P as the solution of the discrete-time algebraic matrix Riccati equation

$$A^\top (P - PB(R + B^\top PB)^{-1}B^\top P) A - P + Q = 0.$$

Finally, $\mathcal{T}$ is chosen as the largest subset of $\mathcal{X}$ for which the linear quadratic regulator can be applied without violating the state or input constraints.

Next, in order to train the neural network, we sampled $M = 6000$ states $x^{(i)}$ in the feasible set $\mathcal{F}$, evaluated $g(x^{(i)})$, and solved the nonlinear program

$$\min_{K,b,L,c} \sum_{i=1}^{M} \left\| \max\{Kx^{(i)} + b\} - \max\{Lx^{(i)} + c\} - g(x^{(i)}) \right\|_2^2$$

locally, where we fixed the size p to either 8 or 16.

The resulting mean squared errors (MSE) with respect to (E.2) are listed in Table E.3.

For $p = 8$, the local optimizers (rounded to two digits after the decimal point) are given by

$$K^\top = \begin{pmatrix} -0.07 & 0.31 & 0.01 & -0.01 & 0.31 & 0.01 & 0.07 & -0.31 \\ -0.52 & -0.32 & -0.30 & -0.40 & 0.68 & 0.52 & -0.41 & -0.64 \end{pmatrix},$$

$$L^\top = \begin{pmatrix} 0.31 & 0.12 & -0.07 & -0.31 & -0.31 & -0.08 & 0.01 & 0.08 \\ 0.68 & -0.03 & -0.52 & 0.36 & -0.64 & 0.48 & 0.52 & 0.01 \end{pmatrix},$$

$$b^\top = \begin{pmatrix} 0.37 & 4.60 & 0.50 & 0.54 & 0.60 & 1.67 & -1.14 & 0.39 \end{pmatrix},$$

$$c^\top = \begin{pmatrix} 0.40 & -0.88 & -1.37 & -4.61 & -0.61 & -1.39 & -0.67 & 0.40 \end{pmatrix}.$$

An illustration of the corresponding function $\hat{g}(x)$ is depicted in Figure E.4.

Now, in order to prepare the secure two-party implementation of the learned controller, we require a suitable integer-based representation of $\hat{g}(x)$. In this context, we mainly need to choose the scaling factors s_1 and s_2 as well as the size of the message space q. A suitable choice should preclude overflow. Hence, the assumption in Corollary E.1 needs to be satisfied for all $x \in \mathcal{F}$. For practical applications, it is useful to start with the choice of $q = 2^l$ since l determines the number format in terms of bits. Here, we consider $l \in \{16, 32\}$. Interestingly, fixing l leads to an upper bound for s_3, i.e., for the product $s_1 s_2$. To see this, we initially note that the proof of Proposition E.1 implies

$$\left\| Kx + b - \frac{1}{s_3}(\mathcal{K}\xi + \beta) \right\|_\infty \leq \frac{1}{2s_3}\left(n\eta + \frac{n}{2} + 1 \right) := \Delta.$$

Based on this bound, one can show that the implication

$$\|Kx + b\|_\infty < \frac{q}{2s_3} - \Delta \quad \implies \quad \mathcal{K}\xi + \beta \in \mathbb{Z}_{pq}^p$$
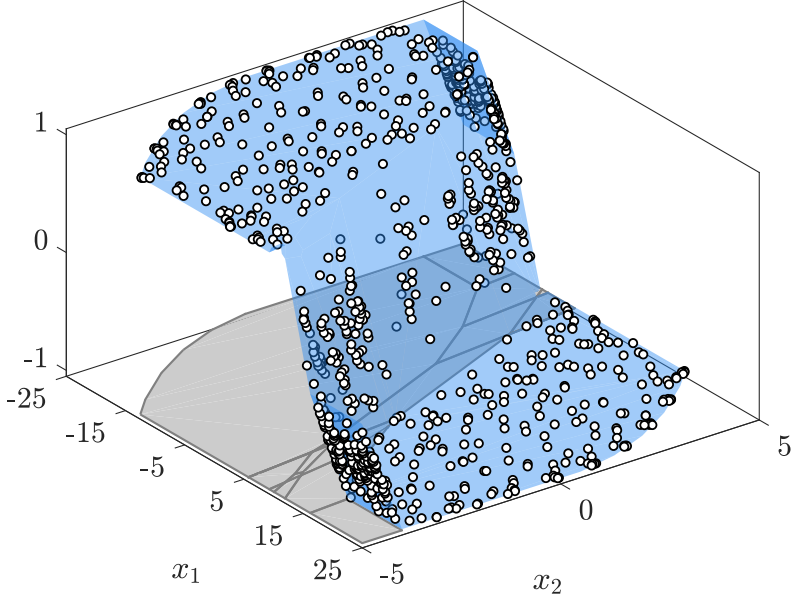
Fig. E.4: Illustration of the trained max-out neural network (blue) and a few sample points $\big(\boldsymbol{x}^{(i)}, g(\boldsymbol{x}^{(i)})\big)$ (white) over the partition (gray) induced by (E.3).

holds. Hence, the assumption in Corollary E.1 is satisfied if

$$s_3 < \frac{2^{l-1}}{\max_{\boldsymbol{x} \in \mathcal{F}} \|\boldsymbol{K}\boldsymbol{x} + \boldsymbol{b}\|_\infty + \Delta}, \tag{E.21}$$

where we used $q = 2^l$. Now, we can easily evaluate $\max_{\boldsymbol{x} \in \mathcal{F}} \|\boldsymbol{K}\boldsymbol{x} + \boldsymbol{b}\|_\infty$ (and $\max_{\boldsymbol{x} \in \mathcal{F}} \|\boldsymbol{L}\boldsymbol{x} + \boldsymbol{c}\|_\infty$) based on linear programming. However, precisely computing Δ is difficult, at this point, since it depends on s_3 and η. Fortunately, we will usually be able to realize a small error bound Δ. Therefore, we can simply choose a reasonable overestimation for Δ and justify (or discard) this choice afterwards.

Here, we assume $\Delta \leq 1$. The resulting upper bounds $s_{3,\max}$ in dependence of p and l are listed in Table E.3. Subsequently, we can freely choose s_1 and s_2 as long as $s_3 = s_1 s_2$ satisfies the corresponding bound. For instance, we choose $s_1 = 20$ and $s_2 = 100$ for $p = 8$ and $l = 16$, where the "asymmetric" choice reflects the fact that s_2 applies to a matrix. Having fixed the scaling factors, we choose the smallest η such that (E.11) holds (for every $\boldsymbol{x} \in \mathcal{F}$). Finally, we can easily verify $\Delta \ll 1$ for all considered combinations of p and l. Further, we can quantify the actual quantization errors, i.e., the left-hand side in (E.12), by computing the MSE based on the sampled states. The results are given in Table E.3.

Now, regarding the secure controller evaluation according to Figure E.3, we note that the quantization, the secret sharing, and the garbled circuits are

Table E.3: Error estimations and key data for the example.

p	$\text{MSE}^{\dagger}$	l	$s_{3,\max}$	$\text{MSE}^{\ddagger}$	#AND	t_{avg}
8	$18.57 \cdot 10^{-6}$	16	$2.23 \cdot 10^{3}$	$4.37 \cdot 10^{-5}$	480	79 ms
		32	$1.51 \cdot 10^{8}$	$5.06 \cdot 10^{-9}$	960	167 ms
16	$1.99 \cdot 10^{-6}$	16	$2.21 \cdot 10^{3}$	$6.45 \cdot 10^{-5}$	992	170 ms
		32	$1.49 \cdot 10^{8}$	$2.46 \cdot 10^{-6}$	1984	348 ms

[†] mean squared error of the max-out approximation w.r.t. (E.2)

[‡] mean squared error of the quantization w.r.t. (E.3)

determined by the scaling factors s_i, the size of the message space q, and the quantities p and l. In particular, the latter quantities determine the size of the garbled circuit according to (E.19) and the corresponding numbers of AND-gates are listed in Table E.3. For the actual garbling, we use 128-bit numbers for each label in the circuit. Further, output labels for each gate are encrypted by the hash of the sum of the associated inputs using SHA-256.

Finally, 1-out-of-2 OT is implemented based on ElGamal encryption. With this setup, a single evaluation of the encrypted controller on an Intel Core i5 with 2.50GHz leads, on average, to the computation times in Table E.3. We note, in this context, that the generation of random numbers (including the Beaver triples) and latency has not been taken into account yet.

5   Conclusions and Outlook

We presented a novel secure implementation of linear MPC over a two-cloud architecture. The key insight that led to our work is that max-out neural networks are well-suited to approximate piecewise affine control laws while providing a structure that allows for an efficient secure implementation. More precisely, the affine preactivations can be evaluated by additive secret sharing and the subsequent evaluation of the max-operations is realized with garbled circuits. Due to this tailored setup, we avoid inefficient multiplications within the garbled circuits. Improvements compared to existing schemes lie in the significantly reduced evaluation times, and the more efficient use of the involved clouds. More precisely, [18] does not approximate (E.3), which results in large values for p, and [22] evaluates the control law partially at the sensor.

Future research may address two aspects. First, max-out networks of the form (E.3) can, in principle, approximate arbitrary (continuous) functions and they consequently may support secure implementations of (some) nonlinear MPC schemes.

Second, our numerical results are obtained from a laptop running unop-

timized python code. Therefore, a more realistic setup with optimized code, a two-cloud network including latency, and a more complex control law will be of interest.

References

[1] A. B. Alexandru, M. Morari, and G. J. Pappas. Cloud-based MPC with encrypted data. In *Proc. of the 57th Conference on Decision and Control*, pages 5014–5019, 2018.

[2] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In *Annual International Cryptology Conference*, pages 420–432. Springer, 1991.

[3] M. Bellare, V.T. Hoang, and P. Rogaway. Foundations of Garbled Circuits. In *Proc. of the Conference on Computer and Communications Security*, pages 784–796, 2012.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[5] J. H. Cheon, K. Han, H. Kim, J. Kim, and H. Shim. Need for controllers having integer coefficients in homomorphically encrypted dynamic system. In *Proc. of 57th Conference on Decision and Control*, pages 5020–5025. IEEE, 2018.

[6] R. Cramer, I. B. Damgård, and J. B. Nielsen. *Secure multiparty computation and secret sharing*. Cambridge University Press, 2015.

[7] F. Farokhi, I. Shames, and N. Batterham. Secure and private control using semi-homomorphic encryption. *Control Engineering Practice*, 67:13–20, 2017.

