Secure and efficient outsourced computation protocols for linear algebra

David Lucas

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : **Mathématiques et Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

David LUCAS

Thèse dirigée par **Clément PERNET**, Université Grenoble Alpes et codirigée par **Jean-Guillaume DUMAS,** Université Grenoble Alpes

préparée au sein du **Laboratoire Jean Kuntzmann**
dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Protocoles de calculs externalisés efficaces et sécurisés pour l'algèbre linéaire

Secure and efficient outsourced computation protocols for linear algebra

Thèse soutenue publiquement le **10 juillet 2020**,
devant le jury composé de :

Monsieur Clément PERNET
Maître de conférences, Université Grenoble Alpes, Directeur de thèse
Monsieur Gilles VILLARD
Directeur de recherche, CNRS, Rapporteur
Madame Melek ÖNEN
Maître de conférences, EURECOM, Rapporteure
Monsieur Jean-Guillaume DUMAS
Professeur des Universités, Université Grenoble Alpes, Co-directeur de thèse
Madame Marie-Laure POTET
Professeur des Universités, Grenoble INP, Examinatrice
Présidente du jury
Monsieur Vincent NEIGER
Maître de conférences, Université de Limoges, Examinateur

PhD Thesis

Secure and Efficient Outsourced Computation Protocols for Linear Algebra

David Lucas

August 24, 2020

Under the direction of:
Jean-Guillaume Dumas and Clément Pernet

Contents

List of Tables

List of verification protocols

List of multiparty protocols

Chapter 1

Introduction

Contents

Presentation and outline of this thesis

Scientific computation. Computation always played a major role in the history of sciences. The development of computers from the 1940s allowed for computations which were faster, easier to reproduce and less prone to errors. The diversity of applications in which computers could be used led to the development of various computation fields: simulations of physical systems could be performed using *numerical computation*, which aims at finding an approximation of the solution to a given problem; while domains such as cryptanalysis or experimental mathematics led to exact computation which aims at finding exact solutions to the problems it tackles. Through the second half of the twentieth century, the need to perform simulations of physical systems became dominating, which strongly influenced the development of architectures, software and algorithmics. Floating point arithmetic, for instance, became a standard in computers in 1985 (with IEEE 754). However, exact computation is essential to a large number of research fields in scientific computing. Cryptography, for instance, widely uses integer numbers or finite field elements and requires highly efficient exact computation algorithms. In some fields it is not possible to approximate a solution: this is the case for instance in experimental mathematics, in graph theory, number theory or combinatorics, which makes the use of efficient exact computation extremely important in these fields. Finally, formal methods also make extensive use of exact computation for instance with computer assisted proof. All these applications make the development of efficient exact computation methods paramount to a wide number of fields.

Linear algebra in computation. Amongst the many tools available to perform efficient computations, linear algebra presents several major advantages: the data structures (vectors, matrices) it uses and the operations performed are relatively easy to handle, allowing to develop algorithms which can be more easily fine tuned and optimised. Linear algebra also enables highly regular computations, allowing to take advantage of efficient parallel programming techniques such as vectorisation. Finally, linear algebra allows for high intensity computation: some operations, such as matrix multiplication have an asymptotically larger time complexity than their space complexity (cubic vs. quadratic). By grouping operations, one can ensure that numerous computations are performed on the same block of data, thus amortising the costly access to computer's memory. Because of these advantages, linear algebra is a central tool in scientific computation, both in fields using exact algebra such as algebraic geometry, graph theory, or cryptography and in numerical computation, for solving partial differential equations or optimisation problems. Thus, any improvement to one of its fundamental operations can be carried over to a large amount of applications, making progresses in linear algebra important to a large number of research fields.

High performance computing. The ability to perform computations as fast as possible, on as many data as possible is paramount in a wide array of applications: in experimental mathematics, for instance, in which being able to test a conjecture on a set of examples as large as possible is an extremely desirable feature, or in cryptography in which the security of the primitives directly depends on the computational power of the fastest available hardware. Many numerical applications require high performance computing, such as weather forecast models or aerospace simulations. A lot of effort has thus been put in developing computing infrastructures and software able to handle such large problems and high performance computing remains an extremely active field to this day.

Computing infrastructures. Infrastructures able to tackle large computational tasks vastly evolved over the years. From the 1960s to the 1980s, the fastest infrastructures available were local machines, such as computing servers or clusters. The democratisation of the Internet and the personal computer from the 1990s has seen the emergence of decentralised infrastructures with the development of peer and volunteer computing, which connects geographically spread resources, owned by a myriad of peers. This new approach to computing gained popularity thanks to its cost model, redistributing the overall price of the infrastructure amongst its peers; its ease of use (as maintaining complex hardware was no longer necessary) and its fast first successes, such as the Mersenne[1] prime number search. Finally, in the mid-2000s, very large companies started to buy state-of-the-art computing infrastructures and developed a pricing system to rent them by the hour to clients all over the Internet. These clients were able to access this hardware via a single transparent interface, allowing them to easily perform tasks on appropriate infrastructures. This model, known as cloud computing spread rapidly thanks to its versatility and ease of use. This evolution, from local, controlled computing hardware to global, outsourced or distributed over many geographically spread machines gave birth to new questions related to confidentiality, security and safety. The question of trust is central to this new model: as users cannot blindly trust the entities who perform computations for them, how can they efficiently verify their outsourced computations? Similarly, users who need to perform outsourced computations on confidential data need to be sure that their data will not be leaked to others. This last application is at the centre of many contemporary applications, such as confidential organ donor/recipient matching (see for instance Wueller et al. (2018), or the EAGER NSF award[2]) or blind auctions in goods trade (Bogetoft et al., 2009). In this context, designing efficient protocols both in terms of communication volume over the network and in client computational cost is essential, as users only have access to limited computational power and the liberal use of expensive mathematical tools could quickly lead to unbearably large client cost.

[1] https://www.mersenne.org/

[2] https://www.nsf.gov/awardsearch/showAward?AWD_ID=1646999

Our thesis and this document's structure. While the spread of third party-owned computing hardware is a fairly recent phenomenon, ensuring confidentiality and correctness concerns of computations performed on outsourced devices is not. Researchers such as L. Babai or S. Goldwasser started to work on outsourced computation verification as early as 1985 and A. Yao's *Millionaire's Problem* from 1982 is considered as the seminal work on secure multi-party computation. Numerous research followed, both on proving the validity or the privacy of outsourced computation. However, many of these research works aim at developing generic tools verifying any computation in a given class of problems. Such genericity usually comes with a heavy price: even if their asymptotic complexities are excellent, they often hide very large constant factors which undermine their practicality. Designing dedicated protocols, tailored to tackle specific problem has thus gained some popularity since 2011 Kaltofen et al. (2011), but a lot of work remain to be done in this area. In this context, we defend the following thesis:

Outsourced exact linear algebra computations do not have to rely on users' trust: we can now verify and ensure confidentiality of its main algorithms.

In this chapter, we formally introduce the notions summarized above. We define the setting of verification protocols for outsourced computation and secure multiparty computation, alongside the metrics used to evaluate their efficiency. We also propose an comparison between the two aforementioned approaches, generic protocols against dedicated ones and give an overview of existing dedicated protocols and assess their efficiency.

In Chapter 2, we present new dedicated protocols for the verification of the computation of a matrix invariant called the rank profile matrix. We achieve better computational and communication volume complexities than generic protocols on the same problem. Using some of the protocols we designed for this result, we also improve on the verification of classical linear algebra properties, matrix determinant and signature.

In Chapter 3, we still focus on verification protocols, but this time for univariate polynomial matrices. We propose dedicated protocols to verify numerous properties of such polynomial matrices, where some verified objects are specific to modules while some others also carry over the vector space whose coefficient domain is the set of rational fractions. In this context, we propose a central tool in a protocol to verify that a vector belongs to a module spanned by a polynomial matrix.

Finally, in Chapter 4, we focus on the privacy aspect with secure multiparty computation. We propose a secure protocol for the shared computation of matrix multiplication based on the recursive Strassen-Winograd algorithm. This new protocol allows us to improve on the state of the art for the communication volume required for multiparty matrix multiplication and has practical applications, as for instance in private trust evaluation.

Notations

Fields and rings We use F to indicate an arbitrary field, $\mathsf{F}[x]$ for the ring of polynomials in one variable x with coefficients in F, and $\mathsf{F}(x)$ for the field of rational fractions, i.e., the fraction field of $\mathsf{F}[x]$. The ring of $m \times n$ matrices, for example over $\mathsf{F}[x]$, is denoted by $\mathsf{F}[x]^{m\times n}$.

Vectors and matrices We use bold font and small letters to represent vectors, as $\boldsymbol{v}$ and bold font and capital letters for matrices $\boldsymbol{A}$. Vectors are considered as columns by default, which means $\boldsymbol{v} \in \mathsf{S}^n$ will be equivalent to $\boldsymbol{v} \in \mathsf{S}^{n\times 1}$ and that $\boldsymbol{v}^T$ is a row vector. Regarding matrices, $\boldsymbol{A}_{i,j}$ represents the element on the i-th row, j-th column of $\boldsymbol{A}$, while $\boldsymbol{A}_{i,*}$ represents the i-th row of $\boldsymbol{A}$. Given a set of row indices $\mathcal{I}$ and column indices $\mathcal{J}$, $\boldsymbol{A}_{\mathcal{I},\mathcal{J}}$ denotes the submatrix extracted from $\boldsymbol{A}$ in these rows and columns.

Asymptotic complexity bounds We use the "soft-oh" notation $\widetilde{O}(\cdot)$ to give asymptotic bounds hiding logarithmic factors. Precisely, for two cost functions f, g, having $f \in \widetilde{O}(g)$ means that $f \in O(g\log(g)^c)$ for some constant $c > 0$.

We write ω for the exponent of matrix multiplication over F, so that any two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathsf{F}^{n\times n}$ can be multiplied using $O(n^\omega)$ field operations; we have $2 < \omega \leqslant 3$ and one may take $\omega < 2.373$ (Coppersmith and Winograd, 1990; Le Gall, 2014).

For a given matrix $\boldsymbol{A}$, we denote by $\mu(\boldsymbol{A})$ the worst case arithmetic cost of multiplying $\boldsymbol{A}$ by a vector.

Protocols In protocols, S is always a finite subset of the base field F which we use to sample field elements uniformly and independently at random. If F is finite and sufficiently large, one can use $\mathsf{S} = \mathsf{F}$. If $\#\mathsf{F}$ is too small, then one may use a field extension, causing up to a logarithmic factor increase in the computation and communication costs.

Following (Dumas and Kaltofen, 2014; Dumas et al., 2016, 2017b) we use the notation $x \stackrel{?}{=} y$ as a placeholder for

$$\textbf{If } x \neq y \textbf{ then} \text{ abort and report failure}$$

to improve the brevity and readability of the protocols. Similarly, we use $x \stackrel{?}{<} y, x \stackrel{?}{\geqslant} y$, $U \stackrel{?}{\subseteq} V$, etc. to check inequalities and set inclusion.

Random sampling We denote by

$$\alpha \xleftarrow{\$} \mathsf{S} \quad \text{and} \quad \boldsymbol{v} \xleftarrow{\$} \mathsf{S}^{n\times 1}$$

respectively the actions of drawing a field element uniformly at random from S and of drawing a vector of n field elements uniformly and independently at random from S.

Encryption and decryption Given a player P, we denote by pk_P their public key and by sk_P their secret key when using an asymmetric cryptosystem. The cleartext space associated with this player will be denoted by $\mathcal{M}_P$ and the ciphertext space by $\mathcal{C}_P$.

Given some scalar u, we denote by $\{u\}_{pk_P}$ the encryption of u using the public key of P. The action of encrypting (resp: decrypting) an element $u \in \mathcal{M}_P$ (resp: $v \in \mathcal{C}_P$) will be denoted by $E(u)_{pk_P}$ (resp: $D(v)_{sk_P}$).

1.1 Verification protocols for outsourced computation

1.1.1 Overview

Let us consider the following situation: a client, with limited computational power, wants to perform some heavy computation. As it would take forever for our client to run this computation by themselves, they will instead rent a server over the Internet and ask this server to do the heavy work. After some time, the server sends a result to the client. While this scenario seems to solve the issue at hand, it contains a major flaw: as the client has absolutely no control over the remote server, they would like to verify the result the server sent back. However, their limited computational power forces them to use specific, efficient verification protocols. In this section, we will describe the so-called interactive proof systems, allowing our client to efficiently verify the result of an operation which has been performed by an untrusted resource. We will specifically present different approaches for verifiable computation, focusing on linear algebra problems, and discuss the metrics that can be used to evaluate the efficiency of such proof systems.

1.1.2 Interactive proof systems and verification protocols

Interactive proof systems

A proof system, usually consisting in a theorem and a proof, allows to verify the theorem by checking the validity of the proof. An interactive proof system, for its part, models an exchange between two parties, the Prover whose goal is to provide the proof, and a Verifier who will check its veracity. The general flow of an interactive proof system is as follows:

1. at the beginning, the Prover and the Verifier share the knowledge of the result of a computation, which they have to verify,
2. the Verifier then sends a Challenge to the Prover, usually consisting of some uniformly sampled random values, to which
3. the Prover replies with a Response, used by the Verifier to ensure the validity of the commitment.

In some cases, several additional rounds of Challenge/Response might be necessary for the Verifier to accept an answer. This interactive approach has been presented by two independent teams in 1985: (Babai, 1985) introduced *Arthur-Merlin* protocols (from the names he respectively affected to the Verifier and the Prover in his paper), where only one round of Challenge/Commitment was allowed and where the random values generated by the Verifier had to be public. (Goldwasser et al., 1985) proposed very similar proof systems, except they allowed several rounds of Challenge/Commitment and they forced the random choices of the Verifier to remain hidden. It was later

proved than hiding or revealing the Verifier's random choices had no impact on the proof (Goldwasser and Sipser, 1986; Babai and Moran, 1988).

Proof systems must satisfy two fundamental properties, **completeness** which states that the Verifier should not reject a true statement provided by the Prover and **soundness** which states that the Verifier should not accept a false statement. More precisely, a protocol is said complete if the probability that the Verifier rejects a true statement can be made arbitrarily small. If the Verifier never rejects such a statement, the protocol is said perfectly complete. Similarly, a protocol is sound if the probability that the Verifier accepts a false statement can be made arbitrarily small, and is perfectly sound if the Verifier never accepts a false statement. As all the protocols presented in this thesis will be perfectly complete, we will describe them just as complete. Note that in practice, a soundness of 1/2 is enough, as the protocol can be repeated several times, *de facto* reducing the probability that the Verifier accepts a wrong statement: a protocol for which the Verifier has probability 1/2 to accept a wrong statement, repeated only five times yields a probability of $1/2^5 \approx 0.03$ for the Verifier to overall accept this wrong statement. These iterations can always be performed in parallel, so that there is a linear scaling in the communication, Verifier and Prover costs (see next section for more details on these metrics), but not in the number of rounds in the protocol.

Finally, we briefly present a few results on problem solving by interactive proofs systems. It has been proven by (Furer et al., 1989) that interactive proofs can be transformed in interactive proof with perfect completeness. They also show that only problems in $\mathcal{NP}$ have interactive proofs with perfect soundness. (Shamir, 1992) proved that the class of problems solvable by an interactive proof ($\mathcal{IP}$) was equal to $\mathcal{PSPACE}$. Finally, the fundamental PCP theorem states that every decision problem from the $\mathcal{NP}$ class has probabilistically checkable proofs, with the verification algorithm using a constant number of bits from the proof and a logarithmic amount of random bits (Babai et al., 1991; Arora and Safra, 1992; Arora et al., 1998).

Verification protocols

In this thesis, we will focus on dedicated *verification protocols*, a particular representation of such proof systems. Their initial name, *certificate* carries different meanings in particular in linear algebra: a certificate often is an algorithm proving the correctness of the output of a randomized algorithm, thus allowing to "transform" a Monte-Carlo algorithm, into a Las Vegas one. Monte-Carlo are randomized algorithms which might silently fail with a certain probability, producing a wrong output; while Las Vegas are randomized algorithms which cannot fail but that use random choices while running which leads to impredictable runtimes. Such certificates have been developed, for instance by (Storjohann, 2009; Mulders and Storjohann, 2004) based on early work by (Giesbrecht et al., 1998). However, the name *certificates* implied that they are static objects. With their generalisation as interactive proof systems does not allow this terminology anymore, we will instead use the name *verification protocol* throughout this thesis.

Here, we will focus on verification protocols as Prover/Verifier protocols, where the

Prover is modelled as a computationally unbounded adversary and where the Verifier has a typically limited amount of computational power.

Generic techniques briefly introduced above prove the evaluation of a boolean or an arithmetic circuit and require that the Verifier either knows the circuit itself or computes certain properties of it. More details on these techniques can be found in Section 1.1.5. The original definition of certificates, was proposed in (Kaltofen et al., 2011, 2012; Dumas and Kaltofen, 2014) and follows:

Definition 1.1.1. *An interactive certificate for a problem that is given by input/output is an input-dependent data structure and an algorithm that computes from that input and its certificate the specified output, and that has lower computational complexity than any known algorithm that does the same when only receiving the input. Correctness of the data structure is not assumed but validated by the algorithm in the adversary-verifier model.*

This definition is not suitable for the verification protocols proposed in this thesis: it states that the Verifier has no *a priori* knowledge of the output of the algorithm. In Chapter 3, we deal with matrices over polynomial rings, which are large data structures. In this particular context, considering that the Prover sends the output of the computation to the Verifier automatically leads to large communication bounds (one of the key efficiency metrics of such certificates, see Section 1.1.3 for more details) which will dominate the responses of the Prover. To circumvent this issue, we consider the output of the computation as an input of the verification protocol, which leads us to this new definition:

Definition 1.1.2. *A verification protocol for a problem $\mathcal{P}$ given by input/output $\mathcal{I}/\mathcal{O}$ is a boolean algorithm which states whether $\mathcal{O}$ is a valid solution for $\mathcal{P}$ on input $\mathcal{I}$. It has a lower computational complexity that any known algorithm that solves $\mathcal{P}$. The verification is done in the adversary-verifier model, in which an untrusted entity provides a set of elements used by the verification protocol to state whether $\mathcal{O}$ is a valid solution for $\mathcal{P}$ or not.*

Such verification protocols can either be non interactive (the Prover sends data to the Verifier only once, and this data is sufficient to verify and/or compute the output), or interactive (following the interaction flow presented at the beginning of this section). Protocol 1.1 gives the generic template we will use in this thesis to represent verification protocols.

Note that our verification protocols might resemble another category of protocols, called Σ-Protocols (Damgård, 2010, Section 2), which present the same communication structure. However, Σ-Protocols are closely related to proofs of knowledge, as such protocols always prove that the Prover knows the result that it verifies (Damgård, 2010, Theorem 1). As the verification protocols presented in this thesis have the data structure to verify as part of the public input, there is no proof of knowledge to provide here.

Protocol 1.1: Template for verification protocol

Public: An algorithm $\mathcal{A}$, a set of input I to $\mathcal{A}$ and a data structure O
Certifies: O is the output to $\mathcal{A}(I)$

Prover		Verifier
1.		Generate a challenge c_1 from public elements
	$\xleftarrow{c_1}$	
2. Compute an answer a_1 to c_1		
	$\xrightarrow{a_1}$	
	$\vdots$	
3.		$O \stackrel{?}{=} \mathcal{A}(I)$ from the set of answers $(a_1, \ldots, a_n)$

1.1.3 Evaluation of the efficiency of a verification protocol

When designing verification protocols, one should be careful of several metrics that are used to assess their efficiency, namely:

Communication: this can be measured using two elements: the volume of data exchanged throughout, without taking into account the input and output of the computation, and the number of Challenge/Response rounds required by the protocol. This duality is necessary to reflect real world constraints: users with a good bandwidth but a poor latency would indeed favour verification protocols with a small (ideally constant) number of rounds. Communication volume can either be measured in number of bits or in field elements. In this thesis, protocols often work over finite fields in which these two units are equivalent. For protocols presented in Chapter 3 which are over a polynomial ring, we will measure communication in field elements. Finally, note that a recent result (Reingold et al., 2016, Theorem 1) shows that any problem which can be solved in bounded space and polynomial time (in the size of the input) can be verified by complete and sound interactive proofs in constant rounds and polynomial communication volume;

Verifier cost: the worst-case number of arithmetic operations performed by the Verifier in the protocol, regardless of the data the Prover sent. Another element to look at for the Verifier efficiency is the minimal size of the field from which the randomly generated challenges are sampled: soundness usually depends on the input field size and some protocols require this field to have a size which depends on the

input data size. Note that the cost of an honest Verifier depends only on the public information — that is, the parameters of the statement being verified — and even a malicious Prover cannot cause the Verifier to do more work than this. However, in the case of very large (or even infinite) fields, the Prover may essentially execute a denial of service attack by sending arbitrary large field elements in the protocol. In this thesis, our analysis is generic in terms of field operations, is not able to capture this weakness, and does not attempt to address it;

Prover cost: the number of operations performed by an honest Prover throughout the protocol. Note that the type of algorithm the Prover runs is also to consider: if there is no deterministic algorithm for the Prover, the random algorithm should be Las Vegas as a Monte Carlo algorithm could lead an honest Prover to handle wrong responses to the Verifier.

As a side note, verification protocols could also be used locally to verify the implementation of a given algorithm. This presents several advantages over the usual testing methods, which often rely on testing against trusted implementations or comparing results against a pre-computed set of inputs/outputs. In this context, there is no longer a Prover and the communication cost metric is replaced by a memory cost: the number of rounds can be seen as the number of memory accesses required to perform the protocol while the volume is the additional memory space required by the verification protocol.