[8] T. K. Frederiksen, M. Keller, E. Orsini, and P. Scholl. A Unified Approach to MPC with Preprocessing using OT. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 711–735. Springer, 2015.

[9] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout Networks. In *Proc. of the 30th International Conference on Machine Learning*, volume 28, pages 1319–1327, 2013.

[10] C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[11] W. Knowles, D. Prince, D. Hutchiso, J. F. P. Disso, and K. Jones. A survey of cyber security management in industrial control systems. *Intl. Journal of Critical Infrastructure Protection*, 9:52–80, 2015.

[12] K. Kogiso and T. Fujita. Cyber-security enhancement of networked control systems using homomorphic encryption. In *Proc. of the 54th Conference on Decision and Control*, pages 6836–6843, 2015.

References

[13] V. Kolesnikov and T. Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. In *International Colloquium on Automata, Languages, and Programming*, pages 486–498. Springer, 2008.

[14] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay-Secure Two-Party Computation System. In *USENIX Security Symposium*, volume 4, page 9. San Diego, CA, USA, 2004.

[15] D. Q. Mayne, J. B. Rawlings, C.V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[16] M. O. Rabin. How To Exchange Secrets with Oblivious Transfer, 2005. Harvard University Technical Report 81.

[17] D. Rathee, T. Schneider, and K. K. Shukla. Improved Multiplication Triple Generation over Rings via RLWE-based AHE. In *Cryptology and Network Security*, pages 347–359, Cham, 2019. Springer International Publishing.

[18] N. Schlüter and M. Schulze Darup. Encrypted explicit MPC based on two-party computation and convex controller decomposition. In *Proc. of the 59th Conference on Decision and Control*, pages 5469–5476, 2020.

[19] N. Schlüter and M. Schulze Darup. Novel convex decomposition of piecewise affine functions. In *Proc. of the 21st IFAC World Congress*, 2020.

[20] M. Schulze Darup, A. B. Alexandru, D. E. Quevedo, and G. J. Pappas. Encrypted control for networked systems–an illustrative introduction and current challenges. *arXiv preprint arXiv:2010.00268*, 2020.

[21] M. Schulze Darup, A. Redder, and D. E. Quevedo. Encrypted cooperative control based on structured feedback. *IEEE Control Systems Letters*, 3(1):37–42, 2019.

[22] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo. Towards encrypted MPC for linear constrained systems. *IEEE Control Systems Letters*, 2(2):195–200, 2018.

[23] Katrine Tjell, Nils Schlüter, Philipp Binfet, and Moritz Schulze Darup. Secure learning-based mpc via garbled circuit. In *2021 IEEE 60th Conference on Decision and Control (CDC)*, August 2021.

References

Paper F

Privacy in Distributed Computations based on Real Number Secret Sharing

Katrine Tjell and Rafael Wisniewski

The layout has been revised.

Abstract

Privacy preservation in distributed computations is an important subject as digitization and new technologies enable collection and storage of vast amounts of data, including private data belonging to individuals. To this end, there is a need for a privacy preserving computation framework that minimises the leak of private information during computations while being efficient enough for practical usage. This paper presents a step towards such a framework with the proposal of a real number secret sharing scheme that works directly on real numbers without the need for conversion to integers which is the case in related schemes. The scheme offers computations like addition, multiplication, and division to be performed directly on secret shared data (the cipher text version of the data). Simulations show that the scheme is much more efficient in terms of accuracy than its counterpart version based on integers and finite field arithmetic. The drawback with the proposed scheme is that it is not perfectly secure. However, we provide a privacy analysis of the scheme, where we show that the leaked information can be upper bounded and asymptotically goes to zero. To demonstrate the scheme, we use it to perform Kalman filtering directly on secret shared data.

1   Introduction

In recent years, there has been a rapid development of technologies for digitization and collection and storage of data. Consequently, various distributed algorithms for the efficient processing of the collected data are being developed in many research communities like signal processing, control, machine learning, and optimization. Simultaneously, concerns about privacy and the possible misuse of the data means a sudden big interest in embedding cryptographic methods into the distributed algorithms to achieve privacy preserving data processing, [1–3].

So far, efficient data processing and privacy preservation are two terms that seems difficult to combine since the cryptographic methods tend to bring a substantial overhead in either communication, computation or both. Moreover, security of cryptographic methods such as secret sharing and homomorphic encryption relies on modular arithmetic, which entails that all data to be protected must be integers and computations on this data must be translated into equivalent computations using finite field arithmetic, [4–6]. The drawbacks of this are, for instance, loss of precision in the solution (because of rounding decimal numbers to integers) and that many operations such as division becomes very intractable.

For some applications, efficient processing, that is not constrained to finite field arithmetic, is crucial. Thus, it becomes relevant to consider a trade-off between privacy and efficiency since after all; limited privacy is better than

none. To this end, we explore distributed computations in the secure multiparty computation [7] setup, where only cipher texts travel between participants and plain texts stay hidden throughout computations. Essentially, what we propose is a *real number secret sharing scheme* that circumvents the disadvantages of using only integers and modular arithmetic and consequently achieves improved performance compared to state-of-the-art methods. The scheme works directly on real numbers and we show straight forward implementations of addition, multiplication and division performed directly on the secret shared data. The shortcoming to our proposed scheme is that it does not guarantee perfect security like its counterpart version based on integers and modular arithmetic. However, we carefully control the amount of leaked information and provide information theoretic results to support our claims.

As a motivating example, we demonstrate the use of the proposed scheme to perform privacy preserving Kalman filtering. That is, we consider a linear dynamical system with state-transition matrix A, control input matrix B, control input u_k, process noise w_k and state vector x_k:

$$x_k = Ax_{k-1} + Bu_k + w_k. \tag{F.1}$$

Observations (or measurements) of the state vector, z_k are modeled as

$$z_k = Hx_k + v_k, \tag{F.2}$$

where H is the observation matrix and v_k is the measurement noise. The objective is to estimate the true state of the system from the noisy observations, which is optimally done using the Kalman filter. The privacy concern emerges from the measurements which could be private data that potentially leaks private information. Scenarios where a problem of this form appears, could for instance be traffic monitoring [8], medical monitoring [9], and consumption forecasting [10]. The problem of privacy preserving Kalman filtering has already been studied for instance in [11] that uses a form of data compression to preserve privacy of measurements, [12] that base the privacy on a combination of homomorphic encryption and secure multiparty computation techniques, and [13] that relies on differential privacy. These existing works all suffer from a degradation in output utility compared to the none-privacy preserving solution due to noise insertion or to the previously mentioned rounding of reals to integers. We will show that a privacy aware Kalman filter based on our real number secret sharing scheme achieves significantly improved output utility. Furthermore, we compare our privacy preserving Kalman filter to the one proposed in [12] and show that ours has a reduction in computation and communication overhead.

1.1   State of the art

The typical way of preserving privacy of real numbers is to simply discard the decimals and keep the integer part which is the suitable representation for most cryptographic methods, [14–16] . The induced error bounds caused by the truncation, can be made small by introducing scaling constants prior to truncation. However, the size of the modular field, in which the cryptographic calculations take place, increases according to the size of the scaling factors and thus cannot be made arbitrarily big.

One of the first more direct ways to deal with non-integers in cryptographic computations, was made in 2010 by Catrina et al. in [17]. Their proposed solution builds on a fixed-point representation of real numbers that allows the use of Shamir's secret sharing scheme as the underlying cryptographic technique. In [18] this solution was applied to privacy preserving linear programming. Along this line of research, [19] proposed in 2013 a similar secure floating-point computation scheme also based on a linear secret sharing framework. In 2016, [20] proposed other techniques for representing secure real numbers suitable for a secret sharing framework with their so-called golden-section and logarithmic number formats.

Apart from secret sharing based secure computation frameworks, there has also been several attempts to secure real number computations in homomorphic encryption based frameworks, [21–23]. Analog to the approach based on secret sharing, the main idea here is to convert the real number into a multi-bit binary integer to achieve a fixed precision presentation of a real number. The drawback with these approaches is the time consuming computational overhead with homomorphic encryption and also that the proposed schemes only offer addition and in some cases multiplication of cipher-texts. This is in contrast to our scheme that allows addition, multiplication and division to be performed efficiently on the cipher-text data.

Finally, our work is closely related to [24] that considers secret sharing schemes (SSS) over infinite domains, e.g. the real number line. Among others, they propose a scheme very similar to ours which is based on polynomials and Lagrange interpolation. However, they consider a game between a dealer and an adversary, which is for the dealer to chose a scheme and a secret such that the adversary has the least probability of guessing the secret. On the contrary, our work assumes that a group of parties would like to perform computations without exposing data belonging to the individual parties. In this sense, the secret is the data, and not something we can chose to our liking. Also, we provide a quantification of the privacy loss of the scheme and propose how to use the scheme for secure multiparty computation (SMPC), which [24] does not.

Table F.1: Comparison of interactive operations (IO) of state-of-the-art protocols, where l_t is the bit-length of the truncated secret and l and k is, respectively, the bit-length of the significant and exponent of the fixed point represented secret.

	IO addition	IO multiplication	IO division	Precision
Shamir's SSS with truncation [14]	0	2	$220l_t + \log 2l_t + 238l_t + 3$	Up to scaling
Shamir's SSS with fixed point representation [19]	$14l + 9k + (\log l)\log\log l$ $+(l+9)\log l + 4\log k + 37$	$8l + 10$	$2\log l(l+2) + 3l + 8$	Up to scaling
Real numbers SSS	0	2	3	Machine precision

1.2 Contribution

The paper puts forth a real number secret sharing scheme which bypasses the usual restrictions to integer secrets and finite field computations. This makes the scheme very practical as solutions can be calculated with high precision and without the need for computations being performed with modular arithmetic. The scheme performs the same or with significant less computation and communication complexity compared to state-of-the-art methods. In Table F.1 the number of interactive operations (IO) are given for a selected number of state-of-the-art protocols. IO's are those that require communication between the participants, and since the time spent on local computations vanishes compared to time spent on IO's, this measure gives both an indication of communication and computation complexity.

The main contribution of the paper can be summarized as:

- To the best of our knowledge, this is the first attempt for a SMPC scheme that works directly on the real number line and consequently offers a trade-off between privacy and practicality.

- The proposed scheme bypasses the requirements for modular arithmetic and integer secrets which is in contrast to state-of-the-art SMPC techniques.

- The scheme allows addition, multiplication and division to be performed directly on shares (ciphertext version of the data), opposed to related schemes that typically only allow addition and in some cases multiplication.

- The paper provides an in-depth analysis of the privacy guaranties of the scheme as well as a quantification of leaked data.

1.3 Outline

The paper proceeds in section 2 by introducing the preliminaries and giving motivation for the work. Section 3 states formally the problem of the paper, while section 4 presents the proposed scheme and the privacy analysis. In section 5 we give a numerical evaluation of the proposed scheme, while section 6 provides simulations of the scheme for Kalman filtering and finally, section 7 concludes the paper.

(a) *n* participants that can communicate privately with each other.

(b) *n* participants and *computing parties* (the grey squares).
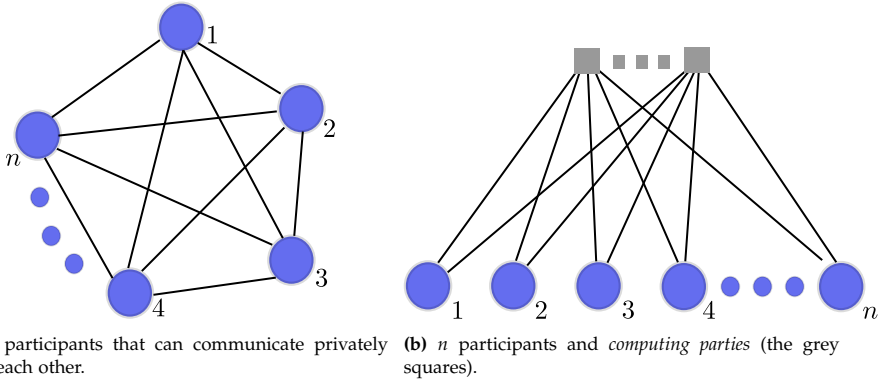
Fig. F.1: Illustration of two scenarios of the communication network. The first scenario (a), each participant can communicate privately with each of the other participant, and all computations are performed by the participants themselves. In (b) each participant can communicate privately with a number of *computing parties* (the grey squares) and each of the computing parties can communicate privately with each of the other computing parties. The computing parties receives shares of the input data from the participants and perform all computations without learning the secret data.

2 Preliminaries and Motivation

In this section, we clarify our notation and terminology and afterwards we give a brief introduction to the concept of *secret sharing* and SMPC, while subsequently discussing their shortcomings which motivates the work in this paper.

2.1 Notation and Terminology

Let $\mathcal{P}$ be an index set of $n > 2$ participants. We assume that each participant $p \in \mathcal{P}$ can communicate privately with each of the other participants $j \in \mathcal{P}$ or alternatively that there exists a number of computing parties that each participant can communicate with. Each of these scenarios is illustrated in Fig. F.1. The advantage of the second scenario is that the computing parties do most of the computations and hence the participants do not have to possess large computation capabilities. Furthermore, the participants need only to communicate with a number of computing parties (which can be as low as 3). In the remaining of the paper we do not make a distinction between these two scenarios, but remark that any presented method can straightforward be used in both.

Concerning notation, let s be a secret value belonging either to a participant or to an external entity providing secret data. We use $\{s[p]\}_{p \in \mathcal{P}}$ to denote the set of so-called *shares* of s. In other words, each share $s[p]$ is a cipher-text version of s. Combining a set $\{s[p]\}_{p \in \mathcal{T}}$ of shares, for $\mathcal{T} \subseteq \mathcal{P}$

where $t < |\mathcal{T}| \le n$ and t is an integer threshold, the shares can be deciphered and s recreated.

2.2   Secret Sharing, SMPC and their shortcomings

Secret sharing in general lets a party "share" a secret among n participants, such that at least $t + 1 \le n$ of the participants must cooperate to learn the secret and opposite; no subset of less than $t + 1$ participants gets information about the secret. There are many different secret sharing schemes, each tailored to different use cases. Perhaps the most simple (and intuitive) secret sharing scheme is the *additive* one [7], where $t = n$ meaning that all shares are needed to reconstruct the secret. In this scheme, the shares $\{s[p]\}_{p \in \mathcal{P}}$ of the secret s satisfy that

$$s = \left(\sum_{p \in \mathcal{P}} s[p] \right) \mod q, \tag{F.3}$$

where q is a large prime number. When choosing $n - 1$ of the shares uniformly on $[0, q - 1]$ and the last share such that (F.3) holds, the modular arithmetic ensures that all shares are uniformly distributed. This means that the scheme is perfectly secure since the uniform distribution holds no information about the secret. The disadvantage is that s must be an element of $\mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field of q elements.

Many secret sharing schemes, like the additive one and Shamirs scheme [25], are very useful in SMPC protocols. These protocols, lets n participants compute a function, that takes as input a private value from each participant, while keeping the private values secret. For instance, for secrets $s_1, s_2 \in \mathbb{F}_q$, the sum $s_1 + s_2$ can be calculated directly on additive shares of each of the secrets;

$$s_1 + s_2 = \left(\sum_{p \in \mathcal{P}} s_1[p] + s_2[p] \right) \mod q, \tag{F.4}$$

where $s_1[p] + s_2[p]$ is computed by the p'th participant.

The drawback is that q must be bigger than $s_1 + s_2$ in order to get the correct result and if no information about the secret data is available, it can be difficult to choose q.

More advanced schemes like Shamir's scheme, also allows multiplication of secrets directly on the shares and in principle also division. However, the division will be *finite field division* [26] and not real number division. As introduced in [14], there are complicated tricks, which usually involve bit-decomposition of the secrets, that will enable the computation of real number division performed on the shares. However, say that the secret to be divided is -3 (which would be represented as $q - 3$ in $\mathbb{F}_q$), what effectively

would happen is the division of $q - 3$ and not -3, which would lead to incorrect results. This is an example of how finite field arithmetic complicates the computations which leads to part of our motivation to introduce a real number secret sharing scheme that does not depend on finite field arithmetic.

3   Problem Statement

Upon the discussion in section 2.2, we conclude that the problem of preserving privacy of real numbers without being limited to finite field arithmetic is indeed a relevant topic in privacy preserving computations. To address this problem, we will propose a real number secret sharing scheme. To this end, we start with the following definition.

> **Definition F.1 (Real Number Secret Sharing Scheme)**
> A real number secret sharing scheme consists of two algorithms; $\texttt{share}$ and $\texttt{recon}$. $\texttt{share}(s, t, \mathcal{P}) = \{s[p]\}_{p \in \mathcal{P}}$ takes a secret $s \in \mathbb{R}$, the threshold $t \in \mathbb{N}$ with $t < n$ and the indices of n participants $\mathcal{P}$ and outputs a share $s[p] \in \mathbb{R}$ for each participant $p \in \mathcal{P}$. The algorithm $\texttt{recon}(\{s[p]\}_{p \in \mathcal{T}}) = s$ outputs the secret s upon inputting at least $t + 1$ shares from any set of participants $p \in \mathcal{T}$, where $\mathcal{T} \subseteq \mathcal{P}$ with $|\mathcal{T}| > t$.

We have the following requirements for the real number secret sharing scheme.

- **Correctness**. A reconstructed secret should be equal to the original secret, that is $s - \texttt{recon}(\{s[p]\}_{p \in \mathcal{T}} = 0$.

- **Privacy.** Only by combining at least t shares of s should it be possible to reconstruct s. A set of fewer than t shares should reveal only very little information about s. We state this formally by using the information theoretic measure called mutual information [27, p.250];

$$I(X; Y) = h(X) - h(X|Y), \tag{F.5}$$

  where $h(X)$ is the entropy of the random variable X and $h(X|Y)$ is the conditional entropy of X given the random variable Y. The mutual information $I(X; Y)$ can be interpreted as the reduction in uncertainty about X one has after learning the outcome of Y (and vice versa since mutual information is symmetric). To this end, we use S and $S[p]$ to denote the random variables that has s and $s[p]$ as outcomes, and we require that for any $\delta > 0$ there exists $\{S[p]\}_{p \in \mathcal{T}'}$ such that

$$I(S; \{S[p]\}_{p \in \mathcal{T}'}) \le \delta, \tag{F.6}$$

where $\mathcal{T}' \subset \mathcal{P}$ with $|\mathcal{T}'| \leq t$.

- **Computations directly on shares**. At least the operations addition, multiplication, and division, should be applicable directly on shares. That is, for any secrets $s_1, s_2 \in \mathbb{R}$ and properly defined protocols $\texttt{add}, \texttt{mult},$ and $\texttt{inv}$, the following should hold

$$\texttt{recon}(\{\texttt{add}(s_1[p], s_2[p])\}_{p \in \mathcal{T}}) = s_1 + s_2 \qquad (\text{F.7})$$

$$\texttt{recon}(\{\texttt{mult}(s_1[p], s_2[p])\}_{p \in \mathcal{T}}) = s_1 s_2 \qquad (\text{F.8})$$

$$\texttt{recon}(\{\texttt{inv}(s_1[p])\}_{p \in \mathcal{T}}) = \frac{1}{s_1} \qquad (\text{F.9})$$

The problem of the paper is to define a real number secret sharing scheme which satisfies the listed requirements assuming that each participant follows the protocol.

4 Proposed Method

As mentioned already, we take great inspiration from Shamir's SSS [25], when proposing our real number SSS. To give some intuition, we explain the derivation of the proposed scheme in comparison to Shamir's scheme.

The approach in Shamir's scheme is to start by choosing t coefficients $\{c_j\}_{j \in T}$, where $T = \{1, \ldots, t\}$, from $\mathbb{F}_q$ uniformly and afterwards defining the polynomial

$$f_s(x) = \left(s + \sum_{j \in T} c_j x^j \right) \mod q, \qquad (\text{F.10})$$

where $s \in \mathbb{F}_q$ as usual is the secret. The shares of s are then defined as

$$\{s[p]\}_{p \in \mathcal{P}} = \{f(p)\}_{p \in \mathcal{P}}. \qquad (\text{F.11})$$

For the real number SSS we want to avoid modular arithmetic and have $s \in \mathbb{R}$ and therefore one idea is to write each share $s[p]$ as

$$s[p] = s + \sum_{j \in T} c_j p^j, \qquad (\text{F.12})$$

where each c_j is Gaussian distributed. We choose the Gaussian distribution because this is the maximum entropy distribution for a random variable on the real number line having a finite mean and variance, [27, p. 413].