1.1.4 From interactive to non interactive protocols and public verifiability

In practice, it might not always be possible for the Verifier to interact directly with the Prover: for instance, a Verifier would like to check the veracity of an already published result of the computation but has no access to a server whatsoever. Then, it is possible to transform any interactive verification protocol into a non-interactive one, using for instance Fiat-Shamir heuristic (Fiat and Shamir, 1987): random values produced by the Verifier are replaced by cryptographic hashes of the input and the previous messages, and the Prover publishes once both the Commitment and Response to the de-randomized challenge. This also solves the issue of *public verifiability*: the fact that anyone can verify and be convinced by the Prover's answers. Indeed, if only the communications between the Prover and the Verifier were made public, one could argue that the Prover and the Verifier might be colluding and crafted the challenges. Using Fiat-Shamir heuristic, doing that would be as hard as breaking the cryptographic hash function. Some efficient dedicated protocols that focus on public verifability do exist : as they are not the main focus of this thesis, we will not focus on them here. Protocols for linear algebra could for instance be found in Elkhiyaoui et al. (2016).

1.1.5 Paradigms for verification protocols

In this section, we go through the major approaches developed for verification protocols. We start by giving an overview of what we call *generic approaches*, whose goal is to propose multi-purpose protocols which are able to certify a wide range of computations. These approaches usually carry large constant factors on the Prover or the Verifier computation time. We then focus on *algorithm-based approaches*, whose goal is to develop protocols for one specific computation, hence trading generality for better efficiency. Finally, we present one *intermediate approach*, which is able to certify any linear algebra computation reducing to matrix multiplication.

Generic approaches

Circuit verification techniques These generic approaches come from (Goldwasser et al., 2008) work, in which they propose a protocol able to verify any problem that can be solved on a parallel computer, that is any problem from the $\mathcal{NC}$ or $\mathcal{AC}$ complexity class.

In a nutshell, the idea here is that the Prover and the Verifier agree on a multi-layered arithmetic circuit $\mathcal{C}$. The Prover first commits a claim on the value of the output layer of this circuit. The Verifier cannot verify this claim without evaluating the circuit, which is very expensive and thus what they want to avoid. What the Verifier will do instead is to engage in an iterative verification sequence, in which a claim on the values at the output of each layer of the circuit can be reduced to a claim about the values on the previous layer of the circuit. The Verifier will use a sum-check protocol (Lund et al., 1992) to perform this reduction, and proceeding backwards from the output layer to the input layer of $\mathcal{C}$, the Verifier eventually needs to only verify a claim on the input of the circuit.

Initially, considering inputs of size n and a circuit of size $S(n)$, depth d, with Prover runtime for this protocol polynomial in $S(n)$ and verifier time $\widetilde{O}(n + d\log(S(n)))$. This has been refined by (Cormode et al., 2012), who managed to achieve a better Prover in $O(S(n)\log(n))$ time. (Thaler, 2013) managed to reduce the Prover time complexity to $O(S(n))$, but only for regular circuits.

QAP-based approach Another approach has been proposed by (Parno et al., 2016) with a system called *Pinocchio* but this time based on converting a circuit into a structure called a quadratic arithmetic program (QAP). In a nutshell, a QAP is a representation of the circuit which contains three sets $\mathcal{L}, \mathcal{R}, \mathcal{O}$ of size n of polynomials and a target polynomial T of degree d. These sets respectively encode the left input of every gate in the circuit, their right input and their output. The target polynomial is associated to the function the circuit is supposed to evaluate and is used to verify that the circuit indeed computes the output of this function. To be more specific, to verify if a set of constants $(c_1, \ldots, c_n)$ (called an assignment) satisfies the QAP, one needs to compute three polynomials $L = \Sigma_{i=1}^{n} c_i \cdot \mathcal{L}_i, R = \Sigma_{i=1}^{n} c_i \cdot \mathcal{R}_i$ and $O = \Sigma_{i=1}^{n} c_i \cdot \mathcal{O}_i$ and

$P = LR - O$. If T divides P, meaning if there exists a polynomial H such that $P = HT$, then the assignment satisfies the QAP. In the context of verifiable computation, this can be applied as follows: the Prover and the Verifier agree on a circuit to evaluate and on the target polynomial T and the Prover computes the four polynomials L, R, O and H. The Verifier then samples a random evaluation point s which they send to the Prover. The Prover replies with the evaluation of their four polynomials at s and the Prover just needs to check that the equality $L(s)R(s) - O(s) = T(s)H(s)$ holds. The soundness comes from DeMillo/Lipton/Zippel-Schwartz lemma which states that the probability of $P(s) = H(s)T(s)$ with $P \neq HT$ is small. However, this is not enough to prove that the assignment indeed satisfies the QAP: the former verifications only guarantee that the polynomials L, R and O have degree at most d and verify $L \times R - O = T \times H$, but not necessarily that they were produced from the assignment. In order to verify that the Prover did not just forge suitable polynomials, the Verifier will create a new polynomial $F = L + X^{d+1} \times R + X^{2(d+1)} \times O$. This new polynomial has been constructed such that its coefficients in degree less than $d + 1$ are exactly the coefficients of L, the next$d + 1$ coefficients are exactly those of R, and the last $d + 1$ are the ones of O. Now, combining the polynomials from the sets $\mathcal{L}, \mathcal{R}$ and $\mathcal{O}$ in a similar manner gives a set of n new polynomials $\mathcal{F}$ such that $F_i \in \mathcal{F} = L_i + X^{d+1} \times R_i + X^{2(d+1)} \times O_i$ for some $i \in \{1, \ldots, n\}$. Now, if F is a linear combination of the F_i polynomials, it means that L, R and O were indeed produced from the assignment. This can be verified by random evaluation: the Verifier picks a random evaluation point s', and sends $s' \times F_1(s), \ldots, s' \times F_n s$ to the Prover who has to reply with $s' \times F(s)$. It they succeed, it means they know how to write F as a linear combination of the F_i, which implies that the L, R and O indeed come from the assignment.

It has been proven by (Gennaro et al., 2013) that any arithmetic circuit can be converted into a QAP whose size (the number of polynomials in each set) depends on the number of multiplication gates and input/output elements in the circuit to transform. This representation is quite powerful, as addition and multiplication by a constant gates can be considered free (they do not impact the size of the QAP) and the correctness of the QAP can be checked by performing cheap polynomial operations. This approach has been further improved in 2015 by (Costello et al., 2015) whose Geppetto compiler improves on Pinocchio by enabling more efficient Provers.

Linear algebra computations Both these approaches can be used to compute classical linear algebra properties, like the rank or the determinant of a matrix, or operations like the matrix-matrix product. However, the additional logarithmic factors take a toll on the Prover time. This cost is illustrated, for instance in (Walfish and Blumberg, 2015) which Figure 5 shows the Prover overhead on two computational tasks (matrix multiplication and PAM clustering - similar to k-means computation) and exhibits a massive overhead (from 1.5 to 9 orders of magnitude) for generic techniques against a native C implementation.

Algorithm-based approaches

In this section, we go through the major verification protocols using algorithm-based techniques to verify linear algebra properties.

We start by the first linear algebra protocol, due to (Freivalds, 1979), which allows to verify the computation of a matrix-matrix product. Given two matrices $\boldsymbol{A}, \boldsymbol{B}$ and a matrix $\boldsymbol{C}$, such that $\boldsymbol{A} \times \boldsymbol{B} = \boldsymbol{C}$, the idea is to perform a right projection using a random vector. If the projection of $\boldsymbol{A} \times \boldsymbol{B}$ and the projection of $\boldsymbol{C}$ are equal, then, with high probability, $\boldsymbol{A} \times \boldsymbol{B} = \boldsymbol{C}$. This verification protocol is given in Protocol 1.2.

Protocol 1.2: Freivalds

Public: $\boldsymbol{A} \in \mathsf{F}^{m \times n}, \boldsymbol{B} \in \mathsf{F}^{n \times \ell}, \boldsymbol{C} \in \mathsf{F}^{m \times \ell}$
Certifies: $\boldsymbol{C} \stackrel{?}{=} \boldsymbol{AB}$

Prover	Verifier
1.	$\boldsymbol{v} \xleftarrow{\$} \mathsf{S}^{\ell \times 1}$ $\boldsymbol{A}(\boldsymbol{B}\boldsymbol{v}) \stackrel{?}{=} \boldsymbol{C}\boldsymbol{v}$

Lemma 1.1.3. *Protocol 1.2 is a perfectly complete and probabilistically sound non interactive protocol. It has Verifier cost* $\mu(\boldsymbol{A}) + \mu(\boldsymbol{B}) + \mu(\boldsymbol{C})$. *The probability that the Verifier incorrectly accepts is at most* $1/\#S$.

In Freivalds' original version, the random projection vector $\boldsymbol{v}$ contained only 1 and -1s, hence yielding a failure probability of $1/2$. (Kimbrel and Sinha, 1993, algorithm 4) proposed a variant to Freivalds' protocol, in which, instead of picking a random $\{-1, 1\}$ vector, the Verifier randomly sampled a single element x from S and used $\boldsymbol{v} = (1, x, x^2, \ldots, x^{n-1})$ as the projection vector instead. This variant requires only $\log(n)$ random bits, as S needs at least n elements. Note that in what follows, we will use Freivalds' protocol as proposed in Protocol 1.2 and not original $\{-1, 1\}$ version or the Kimbrel-Sinha variant.

For the rest of this section, we will give an overview of existing verification protocols using algorithm-based strategies for verifying classical linear algebra properties. Information of the efficiency of these protocols has been summarized in Table 1.1, and improvements over the state of the art alongside the new verification protocols developed in Chapter 2 are presented in Table 2.2.

Matrix rank A first matrix rank dedicated verification protocol was given in 2011 by (Kaltofen et al., 2011, Theorem 2) for square integer matrices, based on the existence of an LU decomposition for the input matrix modulo a random prime. This protocol has $O(n^3)$ communication cost and $O(n^2)$ verification cost. More recently, (Dumas

and Kaltofen, 2014, Corollary 3, Theorem 5) proposed a verification protocol for the rank of both integer matrices and matrices over a finite field. For a matrix $\boldsymbol{A} \in \mathsf{F}^{m\times n}$, their protocol asks the Prover to commit the rank r of $\boldsymbol{A}$, compute two projection matrices, $\boldsymbol{I} \in \mathsf{F}^{r\times m}$ and $\boldsymbol{J} \in \mathsf{F}^{n\times r}$ such that $\boldsymbol{IAJ}$ is non singular, which can be verified, certifying that r is a lower bound on the rank of $\boldsymbol{A}$. They then propose a protocol for an upper bound on the rank of a matrix, which applied to r effectively certifies the rank of $\boldsymbol{A}$. (Eberly, 2015) proposes a variant of these protocols in which the Prover is asked to certify the lower bound on the rank r by recovering a linear combination on r independent columns on the matrix. The upper bound is then proved by the Prover's ability to generate a random vector of the image of $\boldsymbol{A}$ using a linear combination of r columns of $\boldsymbol{A}$. The previous two verification protocols have roughly equivalent complexities, with (Dumas and Kaltofen, 2014) having a less costly Prover in the case of structured or sparse matrices but slightly more expensive communication and Verifier cost overall. Detailed complexities can be found in Table 1.1.

Determinant A first algorithm-based determinant verification protocol was proposed in (Kaltofen et al., 2011, Theorem 3) for dense integer matrices, in which the Prover computes and transmits several LU decompositions of the input matrix $\boldsymbol{A}$ modulo several prime numbers. The verification then consists either in checking the rank of the input matrix if the LU decomposition is not full rank, or in checking the zero equivalence of $\boldsymbol{A}$ and $\boldsymbol{LU}$ and that the product of diagonal coefficients of $\boldsymbol{U}$ is the determinant the Prover provided. This protocol yields a cubic communication volume and a soft quadratic verification time. Then, (Dumas et al., 2016, Theorem 13, 14) proposed an improvement based on minimal polynomial computation. This version is especially efficient in the sparse case where it is Prover optimal, while yielding a linear communication volume and a verification time bounded by the cost of applying a vector to the input matrix. It also retains interesting complexities for structured and dense matrices. Finally, in Section 2.4.1 we will propose new verification protocols for dense matrices with no Prover overhead and the same communication volume and verification time than the latter protocol.

Characteristic and minimal polynomial (Kaltofen et al., 2011, Theorem 4) proposed a first verification protocol for the characteristic polynomial based on similarity residue system and computation modulo prime numbers. This protocol once again yields cubic communication volume and quadratic verification time. (Dumas and Kaltofen, 2014, Section 3) improved on the Frobenius normal form computation, which, combined with the previous protocol, allowed them to reach quadratic communication volume. Finally (Dumas et al., 2016, Section 5) proposed a verification protocol for the minimal polynomial of structured/sparse matrices, in linear communication volume and linear verification time.

Signature The signature of a symmetric matrix is the triple (n_+, n_-, n_0) respectively indicating the number of positive, negative and zero eigenvalues. (Kaltofen et al.,

2011, Corollary 1) verify the signature of a matrix by analyzing the characteristic polynomial of the input matrix. The characteristic polynomial approach can be straightforwardly applied to the signature verification protocol, hence yielding the same communication volume and verification time cost. In Chapter 2, we propose another approach based on the LDLT decomposition which allows the get better practical computation time for the Prover.

1.1.6 Linear algebra intermediate approach

The two approaches presented in previous sections either chose to sacrifice some efficiency in order to gain generality, or to focus on a specific algorithm to achieve better protocols. Here, we describe another technique, proposed in (Kaltofen et al., 2011, Theorem 5) for linear algebra computations which reduce to matrix multiplication. The idea is that the Prover will perform the required computations as normal, but that every time they will need to compute a matrix product, they will store the result of this operation. The Prover's commitment will be all the matrix products' results. The Verifier will then rerun the algorithm the Prover ran, and every time a matrix product needs to be performed, they will verify it using Freivalds' check on the data the Prover provided instead of performing it. This technique yields very interesting properties: while requiring a soft quadratic communication volume, it does not yield any Prover overhead and the verification cost is quadratic. It is also a non interactive protocol. The generic shape of this protocol is given in Protocol 1.3.

Protocol 1.3: KNS11

Public: Input to the algorithm $\mathcal{A}$
Certifies: The output of $\mathcal{A}$

Prover		Verifier
1. Run the algorithm		
	Inter. matrix products →	
2.		Run the algorithm, but replace each matrix product by a call to Freivalds' check

1.1.7 Overview of existing state of the art algorithm-based verification protocols

We finally give in Table 1.1 a summary of the efficiency of the best protocols to verifying the elements described above. We focus on the metrics which were presented in Section 1.1.3. We denote by $\boldsymbol{A}$ the input matrix, which has size $m \times n$ and rank r. Note that signature or characteristic polynomial verification protocols do not appear in the table *per se*. This is because they have the same characteristics as the determinant protocol from (Dumas et al., 2016) and then can be read on the same line as the latter on our table.

1.2 Multi-party computation protocols

1.2.1 Overview

We now consider a new situation, where several people, each of them owning some data want to compute a function which takes of all their data as input. We could imagine those people being owners of databases who want to execute a request on all of their data, bidders at an auction who want to know who made the highest bid or hospitals needing to compute compatibility rate between hosts and donor for organ transplant. They could obviously go public and reveal all their data to each other and then perform the requested computation, but for privacy reasons, they want to perform computations without revealing their input data. Such protocols allowing people to perform such shared computations are called *secure multiparty computation protocols* (abridged MPC in what follows). This has first been investigated by (Yao, 1982), regarding the famous millionaires problem in which two people wants to know who is the richest of them all without revealing their actual wealth to one another. Since then, a lot of effort has been put on this field, and this section will present the major techniques that can be used and discuss their efficiency. As for the previous section on verification protocols, we will focus on the two concurrent paradigms, generic protocols and algorithm-based protocols.

1.2.2 Definitions

An MPC protocol involves a given number of participants (often designated as *players* or *parties*) $P_1, \ldots, P_n$, each with their own private data (or *input*) $x_1, \ldots, x_n$. Players seek to compute the output y of an n-variate public function f, while keeping their own input secret. Note that depending on the structure of y, it can be either public or private: y could for instance be a vector of size n and player i only learns the i-th element of y. In order to perform such a computation *securely*, two conditions must be satisfied: *correctness*, meaning that the correct value of y is computed, and *privacy* meaning that players learn nothing but the output of f.

	Algorithm	Rounds	Prover		Communication	Probabilistic Verifier Time	Min. #S
			Determ.	Time			
RANK	(KNS11) over (CKL13)	0	No	$\widetilde{O}(r^\omega + \mu(\boldsymbol{A}))$	$\widetilde{O}(r^2 + m + n)$	$\widetilde{O}(r^2 + \mu(\boldsymbol{A}))$	2
	(DK14)	2	No	$O(n(\mu(\boldsymbol{A}) + n))$ or $O(mnr^{\omega-2})$	$O(m + n)$	$2\mu(\boldsymbol{A}) + \widetilde{O}(m + n)$	$\Omega(\min\{m, n\} \log(mn))$
	(E15)	2	Yes	$O(mnr^{\omega-2})$	$O(n + r)$	$O(\mu(\boldsymbol{A}) + n)$	2
DET	(F79) & PLUQ	0	Yes	$O(n^\omega)$	$O(n^2)$	$O(n^2) + \mu(\boldsymbol{A})$	2
	(DKTV16) & CHARPOLY	2	No	$O(n\mu(\boldsymbol{A}))$ or $O(n^\omega)$	$O(n)$	$\mu(\boldsymbol{A}) + O(n)$	n^2

Table 1.1: State of the art verification protocols for several linear algebra properties

Regarding the construction of such protocols, it has been proven by (Goldreich et al., 1987) that any function can be securely computed. However, early protocols were not efficient enough to be practical and significant effort has been put in designing protocols that could be used in practice ever since. In the rest of this section, we first present the metrics to consider when designing MPC protocols. We then follow a similar outline to the previous section by first describing classical tools that are used in many protocols and then presenting and comparing concurrent approaches for protocols design.

1.2.3 Evaluation of the efficiency of an MPC protocol

The following metrics should be carefully considered when designing an MPC protocol:

Communication: the volume of data exchanged throughout the protocol. For the same bandwidth vs. latency reasons explained in Section 1.1.3, one should take into account the number of communication rounds required to complete the execution of a protocol.

Computation cost: the worst-case number of arithmetic operations performed by all the players during the execution of the protocol. Another parameter to keep in mind while designing such protocol is computation cost balancing between players: are some players performing much more computations than others? Such discrepancy could be problematic as there is no way to guarantee that the players whose role forces them to perform more computations are using faster machines than the other ones, which could lead to possibly large waiting phases during the protocol. Ideally, computations should be balanced amongst all players.

Security: the type of adversaries against whom the protocol is secure and the optional countermeasures that one needs to deploy to ensure this security. There are two major types of adversaries: **semi-honest** who are not allowed to tamper with the protocol, but can (and will) perform any additional computation on the data they know to try to learn secret information, and **malicious** adversaries (*a la* (Dolev and Yao, 1983)) who are allowed to not follow the protocol. They can, for instance, alter the distribution of the random value they provide or pick any element they want for their input. Furthermore, these adversaries can be working alone, or they can be colluding in which case they will share any information they own. Note that the adversaries can be static in which case they remain adversaries during the whole protocol, or dynamic which means players can become adversaries at any time during the execution of the protocol (there is usually a bound on the number of adversaries existing at the same time). In this thesis, we will only consider static adversaries. Finally, a word on the aforementioned countermeasures: some protocols might require the deployment of

additional techniques to achieve a given security. These countermeasures usually have a cost (either in communication or computation - maybe both) and should ideally be avoided. One can for instance cite secure channels, which guarantee authenticity and integrity of a message, but force digital signature of every message, which induces additional computation cost to verify every message.

1.2.4 Paradigms for MPC protocols

A simple idea for designing MPC protocols could be to assume that there exists some independent, honest and impossible to corrupt entity called a trusted third party. All players could send their private input to this party, who will perform the required computations and send the output back to the players. This solution is both generic and extremely efficient. However, the existence of such an entity is a very strong security model and might not be suitable to some applications. Hence, a lot of effort has been put in designing MPC protocols that do not resort to such trusted parties. In this section, we describe the two main paradigms for MPC protocols design. As for verification protocols, they are split between generic approaches aiming at designing protocols that can be applied to a wide array of problems, and algorithm-based approaches which focus on one specific computation and use its specificities in order to get a very efficient protocol. We will start by giving an overview of the main used tools for designing MPC protocols, before describing the two aforementioned paradigms.

Widely used tools

Secret Sharing. Secret sharing refers to techniques allowing to distribute a secret amongst a group of parties, each receiving one part (or *share*) of the secret. The idea is that the secret can only be reconstructed if a sufficient number of shares are put together. This originates from (Shamir, 1979), who developed a protocol based on polynomial interpolation: given a secret s to be shared amongst n players, and setting the reconstruction threshold to $t < n$, generate a polynomial function f of degree $t-1$ whose degree 0 coefficient is s. Now, each player receives a distinct random evaluation of f, and by Lagrange interpolation t players are needed to reconstruct f, thus revealing s. Other techniques exists, such as one based on plane equations (Blakley, 1979) or Chinese remainder theorem (Mignotte, 1983; Asmuth and Bloom, 2006).

Secret sharing is widely used for MPC applications, usually using a different approach: here, a player will split their input s into as many additive shares as there are players (for n players, $s = \Sigma_{i=1}^{n} s_i$) and distribute these shares amongst players. In order to be secure against trivial collusion attacks, it is necessary to add some extra randomness: consider a protocol with n players, where P_i needs to secretly share their input s_i. Each player $P_j, j \neq i$ will generate a random element r_j that they will send to P_i. P_i will compute $r = \Sigma_j r_j$ and broadcast $s_i - r$. Finally, each party will set their share to $s_i - r + r_j$. This mechanism allows to securely perform arithmetic operations: an addition can be straightforwardly performed on shares of its operands (if two ele-

ments a and b are split into $\{a_i\}$ and $\{b_i\}$, $a+b = \Sigma_i a_i + \Sigma_i b_i$). For multiplication, things are a bit more complicated and require to generate so called "multiplicative triples", a set of three non-reusable elements, in order to be performed. We will not go into details here, but the interested reader can report to (Beaver, 1992) for details on this method.

Oblivious Transfer. Oblivious transfer protocols allow one player, called the sender, to transfer one piece of information they own (amongst possibly many) to another player, the receiver, while remaining oblivious to what piece has been transferred. The first oblivious transfer protocol has been introduced by (Rabin, 2005) and refined by (Even et al., 1985), who designed a new protocol with MPC applications in mind. These first protocols were called 1-2 oblivious transfer as the sender owns two pieces of data and the receiver gets one. A short description of Even et al's protocol could be done as follows: using a public key cryptosystem, the sender generates their key pair and sends the public key to the receiver. They also generate two random messages x_0, x_1, which they send to the receiver. The receiver then picks the random message corresponding to the information they want (say, x_0 for m_0 and scrambles x_0 into v by using a random element k. The sender receives v and uses it to compute two possible values for k, k_0 and k_1. One of these will be the same as the k the receiver picked, but the sender has no way to say which one it is. They will finally generate two messages $m'_0 = m_0 + k_0$ and $m'_1 = m_1 + k_1$ and send these to the receiver, who can learn the message they wanted by subtracting their original k to the message they wanted. Since then, these protocols have been generalized into $1-n$ oblivious transfer, as for instance in (Aiello et al., 2001; Kolesnikov et al., 2016), or even $k-n$ oblivious transfer (Brassard et al., 1987; Ishai and Kushilevitz, 1997) but for what follows we will only be interested in 1-2 oblivious transfer protocols.

Homomorphic Cryptography. Homomorphic cryptography refers to cryptosystems allowing to perform computation on ciphertexts which behaves as if the operation has been performed on the plaintext. This ability is crucial when its deployed in MPC protocols, as it allows players to perform computations on ciphered data without getting information on the clear data. We distinguish three major types of homomorphic cryptosystems: *fully homomorphic* supports an unbounded number of arithmetic operations, *somewhat homomorphic* supports a bounded number of arithmetic operations and *semi-homomorphic* supports only one type of arithmetic operation, the other one taking one plaintext and one ciphertext as operands. Without going into much detail here, we will only mention that the first two are usually considered to be too expensive to be practical, while the latter can be used in practice. The interested reader may refer to sections 3 and 6 of (Mohassel, 2011) to get an overview of several homomorphic cryptosystems. A recent comparison of several modern somewhat homomorphic cryptosystems can be found in (Costache and Smart, 2016). Finally, another interesting reference is the standardisation document from the *Homomorphic Encryption Standardization Consortium* Albrecht et al. (2018) which describes fully homomorphic

cryptosystems and gives recommended parameters to efficiently set them in practice. We will present the specificities of the cryptosystems we used for our work in Chapter 4.

Generic approaches

We present here the main classes of generic approaches, which are both based on circuits.

Secret sharing-based approaches These approaches rely on arithmetic circuit computation, and work as follows: at the beginning of the protocol, each player will split their input into shares and send shares to other players as described above. Then, they will run the algorithm locally, using the shares they received as input to the arithmetic gates, send the output to the appropriate players who can recompose the output of the circuit by using all their shares. These approaches are often split into two phases: the shares generation and exchange, and the actual circuit evaluation. Such secret-sharing based schemes have been widely used and implemented into several libraries over the past decade, see for instance (Bendlin et al., 2011; Damgård and Zakarias, 2013). A recent and highly regarded library using this technique is SPDZ (Damgård et al., 2013), which proposes a quite efficient version of this scheme, with security against active adversaries. This implies the use of zero-knowledge proofs techniques and message authentication codes in order to guarantee that players have been honest regarding their shared values and that they did not corrupt data computed while evaluated the arithmetic circuit. They also use somewhat homomorphic encryption techniques to generate the multiplicative triples required to perform multiplication on shares. This step was improved in the most recent work in line, MASCOT (Keller et al., 2016), which proposed faster oblivious transfer-based techniques for the multiplicative triples generation. However, these approaches still suffer from large overheads: the preprocessing phase requires to compute one distinct triple per multiplication gate per player, which is a time and memory consuming process, and the exchange of shares and gate output amongst players also require a lot of communication. Note that as the preprocessing phase is by far the costlier element of these protocols, communication is not the bottleneck of these approaches.

Garbled circuits These techniques apply to boolean circuit, and were initially developed as two party protocols. The idea is that one player will be in charge of obfuscating (*garbling*) the information passed through the circuit while the other player will be in charge of performing the computation. The first garbled circuit protocol has been proposed by (Yao, 1986). The security of this proposal relies on symmetric cryptography and oblivious transfer and works as follows: for a given gate, let us consider two players Alice and Bob, where Alice is in charge of the garbling. She will start by generating four symmetric keys corresponding to each possible input for the gate, $(k_A^0, k_A^1), (k_B^0, k_B^1)$ and will use these four keys to cipher the four possible output

of the gate. For example, she will use k_A^0, k_B^0 to cipher the output corresponding to $0, 0$. She will then send to Bob the four ciphertexts alongside the key corresponding to her input. Now, Bob needs to get the key corresponding to his input without revealing what it is to Alice, which will be done using a 1-2 oblivious transfer protocol. Bob will now try to decipher each ciphertext, and the one that can be properly deciphered is the actual output to the gate. In order to perform a full evaluation, our players needs to repeat that for all gates in the circuit.

Another garbled circuit approach is due to (Goldreich et al., 1987) and is also a two party protocol. This time, however, instead of using symmetric cryptography to hide the intermediate values computed during circuit evaluation, secret sharing techniques are used: each party splits their inputs into two shares, sending one to the other party and keeping the second to themselves. They will then locally perform the evaluation of each gate, using the shares they own as input. This idea applies straightforwardly to (X)OR or NOT gates, but requires a bit more work for AND gates: in this particular case, the same oblivious transfer-based technique as above has to be performed.

While initially only designed as two-party protocols secure against semi-honest adversaries, both approaches for garbled circuits have been widely studied and improved since then: practicality of Yao's garbled circuits have been proven with, for instance, its Fairplay implementation (Malkhi et al., 2004) and a secure version against malicious adversaries has been achieved in (Kreuter et al., 2012). At first glance, Goldreich et al.'s approach might seem more expensive than Yao's: the latter runs in constant rounds and requires oblivious transfer for only one of the players, while the former requires an oblivious transfer per AND gate (which can be replaced by the use of the aforementioned multiplicative triples for multiplicative secret sharing). However, in (Choi et al., 2012), the authors propose an n-party implementation of the aforementioned Goldreich et al. scheme which outperforms Yao-based techniques for computations involving more than three players. This improvement has been carried out for two players schemes in (Schneider and Zohner, 2013). Security against malicious adversaries has been achieved in (Nielsen et al., 2012).

Despite their genericity making them extremely appealing, these techniques once again involve some large overhead as the need to engage in an oblivious transfer protocol for each gate implies a lot of communication between the two parties.

1.2.5 Algorithm-based approaches

Designing MPC protocols for specific tasks seems to be quite less of fashion than working on generic solutions, and there are not that many articles that focus on this dedicated approach. Nonetheless, we present here a summary of existing algorithm-based approaches for two linear algebra problems, matrix multiplication (and the underlying dot product computation) and linear system solving.

Scalar and matrix product

We need to open this presentation by a fairly long disclaimer: Comparing algorithm-based approaches for these operations is a difficult task. Protocols were designed with different, often incompatible target application in mind which led to very specific models. Initially, researchers were interested in conducting statistical analysis between private data sets (for instance customers buying behaviours or employees HR information). Initially, This model usually involved only two players (see (Du and Atallah, 2001a; Du et al., 2004; Amirbekyan and Estivill-Castro, 2007)), each owning the same input data structure (vector or matrix) but was later extended to support N players (Goethals et al., 2005). In that case, N-party vector product means computing the sum of all the partial coefficient-wise products. Finally, a new model was introduced for partial trust computation (see (Dolev et al., 2010; Dumas et al., 2017a)) in which N players want to perform dot product, one players owning one full vector of size $N-1$ and the other $N-1$ players each own only one coefficient of the other vector. This last model is the one we use in the work presented in Chapter 4.

Thus, considering the difficulty of comparing those approaches, we propose here to summarily describe them, give their main advantages and drawbacks and only focus on their communication cost as it is the usual bottleneck in MPC applications (see Chapter 4 for more details on that point).

The first dedicated protocol for scalar product applied to confidential statistical analysis was proposed by (Du and Atallah, 2001a). As mentioned, their protocol was for two parties only (each party knows a vector) and it was only secure against semi-honest adversaries. They propose two solutions to achieve this result, one based on oblivious transfer and the other one on homomorphic encryption. With security parameter λ, and vectors of size n, the total communication cost for their protocol is $4\lambda n$ field elements. In 2004, Du et al. extended this result, now allowing matrix-matrix product in a semi-honest two-party setting (each players knows a matrix) (Du et al., 2004). Their communication cost is only the size of the input matrix, $m \times n$ field elements. (Goethals et al., 2005) proposed a first n-player dot product protocol Their solution uses homomorphic encryption and masking techniques to achieve security against semi-honest players, and can use very costly zero-knowledge proofs to achieve security against malicious adversaries. While this new protocol is comparable to the previous ones on communication volume, requiring to exchange $n(N-1)$ field elements (with N players and vectors of size n), its ($N > 2$)-player version present the drawback of having asymmetric roles for the players. One "master" player is trusted with the private key to the homomorphic cryptosystem and has to be honest to guarantee privacy. Then, (Amirbekyan and Estivill-Castro, 2007) proposed another scalar product protocol, in $4n$ field elements communication volume (for vectors of size n), but back to the initial two-player only model and achieving only security against semi-honest adversaries.

On trust computation, (Dolev et al., 2010) propose an N-player protocol for weighted average achieving security against any number of semi-honest adversaries at the communication cost of $O(N^3)$ field elements. Finally, (Dumas et al., 2017a) propose an

improvement of the former in $O(N^2)$ field elements. This allowed them to achieve a communication cost of $O(N^3)$ for a matrix product protocol in which a given player P_i knows the i-th row of the input matrices.

Linear system solving

Algorithm-based approaches for solving a linear system of equations $Ax = b$ can be split in two distinct problems: the first one focuses on two party protocols where each party brings an entire matrix and vector as their input, and aim at solving the aforementioned system in which A and b are formed by addition or concatenation of the two parties' inputs. A first solution has been proposed by (Du and Atallah, 2001b) in 2001, and is heavily based on oblivious transfer protocols. The authors are only interested in improving on communication cost and achieve a complexity of $O(\mu \times N^2)$ for matrices of size N and security parameter μ. This has been improved several times since (see for instance (Wu et al., 2005; Yang et al., 2008), with the latest improvement proposed by (Troncoso-Pastoriza et al., 2009). All these protocols are only secure against semi-honest adversaries.

The second class of problems are more in-line with the core of this thesis, as they focus on n-party protocols where each party knows one subset of coefficients of the input matrix A and a set of coefficients of b. On this topic, we can cite a protocol by (Dagdelen and Venturi, 2015). This protocol requires at least three players and can be summarized that way: one player will be in charge of performing computations, one will be in charge of interacting with the former, and the others will be mostly passive. The idea is that the master player will engage into a protocol in which they will learn a masked version A' of the full input matrix A, and a masked vector b' where the "interacting" player will be in charge of picking the masks. The master will then solve the system $A'x = b'$ and send the result to the "interacting player", who can remove the masks and broadcast the solution to the system. The fact that the interacting player is able to gather the inputs of the other players in order to form A' is based on oblivious transfer techniques. This protocol presents nice complexities, with a communication complexity of $O(l + k)$ and a computational complexity of $O(N^2(n + \mu))$ where n is the number of players, μ the security parameter and N the input matrix size. However it does have some disadvantages: the imbalance in player roles forces the master player to perform most of the computation and the security relies on the master player as well: should they be corrupted and colluding with any other party and the players' inputs will immediately be leaked. This protocol is also only secure against semi-honest adversaries.

In the rest of this thesis, we will propose new efficient linear algebra protocols to verify outsourced computation and securely perform multi-party computations. As we showed in this first chapter, these two aspects, verified and secure multi-party computations are two sides of outsourced computation which have many common features, such as their cost metrics, the requirement to interact between several parties

in order to compute a result, and several underlying linear algebra techniques.

Chapter 2

Elimination-based verification protocols for triangular equivalence and rank profiles

Contents

Technical summary and overview of this chapter

Focus This chapter focuses on the design of a verification protocol for the computation of a matrix invariant called the rank profile matrix. Given a matrix $\boldsymbol{A}$, its rank profile matrix carries the information on the pivots of $\boldsymbol{A}$ and of all the pivots of its leading submatrices. These pivots can be used when performing a Gaussian elimination over $\boldsymbol{A}$. As the Gaussian elimination is a central tool for linear algebra and is used in many applications, an efficient dedicated verification protocol for the rank profile matrix is an important for verified computing in linear algebra. The articles associated with this chapter are (Dumas et al., 2017b, 2020).

Model We propose verification protocols as defined in Definition 1.1.2, whose efficiency is measured based on the metrics given in Section 1.1.3.

State of the art – main competition There were no dedicated verification protocols for the rank profile matrix prior to this work. We will thus compare our new protocols to the ones that can be obtained by applying the intermediate approach of Kaltofen et al. (2011) over the rank profile matrix computation algorithm of Dumas et al. (2017c). For a $m \times n$ matrix of rank r, this gives a verification protocol with a deterministic Prover in $O(mnr^{\omega-2})$, $\widetilde{O}(mn)$ communications and $\widetilde{O}(mn)$ Verifier cost. State of the art dedicated verification protocols for the determinant were presented Section 1.1.5 and their cost is given in Table 1.1.

Results We present two verification protocols for the rank profile matrix. The first one has quadratic communication volume and Verifier complexity and thus slightly improves on the state of the art. However, it is a non-interactive certificate, which has the advantage of being straightforwardly publicly verifiable. The second one is the main result of this chapter: a new interactive verification protocol in linear communication volume and quadratic Verifier complexity. This protocol relies on a sub-protocol verifying the triangular equivalence of matrices: two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are said triangular equivalent if there exists a triangular matrix $\boldsymbol{T}$ such that $\boldsymbol{AT} = \boldsymbol{B}$. The structure used in the data exchange for this protocol appear to be a central tool to which several problems could be reduced to. Using this observation, we also provide new, more efficient than the state of the art verification protocols for the determinant and the signature of a matrix.

Outline

We first present in Section 2.1 non-interactive verification protocols for the column/row rank profile and the rank profile matrix. Then, we describe and prove in Section 2.2 a new interactive protocol for the rank profile matrix, and its subprotocols, based on the triangular equivalence problem. After that, in Section 2.3, we show how to have a constant number of communication rounds for our row/column rank profile protocol.

Finally Section 2.4, shows how to use our new protocols to improve Prover efficiency on the verification protocol for the determinant and signature of dense matrices.

Introduction

Gaussian elimination is a central tool used in a wide array of applications in computer algebra, as in the LU decomposition (itself used for linear system solving, or determinant and rank computation), or the computation of basis of vector spaces.

In this chapter, we consider a rectangular matrix $\boldsymbol{A} \in \mathsf{F}^{m\times n}$ of rank r. The *row rank profile* (abridged RRP) is the lexicographically minimal sequence of r indices of independent rows of $\boldsymbol{A}$. The *column rank profile* (CRP) is defined similarly on the columns of $\boldsymbol{A}$. These two sequences describe the shape of the row/column echelon form of $\boldsymbol{A}$ and give the position of row/column pivots that can be used when performing a Gaussian elimination over $\boldsymbol{A}$. The matrix is said to have generic row (resp. column) rank profile if its row (resp. column) rank profile is $(1,\dots,r)$. This information is contained on an invariant of $\boldsymbol{A}$, called its *rank profile matrix* (RPM). The rank profile matrix also contains the information on the row- and column-rank profile of every leading submatrix of $\boldsymbol{A}$. Formally, the rank profile matrix of $\boldsymbol{A}$ is the unique $m \times n$ {0-1}-matrix with r nonzero entries of which every leading submatrix has the same rank as the corresponding submatrix of $\boldsymbol{A}$. This invariant can be computed, either using a deterministic algorithm in $O(mnr^{\omega-2})$, or a Monte-Carlo probabilistic algorithm in $(r^\omega + m + n + \mu(\boldsymbol{A}))^{1+o(1)}$ (Dumas et al., 2017c).

A bottom-line verification protocol from the computation of the rank profile matrix can be immediately derived by applying KNS11 on (for instance) the results of (Dumas et al., 2017c, 2013). However, this approach yields logarithmic factors in the communication volume and the verification cost. Thus, there is room for improvement in designing dedicated protocols for the rank profile matrix. To build our rank profile matrix verification protocol, we show that it comes down to certifying a matrix property called the triangular equivalence, which is, for two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathsf{F}^{m\times n}$, the existence of a triangular matrix $\boldsymbol{T} \in \mathsf{F}^{n\times n}$ such that $\boldsymbol{AT} = \boldsymbol{B}$. We then propose a generalisation of this protocol allowing to verify the generic rank profile-ness of a matrix and the $\boldsymbol{D}$ and $\boldsymbol{P}$ matrices of an LDUP decomposition. Finally, combining these protocols, it is possible to verify the column rank profile and the rank profile matrix of an invertible matrix. Subsequently, by combining these pieces, we propose a rank profile matrix verification protocol for any matrix. Finally, the $\boldsymbol{D}$ and $\boldsymbol{P}$ from LDUP verification, initially built with the rank profile matrix verification in mind allows us to propose more efficient protocols for the determinant and the signature of a matrix.

We introduce some notations that will be used throughout this chapter: the symmetric group and the group of permutation matrices of size n will be denoted by $\mathcal{S}_n$. $\mathcal{D}_n(\mathsf{F})$ is the group of invertible diagonal matrices over the field F, while $\mathcal{D}_n^{(2)}(\mathsf{F})$ represents block diagonal matrices with 1 or 2-dimensional diagonal blocks. Finally, we denote by $\mathbb{P}$ the set of prime numbers.

2.1 Non interactive and quadratic communication verification protocols

In this section, we propose two verification protocols, the first one for the column (resp. row) rank profile of a given matrix, and the second one for the rank profile matrix of a given matrix. While those protocols have a quadratic communication complexity, they have the advantage of being non interactive.

2.1.1 Column rank profile verification protocol

This verification protocol is based on the $\boldsymbol{P}, \boldsymbol{L}, \boldsymbol{U}, \boldsymbol{Q}$ decomposition of the input matrix $\boldsymbol{A}$. The main idea here is that the column rank profile of $\boldsymbol{A}$ can be extracted from its PLUQ decomposition if $\boldsymbol{UQ}$ is in row echelon form, which will be proved in Lemma 2.1.1. The Prover will hence commit the entire PLUQ decomposition to the Verifier, who will verify that $\boldsymbol{A} = \boldsymbol{PLUQ}$ using Freivalds' check (Freivalds) and that $\boldsymbol{UQ}$ is row echelonised. This is proposed in details in NonInteractiveColumnRankProfile.

Protocol 2.1: NonInteractiveColumnRankProfile

Public: $\boldsymbol{A} \in \mathsf{F}^{m\times n}$, $\mathcal{I} \in (1, \ldots, n)$
Certifies: The colum rank profile of $\boldsymbol{A}$

Prover		Verifier
1. Compute a PLUQ decomposition of $\boldsymbol{A}$ s.t. $\boldsymbol{UQ}$ is in row echelon form		
	$\xrightarrow{\boldsymbol{P}, \boldsymbol{L}, \boldsymbol{U}, \boldsymbol{Q}}$	
2.		Check that $\boldsymbol{UQ}$ is in row echelon form
3.	Freivalds($\boldsymbol{A}$, $\boldsymbol{PLUQ}$)	
4.		$\mathcal{I} \stackrel{?}{=} (\boldsymbol{Q}[1], \ldots, \boldsymbol{Q}[r])$

Lemma 2.1.1. *Let $\boldsymbol{A} = \boldsymbol{PLUQ}$ be the $\boldsymbol{P}, \boldsymbol{L}, \boldsymbol{U}, \boldsymbol{Q}$ decomposition of an $m \times n$ matrix $\boldsymbol{A}$ of rank r. If $\boldsymbol{UQ}$ is in row echelon form, then $(\boldsymbol{Q}[1], \ldots, \boldsymbol{Q}[r])$ is the column rank profile of $\boldsymbol{A}$.*

Proof. Write $\boldsymbol{A} = \boldsymbol{P}\begin{bmatrix}\boldsymbol{L}_1 \\ \boldsymbol{L}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2\end{bmatrix}\boldsymbol{Q}$, where $\boldsymbol{L}_1$ and $\boldsymbol{U}_1$ are $r \times r$ lower and upper triangular respectively. If $\boldsymbol{UQ}$ is in echelon form, then $\boldsymbol{R} = \begin{bmatrix}\boldsymbol{I}_r & \boldsymbol{U}_1^{-1}\boldsymbol{U}_2 \\ \boldsymbol{0}_{(m-r)\times n}\end{bmatrix}$ is in reduced

echelon form. Now

$$\begin{bmatrix} \boldsymbol{U}_1^{-1} & \\ & \boldsymbol{I}_{m-r} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_1 & \\ \boldsymbol{L}_2 & \boldsymbol{I}_{m-r} \end{bmatrix}^{-1} \boldsymbol{P}^T \boldsymbol{A} = \begin{bmatrix} \boldsymbol{U}_1^{-1}\boldsymbol{U}\boldsymbol{Q} \\ \boldsymbol{0}_{(m-r)\times n} \end{bmatrix} = \boldsymbol{R}$$

is left equivalent to $\boldsymbol{A}$ and is therefore the echelon form of $\boldsymbol{A}$. Hence the sequence of column positions of the pivots in $\boldsymbol{R}$, that is $(\boldsymbol{Q}[1], \ldots, \boldsymbol{Q}[r])$, is the column rank profile of $\boldsymbol{A}$. □

Theorem 2.1.2. *Let $\boldsymbol{A} \in \mathsf{F}^{m\times n}$ with $r = \operatorname{rank}(\boldsymbol{A})$. Protocol 2.1 is a complete and probabilistically sound non interactive protocol. It requires $O(r(m+n))$ communications and Verifier cost $O(r(m+n))$. The probability that the Verifier incorrectly accepts is at most $1/\#S$. There is a deterministic algorithm for the Prover, which costs $O(mnr^{\omega-2})$*

Proof. If $\boldsymbol{UQ}$ is in row echelon form, the Verifier will never reject it, and if $\boldsymbol{A} = \boldsymbol{PLUQ}$, as Freivalds' protocol is complete, the Verifier will systematically accept the Prover commitment, hence ensuring the completeness of this protocol.