Fig. F.2 depicts the shares of a secret $s = 5.0$ for $n = 11$ participants with $t = 5$ (and for comparison also $t = 10$). For the Gaussian distributed coefficients, we use mean value zero and variance 100. As seen, the shares seem quite systematic which is not advantageous from a privacy point of
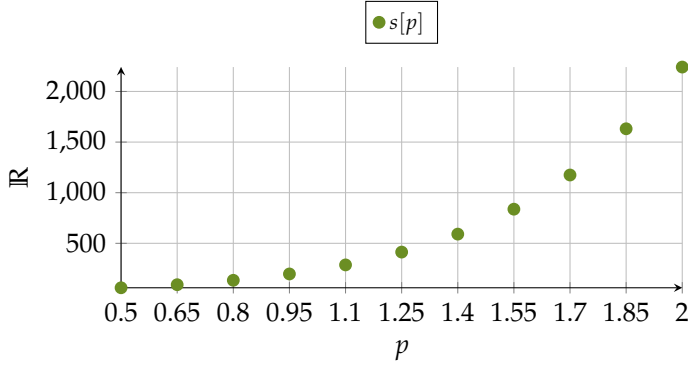
(a) $t = 5$.
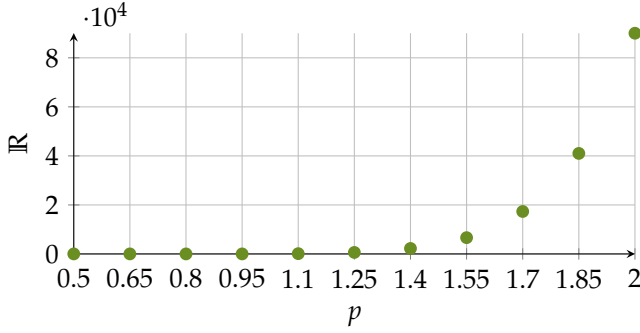
(b) $t = 10$.

Fig. F.2: $n = 11$ shares of the secret $s = 5.0$ with the threshold $t = 5$ for (a) and $t = 10$ for (b), where $s[p] = f(i)$, with $f(x)$ being a polynomial with t coefficients normally distributed with mean value zero and variance 100.

view. Specifically, as seen in (F.12) the random numbers (the coefficients) are scaled according to $p \in \mathcal{P}$. Consequently, less weight are given to the random numbers of the shares constructed with the lower p values. Therefore, the shares tend to be in numerical order as observed in Fig. F.2.

We can information theoretically verify that the information leak caused by a share decreases as the numerical value of $p \in \mathcal{P}$ increases. Consider for instance $\mathcal{P} = \{1, 2, 3\}$ and $t = 2$, then according to (F.12), the shares of s are

$$
\begin{aligned}
s[1] &= s + c_1 + c_2 \\
s[2] &= s + 2c_1 + 4c_2 \\
s[3] &= s + 3c_1 + 9c_2.
\end{aligned}
\tag{F.13}
$$

Then, assuming s, c_1, and c_2 are independent and Gaussian distributed with

mean value zero and variance $\sigma_s^2, \sigma_{c_1}^2$, and $\sigma_{c_2}^2$, respectively, then

$$I(S, S[1]) = \frac{1}{2} \log \left(1 + \frac{\sigma_2^S}{\sigma_{c_1}^2 + \sigma_{c_2}^2} \right)$$

$$I(S, S[2]) = \frac{1}{2} \log \left(1 + \frac{\sigma_2^S}{4\sigma_{c_1}^2 + 16\sigma_{c_2}^2} \right) \tag{F.14}$$

$$I(S, S[3]) = \frac{1}{2} \log \left(1 + \frac{\sigma_2^S}{9\sigma_{c_1}^2 + 81\sigma_{c_2}^2} \right).$$

(F.14) clearly shows that the mutual information, and hence, information leakage about the secret, decreases as p increases. This does not happen in Shamir's SSS because of the modular arithmetic. We therefore need to adjust the method for it to work in a real number SSS.

To make sure each random number carry the same weight across shares, we propose to construct the shares based on Lagrange interpolation [7] and we briefly state this method in our notation.

Consider the points $(\alpha_1, \beta_1), \ldots, (\alpha_t, \beta_t)$ on the plane $\mathbb{R}^2$. A polynomial $f(x)$ of at most degree $t - 1$, that passes through the points, can be found by

$$f(x) = \sum_{j \in T} \beta_j L_j(x), \tag{F.15}$$

where $T = \{1, \ldots, t\}$ and $L_k(x)$ are Lagrange basis polynomials given by

$$L_j(x) = \prod_{k \in T \setminus \{j\}} \frac{x - \alpha_k}{\alpha_j - \alpha_k}. \tag{F.16}$$

To create shares of a secret, we choose t shares at random and interpolate these shares to a degree (at most) t polynomial $f_s(x)$, by also using that $f_s(0) = s$. Using Lagrange basis polynomials stated above, $f_s(x)$ is written as

$$
\begin{aligned}
f_s(x) &= s \underbrace{\prod_{k=1}^{t} \frac{x - x_k}{x_0 - x_k}}_{L_0(x)} + y_1 \prod_{k=0, k \neq 1}^{t} \frac{x - x_k}{x_1 - x_k} + \cdots + y_t \prod_{k=0}^{t-1} \frac{x - x_k}{x_t - x_k} \\
&= s L_0(x) + y_1 \frac{x}{x_1} \prod_{k=2}^{t} \frac{x - x_k}{x_1 - x_k} + \cdots + y_t \frac{x}{x_t} \prod_{k=1}^{t-1} \frac{x - x_k}{x_t - x_k} \\
&= s L_0(x) + y_1 \frac{x}{x_1} L_1(x) + \cdots + y_t \frac{x}{x_t} L_t(x) \\
&= s L_0(x) + \sum_{j \in T} y_j \frac{x}{x_j} L_j(x),
\end{aligned}
\tag{F.17}
$$

4. Proposed Method
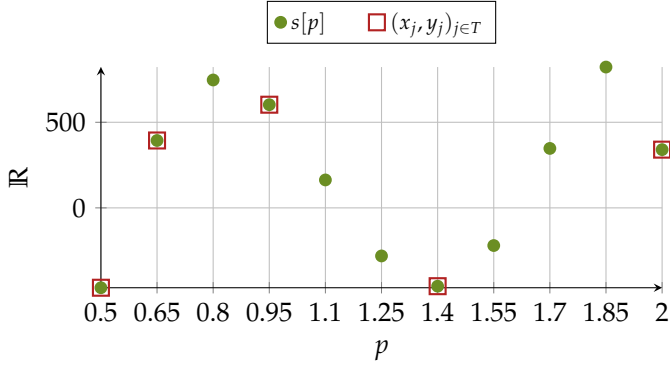
(a) $t = 5$.
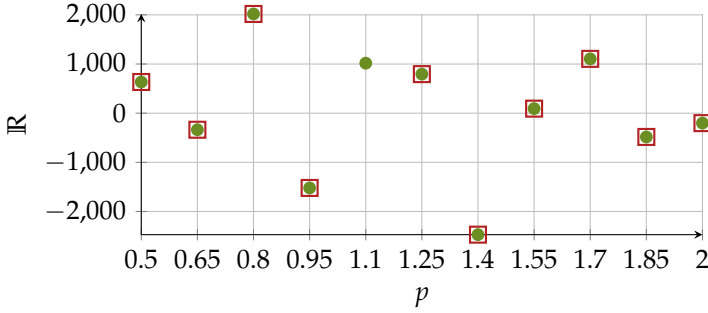
(b) $t = 10$.

Fig. F.3: $n = 11$ shares of the secret $s = 5$ with the threshold $t = 5$ for (a) and $t = 10$ for (b), where $s[p] = f(i)$, with $f(x)$ being a polynomial. t points of $f(x)$ (marked with a □) are normally distributed with mean value zero and variance 100.

where we use that $x_0 = 0$. The shares are then defined as

$$\{s[p]\}_{p \in \mathcal{P}} = \{f_s(p)\}_{p \in \mathcal{P}}.$$

As seen in (F.17), the random numbers (y_j) are normalized and thus have the same weight across shares. Therefore, the shares are much less predictable (especially as t increase) as observed in Fig. F.3.

Moreover, we can information theoretically show that the information leakage of the shares constructed by (F.17) does not depend on the numerical value of p. Consider, $\mathcal{P} = \{1, 2, 3\}$, $t = 2$ and $x_1 = 1$ and $x_2 = 3$. Remark that x_j are chosen each time shares of a secret are constructed and their value is

unknown to any adversary. According to (F.17), the shares of s are written as

$$s[1] = s\frac{1-1}{0-1}\frac{1-3}{0-3} + y_1\frac{1-0}{1-0}\frac{1-3}{1-3} + y_2\frac{1-0}{3-0}\frac{1-1}{3-1} = y_1$$

$$s[2] = s\frac{2-1}{0-1}\frac{2-3}{0-3} + y_1\frac{2-0}{1-0}\frac{2-3}{1-3} + y_2\frac{2-0}{3-0}\frac{2-1}{3-1} = \frac{1}{3}s - y_1 + \frac{1}{3}y_2 \quad \text{(F.18)}$$

$$s[3] = s\frac{3-1}{0-1}\frac{3-3}{0-3} + y_1\frac{3-0}{1-0}\frac{3-3}{1-3} + y_2\frac{3-0}{3-0}\frac{3-1}{3-1} = y_2.$$

Assuming s and y_i are independent and Gaussian distributed with mean zero and variance $\sigma_s^2, \sigma_{Y_1}^2$, and $\sigma_{Y_2}^2$, respectively, the mutual information yields

$$I(S, S[1]) = 0$$

$$I(S, S[2]) = \frac{1}{2}\log(1 + \frac{\frac{1}{9}\sigma_S^2}{\sigma_{Y_1}^2 + \frac{1}{9}\sigma_{Y_2}^2}) \quad \text{(F.19)}$$

$$I(S, S[3]) = 0.$$

Thus, (F.19) shows that the information leakage caused by the shares are independent of the numerical value of p, which is of course important from a privacy perspective. More precisely, the difference between (F.14) and (F.19), is that in the former each participant p knows that you gain most information about the secrets the lower the value of p you have. In the latter, it is unknown to the participants which shares have zero mutual information and which does not, and it is different for each secret.

We state the `share` algorithm of the real number SSS formally in Algorithm F.1 and expand on the privacy analysis of it in section 4.1.