If $\boldsymbol{UQ}$ is not in row echelon form, this will be caught by the Verifier with probability 1. If $\boldsymbol{A} \neq \boldsymbol{PLUQ}$, the Verifier will accept with probability $1/\#\mathsf{S}$ (Freivalds' check), hence the soundness of NonInteractiveColumnRankProfile.

Now, for the complexities: the Prover needs to find a $\boldsymbol{PLUQ}$ decomposition of $\boldsymbol{A}$, where $\boldsymbol{UQ}$ is in row echelon form. Using algorithms provided in (Jeannerod et al., 2013), Prover complexity will be $O(mnr^{\omega-2})$. Verifier checks can be done in $O(r(m+n))$ by Freivalds' check. Finally, communication cost is the one of sending $\boldsymbol{P}$ and $\boldsymbol{Q}$, two permutations matrices and $\boldsymbol{L}$ and $\boldsymbol{U}$, which are respectively $m \times r$ and $r \times n$ matrices, hence the cost of $O(r(m+n))$. □

Note that this holds for the row rank profile of $\boldsymbol{A}$: in that case, the Verifier has to check if $\boldsymbol{PL}$ is in column echelon form.

2.1.2 Non interactive Rank Profile Matrix verification protocol

Lemma 2.1.3. *A decomposition $\boldsymbol{A} = \boldsymbol{PLUQ}$ reveals the rank profile matrix, namely $\mathcal{R}_A = \boldsymbol{P}\left[\begin{smallmatrix} \boldsymbol{I}_r \\ 0 \end{smallmatrix}\right]\boldsymbol{Q}$, if and only if $\boldsymbol{P}\left[\begin{smallmatrix} \boldsymbol{L} \\ 0 \end{smallmatrix}\right]\boldsymbol{P}^T$ is lower triangular and $\boldsymbol{Q}^T\left[\begin{smallmatrix} \boldsymbol{U} \\ 0 \end{smallmatrix}\right]\boldsymbol{Q}$ is upper triangular.*

Proof. The *only if* case is proven in (Dumas et al., 2017c, Th. 21). Now suppose that $\boldsymbol{P}\left[\begin{smallmatrix} \boldsymbol{L} & 0_{m\times(m-r)} \end{smallmatrix}\right]\boldsymbol{P}^T$ is lower triangular. Then we must also have that $\overline{\boldsymbol{L}} = \boldsymbol{P}\left[\begin{smallmatrix} \boldsymbol{L} & \boldsymbol{0} \\ & \boldsymbol{I}_{m-r} \end{smallmatrix}\right]\boldsymbol{P}^T$ is lower triangular and non-singular. Similarly suppose that $\boldsymbol{Q}^T\left[\begin{smallmatrix} \boldsymbol{U} \\ 0 \end{smallmatrix}\right]\boldsymbol{Q}$ is upper triangular so that $\overline{\boldsymbol{U}} = \boldsymbol{Q}^T\left[\begin{smallmatrix} \boldsymbol{U} & \\ 0 & \boldsymbol{I}_{n-r} \end{smallmatrix}\right]\boldsymbol{Q}$ is non-singular upper triangular. We have $\boldsymbol{A} = \overline{\boldsymbol{L}}\boldsymbol{P}\left[\begin{smallmatrix} \boldsymbol{I}_r \\ 0 \end{smallmatrix}\right]\boldsymbol{Q}\overline{\boldsymbol{U}}$. Hence the rank of any (i,j) leading submatrix of $\boldsymbol{A}$ is that of the (i,j) leading submatrix of $\boldsymbol{P}\left[\begin{smallmatrix} \boldsymbol{I}_r \\ 0 \end{smallmatrix}\right]\boldsymbol{Q}$, thus proving that $\mathcal{R}_A = \boldsymbol{P}\left[\begin{smallmatrix} I_r \\ 0 \end{smallmatrix}\right]\boldsymbol{Q}$. □

We use this characterization to verify the computation of the rank profile matrix in the following protocol: Once the Verifier receives $\boldsymbol{P}$, $\boldsymbol{L}$, $\boldsymbol{U}$ and $\boldsymbol{Q}$, they have to check that $A = PLUQ$, using Freivalds' check Protocol 1.2, and check that $\boldsymbol{L}$ is echelonised

Protocol 2.2: NonInteractiveRankProfileMatrix

Public: $\boldsymbol{A} \in \mathsf{F}^{m\times n}$, $\mathcal{R}_A \in \mathsf{F}^{m\times n}$
Certifies: $\mathcal{R}_A$ is the rank profile matrix of $\boldsymbol{A}$

Prover		Verifier
1. Compute a $\boldsymbol{PLUQ}$ decomposition of $\boldsymbol{A}$ revealing $\mathcal{R}_A$		
	$\boldsymbol{P}, \boldsymbol{L}, \boldsymbol{U}, \boldsymbol{Q}$ ⇢	
2.		$\boldsymbol{A} \stackrel{?}{=} \boldsymbol{PLUQ}$
3.		Check that $\boldsymbol{PLP}^T$ is upper triangular
4.		Check that $\boldsymbol{Q}^T\boldsymbol{UQ}$ is lower triangular
5.		$\mathcal{R}_A \stackrel{?}{=} \boldsymbol{P}\begin{bmatrix} \boldsymbol{I}_r & \boldsymbol{0}_{(m-r)\times(n-r)} \end{bmatrix}\boldsymbol{Q}$

by $\boldsymbol{P}$ and $\boldsymbol{U}^T$ by $\boldsymbol{Q}^T$. If successful, the Verifier can just compute the rank profile matrix of $\boldsymbol{A}$ from $\boldsymbol{P}$ and $\boldsymbol{Q}$, as shown in NonInteractiveRankProfileMatrix.

Theorem 2.1.4. *NonInteractiveRankProfileMatrix is a complete and probabilistically sound non interactive protocol. It requires $O(r(m+n))$ communications, and has verifier cost $O(r(m+n)+\mu(\boldsymbol{A}))$ The probability that the verifier incorrectly accepts is at most $1/\#S$. There is a deterministic algorithm for the Prover in $O(mnr^{\omega-2})$*

Proof. If $\boldsymbol{A} = \boldsymbol{PLUQ}$, the Verifier will never reject it. If $\boldsymbol{PLP}^T$ is upper triangular and $\boldsymbol{Q}^T\boldsymbol{UQ}$ is lower triangular, the Verifier will always accept them and by Lemma 2.1.3, the Verifier will successfully extract the rank profile matrix, ensuring the completeness of NonInteractiveRankProfileMatrix.

Now if $\boldsymbol{A} \neq \boldsymbol{PLUQ}$, the Verifier will reject that with probability $1/\#\mathsf{S}$ by Freivalds' check. And finally, if the provided $\boldsymbol{PLUQ}$ decomposition does not reveal the rank profile matrix of $\boldsymbol{A}$, Lemma 2.1.3 implies that either $\boldsymbol{PLP}^T$ or $\boldsymbol{Q}^T\boldsymbol{UQ}$ will not have the expected shape, which the Verifier will always detect.

Now, for the complexities: the Prover needs to find a rank profile matrix-revealing $\boldsymbol{PLUQ}$ decomposition of $\boldsymbol{A}$, which can be computed in $O(mnr^{\omega-2})$ operations using algorithms provided in (Dumas et al., 2013). Verifier checks can be done in $O(r(m+n)) + \mu(A)$. Finally, communication cost is the one of sending $\boldsymbol{P}$ and $\boldsymbol{Q}$, two permutations matrices and $\boldsymbol{L}$ and $\boldsymbol{U}$, which are respectively $m \times r$ and $r \times n$ matrices, hence the cost of $O(r(m+n))$. □

2.2 An interactive verification protocol for the rank profile matrix

2.2.1 Triangular one sided equivalence

Two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathsf{F}^{m\times n}$ are right (resp. left) equivalent if there exists an invertible $n \times n$ matrix $\boldsymbol{T}$ such that $\boldsymbol{AT} = \boldsymbol{B}$ (resp. $\boldsymbol{TA} = \boldsymbol{B}$). If in addition $\boldsymbol{T}$ is a lower triangular matrix, we say that $\boldsymbol{A}$ and $\boldsymbol{B}$ are lower triangular right (resp. left) equivalent. The upper triangular (resp. left) equivalence is defined similarly. We propose a protocol verifying that two matrices are left or right equivalent. These two matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$, will be the input of the protocol. We force $\boldsymbol{A}$ to be a regular (full rank) matrix. A naive protocol would be the matrix $\boldsymbol{T}$ itself, and the Verifier would check that $\boldsymbol{AT} = \boldsymbol{B}$ using Freivalds' check. However, this protocol requires a quadratic amount of communication. Here, we instead propose a protocol allowing to verify triangular equivalence in a linear $(2n)$ amount of communication. We essentially use Freivalds' check, but with a constrained interaction pattern in the way the challenge and response vectors are communicated. This pattern imposes a triangular structure in the way the Prover's responses depend on the Verifier's challenges.TriangularEquivalence shows the full protocol.

Theorem 2.2.1. *TriangularEquivalence is a complete and probabilistically sound interactive protocol. It requires $2n$ communications, and has Verifier cost $\mu(A) + \mu(B)$. The probability that the Verifier incorrectly accepts is at most $1/\#S$. There is a deterministic algorithm for the Prover in $O(mn^{\omega-1})$.*

Proof. If $\boldsymbol{T}$ exists, then the Prover can compute it, and answer with appropriate y_i, such that the Verifier check $\boldsymbol{Ay} = \boldsymbol{Bx}$ will always hold, hence the completeness of this protocol.

Now, if there is no such matrix, we have two distinct cases: either $\boldsymbol{AT} \neq \boldsymbol{B}$ for any $\boldsymbol{T}$, or there exists at least one such $\boldsymbol{T}$, but it is not triangular. In the rest of this proof, we replace the random values $x_1, \ldots, x_n$ by algebraically independent variables $X_1, \ldots, X_n$.

In the first case, if $\boldsymbol{AT} \neq \boldsymbol{B}$, then there is at least one inconsistent column in $\boldsymbol{B}$ (say the j-th). Thus, there exists a Farkas' certificate of inconsistency (a vector $\boldsymbol{z}$ such that $\boldsymbol{z}^T\boldsymbol{A} = \boldsymbol{0}$ and $\boldsymbol{z}^T\boldsymbol{B}_{*,j} \neq \boldsymbol{0}$ for that column. This means that there exists a vector $\boldsymbol{z}$ such that $\boldsymbol{z}^T\boldsymbol{Ay} = 0$ for any $\boldsymbol{y}$, but $\boldsymbol{z}^T\boldsymbol{B}[X_1, \ldots, X_n]^T$ is a not identically zero polynomial (at least the coefficient of X_j is non zero) of degree 1. Therefore, by the DeMillo-Lipton/Schwartz/Zippel lemma (Demillo and Lipton, 1978; Zippel, 1979; Schwartz, 1980), its evaluation will be zero with probability at most $1/\#\mathsf{S}$.

In the second case, $\boldsymbol{AT} = \boldsymbol{B}$ but $\boldsymbol{T}$ is not triangular. Since $\boldsymbol{A}$ is regular, there is a unique $n \times n$ matrix $\boldsymbol{T}$ (that is, $\boldsymbol{T} = \boldsymbol{A}^{-1}_{\text{left}}\boldsymbol{B}$, for any $\boldsymbol{A}^{-1}_{\text{left}}$ left inverse of $\boldsymbol{A}$). For the same reason, the equality $\boldsymbol{Ay} = \boldsymbol{Bx} = \boldsymbol{ATx}$ implies $\boldsymbol{y} = \boldsymbol{Tx}$. if $\boldsymbol{T}$ is not lower triangular, there is a row index i such that the entry $t_{i,j_m} \neq 0$ for some $j_m > i$. The test $\boldsymbol{y} = \boldsymbol{Tx}$ only succeeds if $y_i = \sum_{j=0}^{n} t_{i,j}x_j$. Now the Prover selects y_i before x_{j_m} is

Protocol 2.3: TriangularEquivalence

Public: $\boldsymbol{A}, \boldsymbol{B} \in \mathsf{F}^{m\times n}$ $\boldsymbol{A}$ is regular, and $m \geqslant n$. $\boldsymbol{T}$ exists
Certifies: There exists a matrix $\boldsymbol{T}$ such that $\boldsymbol{AT} = \boldsymbol{B}$

Prover		Verifier
1. Find a lower triangular matrix $\boldsymbol{T}$ such that $\boldsymbol{AT} = \boldsymbol{B}$		
2.		$x_i \xleftarrow{\$} \mathsf{S}$
	$\xleftarrow{x_1}$	
3. $y_1 = \boldsymbol{T}_{1,*} \begin{bmatrix} x_1 \\ 0 \\ \vdots \end{bmatrix}$		
	$\xrightarrow{y_1}$	
$\vdots$	$\vdots$	
4.		
	$\xleftarrow{x_n}$	
5. $y_n = \boldsymbol{T}_{n,*} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$		
	$\xrightarrow{y_n}$	
6.		$\boldsymbol{y} = [y_1 \dots y_n]^T$ $\boldsymbol{Ay} \stackrel{?}{=} \boldsymbol{Bx}$

revealed. Therefore, with probability no more than $1/\#\mathsf{S}$ the Verifier selects the field element $x_{j_m} = 1/t_{i,j_m}(y_i - \sum_{j \neq j_m} t_{i,j}x_j)$, and the test succeeds for a wrong $\boldsymbol{T}$.

Now for the complexities: the Verifier has to $\boldsymbol{Ay}$ and $\boldsymbol{By}$, whose computational cost is $\mu(\boldsymbol{A}) + \mu(\boldsymbol{B})$. The Prover has to compute $\boldsymbol{T}$, which can be done by a PLUQ elimination on $\boldsymbol{A}$ followed by a triangular system solve, both in $O(mn^{\omega-1})$. Then, $\boldsymbol{y} = \boldsymbol{Tx}$ requires only $O(n^2)$ operations, hence the cost given on the theorem statement. Finally, this verification protocol requires to transmit $\boldsymbol{x}$ and $\boldsymbol{y}$ two size n vectors, for a total communication volume of $2n$.

□

Note that the case where $\boldsymbol{T}$ is upper triangular works similarly: the Verifier needs to transmit $\boldsymbol{n}$ in reverse order, starting by x_n.

2.2.2 Generic rank profile-ness

Here, we propose a verification protocol which verifies whether a non-singular input matrix $\boldsymbol{A} \in \mathsf{F}^{m\times n}$ has generic rank profile. As this matrix needs to be non-singular, one needs to verify that beforehand, which can be achieved using the protocol proposed in (Dumas and Kaltofen, 2014, Fig. 2) or RankLowerBound thereafter. A matrix $\boldsymbol{A}$ has generic rank profile if and only if it has an LU decomposition $\boldsymbol{A} = \boldsymbol{LU}$, with $\boldsymbol{L}$ (resp. $\boldsymbol{U}$) a non-singular lower (resp. upper) triangular. This protocol picks random vectors $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\lambda}$ and asks the Prover to provide the vectors $\boldsymbol{z}^T = \boldsymbol{\lambda}^T\boldsymbol{L}, \boldsymbol{x} = \boldsymbol{U\phi}, \boldsymbol{y} = \boldsymbol{U\psi}$ on the fly, while receiving the coefficients of $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\lambda}$ one at a time. The Verifier will then check that these vectors satisfy the fundamental equations $\boldsymbol{z}^T\boldsymbol{x} = \boldsymbol{\lambda A\phi}$ and $\boldsymbol{z}^T\boldsymbol{y} = \boldsymbol{\lambda}^T\boldsymbol{A\psi}$. NonSingularGenericRankProfile gives an overview of this verification protocol.

Theorem 2.2.2. *NonSingularGenericRankProfile is a complete and probabilistically sound interactive protocol. It requires $6n$ communications, and has Verifier cost $\mu(\boldsymbol{A}) + 8n$. The probability that the Verifier incorrectly accepts is at most $1 - (1 - 1/\#S)^{2n}$. There is a deterministic algorithm for the Prover in $O(mn^{\omega-1})$.*

Proof. If $\boldsymbol{A}$ has generic rank profile, then $\boldsymbol{A} = \boldsymbol{LU}$, thus $\boldsymbol{z}^T \begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} = \boldsymbol{\lambda}^T\boldsymbol{LU} \begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix} = \boldsymbol{\lambda}^T\boldsymbol{A} \begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$, which means the Verifier will never reject the Prover's answer. This proves the completeness of NonSingularGenericRankProfile.

Now, for any i such that the $(i-1)\times(i-1)$ leading submatrix of $\boldsymbol{A}$ has generic rank profile, we can write a partial LU decomposition of $\boldsymbol{A}$ with the following notations:

$$\boldsymbol{A} = \underbrace{\begin{bmatrix}\boldsymbol{L}^{\langle i\rangle} & 0\\ \boldsymbol{B}^{\langle i\rangle} & \boldsymbol{I}_{n-i+1}\end{bmatrix}}_{\boldsymbol{B}} \underbrace{\begin{bmatrix}\boldsymbol{U}^{\langle i\rangle} & \boldsymbol{V}^{\langle i\rangle}\\ 0 & \boldsymbol{C}^{\langle i\rangle}\end{bmatrix}}_{\boldsymbol{C}}, \tag{2.2.1}$$

where $\boldsymbol{L}^{\langle i\rangle} \in \mathsf{F}^{(i-1)\times(i-1)}$ is non-singular lower triangular, $\boldsymbol{U}^{\langle i\rangle} \in \mathsf{F}^{(i-1)\times(i-1)}$ is non-singular upper triangular, $\boldsymbol{B}^{\langle i\rangle} \in \mathsf{F}^{(n-i+1)\times(i-1)}$, $\boldsymbol{V}^{\langle i\rangle} \in \mathsf{F}^{(i-1)\times(n-i+1)}$, $\boldsymbol{C}^{\langle i\rangle} \in \mathsf{F}^{(n-i+1)\times(n-i+1)}$.

Protocol 2.4: NonSingularGenericRankProfile

Public: $\boldsymbol{A} \in \mathsf{F}^{m \times n}$, non-singular, $\boldsymbol{A}$ has generic rank profile
Certifies: $\boldsymbol{A}$ indeed has generic rank profile

	Prover		Verifier
1.	$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$		
		for i from n to 1	
2.			$(\phi_i, \psi_i) \xleftarrow{\$} \mathsf{S}^2$
		$\xleftarrow{\phi_i, \psi_i}$	
3.	$\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} = \boldsymbol{U}\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$		
		$\xrightarrow{x_i, y_i}$	
4.			$\lambda_i \xleftarrow{\$} \mathsf{S}$
		$\xleftarrow{\lambda_i}$	
5.	$\boldsymbol{z}^T = \boldsymbol{\lambda}^T\boldsymbol{L}$		
		$\xrightarrow{z_i}$	
6.			$\boldsymbol{z}^T\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} \stackrel{?}{=} (\boldsymbol{\lambda}^T\boldsymbol{A})\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$

Let $\boldsymbol{v}^{[i\dots n]} = [v_i, \dots, v_n]^T \in \mathsf{F}^{1\times n-i+1}$ for a vector $\boldsymbol{v} \in \mathsf{F}^{n\times 1}$, and let

$$\eta_i = (\boldsymbol{\lambda}^{[i\dots n]})^T \boldsymbol{C}^{\langle i\rangle} \boldsymbol{\phi}^{[i\dots n]}, \quad \xi_i = (\boldsymbol{\lambda}^{[i\dots n]})^T \boldsymbol{C}^{\langle i\rangle} \boldsymbol{\psi}^{[i\dots n]}. \tag{2.2.2}$$

Consider the following predicate:

$$H_i\colon \eta_i{=}(\boldsymbol{z}^{[i\dots n]})^T \boldsymbol{x}^{[i\dots n]} \text{ and } \xi_i{=}(\boldsymbol{z}^{[i\dots n]})^T \boldsymbol{y}^{[i\dots n]}. \tag{2.2.3}$$

Note that H_1 is what the Verifier checks because then $\boldsymbol{B} = \boldsymbol{I}_n$. Note also that when $\boldsymbol{A}$ is in generic rank profile with $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$ and $\boldsymbol{z}^T = \boldsymbol{\lambda}^T\boldsymbol{L}$ and $\boldsymbol{x} = \boldsymbol{U}\phi$ and $\boldsymbol{y} = \boldsymbol{U}\psi$ then H_i is true for all i. To see this consider an LU-factorization $\boldsymbol{C}^{\langle i\rangle} = \bar{\boldsymbol{L}}^{\langle i\rangle}\bar{\boldsymbol{U}}^{\langle i\rangle}$
and the identity

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{L}^{\langle i\rangle} & 0 \\ \boldsymbol{B}^{\langle i\rangle} & \boldsymbol{I}_{n-i+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}^{\langle i\rangle} & \boldsymbol{V}^{\langle i\rangle} \\ 0 & \boldsymbol{C}^{\langle i\rangle} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}^{\langle i\rangle} & 0 \\ \boldsymbol{B}^{\langle i\rangle} & \bar{\boldsymbol{L}}^{\langle i\rangle} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}^{\langle i\rangle} & \boldsymbol{V}^{\langle i\rangle} \\ 0 & \bar{\boldsymbol{U}}^{\langle i\rangle} \end{bmatrix} = \boldsymbol{L}\boldsymbol{U}. \tag{2.2.4}$$

Then $(\boldsymbol{z}^{[i\dots n]})^T = (\boldsymbol{\lambda}^{[i\dots n]})^T \bar{\boldsymbol{L}}^{\langle i\rangle}$ and $\boldsymbol{x}^{[i\dots n]} = \bar{\boldsymbol{U}}^{\langle i\rangle}\boldsymbol{\phi}^{[i\dots n]}$ and $\boldsymbol{y}^{[i\dots n]} = \bar{\boldsymbol{U}}^{\langle i\rangle}\boldsymbol{\psi}^{[i\dots n]}$ verify H_i.