Algorithm F.1: `share`$(s, t, \mathcal{P}) = \{s[p]\}_{p \in \mathcal{P}}$

Input: s is the secret, t is the threshold and $\mathcal{P}$, with $|\mathcal{P}| = n$ is the index set of the participants.
Output: $\{s[p]\}_{p \in \mathcal{P}}$ is the set of shares of s.
1: Draw distinct $\{x_j\}_{j \in T}$ from $\mathcal{P}$, where $T = \{1, \dots, t\}$.
2: $y_j \overset{\text{iid}}{\sim} \mathcal{N}(\mu_Y, \sigma_Y^2)$ for $j \in T$, where μ_Y and σ_Y^2 are chosen privacy parameters.
3: $f_s(x) = sL_0 + \sum_{j \in T} y_j \frac{x}{x_j} L_j(x)$.
4: $\{s[p]\}_{p \in \mathcal{P}} = \{f_s(p)\}_{p \in \mathcal{P}}$.

Remark, that Algorithm F.1 has two privacy parameters μ_Y and σ_Y^2 which can be chosen by the party constructing shares of its secret. The mean value does not have a significant effect on the privacy and could in principle be chosen randomly (or as zero as we do throughout the paper). In section 4.1 the impact of σ_Y^2 becomes clear.

4. Proposed Method

The reconstruct algorithm of the proposed real number secret sharing scheme, is almost identical to the one of Shamir's scheme (the only difference is the lacking of modular arithmetic). Since the algorithm consist solely of Lagrange interpolation, we state it without further introduction.

Algorithm F.2: $\texttt{recon}(\{s[p]\}_{p\in\mathcal{T}}) = \hat{s}$

Input: $\{s[p]\}_{p\in\mathcal{T}}$, with $|\mathcal{T}| > t$, is a set of at least $t+1$ shares of s.
Output: $\hat{s}$, the reconstructed secret.
1: Define $\{(p, s[p])\}_{p\in\mathcal{T}}$ as the set of points to interpolate.
2: $f_r(x) = \sum_{p\in\mathcal{T}} s[p] L_p(x)$.
3: $\hat{s} = f_r(0)$.

To be clear, our proposed real number secret sharing scheme, consists of the algorithms share and recon stated in Algorithm F.1 and Algorithm F.2, respectively. To give intuition about the proposed method, Example F.1 gives an example of using it.

> **Example F.1 (Real number secret sharing)**
> Let $s = 5.0$ be a secret and $\mathcal{P}=\{0.5,0.65,0.8,0.95,1.1,1.25,1.4,1.55,1.7,1.85,2\}$ the index of the participants. Consider $\texttt{share}(s,t,\mathcal{P})$ in Algorithm F.1 to create shares of s for $n = 11$ participants. We perform the following steps with $t = 5$:
>
> 1. $\{x_j\}_{j\in T} = \{0.5, 0.65, 0.95, 1.4, 2\}$.
>
> 2. $\{y_j\}_{j\in T} = \{-466.506, 393.646, 602.653, -457.489, 340.160\}$. See $(x_j, y_j)_{j\in T}$ in Fig. F.3a marked with $\square$.
>
> 3. Define $f(x) = sL_0 + \sum_{j\in T} y_j \frac{x}{x_j} L_j(x)$.
>
> 4. Define $\{s[p]\}_{p\in\mathcal{P}} = \{f(p)\}_{p\in\mathcal{P}}$. See $\{s[p]\}_{p\in\mathcal{P}}$ in Fig. F.3a marked with $\bullet$.
>
> For comparison, we perform the same steps for $t = 10$, where $(x_j, y_j)_{j\in T}$ are seen in Fig. F.3b marked with $\square$ and $\{s[p]\}_{p\in\mathcal{P}}$ are seen in Fig. F.3b marked with $\bullet$.

We will now show that the scheme satisfies the requirements listed in section 3. We start by noting that the proof of Lagrange interpolation also proves the correctness of the scheme. Therefore, we immediately analyse the privacy of the scheme in the following section.

4.1 Privacy Analysis

We start out the analysis by considering one participant $p \in \mathcal{P}$, who does not know s, but learns $s[p]$. That is, from the view of p, s can be modeled as the outcome of the random variable S having some distribution. The uncertainty p has about s can be stated as the differential entropy $h(S)$ of S. Also $s[p]$ is the outcome of a random variable $S[p]$. To see the relation between S and $S[p]$, consider the rewrite of $s[p]$

$$s[p] = sL_0(p) + \underbrace{\sum_{j \in T} y_j \frac{p}{x_j} L_j(p)}_{b(p)} = sL_0(p) + b(p), \tag{F.20}$$

To this end, we have that

$$S[p] = SL_0(p) + B(p). \tag{F.21}$$

We choose to model the L_j values as constants even though it can be argued that they are indeed random variables because each x_k from step 1. of Algorithm F.1 are randomly chosen. However, since t is generally close to n and $\mathcal{P}$ is public, there is not an insignificant probability of guessing the x_k values. Consider for instance $\mathcal{P}$ given in Example F.1 and let $t = 10$. Then we know that $x_1 \in \{0.5, 0.65\}$ because the 9 remaining x_k values must also be distinct elements of $\mathcal{P}$. Consequently, for each x_k there are generally only a few possible values it can take and thus in our analysis we choose to treat each L_j value as a constant. To this end, $B(p)$ is normally distributed with mean $t\mu_Y$ and variance

$$\sigma^2_{B(p)} = \sigma^2_Y \sum_{j \in T} \left(\frac{p}{x_j} L_j(p) \right)^2. \tag{F.22}$$

Consider now the mutual information $I(S; S[p])$ between S and $S[p]$;

$$\begin{aligned} I(S; S[p]) &= h(S[p]) - h(S[p]|S) \\ &= h(SL_0(p) + B(p)) - h(SL_0(p) + B(p)|S) \\ &= h(SL_0(p) + B(p)) - h(B(p)), \end{aligned} \tag{F.23}$$

where we use that $I(X, Y)$ is symmetric, that L_0 is a constant, and that S and $B(p)$ are independent. Before we proceed, we remark that when $f_s(x)$ given in step 3. of Algorithm F.1, is evaluated in one of the x_k values chosen in step 1., y_k is outputted. That is, $f_s(x_k) = y_k$, see (F.18). Recall that each y_k is Gaussian distributed and since $x_k \in \mathcal{P}$, we have that exactly t shares are completely independent of the secret s. Thus, in this best case scenario, which is true for t shares, $I(S; S[p]) = 0$ and there is no leak of information.

To analyse the information leakage of the remaining $n - t$ shares, we take the same approach as in [28] and consider again (F.23). As discussed, $B(p)$ are Gaussian distributed and according to [27, p. 244], the differential entropy of a Gaussian distributed variable X can be written as

$$h(X) = \frac{1}{2} \log_2(2\pi e \sigma_X^2), \tag{F.24}$$

where e is the Euler number. On the other hand, we do not make an assumption of the distribution of $SL_0(p) + B(p)$, since this can vary from application to application. Instead, we note that a high entropy of $SL_0(p) + B(p)$, results in a higher $I(S; S[p])$ in equation (F.23). Thus, by using the maximum entropy distribution (which is the Gaussian distribution) as the distribution of $SL_0(p) + B(p)$, we establish an upper bound on the mutual information.

$$
\begin{aligned}
I(S; S[p]) &= h(SL_0(p) + B(p)) - h(B(p)) \\
&\leq \frac{1}{2} \log(2\pi e(\sigma_{S_{L_0(p)}}^2 + \sigma_{B(p)}^2)) - \frac{1}{2} \log(2\pi e \sigma_{B(p)}^2) \\
&= \frac{1}{2} \log\left(1 + \frac{\sigma_{SL_0(p)}^2}{\sigma_{B(p)}^2}\right),
\end{aligned}
\tag{F.25}
$$

where we use that since S and $B(p)$ are independent, the variance of $S[p]$ can be written as

$$\sigma_{S[p]}^2 = \sigma_{SL_0(p)}^2 + \sigma_{B(p)}^2. \tag{F.26}$$

In conclusion, choosing for instance $\sigma_{B(p)}^2$ 100 times larger than the variance of $\sigma_{SL_0(p)}^2$, the leaked information is at most 0.0072 bits (no matter the real distribution of $SL_0(p) + B(p)$), which is to be read in the way that *on the average* one share of s leaks 0.0072 bits. For comparison, if the secret indeed is Gaussian distributed with variance 10, the uncertainty about it is 3.7080 bits and after learning $s[p]$, the uncertainty is 3.7008 bits. Hence, each share $s[p]$ leaks only very little information about s, when choosing the variance σ_Y^2 large enough.

To continue this analysis, note that in the problem statement, we require that a set of at most t shares should reveal very little information about the secret. Thus, we now address the mutual information between s and a set of t shares. That is,

$$I(S; S[1], \ldots, S[t]) = h(S[1], \ldots, S[t]) - h(B(1), \ldots, B(t)) \tag{F.27}$$

Again, we notice that in the best case scenario, the set of t shares is exactly the set of normally distributed values y_j chosen in step 2. of Algorithm F.1, i.e. $\{s[p]\}_{p \in \mathcal{T}'} = \{y_j\}_{j \in T}$, where $T = 1, \ldots, t$. In this case, all t shares are independent of the secret s and thus we have no leak of information. This

case happens with a high probability if t is close to n. However, due to properties of the scheme, which we will explore in the following section, t might be chosen less than $\lfloor \frac{n}{2} \rfloor$. In this case, we may have that none of the t shares are independent of the secret. This would be the worst case scenario, which we address now by establishing an upper bound for $I(S; S[1], \dots, S[t])$ by using the same trick as previously. Namely, we choose the t-variate Gaussian distribution for $X_S = (S[1], \dots, S[t])$, which is the maximum entropy distribution. Since the sum of two Gaussian distributions is still Gaussian, we have that $X_B = (B(1), \dots, B(t))$ also follows a t-variate Gaussian distribution. The entropy of a N-variate Gaussian distributed variable X is given as [27, p.249]

$$h(X) = \frac{1}{2} \log \left((2\pi e)^N \det(C_X) \right), \tag{F.28}$$

where C_X is the covariance matrix for X. This expression can be used directly in (F.27), yielding

$$\begin{aligned} I(S; X_S) &\leq \frac{1}{2} \log \left((2\pi e)^t \det(C_{X_S}) \right) - \frac{1}{2} \log \left((2\pi e)^t \det(C_{X_B}) \right) \\ &= \frac{1}{2} \log \left(\frac{\det(C_{X_S})}{\det(C_{X_B})} \right) \end{aligned} \tag{F.29}$$

Using (F.21), we can write the (i,j)'th term of the covariance matrix C_{X_S}, as

$$\begin{aligned} c_{X_S}(i,j) &= \mathrm{cov}\left(SL_0(i) + B(i); SL_0(j) + B(j) \right) \\ &= \mathrm{cov}\left(SL_0(i); SL_0(j) + B(j) \right) + \mathrm{cov}\left(B(i); SL_0(j) + B(j) \right) \\ &= \mathrm{cov}\left(SL_0(i); SL_0(j) \right) + \mathrm{cov}\left(SL_0(i); B(j) \right) \\ &\quad + \mathrm{cov}\left(B(i); SL_0(j) \right) + \mathrm{cov}\left(B(i); B(j) \right) \\ &= \mathrm{cov}\left(SL_0(i); SL_0(j) \right) + \mathrm{cov}\left(B(i); B(j) \right), \end{aligned} \tag{F.30}$$

where we use that $SL_0(i)$ and $B(i)$ are independent. Therefore, we have

$$C_{X_S} = C_{X_{SL_0}} + C_{X_B}, \tag{F.31}$$

where $X_{SL_0} = (SL_0(1), \dots, SL_0(t))$.

Thus, analogue to the previous result, the leaked information is controlled by the relation between the variance of S and the variance of Y. By choosing σ_Y^2 large compared to the variance of S, the determinant of C_{X_S} will be only slightly larger than the determinant of C_{X_B} and we have that asymptotically, the leaked information goes to zero bits.

We can therefore make the following proposition, stating that the scheme fulfills the privacy requirement.

Proposition F.1

The real number secret sharing scheme comprised of the algorithms `share` and `recon` stated in Algorithm F.1 and Algorithm F.2, respectively, satisfy that for any $\delta > 0$ there exists the covariance matrix C_{X_B} such that

$$I(S; \{S[p]\}_{p \in \mathcal{T}'}) \leq \delta, \tag{F.32}$$

for a secret s being the outcome a random variable S and shares $\{s[p]\}_{p \in \mathcal{T}'}$ being the outcome of the random variables $\{S[p]\}_{p \in \mathcal{T}'}$.

Proof. We use (F.29) and (F.31). For short, we write $A = C_{X_{S_{L_0}}}$, and $B = C_{X_B}$. Since A and B are symmetric, in fact positive semi-definite, they can be simultaneously diagonalizable. We denote the eigenvalues of A by λ_A^i, and B by λ_B^i. Let $\overline{\lambda}_A, \overline{\lambda}_B$ be the maximal eigenvalue of A, B respectively, and $\underline{\lambda}_B$ be the minimal eigenvalue of B. Specifically,

$$\frac{\det(C_{X_S})}{\det(C_{X_B})} = \frac{\prod_{i=1}^{N}(\lambda_A^i + \lambda_B^i)}{\prod_{i=1}^{N} \lambda_B^i} \leq \frac{\overline{\lambda}_A + \overline{\lambda}_B}{\underline{\lambda}_B} = \frac{\overline{\lambda}_A}{\underline{\lambda}_B} + \frac{\overline{\lambda}_B}{\underline{\lambda}_B}.$$

Hence, by rescaling $\det(C_{X_B})$ by sufficiently large coefficient, the mutual information $I(S; S(1), \dots, S(t))$ can be made arbitrarily small. $\square$

In the next section, we will show that the real number secret sharing scheme also satisfies that final requirement.

4.2 Computations on Shares

In this section, we will define the algorithms `add`, `mult`, and `inv`, which perform addition, multiplication and inverse of secrets directly on the shares and, thus, does not leak any secrets. To improve readability and intuition, we present the operations using scalars, however the methods are easily extendable to matrices as well. To this end, we start by defining what we mean by shares of a matrix (and equivalent; a vector).

> **Definition F.2 (Secret shared matrix)**
> Let $A \in \mathbb{R}^{m_1 \times m_2}$ be a matrix and let each entry of A be secret shared using `share`. To this end, $A[p]$ denotes the matrix consisting of the p'th share of each element in A, respectively.

For the rest of this section, assume that $s, a \in \mathbb{R}$ are secrets and that each participant $p \in \mathcal{P}$ holds the shares $s[p]$ and $a[p]$, respectively.

Addition

We start out with the simplest operation, which is addition. The output of the addition algorithm is that each participant p holds a share $c[p]$, where $c = s + a$. Note that since each share is a point on a polynomial, it can be written as

$$s[p] = s + \alpha_1 p + \alpha_2 p^2 + \cdots + \alpha_t p^t, \tag{F.33}$$

and

$$a[p] = a + \beta_1 p + \beta_2 p^2 + \cdots + \beta_t p^t, \tag{F.34}$$

where $\alpha_j, \beta_j \in \mathbb{R}$ are coefficients. Adding the above expressions yields

$$c[p] = s[p] + a[p] = (s + a) + (\alpha_1 + \beta_1)p + \cdots + (\alpha_t + \beta_t)p^t. \tag{F.35}$$

Hence, by participant p performing $s[p] + a[p]$, it now holds a share $c[p]$, where $c = s + a$. To denote the computation of adding shares we simply use the '+' sign or we write $\mathtt{add}(s[p], a[p]) = c[p]$. Note that subtraction is performed on the shares equivalently, which we simply denote by '−'.

Multiplication

Multiplying shares is somewhat more complicated. If we attempted to simply multiply the polynomials like we added them previously, we would find that the degree of the resulting polynomial is $2t$. In this case we need $2t + 1$ shares to reconstruct the secret. To avoid the growing degree of the polynomial, we use a well-know trick called Beavers' trick, [29]. It uses so-called triplets, $\{r_1[p], r_2[p], r_1r_2[p]\}_{p \in \mathcal{P}}$ of shares of (unknown) random numbers r_1 and r_2, and their product r_1r_2. To create the triplets, it is typically required that $t < \lfloor \frac{n}{2} \rfloor$, however there are ongoing research in efficient methods of generating Beaver triplets for larger values of t, [30].

We state formally $\mathtt{mult}$ in Algorithm F.3.

Algorithm F.3: $\mathtt{mult}(s[p], a[p]) = sa[p]$

Input: $\{s[p]\}_{p \in \mathcal{P}}, \{a[p]\}_{p \in \mathcal{P}}$ shares of the secrets and $\{r_1[p], r_2[p], r_1r_2[p]\}_{p \in \mathcal{P}}$ shares of the unknown Beavers triplet.

Output: $\{sa[p]\}_{p \in \mathcal{P}}$, shares of the product of the secrets.

1:

$$d = \mathtt{recon}(\{s[p] - r_1[p]\}_{p \in \mathcal{T}}) \tag{F.36}$$

$$e = \mathtt{recon}(\{a[p] - r_2[p]\}_{p \in \mathcal{T}}), \tag{F.37}$$

2:

$$sa[p] = de + dr_2[p] + r_1[p]e + r_1r_2[p], \tag{F.38}$$

To see that the multiplication protocol in Algorithm F.3 is correct, perform the following rewrite

$$s = d + r_1$$
$$a = e + r_2, \tag{F.39}$$

to see that

$$sa[p] = (d + r_1[p])(e + r_2[p]). \tag{F.40}$$

Note that a public constant (like e and d in this case) can be directly multiplied on the shares by each participant. This can easily be verified by using the same approach as showing that the add protocol is correct.

We also remark that $d = s + r_1$ and $e = a + r_2$ are revealed in plain text in Algorithm F.3. Since r_1 and r_2 are Gaussian distributed, d and e does not leak more than a share of the secrets. However, we give here the upper bound of the information leak of knowing both t shares of s and also d.

$$\begin{aligned}
I(S; \{S[p]\}_{p \in \mathcal{T}'}, S + R_1) \\
= h(\{S[p]\}_{p \in \mathcal{T}'}, S + R_1) - h(\{S[p]\}_{p \in \mathcal{T}'}, S + R_1 | S) \\
= h(\{S[p]\}_{p \in \mathcal{T}'}, S + R_1) \\
- h(\{SL_0(p) + B(p)\}_{p \in \mathcal{T}'}, S + R_1 | S) \\
= h(\{S[p]\}_{p \in \mathcal{T}'}, S + R_1) - h(\{B(p)\}_{p \in \mathcal{T}'}, R_1)
\end{aligned} \tag{F.41}$$

To find an upper bound on the information leakage we use the maximal entropy distribution for the distribution of $X_{SR_1} = (\{S[p]\}_{p \in \mathcal{T}'}, S + R_1)$. By design, $X_{BR_1} = (\{B(p)\}_{p \in \mathcal{P}}, R_1)$ are distributed according to a multivariate Gaussian distribution. To this end we have,

$$I(S; X_{SR_1}) \leq \frac{1}{2} \log \left((2\pi e)^t \frac{\det(C_{X_{SR_1}})}{\det(C_{X_{BR_1}})} \right), \tag{F.42}$$

where $C_{X_{SR_1}}, C_{X_{BR_1}}$ are the covariance matrices of X_{SR_1} and X_{BR_1}, respectively. As seen, the result in (F.42) is very similar to the one obtained in (F.29). To demonstrate the (at most) revealed data using `mult`, Example F.2 demonstrates the multiplication of two secrets. Note that in section 5 we numerically estimate the leak of information caused by the multiplication protocol.

> **Example F.2 (Multiplication of shares)**
> Let the number of participants $n = 7$, $\mathcal{P} = \{1, 2, \ldots, 7\}$, and the threshold $t = 3$. Consider two secret $s_1 = 34.5$ and $s_2 = 3.42$ and the multiplication of them performed on their shares. To demonstrate the (small) information leak caused by `mult`, Fig. F.4 depicts $t = 3$ shares of each secret and the values d and e revealed by the algorithm. As seen, it is very hard to deduce the true values of the secrets using the revealed information.
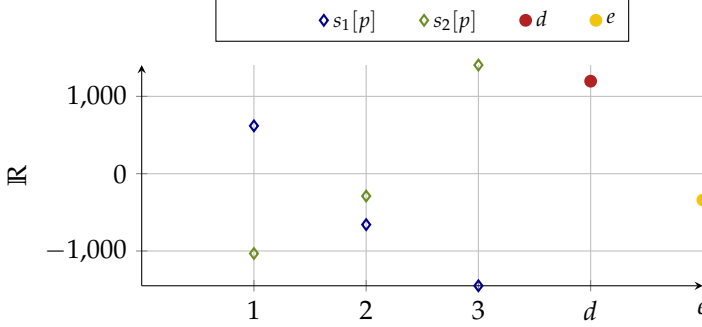
Fig. F.4: Example of the information known about two secrets s_1 and s_2 after executing the `mult` algorithm. In the worst case, the adversary knows $t = 3$ shares of each of the secrets and the values d and e revealed by `mult`. In this example $s_1 = 34.5$ and $s_2 = 3.42$, which is very hard to deduce from the revealed information.