Note that the conditions are only tested by the Verifier for $i = 1$.

At stage i, let Λ_i, Φ_i and Ψ_i be variables for the random choices for λ_i, ϕ_i and ψ_i and Z_i be a variable for the Prover's choice of z_i. Then H_i in Equation (2.2.3) expands as:

$$\begin{cases} x_i Z_i &= \Big(d\Phi_i + \overbrace{\sum_{j=i+1}^{n} \boldsymbol{C}^{\langle i\rangle}_{1,j-i+1}\phi_j}^{e}\Big)\Lambda_i + a\Phi_i + f, \\ y_i Z_i &= \Big(d\Psi_i + \underbrace{\sum_{j=i+1}^{n} \boldsymbol{C}^{\langle i\rangle}_{1,j-i+1}\psi_j}_{g}\Big)\Lambda_i + a\Psi_i + h, \end{cases} \tag{2.2.5}$$

where $d = \boldsymbol{C}^{\langle i\rangle}_{1,1}$ and $a = \sum_{k=i+1}^{n} \lambda_k \boldsymbol{C}^{\langle i\rangle}_{k-i+1,1}$, or equivalently

$$\begin{bmatrix} -(d\Phi_i + e) & x_i \\ -(d\Psi_i + g) & y_i \end{bmatrix} \begin{bmatrix} \Lambda_i \\ Z_i \end{bmatrix} = \begin{bmatrix} a\Phi_i + f \\ a\Psi_i + h \end{bmatrix}. \tag{2.2.6}$$

Suppose now that $\boldsymbol{A}$ is not in generic rank profile, and let i_0 be minimal such that the leading $i_0 \times i_0$ minor of $\boldsymbol{A}$ is equal to 0. On any corresponding partial LU decomposition this means that $d = \boldsymbol{C}^{\langle i_0\rangle}_{1,1} = 0$. Furthermore, because $\boldsymbol{A}$ is assumed to be non-singular, there exist indices k_0 with $2 \leqslant k_0 \leqslant n - i_0 + 1$, and j_0 with $2 \leqslant j_0 \leqslant n - i_0 + 1$ such that $\boldsymbol{C}^{\langle i_0\rangle}_{k_0,1} \neq 0$ and $\boldsymbol{C}^{\langle i_0\rangle}_{1,j_0} \neq 0$.

We will now prove the two following statements:

1. H_{i_0} is false with probability $\geqslant (1 - 1/\#\mathsf{S})^4$;
2. If H_{i+1} is false then H_i is false with probability $\geqslant (1 - 1/\#\mathsf{S})^2$ for $1 \leqslant i < i_0$.

Informally, this means that the Prover cannot achieve H_{i_0} with any choice of returned values $x_1, \ldots, z_{i_0}$, with high probability and then this failure propagates with high probability to H_1 which is checked by the Verifier. By induction, this leads to a probability of $\geqslant (1 - 1/\#\mathsf{S})^{4+2(i_0-1)}$ that the Verifier check will fail when the matrix $\boldsymbol{A}$ is not in generic rank profile. Since $\boldsymbol{A}$ is non-singular, $i_0 \leqslant n-1$, and therefore this probability is $\geqslant (1 - 1/\#\mathsf{S})^{2n}$.

First, we prove Statement 1, that is the case when $d = 0$. The Verifier selects a random λ_{i_0}, and then the Prover a z_{i_0}. If the coefficient matrix in Equation (2.2.6) is non-singular, there is a unique solution for Λ_{i_0}, which the Verifier will choose with probability $\leqslant 1/\#\mathsf{S}$. Otherwise, the coefficient matrix is singular and the only way for the system to have a solution is that the determinant

$$\Delta = \begin{vmatrix} -e & a\Phi_{i_0} + f \\ -g & a\Psi_{i_0} + h \end{vmatrix} = -e(a\Psi_{i_0} + h) + g(a\Phi_{i_0} + f)$$

is equal to 0, which exactly happens in the three following cases:

a. $\begin{bmatrix} e & g \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$, which happens with probability $\leqslant 1/\#\mathsf{S}^2$ as $\boldsymbol{C}^{\langle i_0 \rangle}_{1,j_0} \neq 0$;

b. $a = 0$ (which happens with probability $\leqslant 1/\#\mathsf{S}$ as $\boldsymbol{C}^{\langle i_0 \rangle}_{k_0,1} \neq 0$) and $\begin{vmatrix} -e & f \\ -g & h \end{vmatrix} = 0$;

c. otherwise, $ea \neq 0$ or $ga \neq 0$ and Δ is a nonzero polynomial of degree 1 in Φ_{i_0}, Ψ_{i_0} and evaluates to 0 for the random choices ϕ_{i_0}, ψ_{i_0} with probability $\leqslant 1/\#\mathsf{S}$.

Overall, H_{i_0} is false with probability

$$\geqslant \big(1 - \frac{1}{\#\mathsf{S}^2}\big)\big(1 - \frac{1}{\#\mathsf{S}}\big)^3 \geqslant \big(1 - \frac{1}{\#\mathsf{S}}\big)^4 \geqslant 1 - \frac{4}{\#\mathsf{S}}$$

based on the random choices of the Verifier: ϕ_{j_0}, ψ_{j_0} yield $\begin{bmatrix} e & g \end{bmatrix} \neq \begin{bmatrix} 0 & 0 \end{bmatrix}$; λ_{k_0} yields $a \neq 0$; ϕ_{i_0}, ψ_{i_0} yield $\Delta \neq 0$; λ_{i_0} avoids the unique solution to Equation (2.2.6).

For Statement 2, consider the predicate H_i Equation (2.2.3) at $i < i_0$, that is $d \neq 0$. Similarly, if the coefficient matrix in Equation (2.2.6) is non-singular, there is a unique solution for Λ_i, which the Verifier will choose with probability $\leqslant 1/|S|$. Otherwise, the coefficient matrix is singular and the only way for the system to have a solution is that the following determinant is equal 0:

$$0 = \Delta = \begin{vmatrix} -(d\Phi_i + e) & a\Phi_i + f \\ -(d\Psi_i + g) & a\Psi_i + h \end{vmatrix} = (df - ae)\Psi_i - (dh - ag)\Phi_i - eh + gf.$$

We block decompose the bottom right block in the incomplete right factor in Equation (2.2.1) $\boldsymbol{C}^{\langle i \rangle} = \begin{bmatrix} d & r^T \\ s & W \end{bmatrix}$, where $d = \boldsymbol{C}^{\langle i \rangle}_{1,1} \neq 0$. We have $\boldsymbol{C}^{\langle i+1 \rangle} = W - \frac{1}{d} s r^T$. Now since $a = (\lambda^{[i+1\ldots n]})^T s$, $e = r^T \phi^{[i+1\ldots n]}$, we have $ae = (\lambda^{[i+1\ldots n]})^T s r^T \phi^{[i+1\ldots n]}$ and

$$\begin{aligned} f - \frac{ae}{d} &= (\lambda^{[i+1\ldots n]})^T C^{\langle i+1 \rangle} \phi^{[i+1\ldots n]} - (z^{[i+1\ldots n]})^T x^{[i+1\ldots n]} \\ &= \eta_{i+1} - (z^{[i+1\ldots n]})^T x^{[i+1\ldots n]}. \end{aligned}$$

Similarly, $h - \frac{ag}{d} = \xi_{i+1} - (z^{[i+1\ldots n]})^T x^{[i+1\ldots n]}$, and these two quantities are not equal to 0 simultaneously, for otherwise H_{i+1} would be true. Therefore Δ is a nonzero polynomial of degree 1 in Φ and Ψ. It is equal to 0 with probability $\leqslant 1/\#\mathsf{S}$. Overall, H_i is false with probability $\geqslant (1 - 1/\#\mathsf{S})^2$ based on the random choices for λ_i, ϕ_i and ψ_i made by the Verifier.

Finally, for the complexity, the Prover needs one Gaussian elimination to compute $\boldsymbol{LU}$ in time $O(n^\omega)$, then her extra work is just three triangular solve in $O(n^2)$. The extra communication is six vectors, $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$, and the Verifier's work is four dot-products and one multiplication by the initial matrix $\boldsymbol{A}$ (certifying the transposed to have a single matrix times $\boldsymbol{\lambda}$-vector product). □

2.2.3 LDUP decomposition

With NonSingularGenericRankProfile, when the input matrix does not have generic rank profile, any attempt to prove that it has generic rank profile will be detected with high probability, as this protocol is sound. However, when it does have generic rank profile, the Verifier will accept many possible vectors $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$: any scaling of z_i by α_i and x_i, y_i by $1/\alpha_i$ would be accepted, for any non zero constant α_i. This slack corresponds to our lack of specification on the diagonals in the LU decomposition. Indeed, for any diagonal matrix with non zero elements, $\boldsymbol{LD} \times \boldsymbol{D}^{-1}\boldsymbol{U}$ is also a valid LU decomposition and yields $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{z}$ scaled as above. As specifying these diagonals was not necessary to prove generic rank profile-ness, we left it as is for NonSingularGenericRankProfile.

However, for the determinant or the rank profile matrix verification protocols which follow (Section 2.4.1, InvertibleRankProfileMatrix) we need to ensure that this scaling is independent from the choice of vectors $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\lambda}$. Hence, we propose an updated protocol where $\boldsymbol{L}$ has to be unit triangular and the Prover has to first commit the main diagonal $\boldsymbol{D}$ of $\boldsymbol{U}$.

For a non-singular upper triangular matrix $\boldsymbol{U}$ with diagonal $\boldsymbol{D} = \mathrm{Diag}(d_1, \ldots, d_n)$, the matrix $\boldsymbol{U}_1 = \boldsymbol{D}^{-1}\boldsymbol{U}$ is unit triangular. Thus, for any $\psi = \begin{bmatrix} \boldsymbol{\psi}_1 \\ \widetilde{\boldsymbol{\psi}} \end{bmatrix} \in \mathsf{F}^{n\times 1}$: $\boldsymbol{U\psi} = \boldsymbol{DU}_1\boldsymbol{\psi} = \boldsymbol{D}\left(\boldsymbol{\psi} + \begin{bmatrix} \widetilde{U_1}\widetilde{\psi} \\ 0 \end{bmatrix}\right)$, where $\widetilde{\boldsymbol{U}_1} = (\boldsymbol{U}_1 - \boldsymbol{I}_n)_{\{1,\ldots,n-1\},\{2,\ldots,n\}}$ upper triangular in $\mathsf{F}^{n-1\times n-1}$. So the idea is that the Prover will commit $\boldsymbol{D}$ beforehand, and that within a generic rank profile verification protocol, the Verifier will only communicate $\widetilde{\boldsymbol{\phi}}, \widetilde{\boldsymbol{\psi}}$ and $\widetilde{\boldsymbol{\lambda}}$ to obtain $\overline{\boldsymbol{z}} = \widetilde{\boldsymbol{\lambda}}^T\widetilde{\boldsymbol{L}}$, $\overline{\boldsymbol{x}} = \widetilde{\boldsymbol{U}}_1\widetilde{\boldsymbol{\phi}}$ and $\overline{\boldsymbol{y}} = \widetilde{\boldsymbol{U}}_1\widetilde{\boldsymbol{\psi}}$, where $\widetilde{\boldsymbol{L}} = (\boldsymbol{L} - \boldsymbol{I}_n)_{\{2,\ldots,n\},\{1,\ldots,n-1\}}$ lower triangular in $\mathsf{F}^{n-1\times n-1}$. Then the Verifier will compute by himself the complete vectors. This ensures that $\boldsymbol{L}$ is unit triangular and that $\boldsymbol{U} = \boldsymbol{DU}_1$ with $\boldsymbol{U}_1$ unit triangular.

Finally, if an invertible matrix does not have generic rank profile, we note that it is also possible to incorporate the permutations, by committing them in the beginning and reapplying them to the matrix during the checks. The full verification protocol is given in Protocol LDUP.

Theorem 2.2.3. *Protocol LDUP is a complete and probabilistically sound interactive protocol. It requires $6n$ communications and has Verifier cost $\mu(\boldsymbol{A}) + 12n + o(n)$. The*

Protocol 2.5: Protocol LDUP

Public: $\boldsymbol{A} \in \mathsf{F}^{n\times n}$, non singular, $\boldsymbol{P} \in \mathsf{F}^{n\times n}$, $\boldsymbol{D} \in \mathsf{F}^{n\times n}$
Certifies: That $\boldsymbol{D}$ and $\boldsymbol{P}$ come from an LDUP decomposition of $\boldsymbol{A}$

	Prover		Verifier
	$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{U}_1\boldsymbol{P}$		
1.	$\widetilde{\boldsymbol{U}_1} = (\boldsymbol{U}_1 - \boldsymbol{I}_n)_{\{1,\ldots,n-1\},\{2,\ldots,n\}}$ $\widetilde{\boldsymbol{L}} = (\boldsymbol{L} - \boldsymbol{I}_n)_{\{2,\ldots,n\},\{1,\ldots,n-1\}}$		
2.			$\boldsymbol{P} \stackrel{?}{\in} \mathcal{S}_n, \boldsymbol{D} \stackrel{?}{\in} \mathcal{D}_n(\mathsf{F}^*)$
		for i from n to 2	
3.			$\phi_i, \psi_i \xleftarrow{\$} \mathsf{S}^2$
		$\xleftarrow{\phi_i, \psi_i}$	
4.	$\begin{bmatrix}\overline{\boldsymbol{x}} & \overline{\boldsymbol{y}}\end{bmatrix} = \widetilde{\boldsymbol{U}_1}\begin{bmatrix}\widetilde{\boldsymbol{\phi}} & \widetilde{\boldsymbol{\psi}}\end{bmatrix}$		
		$\xrightarrow{\overline{\boldsymbol{x}}_{i-1}, \overline{\boldsymbol{y}}_{i-1}}$	
5.			$\lambda_i \xleftarrow{\$} \mathsf{S}$
		$\xleftarrow{\lambda_i}$	
6.	$\overline{\boldsymbol{z}} = \widetilde{\boldsymbol{\lambda}}^T\widetilde{\boldsymbol{L}}$		
		$\xrightarrow{\overline{\boldsymbol{z}}_{i-1}}$	
			$\phi_1, \psi_1, \lambda_1 \xleftarrow{\$} \mathsf{S}^3$ $\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} =$ $\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix} + \begin{bmatrix}\overline{\boldsymbol{x}} & \overline{\boldsymbol{y}} \\ 0 & 0\end{bmatrix}$
7.			$\boldsymbol{z}^T = \left(\boldsymbol{\lambda}^T + \begin{bmatrix}\overline{\boldsymbol{z}}^T & 0\end{bmatrix}\right)$ $\boldsymbol{z}^T\boldsymbol{D}\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} \stackrel{?}{=}$ $(\lambda^T\boldsymbol{A})\boldsymbol{P}^T\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$

probability that the verifier incorrectly accepts is at most $1-(1-1/\#S)^{2n}$. *There is a deterministic algorithm for the Prover in* $O(n^\omega)$.

Proof. If $\boldsymbol{A}$ has an LDUP decomposition, then $\boldsymbol{A} = \boldsymbol{LUP} = \boldsymbol{LDU}_1\boldsymbol{P}$, so that for any choice of $\boldsymbol{\lambda}$ and $\boldsymbol{\psi}$ we have: $\boldsymbol{\lambda}^T\boldsymbol{AP}^T\boldsymbol{\psi} = \boldsymbol{\lambda}^T\boldsymbol{LDU}_1\boldsymbol{\psi}$, that is:

$$\begin{aligned}\boldsymbol{z}^T\boldsymbol{Dy} &= (\boldsymbol{\lambda}^T + \begin{bmatrix}\overline{\boldsymbol{z}}^T & 0\end{bmatrix})\boldsymbol{D}\left(\boldsymbol{\psi} + \begin{bmatrix}\overline{\boldsymbol{y}}\\ 0\end{bmatrix}\right)\\ &= \begin{bmatrix}\boldsymbol{\lambda}_1 & \widetilde{\boldsymbol{\lambda}}^T\end{bmatrix}\left(\boldsymbol{I} + \begin{bmatrix}0 & 0\\ \widetilde{\boldsymbol{L}} & 0\end{bmatrix}\right)\boldsymbol{D}\left(\begin{bmatrix}0 & \widetilde{\boldsymbol{U}_1}\\ 0 & 0\end{bmatrix} + \boldsymbol{I}\right)\begin{bmatrix}\boldsymbol{\psi}_1\\ \widetilde{\boldsymbol{\psi}}\end{bmatrix}.\end{aligned}$$

The same is true for $\boldsymbol{\lambda}$ and $\boldsymbol{\phi}$, which proves the completeness.

Now, the last part of Protocol LDUP is actually a verification that $\boldsymbol{AP}^T$ has generic rank profile, in other words that there exists lower and upper triangular matrices $\boldsymbol{L}^*$ and $\boldsymbol{U}^*$ such that $\boldsymbol{AP}^T = \boldsymbol{L}^*\boldsymbol{U}^*$. This verification is sound by Theorem 2.2.2. Next, the multiplication by the diagonal $\boldsymbol{D}$ is performed by the Verifier, in order to be actually convinced that there exists lower and upper triangular matrices $\boldsymbol{L}^*$ and $\boldsymbol{U}_1^*$ such that $\boldsymbol{AP}^T = \boldsymbol{L}^*\boldsymbol{DU}_1^*$. Finally, the construction of the vectors with the form $\boldsymbol{a} + \begin{bmatrix}\tilde{\boldsymbol{b}}\\ 0\end{bmatrix}$ is also done by the Verifier, in order to have in fact a guarantee that $\boldsymbol{L}^*$ and $\boldsymbol{U}_1^*$ are unit triangular.

Overall, if the matrix $\boldsymbol{AP}^T$ does not have generic rank profile, the Verifier will catch him with the probability of Theorem 2.2.2.

Finally, for the complexity bounds, the extra communications are the 6 vectors $\widetilde{\boldsymbol{\lambda}}$, $\widetilde{\boldsymbol{\phi}}$, $\widetilde{\boldsymbol{\psi}}$ and $\overline{\boldsymbol{z}}$, $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$. That is $6(n-1)$ field elements. The arithmetic computations of the Verifier are one multiplication by a diagonal matrix, 3 vector sums, 4 dot-products and one vector-matrix multiplication by $\boldsymbol{A}$ (for $(\boldsymbol{\lambda}^T\boldsymbol{A})$), that is $n+3(n-1)+4(2n-1)$. □

We also give the following proposition, which, while not being need to prove Protocol LDUP, gives some guaranties on the values of $\boldsymbol{D}$ and $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$.

Proposition 1. *In Protocol LDUP, if* $\boldsymbol{AP}^T$ *is not in generic rank profile, or else if the committed diagonal* $\boldsymbol{D}$ *does not correspond to the unique decomposition* $\boldsymbol{AP}^T = \boldsymbol{LDU}_1$ *or* $\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} \neq \boldsymbol{U}_1\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$ *or* $\boldsymbol{z}^T \neq \boldsymbol{\lambda}^T\boldsymbol{L}$, *then the verification will fail with probability* $\geqslant (1-\frac{1}{\#S})^{2n}$, *and therefore Protocol LDUP is sound.*

Proof. If $\boldsymbol{A}$ does not have an LDUP decomposition, either

(i) $\boldsymbol{AP}^T$ is not in generic rank profile, then Protocol LDUP will detect it with the probability of Theorem 2.2.3;

(ii) or the Prover could still try to send modified vectors $\overline{\boldsymbol{x}}$, $\overline{\boldsymbol{y}}$, $\overline{\boldsymbol{z}}$ or diagonal $\boldsymbol{D}$.