We use $\mathtt{mult}(s[p], a[p]) = sa[p])$ to denote the computation of multiplying shares using Beaver's trick. In continuation, we note that `mult` can easily take two matrices as input, for instance $A[p]$ with $A \in \mathbb{R}^{m_1 \times m_2}$ and $B[p]$ with $B \in \mathbb{R}^{m_2 \times m_3}$. In this case the Beavers triplet is also matrices; $R_1 \in \mathbb{R}^{m_1 \times m_2}$, $R_2 \in \mathbb{R}^{m_2 \times m_3}$ and $R_1 R_2$ is the matrix-matrix product. The rest of algorithm F.3 remains unchanged.

Division

We consider the inversion s^{-1} and note that one could afterwards use `mult` to compute a secret divided by another secret. We propose to compute this operation efficiently on the shares, by noting that

$$s^{-1} = \frac{1}{sr}r, \tag{F.43}$$

where $r \in \mathbb{R}$ is a random number. To this end, we propose to use a normally distributed random variable r which is unknown to the participants. This r can be constructed in the following way; each participant p chooses a Gaussian distributed value r_p and distributes the shares $r_p[j]$ to participant $j \in \mathcal{P}$. Each participant p then computes its share of r by $r[p] = \sum_{j \in \mathcal{P}} r_j[p]$.

To calculate (F.43), the participants use $\mathtt{recon}(\mathtt{mult}(s[p], r[p])) = sr$ to learn in plain text the product sr. Subsequently, they each compute $s^{-1}[p] = \frac{1}{sr}r[p]$ to learn individual shares of s^{-1}. To improve readability, we state the division algorithm in Algorithm F.4.

Algorithm F.4: $\mathtt{inv}(s[p]) = s^{-1}[p]$

Input: $\{s[p]\}_{p \in \mathcal{P}}$ shares of the secret and $\{r[p]\}_{p \in \mathcal{P}}$ shares of an unknown random value $r \in \mathbb{R}$.

Output: $\{s^{-1}[p]\}_{p \in \mathcal{P}}$, shares of the inverse secret.

1: $sr = \mathtt{recon}(\mathtt{mult}(s[p], r[p])$.

2: $s^{-1}[p] = (sr)^{-1} r[p]$.

We remark that the plain text sr does reveal some information about s. However, this information leak can be upper bounded. We here compute the maximal information leak about s from a set of t shares of s joint with sr.

$$
\begin{aligned}
I(S; \{S[p]\}_{p \in \mathcal{T}'}, SR) &= h(\{S[p]\}_{p \in \mathcal{T}'}, SR) - h(\{S[p]\}_{p \in \mathcal{T}'}, SR \mid S) \\
&= h(\{S[p]\}_{p \in \mathcal{T}'}, SR) - h(\{SL_0(p) + B(p)\}_{p \in \mathcal{T}'}, SR \mid S) \\
&= h(\{S[p]\}_{p \in \mathcal{T}'}, SR) - h(\{B(p)\}_{p \in \mathcal{T}'}, R)
\end{aligned}
$$
(F.44)

No assumptions on the joint distribution of $X_{SR} = (\{S[p]\}_{p \in \mathcal{P}}, SR)$ is made, thus we make an upper bound for the mutual information by choosing the maximal entropy distribution. By design, $X_{BR} = (\{B(p)\}_{p \in \mathcal{P}}, R)$ are distributed according to a multivariate Gaussian distribution. To this end, we have

$$
I(S; X_{SR}) \le \frac{1}{2} \log \left((2\pi e)^t \frac{\det(C_{X_{SR}})}{\det(C_{X_{BR}})} \right),
$$
(F.45)

which is a very similar to the result obtained in (F.29).

We denote the computation of $s^{-1}[p]$ as $\mathtt{inv}(s[p]) = s^{-1}[p]$ and note that also $\mathtt{inv}$ can take a matrix as input. In this case, the random value r is simply a random matrix of suitable dimension and the rest of the algorithm remains the same.

5   Numerical Evaluation

In this section, we evaluate the numerical performance of the proposed real number secret sharing scheme. To this end, we have implemented the scheme on a laptop PC in the programming language Python that uses the IEEE 754 floating point standard. We start by evaluating the accuracy of the scheme in terms of the variance of the Gaussian distributed y_j values in Algorithm F.1. The parameters we have chosen are $n = 11$ participants, $t = 5$, and the secrets $s_1 = 5.5$ and $s_2 = 34.7$. We simulate both $\mathtt{recon}$ (Algorithm F.2), $\mathtt{add}$, $\mathtt{mult}$ (Algorithm F.3), and $\mathtt{inv}$ (Algorithm F.4), where we start by generating shares of the secrets using Algorithm F.1. Afterwards, we either directly reconstruct the secret using $\mathtt{recon}$ in Algorithm F.2 or use respectively, $\mathtt{add}$, $\mathtt{mult}$, or $\mathtt{div}$
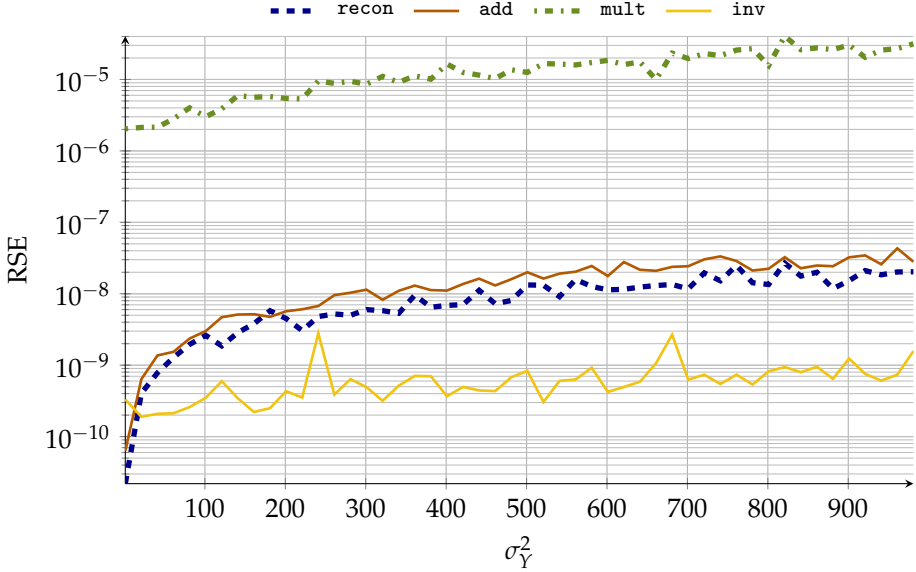
Fig. F.5: Accuracy of the algorithms `recon`, `add`, `mult`, and `inv` in terms of the variance of the Gaussian distributed y_j values in `share` (Algorithm F.1). The loss of accuracy is due to numerical errors.

on the shares before reconstruction. To evaluate the accuracy, we use the root square error (RSE) between the expected result v and the reconstructed result $\hat{v}$, which is defined as

$$\text{RSE} = \sqrt{(v - \hat{v})^2}.$$

Fig. F.5 depicts the RSE between v and $\hat{v}$ as a function of σ_Y^2, for all four algorithms. As seen, as σ_Y^2 is increased, the accuracy slowly decreases. This is purely due to numerical errors, because as the y_j values in Algorithm F.1 increases, the shares grow exponentially large and consequently loose precision due to the floating point representation. The reason why `inv` achieves such high precision, is because the outputted shares are relatively small due to the reciprocal operation of the algorithm.

Finally, we numerically evaluate the privacy properties of the scheme. That is, we estimate the privacy loss of the secret from one share, from t shares and from performing multiplication. In particular, we estimate $I(S; S[1])$ in (F.25), $I(S; S[1], \ldots, S[t])$ in (F.27), and $I(S; S[1], \ldots, S[t], S + R_1)$ in (F.41), based on simulated data. These estimations are a product of statistical analysis, thus we generate a large sample size of each relevant variable for each estimation. We simulate in each case the secret $S \sim \mathcal{N}(0, 1)$ and the remaining variables are computed based on the secret. Fig. F.6 depicts all
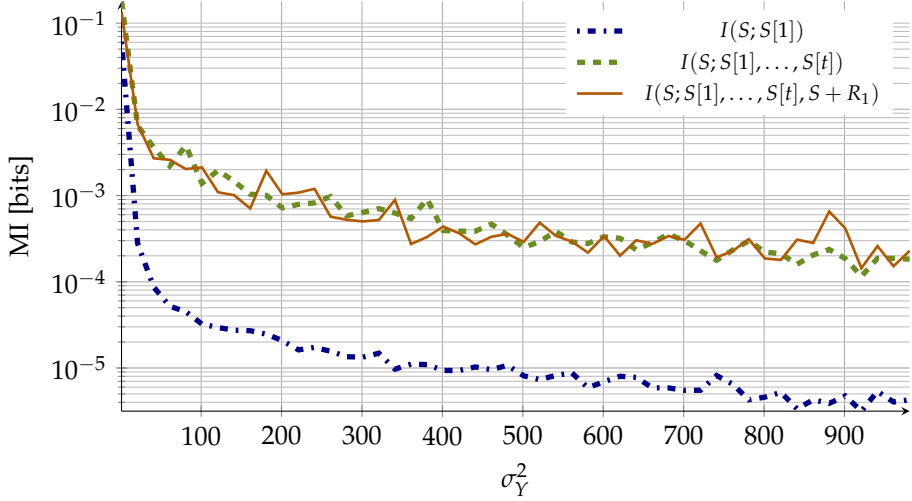
Fig. F.6: Mutual information (MI) between a standard normal distributed secret, S and, respectively, one share of S, t shares of S, and t shares of S joint with $S + R_1$ for a normal distributed variable R_1 (see Algorithm F.3).

three estimations and as expected, one share of the secret leaks very little information while t shares clearly has a greater leak. As seen, these numerical results validate the theoretical results.