Let then $\boldsymbol{D}^*$, $\boldsymbol{x}^* = \phi + \begin{bmatrix}\overline{\boldsymbol{x}}^*\\ 0\end{bmatrix} = \boldsymbol{U}_1\boldsymbol{\phi}$, $\boldsymbol{y}^* = \psi + \begin{bmatrix}\overline{\boldsymbol{y}}^*\\ 0\end{bmatrix} = \boldsymbol{U}_1\boldsymbol{\psi}$ and $\boldsymbol{z}^* = \begin{bmatrix}\overline{\boldsymbol{z}}^*\\ 0\end{bmatrix} + \boldsymbol{\lambda} = \boldsymbol{L}^T\boldsymbol{\lambda}$ be the expected diagonal and vectors. Let also $i_0 \leqslant n$ be the largest index such that there is at least one discrepancy in d_{i_0}, $\overline{\boldsymbol{x}}_{i_0}$, $\overline{\boldsymbol{y}}_{i_0}$ or $\overline{\boldsymbol{z}}_{i_0}$ that makes at least one of them respectively different from $d^*_{i_0}$, $\overline{\boldsymbol{x}}^*_{i_0}$, $\overline{\boldsymbol{y}}^*_{i_0}$ or $\overline{\boldsymbol{z}}^*_{i_0}$ ($\overline{\boldsymbol{x}}_n = \overline{\boldsymbol{x}}^*_n = 0, \overline{\boldsymbol{y}}_n = \overline{\boldsymbol{y}}^*_n = 0, \overline{\boldsymbol{z}}_n = \overline{\boldsymbol{z}}^*_n = 0$ by default). Then H_i of Equation (2.2.3) is true for all i such that $n \geqslant i > i_0$, and thus

in particular H_{i_0+1} is true (H_{n+1} is true by default). Now, H_{i_0} is also true if and only if we have both:

$$\begin{cases} z_{i_0} d_{i_0} x_{i_0} = z^*_{i_0} d^*_{i_0} x^*_{i_0}, \\ z_{i_0} d_{i_0} y_{i_0} = z^*_{i_0} d^*_{i_0} y^*_{i_0}. \end{cases} \tag{2.2.7}$$

Indeed, H_{i_0} is $(\boldsymbol{z}^{[i_0\dots n]})^T \boldsymbol{D}^{[i_0\dots n]} \boldsymbol{x}^{[i_0\dots n]} = (\boldsymbol{z}^{*[i_0\dots n]})^T \boldsymbol{D}^{[i_0\dots n]} \boldsymbol{x}^{*[i_0\dots n]}$ and similarly H_{i_0+1} is $(\boldsymbol{z}^{[i_0+1\dots n]})^T \boldsymbol{D}^{[i_0+1\dots n]} \boldsymbol{x}^{[i_0+1\dots n]} = (\boldsymbol{z}^{*[i_0+1\dots n]})^T \boldsymbol{D}^{[i_0+1\dots n]} \boldsymbol{x}^{*[i_0+1\dots n]}$. Further, Equation (2.2.7), with $a = \overline{\boldsymbol{z}}^*_{i_0} d^*_{i_0} \overline{\boldsymbol{x}}^*_{i_0} - \overline{\boldsymbol{z}}_{i_0} d_{i_0} \overline{\boldsymbol{x}}_{i_0}$, and $b = \overline{\boldsymbol{z}}^*_{i_0} d^*_{i_0} \overline{\boldsymbol{y}}^*_{i_0} - \overline{\boldsymbol{z}}_{i_0} d_{i_0} \overline{\boldsymbol{y}}_{i_0}$, is equivalent to:

$$\begin{cases} \lambda_{i_0}\phi_{i_0}(d_{i_0} - d^*_{i_0}) + \lambda_{i_0}(d_{i_0}\overline{\boldsymbol{x}}_{i_0} - d^*_{i_0}\overline{\boldsymbol{x}}^*_{i_0}) + \phi_{i_0}(d_{i_0}\overline{\boldsymbol{z}}_{i_0} - d^*_{i_0}\overline{\boldsymbol{z}}^*_{i_0}) - a = 0, \\ \lambda_{i_0}\psi_{i_0}(d_{i_0} - d^*_{i_0}) + \lambda_{i_0}(d_{i_0}\overline{\boldsymbol{y}}_{i_0} - d^*_{i_0}\overline{\boldsymbol{y}}^*_{i_0}) + \psi_{i_0}(d_{i_0}\overline{\boldsymbol{z}}_{i_0} - d^*_{i_0}\overline{\boldsymbol{z}}^*_{i_0}) - b = 0. \end{cases} \tag{2.2.8}$$

However, $\lambda_{i_0}, \phi_{i_0}, \psi_{i_0}$ are chosen by the Verifier after d_{i_0}, $\overline{\boldsymbol{x}}_{i_0}$, $\overline{\boldsymbol{y}}_{i_0}$ and $\overline{\boldsymbol{z}}_{i_0}$ have been committed. Hence, on the one hand, if $d_{i_0} \neq d^*_{i_0}$ then the coefficient of λ_{i_0} in one of the two polynomials is not equal to 0 for a random ϕ_{i_0} with probability $\geqslant 1 - 1/\#\mathsf{S}^2$ and then that polynomial does not vanish for a random λ_{i_0} with probability $\geqslant (1 - 1/\#\mathsf{S}^2)(1 - 1/\#\mathsf{S})$, based on the random choices made by the Verifier, and H_{i_0} is violated.

On the other hand, if $d_{i_0} = d^*_{i_0} \neq 0$, they can be removed from Equation (2.2.8) which then simplifies (for $i_0 < n$) as:

$$\begin{cases} \lambda_{i_0}(\overline{\boldsymbol{x}}_{i_0} - \overline{\boldsymbol{x}}^*_{i_0}) + \phi_{i_0}(\overline{\boldsymbol{z}}_{i_0} - \overline{\boldsymbol{z}}^*_{i_0}) - (\overline{\boldsymbol{z}}^*_{i_0}\overline{\boldsymbol{x}}^*_{i_0} - \overline{\boldsymbol{z}}_{i_0}\overline{\boldsymbol{x}}_{i_0}) = 0, \\ \lambda_{i_0}(\overline{\boldsymbol{y}}_{i_0} - \overline{\boldsymbol{y}}^*_{i_0}) + \psi_{i_0}(\overline{\boldsymbol{z}}_{i_0} - \overline{\boldsymbol{z}}^*_{i_0}) - (\overline{\boldsymbol{z}}^*_{i_0}\overline{\boldsymbol{y}}^*_{i_0} - \overline{\boldsymbol{z}}_{i_0}\overline{\boldsymbol{y}}_{i_0}) = 0. \end{cases} \tag{2.2.9}$$

When there is at least one discrepancy with the expected vector coefficients, then Equation (2.2.9) can be considered as 2 polynomials that are not simultaneously identically zero. Thus they both vanish with probability $\leqslant 1/\#\mathsf{S}$ based on the random choices made by the Verifier. H_{i_0} is thus false with probability $\geqslant (1 - 1/\#\mathsf{S})$. As in the proof of Theorem 2.2.2, this propagates with high probability, to H_1 and the dishonest Prover is detected with probability $\geqslant (1 - 1/\#\mathsf{S})^{2(n-1)}(1 - 1/\#\mathsf{S}^2)(1 - 1/\#\mathsf{S}) \geqslant (1 - 1/\#\mathsf{S})^{2n}$. Overall, both (i), AP^T is not GRP, or (ii), $\boldsymbol{A}\boldsymbol{P}^T$ is GRP but some diagonal or vector elements is wrong, are detected with probability $\geqslant (1 - 1/\#\mathsf{S})^{2n}$. □

2.2.4 Column or row rank profile verification protocol

Here, we start by recalling two linear time and space verification protocol for an upper and lower bound on the rank of a matrix (RankUpperBound and RankLowerBound). These two protocols combined make a rank verification protocol. We chose to present the variants proposed in (Eberly, 2015, paragraph 2) of the protocol from (Dumas and Kaltofen, 2014). An upper bound R on the rank is certified by the ability for the Prover to generate any vector sampled from the image of $\boldsymbol{A}$ by a linear combination of R columns of $\boldsymbol{A}$ ($\|\gamma\|_H$ denotes the Hamming weight of the vector γ). A lower bound ρ is certified by the Prover ability to recover the unique coefficients of ρ linearly

Protocol 2.6: RankUpperBound

Public: $\boldsymbol{A} \in \mathsf{F}^{m \times n}$, a positive integer R
Certifies: R is an upper bound on the rank of $\boldsymbol{A}$

	Prover		Verifier
1.			$\boldsymbol{v} \xleftarrow{\$} \mathsf{S}^n, \boldsymbol{w} = \boldsymbol{A}\boldsymbol{v}$
		$\boldsymbol{w}$ ⟵	
2.	$\boldsymbol{A}\boldsymbol{\gamma} = \boldsymbol{w}$		
		⟶ $\boldsymbol{\gamma}$	
3.			$\|\boldsymbol{\gamma}\|_H \stackrel{?}{=} R$ $\boldsymbol{A}\boldsymbol{\gamma} \stackrel{?}{=} \boldsymbol{w}$

independent columns of $\boldsymbol{A}$. $LINSYS(r)$ denotes a complexity bound for solving a linear system of rank r.

Theorem 2.2.4. *RankUpperBound is a complete and probabilistically sound interactive protocol. It requires $m+n$ communications, and has Verifier cost $2\mu(\boldsymbol{A})+n$. The probability that the Verifier incorrectly accepts is at most $1/\#S$. There is a deterministic algorithm for the Prover in $LINSYS(r)$.*

Protocol 2.7: RankLowerBound

Public: $\boldsymbol{A} \in \mathsf{F}^{m \times n}$, a set $\mathcal{J}$ of ρ column indices of $\boldsymbol{A}$
Certifies: ρ is a lower bound on the rank of $\boldsymbol{A}$

	Prover		Verifier
1.			$\boldsymbol{\alpha} = \begin{cases} \alpha_{c_j} \xleftarrow{\$} \mathsf{S} \\ 0 \text{ otherwise} \end{cases}$ $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{\alpha}$
		$\boldsymbol{v}$ ⟵	
2.	Solve $\boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{v}$		
		⟶ $\boldsymbol{\beta}$	
3.			$\boldsymbol{\beta} \stackrel{?}{=} \boldsymbol{\alpha}$

Theorem 2.2.5. *RankLowerBound is a complete and probabilistically sound protocol. It requires $m + \rho$ communications, and has Verifier cost $\mu(A) + \rho$. The probability that the Verifier incorrectly accepts is at most $1/\#S$. There is a deterministic algorithm for the Prover in $LINSYS(r)$.*

Note that the communication in RankLowerBound involves sending m field elements for vector $\boldsymbol{v}$, and only ρ field elements for vector $\boldsymbol{\beta}$, as it has only ρ non-zero coefficients which positions are already indicated by $\mathcal{J}$. Hence the total communication cost is $m + \rho$.

We now consider a column rank profile verification protocol: the Prover is given a matrix $\boldsymbol{A}$, and answers the column rank profile of $\boldsymbol{A}$, $\mathcal{J} = (c_1, \ldots, c_r)$. In order to certify this column rank profile, we need to certify two properties:

1. the columns given by $\mathcal{J}$ are linearly independent; and
2. the columns given by $\mathcal{J}$ form the lexicographically smallest set of independent columns of $\boldsymbol{A}$.

Property 1 is verified by RankLowerBound, as it checks whether a set of columns are indeed linearly independent. Property 2 could be certified by successive applications of RankUpperBound: at step i, checking that the rank of $\boldsymbol{A}_{*,(0,\ldots,c_i-1)}$ is at most $i-1$ would certify that there is no column located between c_{i-1} and c_i in $\boldsymbol{A}$ which increases the rank of $\boldsymbol{A}$. Hence, it would prove the minimality of $\mathcal{J}$. However, this method requires $O(nr)$ communication space.

Instead, one can reduce the communication by seeding all challenges from a single n dimensional vector, and by compressing the responses with a random projection. The right triangular equivalence verification protocol plays here a central role, ensuring the lexicographic minimality of $\mathcal{J}$. More precisely, the Verifier chooses a vector $v \in \mathsf{F}^n$ uniformly at random and sends it to the Prover. Then, for each index $c_k \in \mathcal{J}$ the Prover computes the linear combination of the first $c_k - 1$ columns of $\boldsymbol{A}$ using the first $c_k - 1$ coefficients of $\boldsymbol{v}$ and has to prove that it can be generated from the $k-1$ columns $c_1, \ldots, c_{k-1}$. This means, find a vector $\boldsymbol{\gamma}^{(k)}$ solution to the system:

$$\begin{bmatrix} \boldsymbol{A}_{*,c_1} & \boldsymbol{A}_{*,c_2} & \ldots & \boldsymbol{A}_{*,c_{k-1}} \end{bmatrix} \gamma^{(k)} = \boldsymbol{A} \begin{bmatrix} v_1 \\ \vdots \\ v_{c_k-1} \\ 0 \\ \vdots \end{bmatrix}.$$

Equivalently, find an upper triangular matrix $\boldsymbol{\Gamma}$ such that:

$$\begin{bmatrix} \boldsymbol{A}_{*,c_1} & \boldsymbol{A}_{*,c_2} & \ldots & \boldsymbol{A}_{*,c_{r-1}} \end{bmatrix} \boldsymbol{\Gamma} = \boldsymbol{A} \underbrace{\begin{bmatrix} v_1 & v_1 & \cdots & \cdots & v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{c_1-1} & \vdots & \vdots & \vdots & \vdots \\ 0 & v_{c_2-1} & \vdots & \vdots & \vdots \\ 0 & 0 & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & v_{c_r-1} & \vdots \\ 0 & 0 & 0 & 0 & v_n \end{bmatrix}}_{\boldsymbol{V}}. \tag{2.2.10}$$

Note that $\boldsymbol{V} = \text{Diag}(v_1, \ldots, v_n)\boldsymbol{W}$ where $\boldsymbol{W} = [\mathbb{1}_{i<c_{j+1}}]_{i,j}$ (with $c_{r+1} = n+1$ by convention) In order to avoid having to transmit the whole $r \times r$ upper triangular matrix $\boldsymbol{\Gamma}$, the Verifier only checks a random projection $\boldsymbol{x}$ of it, using the triangular equivalence TriangularEquivalence. We then propose this verification protocol in ColumnRankProfile.

Protocol 2.8: ColumnRankProfile

Public: $\boldsymbol{A} \in \mathsf{F}^{m\times n}$, a set $\mathcal{J}$ of r column indices for $\boldsymbol{A}$
Certifies: $\mathcal{J}$ is the column rank profile of $\boldsymbol{A}$

	Prover		Verifier
1.		RankLowerBound$(\boldsymbol{A}, \mathcal{J})$	
2.			$\boldsymbol{v} \xleftarrow{\$} \mathsf{S}^n$ $\boldsymbol{W} = [\mathbb{1}_{i<c_{j+1}}] \in \mathsf{F}^{n\times r}$
		$\boldsymbol{v}$ ⟵	
	$\boldsymbol{V} = \text{Diag}(v_1, \ldots, v_n)\boldsymbol{W}$		
3.	$\boldsymbol{\Gamma}$ upper tri. s.t. $\boldsymbol{A}_{*,\{c_1,..,c_r\}}\boldsymbol{\Gamma} = \boldsymbol{AV}$		
4.		TriangularEquivalence$(\boldsymbol{A}_{*,\{c_1,..,c_r\}}, \boldsymbol{AV})$	
5.			Learns $\boldsymbol{x}, \boldsymbol{y}$, respective challenge and response from the previous call to TriangularEquivalence $\boldsymbol{z} =$ $\text{Diag}(v_1, \ldots, v_n)(\boldsymbol{Wx})$
6.			$\boldsymbol{z}_{c_j} = \boldsymbol{z}_{c_j} - \boldsymbol{y}_j$ for $j = 1..r$ $\boldsymbol{Az} \stackrel{?}{=} 0$

Theorem 2.2.6. *ColumnRankProfile is a complete and probabilistically sound protocol. It requires $m+n+3r$ communications, and has Verifier cost $2\mu(\boldsymbol{A})+n+3r$. The probability that the Verifier incorrectly accepts is at most $1/\#S$. There is a deterministic algorithm for the Prover in $O(mnr^{\omega-2})$.*

Proof. If $\mathcal{J}$ is the column rank profile of $\boldsymbol{A}$, ColumnRankProfile corresponds first to an application of Theorem 2.2.5 to certify that $\mathcal{J}$ is a set of independent columns. This protocol is perfectly complete. Second the protocol also uses challenges from RankUpperBound, which is perfectly complete, together with TriangularEquivalence, which is perfectly complete as well. The latter verification protocol is used on $\boldsymbol{A}_{*,\mathcal{J}}$, a regular submatrix, as $\mathcal{J}$ is a set of independent columns of $\boldsymbol{A}$. The final check then corre-

sponds to $\boldsymbol{A}(\boldsymbol{D}(\boldsymbol{W}\boldsymbol{x})) - \boldsymbol{A}_{*,\{c_1,..c_r\}}\boldsymbol{y} \stackrel{?}{=} 0$ and, overall, ColumnRankProfile is perfectly complete.

Now, if $\mathcal{J}$ is not the CRP of $\boldsymbol{A}$, then either the set of columns in $\mathcal{J}$ are not linearly independent, which will be caught by the Verifier with probability at least $1 - \frac{1}{\#\mathsf{S}}$, from Theorem 2.2.5, or $\mathcal{J}$ is not lexicographically minimal, or the rank of $\boldsymbol{A}$ is not r. If the rank is wrong, it will not be possible for the Prover to find a suitable $\boldsymbol{\Gamma}$. This will be caught by the Verifier verifier with probability $1-\frac{1}{\#\mathsf{S}}$, from Theorem 2.2.1. Finally, if $\mathcal{J}$ is not lexicographically minimal, there exists at least one column $c_k \notin \mathcal{J}, c_i < c_k < c_{i+1}$ for some fixed i such that $\{c_1, \ldots, c_i\} \cup \{c_k\}$ form a set of linearly independent columns of $\boldsymbol{A}$. This means that $\mathrm{rank}(A_{*,1,\ldots,c_{i+1}-1}) = i+1$, whereas it was expected to be i. Thus, the Prover cannot reconstruct a suitable triangular $\boldsymbol{\Gamma}$ and this will be detected by the Verifier also with probability $1 - \frac{1}{\#\mathsf{S}}$, as shown in Theorem 2.2.1.

The Prover's time complexity is that of computing a PLUQ decomposition of $\boldsymbol{A}$. The transmission of $\boldsymbol{v}, \boldsymbol{x}$ and $\boldsymbol{y}$ yields a communication cost of $n+2r$, which adds up to the $m+r$ communication cost of RankLowerBound. Finally, in addition to RankLowerBound, the Verifier computes $\boldsymbol{W}\boldsymbol{x}$ as a prefix sum with $r-1$ additions, multiplies it by $\boldsymbol{D}$, then subtracts y_i at the r correct positions and finally multiplies by $\boldsymbol{A}$ for a total cost bounded by $2\mu(\boldsymbol{A}) + n + 3r - 1$. □

2.2.5 Rank profile matrix verification protocol

We propose an interactive protocol for the rank profile matrix based on (Dumas et al., 2017c, Algorithm 4): first computing the row and column support of the rank profile matrix, using ColumnRankProfile twice for the row and column rank profiles, then computing the rank profile matrix of the invertible submatrix of $\boldsymbol{A}$ lying on this grid.

In the following we then only focus on a verification protocol for the rank profile matrix of an invertible matrix. It relies on an LUP decomposition that reveals the rank profile matrix. From Lemma 2.1.3, this is the case if and only if $\boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P}$ is upper triangular. InvertibleRankProfileMatrix thus gives an interactive protocol that combines Protocol LDUP for a LDUP decomposition with a check that $\boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P}$ is upper triangular. The latter is achieved by TriangularEquivalence showing that $\boldsymbol{P}^T$ and $\boldsymbol{P}^T\boldsymbol{U}$ are left upper triangular equivalent, but since $\boldsymbol{U}$ is unknown to the Verifier, the verification is done on a random right projection with the vector $\boldsymbol{\phi}$ used in Protocol LDUP.

Theorem 2.2.7. *InvertibleRankProfileMatrix is a complete and probabilistically sound protocol. It requires $8n$ communication and has Verifier cost $\mu(\boldsymbol{A}) + 16n + o(n)$. The probability the Verifier incorrectly accepts is $1 - (1 - 1/\#S)^{2n}$. There is a deterministic algorithm for the Prover in $O(n^\omega)$.*

Proof. If $\boldsymbol{P}$ is the rank profile matrix of $\boldsymbol{A}$, then $\boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P}$ is upper triangular, and this protocol is complete by the completeness of its sub-protocols

Now, if $\boldsymbol{P}$ is not the rank profile matrix of $\boldsymbol{A}$ and $\overline{\boldsymbol{U}} = \boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P}$ is not upper triangular, then let (i, j) be the lexicographically minimal coordinates such that $i > j$ and $\overline{\boldsymbol{U}}_{i,j} \neq 0$.

Protocol 2.9: InvertibleRankProfileMatrix

Public: $\boldsymbol{A} \in \mathsf{F}^{m\times n}$, invertible, $\boldsymbol{P} \in \mathsf{F}^{n\times n}$, $\boldsymbol{D} \in \mathsf{F}^{n\times n}$
Certifies: $\boldsymbol{P}$ is the rank profile matrix of $\boldsymbol{A}$

	Prover		Verifier
1.	$\boldsymbol{A} = \boldsymbol{LDUP}$, with public $\boldsymbol{P}$ and $\boldsymbol{D}$		
2.			$\boldsymbol{P} \stackrel{?}{\in} \mathcal{S}_n, \boldsymbol{D} \stackrel{?}{\in} \mathcal{D}_n(\mathsf{F})$
3.		TriangularEquivalence$(\boldsymbol{P}^T, \boldsymbol{P}^T\boldsymbol{U})$	
4.	$\overline{\boldsymbol{U}} = \boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P}$		
		for i from 1 to n:	
5.			$e_i \stackrel{\$}{\leftarrow} \mathsf{S}$
		$\xleftarrow{e_1, \ldots, e_n}$	
6.	$\boldsymbol{f}^T = \boldsymbol{e}^T\overline{\boldsymbol{U}}$		
		$\xrightarrow{f_1, \ldots, f_n}$	
7.		Protocol LDUP$(\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{P})$	
8.			Learns $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{x}, \boldsymbol{y}$ from the call to Protocol LDUP $\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix}$ is $\boldsymbol{U}\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$
9.			$\boldsymbol{e}^T\boldsymbol{P}^T\boldsymbol{x} \stackrel{?}{=} \boldsymbol{f}\boldsymbol{P}\boldsymbol{\phi}$

Now either $\begin{bmatrix}\boldsymbol{x} & \boldsymbol{y}\end{bmatrix} \neq \boldsymbol{U}\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix}$, and the verification will then fail to detect it with probability less than $(1-\frac{1}{\#\mathsf{S}})^{2n}$, from Proposition 1. Or one can write $\boldsymbol{e}^T\boldsymbol{P}^T\boldsymbol{x} - \boldsymbol{f}^T\boldsymbol{P}^T\boldsymbol{\phi} = (\boldsymbol{e}^T\overline{\boldsymbol{U}} - \boldsymbol{f}^T)\boldsymbol{P}\boldsymbol{\phi} = 0$.

If

$$\boldsymbol{e}^T\boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P} - \boldsymbol{f}^T = 0. \tag{2.2.11}$$

is not satisfied, then a random ϕ will fail to detect it with probability less than $\frac{1}{\#\mathsf{S}}$, since $\boldsymbol{e}, \overline{\boldsymbol{U}}$ and $\boldsymbol{f}$ are set before choosing for $\boldsymbol{\phi}$. At the time of committing f_j, the value of e_i is still unknown, hence f_j is constant in the symbolic variable E_i. Thus the j-th coordinate in (2.2.11) is a nonzero polynomial in E_j and therefore vanishes with probability $1/\#\mathsf{S}$ when sampling the values of $\boldsymbol{e}$ uniformly. Hence, overall if $\boldsymbol{P}^T\boldsymbol{U}\boldsymbol{P}$ is not upper triangular, the verification will detect it with probability $\geqslant (1-\frac{1}{\#\mathsf{S}})^{2n}$.

The Verifier's cost is that of Protocol LDUP with two additional dot products for the last step, which is $\mu(\boldsymbol{A}) + 16n + o(n)$. Similarly, the communication cost is that of Protocol LDUP plus the size of $\boldsymbol{e}$ and $\boldsymbol{f}$ for a total of $8n$. The Prover remains unchanged. □

Finally, we use (Dumas et al., 2017c, Algorithm 4) to certify the rank profile matrix of any matrix, even a singular one. To do so, we need to verify the row rank profile and the column rank profile of the input matrix, which can be done with two applications of ColumnRankProfile. Then, we certify the rank profile matrix of the $r \times r$ selection of lexicographically minimal independent rows and columns we obtained before. This is done by an application of InvertibleRankProfileMatrix. We now define $\mathcal{E}_{m,\{i_1,\ldots,i_n\}}$ as the $m \times n$ matrix whose j-th column is the i_j-th vector of the m-dimensional canonical basis. This verification protocol is detailed in RankProfileMatrix, in the case where $m \leqslant n$. If $n < m$, one should first apply ColumnRankProfile on $\boldsymbol{A}$ to compute its column rank profile, and then apply the verification steps of the same protocol for the row rank profile of $\boldsymbol{A}$. The application of InvertibleRankProfileMatrix remains unchanged.

Theorem 2.2.8. *RankProfileMatrix is a complete and probabilistically sound protocol. It requires $m+n+\min(m,n)+15r$ communications and has Verifier cost $4\mu(\boldsymbol{A})+m+n+21r$. The probability the Verifier incorrectly accepts is $1-(1-1/\#S)^{2n}$. The is a deterministic algorithm for a Prover, which costs $O(mnr^{\omega-2})$.*

Proof. If the Prover is honest, $\mathcal{I}$ is the row rank profile of $\boldsymbol{A}$ and $\mathcal{J}$ is the column rank profile of A. Then, the application of InvertibleRankProfileMatrix will output the correct rank profile matrix of $\boldsymbol{A}_{\mathcal{I},\mathcal{J}}$ which will lead the Verifier to the correct rank profile matrix of $\boldsymbol{A}$, as described in (Dumas et al., 2017c, Theorem 37). Note that one only needs to verify the lower bound on the rank of $\boldsymbol{A}$ once, which is why ColumnRankProfile is fully executed once, while the second run only verifies that the committed rank profile is a rank profile indeed.

Now, for the soundness, Prover has a probability $\geqslant 1 - 1/\#\mathsf{S}$ to be caught when cheating while running ColumnRankProfile, and a probability $\geqslant (1-\frac{1}{\#\mathsf{S}})^{2n}$ to be caught when cheating while running InvertibleRankProfileMatrix. Overall, this makes a probability $\geqslant (1-\frac{1}{\#\mathsf{S}})^{2n}$ for the Verifier to catch a cheating Prover during the execution of RankProfileMatrix.

For the complexities, Prover time complexity is bounded by the complexity of performing a PLUQ decomposition of the input matrix, $O(mnr^{\omega-2})$. The Verifier complexity is the one of one full application of ColumnRankProfile and one application of ColumnRankProfile without applying RankLowerBound, which makes $3\mu(\boldsymbol{A})+n+m+5r$, plus one application of InvertibleRankProfileMatrix over an $r \times r$ matrix for a cost of $\mu(\boldsymbol{A}) + 16r + o(r)$, the computation of $\boldsymbol{\mathcal{R}}_A$ only consists of memory operations, hence a total cost of $4\mu(A) + m + n + 21r + o(r)$ field operations. Communication space is computed as follows: a full application of ColumnRankProfile on $\boldsymbol{A}$ if $m \geqslant n$, on $\boldsymbol{A}^T$ otherwise, an application of the same Protocol without the underlying RankLowerBound which makes $n + m + min(m, n) + 7r$ and the same application of InvertibleRankProfileMatrix as above, for a cost of $8r$, hence a total communication space of $m + n + min(m, n) + 15r$. □

Protocol 2.10: RankProfileMatrix

Public: $\boldsymbol{A} \in \mathsf{F}^{m\times n}, m \leqslant n$, $\boldsymbol{\mathcal{R}}_{\boldsymbol{A}} \in \mathsf{F}^{m\times n}$
Certifies: $\boldsymbol{\mathcal{R}}_{\boldsymbol{A}}$ is the rank profile matrix of $\boldsymbol{A}$

	Prover		Verifier
1.			Extract the row support of $\boldsymbol{\mathcal{R}}_{\boldsymbol{A}}$ as $\mathcal{I}$, its column support as $\mathcal{J}$
2.		ColumnRankProfile$(\boldsymbol{A}^T, \mathcal{I})$	
3.		ColumnRankProfile$(\boldsymbol{A}, \mathcal{J})$ without its Step 1	
4.	$\mathcal{R}_r = \boldsymbol{A}_{\mathcal{I},\mathcal{J}}$		
		$\xrightarrow{\mathcal{R}_r}$	
5.		InvertibleRankProfileMatrix$(\boldsymbol{A}_{\mathcal{I},\mathcal{J}}, \mathcal{R}_r)$	
6.			$\mathcal{R}_{\boldsymbol{A}} \stackrel{?}{=} \mathcal{E}_{m,\mathcal{I}}\mathcal{R}_r\mathcal{E}^T_{n,\mathcal{J}}$

2.3 Constant rounds verification protocols

As previously mentioned in Section 1.1.3, if the network has a poor latency, it is interesting to reduce the number of rounds atop the overall communication volume. We therefore propose in this section a protocol with a constant number of rounds for triangular equivalence, still preserving Prover efficiency as well as linear communication

volume and Verifier cost. This applies directly, as previously shown, to row or column rank profiles. However, it fails to apply to the generic rank profile, at least in a straightforward manner, and we were unable to produce such a verification protocol in constant rounds for this task.

2.3.1 Representative Laurent polynomial of a matrix

Following a technique in (Mulmuley, 1987), we first define the *representative Laurent polynomial*, $P_A(X)$ of an $m \times n$ matrix $\boldsymbol{A}$ as :

$$P_A(X) = \begin{bmatrix} 1 & X & X^2 & \dots & X^{m-1} \end{bmatrix} \cdot \boldsymbol{A} \cdot \begin{bmatrix} 1 \\ X^{-1} \\ \vdots \\ X^{1-n} \end{bmatrix} = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{i,j} X^{i-j}$$

Therefore, if a matrix is lower triangular, then its representative Laurent polynomial cannot have negative powers and it is therefore a polynomial of degree at most $m-1$. The converse is not true, consider for instance an upper diagonal with two opposite coefficients : $A_{i,i+1} = 0$ for all i except $A_{1,2} = -A_{2,3}$. Generically, if one pre-multiplies $\boldsymbol{A}$ on the right by a random non-zero diagonal matrix, these cancellations will not occur as in general $d_1 A_{1,2} \neq -d_2 A_{2,3}$ unless $A_{1,2} = A_{2,3} = 0$.

2.3.2 Constant rounds triangular equivalence verification protocol

From this representation, we can obtain a triangular equivalence protocol that requires only a constant number of rounds: the Prover commits the Laurent polynomial of the triangular matrix, then the Verifier will evaluate it a random and compare this to the actual projections. However, the field size must be sufficiently large so that the polynomial identity testing does not fail. The full protocol is given in ConstantRoundTriangularEquivalence. It requires that the Prover solves a regular system (this is checked deterministically by reapplying the resulting vector), and a diagonal preconditioning on the triangular matrix needs to be applied to prevent cancellations.

Theorem 2.3.1. *ConstantRoundTriangularEquivalence is a complete and probabilistically sound protocol. It requires $3n+1$ communications and has Verifier cost $\mu(\boldsymbol{A}) + \mu(\boldsymbol{B}) + 7n$. The probability the Verifier incorrectly accepts is at most $2(n-1)/\#S$. There is an algorithm for the Prover in $O(mn^{\omega-1})$.*

Proof. Let $\boldsymbol{x} = \boldsymbol{D} \begin{bmatrix} 1 \\ \lambda^{-1} \\ \vdots \\ \lambda^{1-n} \end{bmatrix}$. If $\boldsymbol{L}$ exists, as $\boldsymbol{A}$ is regular, there is only one solution $\boldsymbol{y}$ to $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{B}\boldsymbol{x}$, and $\boldsymbol{y} = \boldsymbol{L}\boldsymbol{x}$. Therefore $\begin{bmatrix} 1 & \lambda & \dots & \lambda^{n-1} \end{bmatrix} \cdot \boldsymbol{y} = \begin{bmatrix} 1 & \lambda & \dots & \lambda^{n-1} \end{bmatrix} \cdot$

Protocol 2.11: ConstantRoundTriangularEquivalence

Public: $\boldsymbol{A}, \boldsymbol{B}, \in \mathsf{F}^{m \times n}$, $\boldsymbol{A}$ is regular, $m \geqslant n$, there exists a lower triangular matrix $\boldsymbol{L}$ s.t. $\boldsymbol{AL} = \boldsymbol{B}$
Certifies: $\boldsymbol{L}$ indeed exists

	Prover		Verifier
1.	Find $\boldsymbol{L}$ lower triangular s.t. $\boldsymbol{AL} = \boldsymbol{B}$		
2.			$\boldsymbol{D} \xleftarrow{\$} \mathcal{D}_n(\mathsf{S} \backslash \{0\})$
		$\xleftarrow{\boldsymbol{D}}$	
3.	$g(X) = P_{LD}(X)$		
		$\xrightarrow{g(X)}$	
4.			$g \overset{?}{\in} \mathsf{F}[X]_{\geqslant n-1}$ $\lambda \xleftarrow{\$} \mathsf{S}$
		$\xleftarrow{\lambda}$	
5.	$\boldsymbol{y}$, s.t. $\boldsymbol{A} \cdot \boldsymbol{y} = \boldsymbol{B} \cdot \boldsymbol{D} \cdot \begin{bmatrix} 1 \\ \lambda^{-1} \\ \vdots \\ \lambda^{1-n} \end{bmatrix}$		
		$\xrightarrow{\boldsymbol{y}}$	
6.			$\boldsymbol{A} \cdot \boldsymbol{y} \overset{?}{=} \boldsymbol{B} \cdot \boldsymbol{D} \cdot \begin{bmatrix} 1 \\ \lambda^{-1} \\ \vdots \\ \lambda^{1-n} \end{bmatrix}$ $g(\lambda) \overset{?}{=}$ $\begin{bmatrix} 1 & \lambda & \dots & \lambda^{n-1} \end{bmatrix} \cdot \boldsymbol{y}$

$\boldsymbol{LD}\begin{bmatrix}1\\ \lambda^{-1}\\ \vdots\\ \lambda^{1-n}\end{bmatrix} = P_{LD}(\lambda)$ which proves the correctness.

Now, if $\boldsymbol{L}$ does not exist:

- As $\boldsymbol{A}$ is regular, there is only one solution $\boldsymbol{y}$ to $\boldsymbol{Ay} = \boldsymbol{Bx}$, thus that check ensures that $\boldsymbol{y}$ is correct, unless not all columns in $\boldsymbol{B}$ are in the column space of $\boldsymbol{A}$, which is handled as in the proof of Theorem 2.2.1.
- If $\boldsymbol{L}$ is not triangular then its upper part is not identically zero. Therefore by considering $\boldsymbol{D}$ as a diagonal matrix of indeterminates, at least one coefficient of negative degree of the representative rational fraction $\boldsymbol{LD}$ will be non identically zero. As those are of degree 1 in the indeterminates of $\boldsymbol{D}$, for a random diagonal D, the representative rational fraction of $\boldsymbol{LD}$ will not be a polynomial with probability at least $1 - \frac{1}{\#\mathsf{S}-1}$.
- If g is not a polynomial of degree at most $n-1$, it is not the representative of a triangular matrix.
- If g is not the representative polynomial of $\boldsymbol{LD}$ then by the DeMillo-Lipton/ Schwartz/Zippel lemma (Demillo and Lipton, 1978; Zippel, 1979; Schwartz, 1980), its evaluation at λ will fail with probability $1 - 2\frac{n-1}{\#\mathsf{S}}$ (since $X^{n-1}(g - P_{LD})(X)$ is a polynomial of degree at most $2(n-1)$).

For the complexity, the Prover computes $\boldsymbol{L}$, in $O(mn^{\omega-1})$. Then $P_{LD}(X)$ requires one pass over the coefficients of $\boldsymbol{L}$, and finally $\boldsymbol{y} = \boldsymbol{LD}\begin{bmatrix}1\\ \lambda^{-1}\\ \vdots\\ \lambda^{1-n}\end{bmatrix}$. The communication cost is $\boldsymbol{D}$, $g(X)$, $\boldsymbol{y}$ all of size n, and λ. The Verifier cost is, $\mu(\boldsymbol{A})+\mu(\boldsymbol{B})$ to apply $\boldsymbol{A}$ and $\boldsymbol{B}$, as well as $2n-3$ to compute $\begin{bmatrix}1 & \lambda & \dots & \lambda^{n-1}\end{bmatrix}$ and their inverses, $n-2$ to multiply by $\boldsymbol{D}$, $2(n-1)$ to evaluate g, and $2(n-1)$ to compute the dotproduct $\begin{bmatrix}1 & \lambda & \dots & \lambda^{n-1}\end{bmatrix}\cdot\boldsymbol{y}$. □

2.3.3 Constant round verification protocols for the row and column rank profiles

Now we can combine the lower rank RankLowerBound, with the constant-round ConstantRoundTriangularEquivalence for triangular equivalence, as a replacement of TriangularEquivalence, within the column rank profile ColumnRankProfile, in order to get the constant-round ConstantRoundColumnRankProfile for column rank profile. It remains Prover efficient, linear in communication volume and Verifier time.

Corollary 2.3.2. *ConstantRoundColumnRankProfile is a complete and probabilistically sound interactive protocol. It requires $m + n + 4r + 1$ communications and has Verifier cost $2\mu(A) + n + 9r$.*

Protocol 2.12: ConstantRoundColumnRankProfile

Public: $\boldsymbol{A} \in \mathsf{F}^{m \times n}$, $\mathcal{J}$ a set of r column indices of $\boldsymbol{A}$
Certifies: $\mathcal{J}$ is the column rank profile of $\boldsymbol{A}$

	Prover		Verifier
1.		RankLowerBound$(\boldsymbol{A}, \mathcal{J})$	
2.			$\boldsymbol{v} \xleftarrow{\$} \mathsf{S}^n$ $\boldsymbol{W} = [\mathbb{1}_{i<c_{j+1}}] \in \mathsf{F}^{n \times r}$ $\boldsymbol{V} = \mathrm{Diag}(v_1, \ldots, v_n)\boldsymbol{W}$
		$\xleftarrow{\boldsymbol{v}}$	
3.	$\boldsymbol{W} = [\mathbb{1}_{i<c_{j+1}}] \in \mathsf{F}^{n \times r}$ $\boldsymbol{V} = \mathrm{Diag}(v_1, \ldots, v_n)\boldsymbol{W}$		
4.		ConstantRoundTriangularEquivalence$(\boldsymbol{A}_{*,\mathcal{J}}, \boldsymbol{AV})$	
5.			Learns $\boldsymbol{D}$, g, λ and $\boldsymbol{y}$ from the call to ConstantRoundTriangularEquivalence $\boldsymbol{z} = \mathrm{Diag}(v_i)\boldsymbol{W}\boldsymbol{D}\begin{bmatrix} 1 \\ \lambda^{-1} \\ \vdots \\ \lambda^{1-r} \end{bmatrix}$
6.			$\boldsymbol{z}_{c_j} = \boldsymbol{z}_{c_j} - \boldsymbol{y}_j$ for $j = 1..r$ $\boldsymbol{Az} \stackrel{?}{=} \boldsymbol{0}$ $g(\lambda) \stackrel{?}{=} \begin{bmatrix} 1 & \lambda & \ldots & \lambda^{r-1} \end{bmatrix} \boldsymbol{y}$

2.4 Some additional verification protocol

In this final section, we present two new protocols for classical linear algebra problems, the determinant of a matrix and the signature of a symmetric matrix. Using the protocols presented before, our new verification protocols achieve better practical Prover costs than the existing ones for the same problems.

2.4.1 Linear communication verification protocols for the determinant

Existing protocols for the determinant are either optimal for the Prover in the dense case, using the strategy of (Kaltofen et al., 2011, Theorem 5) over a PLUQ decomposition, but quadratic in communication; or linear in communication, using (Dumas et al., 2016, Theorem 14), but using a reduction to the characteristic polynomial. In the sparse case the determinant and the characteristic polynomial both reduce to the same minimal polynomial computations and therefore the latter protocol is currently optimal for the Prover. Now in the dense case, while the determinant and characteristic polynomial both reduce to matrix multiplication, the determinant, via a single PLUQ decomposition is more efficient in practice (Pernet and Storjohann, 2007). Therefore, we propose here an alternative in the dense case: use only one PLUQ decomposition for the Prover while keeping linear extra communications and $O(n) + \mu(\boldsymbol{A})$ operations for the Verifier, for a square matrix $\boldsymbol{A}$ of size n. The idea is to extract the information of an LDUP decomposition without communicating it: one uses Protocol LDUP for $\boldsymbol{A} = \boldsymbol{LDUP}$ with $\boldsymbol{L}$ and $\boldsymbol{U}$ unit diagonal, but kept on the Prover side, and then the Verifier only has to compute $\det(\boldsymbol{A}) = \det(\boldsymbol{D})\det(\boldsymbol{P})$, with $n-1$ additional field operations.

Corollary 2.4.1. *For an $n \times n$ matrix, there exists a sound and perfectly complete protocol for the determinant over a field using less than $8n$ extra communications and with computational cost for the Verifier bounded by $\mu(\boldsymbol{A}) + 13n + o(n)$.*

As a comparison, the protocol of (Dumas et al., 2016, Theorem 14) reduces to CHARPOLY instead of PLUQ for the Prover. In theory, both approaches are similar: Prover asymptotic cost is the same, $O(n^\omega)$, Verifier time is the same and communication volume is $8n$ for PLUQ and $5n$ for CHARPOLY. Moreover, CHARPOLY is better than PLUQ on the number of rounds (constant rounds against $O(n)$) while PLUQ works on smaller fields than CHARPOLY (minimum size 2 against n^2). However, PLUQ is much more efficient in practice. Table 2.1 shows timings for PLUQ and CHARPOLY computations using the FFLAS-FFPACK library (version 2.3.1) (The FFLAS-FFPACK group, 2019) on an Intel i7-6700U@3.4GHz. Communications were measured between two workstations over an Ethernet Cat. 6@1GB/s network cable. On $50k \times 50k$ matrices, the characteristic polynomial of a random dense matrix can be computed in about 134 minutes, while it takes approximately 5.5 less time to perform the PLUQ decomposition (24 minutes). The Verifier time, which is the one of performing a matrix-vector multiplication (the `fgemv` routine) is approximately 1s.

Dimension	$2k$	$10k$	$50k$
PLUQ	0.28s	17.99s	1448.16s
CHARPOLY	1.96s	100.37s	8047.56s
Linear comm.	0.50s	0.50s	0.50s
Quadratic comm.	1.50s	7.50s	222.68s
`fgemv`	0.0013s	0.038s	1.03s

Table 2.1: Communication of 64 bit words versus computation modulo 131071

2.4.2 Verification protocol for the signature of an integer matrix

The signature of a symmetric matrix is the triple (n_+, n_-, n_0) indicating the number of positive, negative, and zero eigenvalues, respectively. Just like (Dumas and Kaltofen, 2014, Theorem 5), the idea is that the Prover commits the list of eigenvalues, and then certifies it modulo a Verifier chosen prime. This works directly for the signature algorithm in (Kaltofen et al., 2011, Corollary 1) together with the CHARPOLY protocol of (Dumas et al., 2016, Theorem 14). As in Section 2.4.1, in the dense case we propose here to replace the CHARPOLY computation with a symmetric Gaussian elimination.