6 Application to Kalman filtering

To demonstrate our proposed privacy preserving computation framework, we use the Kalman filter [31] to estimate $\hat{x}_k$ of (C.1) when given only real number secret shared versions of the observations in (F.2). The Kalman filter consists of the following 5 equations, where P is the covariance matrix of the estimate, K is the Kalman gain and Q and R are covariance matrices of the process and measurement noise respectively,

$$
\begin{aligned}
\tilde{x}_k &= A\hat{x}_{k-1} + Bu_k \\
\tilde{P}_k &= AP_{k-1}A^\top + Q_k \\
K_k &= \tilde{P}_k H^\top (H_k \tilde{P}_k H^\top + R_k)^{-1} \\
\hat{x}_k &= \tilde{x}_k + K_k(z_k - H\tilde{x}_k) \\
P_k &= \tilde{P}_k - K_k H_k \tilde{P}_k.
\end{aligned}
\tag{F.46}
$$

We consider the following scenario. Assume that n none-colluding entities are used as computing units, hereafter referred to as computing parties. That is, the computing parties perform all computations given only shares of

the data. Each time the computing parties receive shares of a new measurement, they compute a new update of the state estimate. We do not specify who delivers these measurements, but it could likely be from a collection of nodes or from a set of other participants. The computing parties are not allowed to learn any clear text data and they only output shares (which can afterwards be reconstruct to the clear text output).

In Algorithm F.5, we state a privacy preserving Kalman filter based on the proposed real number secret sharing scheme, from the view of computing party $p \in \mathcal{P}$.

Algorithm F.5: `privKalman()`

Input: $u_k[p]$ for all k, are shares of the observations, P_0 and K_0 can be initialized as identity matrices.

Output: $\hat{x}_k[p]$; the estimate of the k'th state of the system.

1: **for** all k **do**
2: $\quad \tilde{x}_k[p] = \texttt{mult}(A[p], \hat{x}_{k-1}[p]) + \texttt{mult}(B[p], u_k[p])$
3: $\quad V_k[p] = \texttt{mult}(P_{k-1}[p], A^\top[p])$
4: $\quad \tilde{P}_k[p] = \texttt{mult}(A[p], V_k[p]) + Q_k[p]$
5: $\quad S_k[p] = \texttt{mult}(H[p], \texttt{mult}(\tilde{P}_k[p], H^\top[p])) + R_k[p]$
6: $\quad K_k[p] = \texttt{mult}(\texttt{mult}(\tilde{P}_k[p], H^\top[p]), \texttt{inv}(S_k[p]))$
7: $\quad y_k[p] = z_k[p] - \texttt{mult}(H_k[p], \tilde{x}_k[p])$
8: $\quad \hat{x}_k[p] = \tilde{x}_k[p] + \texttt{mult}(K_k[p], y_k[p])$
9: $\quad P_k[p] = \tilde{P}_k[p] - \texttt{mult}(K_k[p], \texttt{mult}(H_k[p], \tilde{P}_k[p]))$
10: **end for**

Remark that Algorithm F.5 does not reveal the result or any intermediate results.

6.1 Simulation

We have simulated Algorithm F.5 and compared its estimation performance to the algorithm in (F.46) which does not provide any privacy. Thus, we want to evaluate the sacrifice in output utility when using the privacy preserving algorithm. We thus simulate both algorithms solving the same problem and compare the results. We conduct the simulation on a laptop PC based on a Python implementation of the algorithms. We use $n = 3$ computing parties and $t = 1$. For the `sharing` algorithm we use mean value zero and variance 1000 for the Gaussian distributed shares.

We use the RSE between the result from Algorithm F.5, $\hat{x}_k^{(\text{priv})}$, and (F.46), $\hat{x}_k$, which at time k is defined as

$$\text{RSE}_k = \sqrt{(\hat{x}_k^{(\text{priv})} - \hat{x}_k)^2}, \quad \text{for } k = 1, 2, \dots.$$
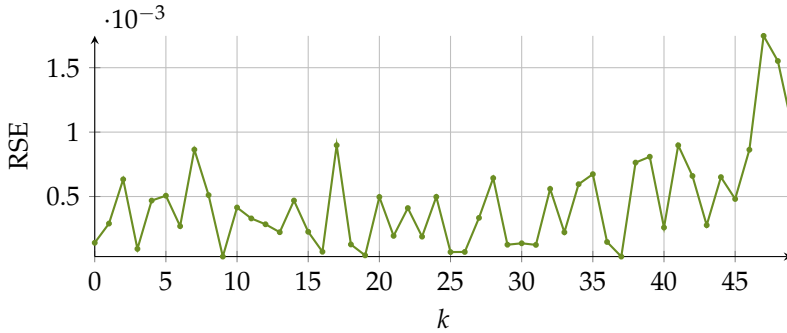
Fig. F.7: RSE between simulated result from Algorithm F.5 and (F.46).

In Fig. F.7 it is seen that the difference in result from the privacy preserving solution and the non-private solution lies around the third decimal. In comparison, the difference for the solution in [12] lies before the decimal point.

Regarding the complexities, as seen, Algorithm F.5 uses 12 multiplications and one inversion, which amounts to 27 interactive operations, independent of the dimension of the matrices. [12] does not provide the complexity for their solution, thus, we provide here an underestimation of the number of interactive operations which lies around $10M + l + 1$, where M is the dimension of the matrix R in (F.46) and l is the number of bits used to represent the numbers (which in the simulations by [12] is at least 24 bits).

7 Conclusion

The paper presents a real number secret sharing scheme that bypasses the requirements on integer shares and modular arithmetic which is used in state-of-the-art secure multiparty computation schemes. That the scheme does not use modular arithmetic, makes it very useful for computations directly on shares including division. The trade-off is that the proposed scheme is not perfectly secure, however, we show that the information leak can be upper bounded and demonstrate with examples how small the leak is. We see the proposed scheme with its high level accuracy and privacy properties and its low communication complexity as offering a relevant trade-off between privacy of the distributed computations and practicality of the scheme. Numerical evaluations of the proposed scheme as well as simulations of the scheme to perform Kalman filtering with privacy preservation verify the theoretic results.

Acknowledgement

References

[1] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Engineering Practice*, vol. 67, pp. pp. 13 – 20, 2017.

[2] Q. Li, I. Cascudo, and M. G. Christensen, "Privacy-preserving distributed average consensus based on additive secret sharing," in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[3] K. Tjell and R. Wisniewski, "Private aggregation with application to distributed optimization," *IEEE Control Systems Letters*, vol. 5, pp. pp. 1591–1596, 2021.

[4] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *Automatica*, vol. 96, pp. pp. 314 – 325, 2018.

[5] Q. Li and M. G. Christensen, "A privacy-preserving asynchronous averaging algorithm based on shamir's secret sharing," in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[6] K. Tjell and R. Wisniewski, "Privacy preservation in distributed optimization via dual decomposition and admm," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 7203–7208.

[7] R. Cramer, I. B. Damgaard, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*, 1st ed.   Cambridge University Press, 2015.

[8] J. Lint and T. Djukic, *Applications of Kalman Filtering in Traffic Management and Control*, 10 2012, pp. 59–91.

[9] K. Gordon, "The multi-state kalman filter in medical monitoring," *Computer Methods and Programs in Biomedicine*, vol. 23, no. 2, pp. pp. 147–154, 1986.

[10] M. Nasseri, A. Moeini, and M. Tabesh, "Forecasting monthly urban water demand using extended kalman filter and genetic programming," *Expert Syst. Appl.*, vol. 38, no. 6, p. pp. 7387–7395, Jun. 2011.

[11] Y. Song, C. X. Wang, and W. P. Tay, "Privacy-aware kalman filtering," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4434–4438.

[12] F. J. Gonzalez-Serrano, A. Amor-Martın, and J. Casamayon-Anton, "State estimation using an extended kalman filter with privacy-protected observed inputs," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2014, pp. 54–59.

References

[13] J. Le Ny and G. J. Pappas, "Differentially private filtering," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. pp. 341–354, 2014.

[14] K. Tjell, I. Cascudo, and R. Wisniewski, "Privacy preserving recursive least squares solutions," in *2019 18th European Control Conference (ECC)*, 2019, pp. 3490–3495.

[15] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted mpc for linear constrained systems," *IEEE Control Systems Letters*, vol. 2, no. 2, pp. pp. 195–200, 2018.

[16] Q. Li, I. Cascudo, and M. Christensen, "Privacy-preserving distributed average consensus based on additive secret sharing," in *EUSIPCO 2019 - 27th European Signal Processing Conference*, ser. Proceedings of the European Signal Processing Conference. IEEE Signal Processing Society, Sep. 2019.

[17] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *Financial Cryptography and Data Security*, R. Sion, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 35–50.

[18] O. Catrina and S. de Hoogh, "Secure multiparty linear programming using fixed-point arithmetic," in *Computer Security – ESORICS 2010*, D. Gritzalis, B. Preneel, and M. Theoharidou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 134–150.

[19] M. Aliasgari, M. Blanton, Y. Zhang, and A. Steel, "Secure computation on floating point numbers," in *In NDSS*, 2013.

[20] V. Dimitrov, L. Kerik, T. Krips, J. Randmets, and J. Willemson, "Alternative implementations of secure real numbers," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 553–564.

[21] L. Bai and Y. Lan, "A homomorphic arithmetic scheme on real number with fixed precision," *AIP Conference Proceedings*, vol. 1839, no. 1, 2017.

[22] K. Gai, M. Qiu, Y. Li, and X. Liu, "Advanced fully homomorphic encryption scheme over real numbers," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2017, pp. 64–69.

[23] J. Basilakis and B. Javadi, "Efficient parallel binary operations on homomorphic encrypted real numbers," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. pp. 507–519, 2021.

[24] A. Dibert and L. Csirmaz, "Infinite secret sharing – examples," *Journal of Mathematical Cryptology*, vol. 8, no. 2, pp. pp. 141 – 168, 2014.

[25] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. pp. 612–613, Nov. 1979.

References

[26] J. Justesen and T. Hoholdt, *A Course in Error-Correcting Codes (EMS Textbooks in Mathematics)*. European Mathematical Society, 2004.

[27] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. USA: Wiley-Interscience, 1991.

[28] Q. Li, R. Heusdens, and M. G. Christensen, "Privacy-preserving distributed optimization via subspace perturbation: A general framework," *IEEE Transactions on Signal Processing*, vol. 68, p. pp. 5983–5996, 2020.

[29] D. Beaver, "Efficient multiparty protocols using circuit randomization," vol. 576, 08 1991, pp. 420–432.

[30] A. Ben-Efraim, M. Nielsen, and E. Omri, "Turbospeedz: Double your online spdz! improving spdz using function dependent preprocessing," in *Applied Cryptography and Network Security*, R. H. Deng, V. Gauthier-Umaña, M. Ochoa, and M. Yung, Eds. Cham: Springer International Publishing, 2019, pp. 530–549.

[31] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. pp. 35–45, 03 1960.

AALBORG UNIVERSITY PRESS