Over the rationals, an algorithm for the Prover could be to first compute and certify the rank of $\boldsymbol{A}$, and to compute a permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$ has generic rank profile: for instance compute a $\boldsymbol{P}\boldsymbol{L}_p\Delta_p\boldsymbol{L}_p^T\boldsymbol{P}^T$ factorization modulo a sufficiently large prime p. A symmetric matrix always has such a decomposition, where Δ is a block diagonal matrix with square blocks of size 1 or 2 (Bunch and Kaufman, 1977). Then $\boldsymbol{B} = [\boldsymbol{I}_r|0]\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}\left[\begin{smallmatrix}\boldsymbol{I}_r\\0\end{smallmatrix}\right]$ is symmetric and non-singular. It is then sufficient to lift or reconstruct only the block diagonal matrix Δ over $\mathbf{Q}$ of a non-pivoting symmetric factorization of B (the unit triangular matrix over $\mathbf{Q}$ need not be computed). Compared to an integer characteristic polynomial computation this gains in practice an order of magnitude in efficiency for the Prover as shown on the logscale Figure 2.1, using FFLAS-FFPACK-2.3.1, LinBox-1.5.1 (The LinBox group, 2019) on a single core of an Intel i7-6700U@3.4GHz.

For the verification, the block diagonal matrix $\boldsymbol{\Delta}$, and the permutation $\boldsymbol{P}$ are committed. The Verifier then randomly chooses a prime q and enters an interactive certification process for $\boldsymbol{P}$ and $\boldsymbol{\Delta} \mod q$ using Protocol LDUP, as shown on Protocol Signature.

From (Dumas and Kaltofen, 2014, Theorem 5), we let $h = \log_2(\sqrt{n}^n||\boldsymbol{A}||_\infty^n)$ be the logarithm of Hadamard's bound for the invariant factors of $\boldsymbol{A}$. There cannot be more than h primes reducing the rank. Therefore it is possible to sample $c \cdot h$ distinct primes of magnitude bounded by $O(h\log(h))$ for any constant $c > 2$ and select q_1 from that set S_1. Once the rank is certified, the Prover can compute the permutation and lift the diagonal. Finally the rational $\boldsymbol{P}\boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}^T\boldsymbol{P}^T$ factorization of the full rank matrix can be similarly verified modulo a prime q_2. As for the determinant, no more than h primes can reduce the rank of Δ and q_2 can be selected from the same kind of set. We have

Protocol 2.13: Protocol Signature

Public: $\boldsymbol{A} \in \mathbf{Z}^{n\times n}$, symmetric, a triple of integers (n_+, n_-, n_0)
Certifies: The signature of $\boldsymbol{A}$ is (n_+, n_-, n_0)

	Prover		Verifier
1.	$\mathcal{I} = RRP(\boldsymbol{A})$		
		$\xrightarrow{\mathcal{I}}$	
2.			$q_1 \xleftarrow{\$} S \subset \mathbb{P}$
		$\xleftarrow{q_1}$	
3.		ColumnRankProfile$(\boldsymbol{A} \mod q_1, \mathcal{I})$	
			Is convinced that $\mathcal{I}$ is the RRP and CRP of $\boldsymbol{A}$ mod q_1
4.			Is convinced that $\lvert\mathcal{I}\rvert = \operatorname{rank}(\boldsymbol{A}) \bmod q_1$
5.	$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}^T\boldsymbol{P}^T$		
		$\xrightarrow{\boldsymbol{P}, \boldsymbol{\Delta}}$	
6.			$\boldsymbol{P} \stackrel{?}{\in} \mathcal{S}_r, \boldsymbol{\Delta} \stackrel{?}{\in} \mathcal{D}_r^{(2)}(\mathbf{Q})$ $q_2 \xleftarrow{\$} \mathsf{S}_2 \subset \mathbb{P}$
7.		Protocol LDUP$(\boldsymbol{P}^T\boldsymbol{A}_{\mathcal{I},\mathcal{I}}\boldsymbol{P} \mod q_2, \boldsymbol{I}_r, \boldsymbol{\Delta})$	
			Learns $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\psi}, \boldsymbol{\phi}$ from the call to Protocol LDUP
8.			$\boldsymbol{z}^T\boldsymbol{\Delta}\begin{bmatrix}\boldsymbol{x}\boldsymbol{y}\end{bmatrix} \stackrel{?}{=} (\boldsymbol{\lambda}^T\boldsymbol{P}^T\boldsymbol{A}_{\mathcal{I}})\boldsymbol{P}\begin{bmatrix}\boldsymbol{\phi} & \boldsymbol{\psi}\end{bmatrix} \bmod q_2$
9.			Compare (n_+, n_-, n_0) with the number of positive, negative and zero elements in Δ

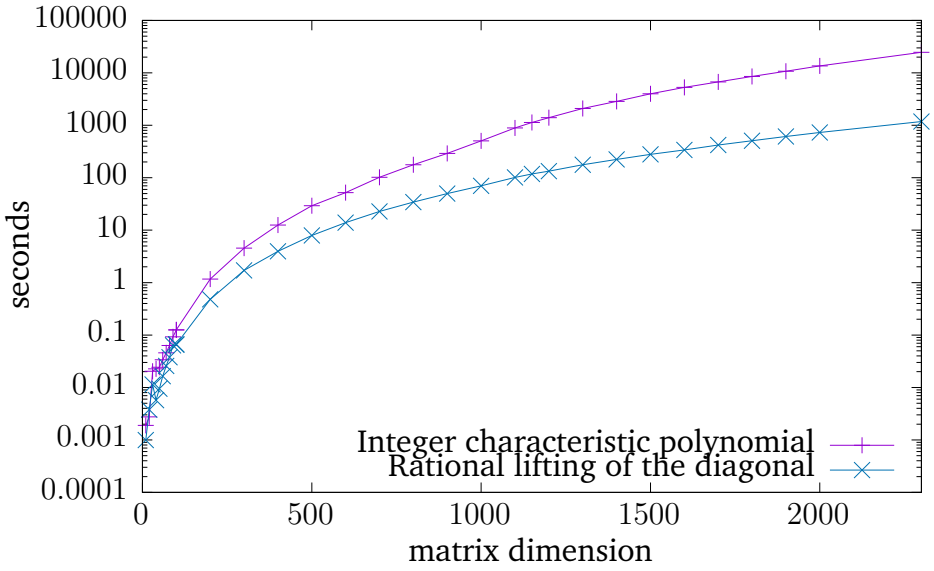

Figure 2.1: (Verifiable) signature computation on an Intel i7-6700U@3.4GHz.

proven:

Corollary 2.4.2. *For a symmetric matrix $\boldsymbol{A} \in \mathbf{Z}^{n\times n}$, Protocol Signature for its signature is sound and perfectly complete.*

The communication comprise that of the Protocol LDUP, the permutation matrix $\boldsymbol{P}$, all of size n, as well as small primes bounded by h, and finally $\boldsymbol{\Delta}$. Just like that of the characteristic polynomial, the size of $\boldsymbol{\Delta}$ can be quadratic and therefore the whole protocol is not linear. Thus a simpler quadratic protocol communicating the triangular matrix $\boldsymbol{L}$ modulo q_2, and checking the decomposition $\boldsymbol{A} - \boldsymbol{L}\boldsymbol{\Delta}\boldsymbol{L}^T$ via Freivalds' check might also work. But then the communication and Verifier time would always be quadratic. Instead, Protocol Signature, just like the Protocol using the characteristic polynomial, is better if the size of the determinant is small, as then the size of $\boldsymbol{\Delta}$ might be much less than that of $\boldsymbol{L}$ (for instance linear if the determinant is a constant). Protocol Signature is also interesting if $\mu(\boldsymbol{A})$ is less than quadratic.

Conclusion

A summary of our contributions is given in Table 2.2. In this chapter, we provided verification protocols that can save overall computational time for the Prover, and an order of magnitude in terms of communication volume or number of rounds. In particular, in Table 2.1 which compares both linear and quadratic communication and sub-cubic or quadratic matrix operations, one can see that it is interesting to use linear space protocols even when they have quadratic verification time. This table also exhibits a practical factor of about 5 between PLUQ and CHARPOLY computations.

The key idea of this chapter was to certify the existence of a triangular matrix in an equivalence relation, by having an n round protocol where data dependency matches

the triangular shape of the unknown matrix factor. This approach was adapted to the verification of generic rank profile-ness in the LU decomposition, where two triangular unknown factors are considered.

Mulmuley's Laurent's polynomial representation of a matrix successfully replaces the former technique to certify triangular equivalence, and consequently row or column rank profiles, reducing the number of rounds from linear to constant. However, we were unable to adapt this technique for the generic rank profile-ness protocol and the rank profile matrix protocol.

The use of symmetric Gaussian elimination allowed us to achieve a more practical protocol for the signature of symmetric integer matrices. Even though it is based on LDLT verification with linear communication modulo a prime, the diagonal of rational eigenvalues remains quadratic in size, and full precision was required to recover their sign. Designing a linear communication and Prover efficient protocol for the signature is the other major open problem which should be investigated.

Algorithm	Rounds	Prover		Communication	Probabilistic Verifier Time	#S
		Determ.	Time			
CRP/RRP § 2.1.1	0	Yes	$O(mnr^{\omega-2})$	$O(r(m+n))$	$O(r(m+n))+\mu(A)$	$\geqslant 2$
CRP/RRP § 2.2.4	$O(n)$	Yes	$O(mnr^{\omega-2})$	$O(m+n)$	$2\mu(A)+O(m+n)$	$\geqslant 2$
CRP/RRP § 2.3.3	3	Yes	$O(mnr^{\omega-2})$	$O(m+n)$	$2\mu(A)+O(m+n)$	$\geqslant 2n-1$
RPM § 2.1.2	0	Yes	$O(mnr^{\omega-2})$	$O(r(m+n))$	$O(r(m+n))+\mu(A)$	$\geqslant 2$
RPM § 2.2.5	$O(n)$	Yes	$O(mnr^{\omega-2})$	$O(m+n)$	$4\mu(A)+O(m+n)$	$\Omega(n)$
DET § 2.4.1 & PLUQ	$O(n)$	Yes	$O(n^{\omega})$	$O(n)$	$\mu(A)+O(n)$	$\Omega(n)$

Table 2.2: Contributions

Chapter 3

Verification protocols with sub-linear communication for polynomial matrix operations

Contents

Technical summary and overview of this chapter

Focus This chapter focuses on the design of verification protocols for various computations on matrices over the ring of univariate polynomials over a field F. The article associated with this chapter is (Lucas et al., 2019).

Model We propose verification protocols as defined in Definition 1.1.2, whose efficiency is measured based on the metrics given in Section 1.1.3.

State of the art – main competition There were no dedicated verification protocols for the properties we consider on polynomial matrices prior to this work.

Results The first part of this chapter focuses on verification protocols for classical linear algebra properties of matrices such as the rank, the determinant or the singularity. We also give verification protocols for matrix product and system solving. As these properties can be seen as vector space problems, the protocols we propose are heavily based on evaluation techniques: we evaluate the polynomial matrix at a random point and use vector space verification problems for the property to verify. These protocols are thus similar to those of Kaltofen et al. (2011), in which they were using similar techniques of reduction modulo a prime number for integer matrices. In the second part of this chapter, we focus on module problems. For these problems, the evaluation technique no longer works and we can no longer rely on vector space verification protocols. We first present a new verification protocol to verify that a given vector belongs to the row space of a given full row rank polynomial matrix. Using this new protocol, we then design a verification protocol for the same problem for matrices of arbitrary rank. This new row space membership verification protocol proves to be a central tool for the verification of many module properties of polynomial matrices: it allows us to verify that the row space of a matrix is contained in the row space of another, which in turn leads to verifying the row space equality of two matrices. With this, we can in turn verify normal forms of polynomial matrices, such as the Hermite form or the shifted Popov form. Finally, we verify properties related to the saturation and the kernels of polynomial matrices.

Outline

After giving some preliminary material in section 3.1, we propose verification protocols for classical properties of polynomial matrices — singularity, rank, determinant and matrix product — with sub-linear communication cost with respect to the input size (section 3.2). Those protocols are based on evaluating considered matrices at random points, which allows us to reduce the communication space and to use existing verification protocols for matrices over fields. Then, in section 3.3 we give the main result of this chapter, which is certifying that a given polynomial row vector is in the row space of a given polynomial matrix, which can either have full rank or be

rank-deficient. section 3.4 shows how to use this result to certify that for two given polynomial matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the row space of $\boldsymbol{A}$ is contained in the row space of $\boldsymbol{B}$, and then gives verification protocols for some classical normal forms of polynomial matrices. In section 3.5, we present verification protocols related to saturations and kernels of polynomial matrices. Finally, section 3.5.2 gives a conclusion and comments on a few perspectives.

Introduction

In this chapter, we propose new *verification protocols* for computations performed on univariate polynomial matrices. Generically, we consider protocols where a Prover performs computations and provides additional data structures to or exchanges with a Verifier, who will use these to check the validity of a result, at a lower cost than by recomputing it.

This chapter deals with computations on matrices whose entries are univariate polynomials. While certification for matrices over fields and over integer rings have been studied over the past twenty years, there are only few results on polynomial matrices (Dumas, 2018; Giorgi and Neiger, 2018).

A *polynomial matrix* is a matrix $\boldsymbol{M} \in \mathsf{F}[x]^{m\times n}$ whose entries are univariate polynomials over a field F. There is an isomorphism with *matrix polynomials* (univariate polynomials with matrices as coefficients) which we will sometimes use implicitly, such as when considering the evaluation $\boldsymbol{M}(\alpha) \in \mathsf{F}^{m\times n}$ of $\boldsymbol{M}$ at a point $\alpha \in \mathsf{F}$.

One general approach for computing with polynomial matrices is based on evaluation and interpolation. The basic idea is to first evaluate the polynomial matrix, say $\boldsymbol{M} \in \mathsf{F}[x]^{n\times n}$ at a set of points $\alpha_1, \alpha_2, \ldots \in \mathsf{F}$ in the ground field, then to separately perform the desired computation on each $\boldsymbol{M}(\alpha_i)$ over $\mathsf{F}^{n\times n}$, and finally reconstruct the entries of the result using fast polynomial interpolation. This kind of approach works well for operations such as matrix multiplication (Bostan and Schost, 2005, Section 5.4) or determinant computation. These computations essentially concern the *vector space* in the sense that $\boldsymbol{M}$ may as well be seen as a matrix over the fractions $\mathsf{F}(x)$ without impact on the results of the computations.

Other computational problems with polynomial matrices intrinsically concern $\mathsf{F}[x]$-*modules* and thus cannot merely rely on evaluation and interpolation. Classic and important such examples are that of computing normal forms such as the Popov form and the Hermite form (Popov, 1972; Villard, 1996; Neiger et al., 2018) and that of computing modules of relations such as approximant bases (Beckermann and Labahn, 1994; Giorgi et al., 2003; Neiger and Vu, 2017). The algorithms in this case must preserve the module structure attached to the matrix and thus deal with the actual polynomials in some way; in particular, an algorithm which works only with evaluations of the matrix at points $\alpha \in \mathsf{F}$ is oblivious to this module structure.

A summary of our contributions is given in table 3.1, based on the following notations: the input matrix has rank r and size $n \times n$ if it is square or $m \times n$ if it can be rectangular; if there are several input matrices, then r stands for the maximum of their

ranks, m for the maximum of their row dimensions, and n for the maximum of their column dimensions. Where appropriate, r is the maximum of the actual ranks of the matrices and the claimed rank by the Prover. We write d for the maximum degree of any input matrix or vector.

The Prover and Verifier costs are in arithmetic operations over the base field F. We use $\widetilde{O}(\cdot)$ for asymptotic cost bounds with hidden logarithmic factors, and $\omega \leqslant 3$ is the exponent of matrix multiplication, so that the multiplication of two $n \times n$ matrices over F uses $O(n^\omega)$ operations in F; see Section 3.1 for more details and references.

The last column indicates the smallest size of the ground field F needed to ensure both perfect completeness of the protocol and soundness with probability at least $\frac{1}{2}$. If this lower bound is not met, an extension field may be agreed on in advance by the Prover and Verifier, for a (logarithmic) increase in arithmetic and communication costs. For all protocols, an arbitrary low probability p of failure can be achieved by simply iterating the protocol at most $\lceil \log_2(1/p) \rceil$ times.

3.1 Preliminaries

Asymptotic complexity bounds We complete the notations and notions given in Chapter 1, with specific details for polynomial matrices. Cantor and Kaltofen (1991) have shown that multiplying two univariate polynomials of degree $\leqslant d$ over any algebra uses $\widetilde{O}(d)$ additions, multiplications, and divisions in that algebra. In particular, multiplying two matrices in $\mathsf{F}[x]^{n\times n}$ of degree at most d uses $\widetilde{O}(n^\omega d)$ operations in F.

Rational fractions For a rational fraction $f \in \mathsf{F}(x)$, define its *denominator* $\operatorname{denom}(f)$ to be the unique monic polynomial $g \in \mathsf{F}[x]$ of minimal degree such that $gf \in \mathsf{F}[x]$. Correspondingly, define its *numerator* $\operatorname{numer}(f) = f \cdot \operatorname{denom}(f)$. Note that $\operatorname{denom}(a) = 1$ if and only if $a \in \mathsf{F}[x]$. More generally, for a matrix of rational fractions $\boldsymbol{A} \in \mathsf{F}(x)^{m\times n}$, define $\operatorname{denom}(\boldsymbol{A})$ to be the unique monic polynomial $g \in \mathsf{F}[x]$ of minimal degree such that $g\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, and again write this polynomial matrix $g\boldsymbol{A}$ as $\operatorname{numer}(\boldsymbol{A})$. Note that we have the identity $\operatorname{denom}(\boldsymbol{A}) = \operatorname{lcm}_{i,j}(\operatorname{denom}(\boldsymbol{A}_{i,j}))$.

Row space, kernel, and row basis For a given matrix $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, two basic sets associated to it are its row space

$$\operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \{\boldsymbol{p}\boldsymbol{A},\ \boldsymbol{p} \in \mathsf{F}[x]^{1\times m}\},$$

and its left kernel

$$\{\boldsymbol{p} \in \mathsf{F}[x]^{1\times m} \mid \boldsymbol{p}\boldsymbol{A} = \boldsymbol{0}\}.$$

Accordingly, a *row basis* of $\boldsymbol{A}$ is a matrix in $\mathsf{F}[x]^{r\times n}$ whose rows form a basis of the former set, where r is the rank of $\boldsymbol{A}$, while a *left kernel basis* of $\boldsymbol{A}$ is a matrix in $\mathsf{F}[x]^{(m-r)\times n}$ whose rows form a basis of the latter set. We use similar notions and notations for column spaces and column bases, and for right kernels and right kernel

	Prover		Comm.	Verifier	Minimum #F
	Deter.	Cost		Cost	
Protocol Singularity	Yes	$O(nr^{\omega-1}+n^2d)$	$O(n)$	$O(n^2d)$	$2nd$
Protocol NonSingularity	Yes	$\widetilde{O}(n^\omega d)$	$O(n)$	$O(n^2d)$	$nd+1$
RankLowerBound	No	$O(mnr^{\omega-2}+mnd)$	$O(r)$	$O(r^2d)$	$rd+1$
RankUpperBound	Yes	$O(mnr^{\omega-2}+mnd)$	$O(n)$	$O(mnd)$	$2rd+2$
Protocol Rank	No	$O(mnr^{\omega-2}+mnd)$	$O(n)$	$O(mnd)$	$2rd+2$
Protocol Determinant	Yes	$O(n^2d+n^\omega)$	$O(n)$	$O(n^2d)$	$2nd+2$
SystemSolve	N/A	N/A	0	$O(n^2d)$	$4d$
MatMul	N/A	N/A	0	$O(n^2d)$	$4d+2$
FullRankRowSpaceMembership	Yes	$\widetilde{O}(nm^{\omega-1}d)$	$O(md)$	$O(mnd)$	$6md+2d+2$
RowSpaceMembership	No	$\widetilde{O}(mnr^{\omega-2}d)$	$\widetilde{O}(md+n)$	$\widetilde{O}(mnd)$	$6md+2d+2$
RowSpaceSubset	No	$\widetilde{O}(mnr^{\omega-2}d)$	$\widetilde{O}(md+n)$	$\widetilde{O}(mnd)$	$8rd+2d+4$
RowSpaceEquality	No	$\widetilde{O}(mnr^{\omega-2}d)$	$\widetilde{O}(md+n)$	$\widetilde{O}(mnd)$	$8rd+2d+4$
RowBasis	No	$\widetilde{O}(mnr^{\omega-2}d)$	$\widetilde{O}(md+n)$	$\widetilde{O}(mnd)$	$8rd+2d+6$
HermiteForm	No	$\widetilde{O}(mnr^{\omega-2}d)$	$\widetilde{O}(md+n)$	$\widetilde{O}(mnd)$	$8rd+2d+4$
ShiftedPopovForm	No	$\widetilde{O}(mnr^{\omega-2}d)$	$\widetilde{O}(md+n)$	$\widetilde{O}(mnd)$	$8rd+2d+4$
Saturated ($m \leqslant n$)	No	$\widetilde{O}(nm^{\omega-1}d)$	$\widetilde{O}(nd)$	$\widetilde{O}(mnd)$	$8md+4$
Saturated ($m > n$)	No	$\widetilde{O}(mn^{\omega-1}d)$	$\widetilde{O}(md)$	$\widetilde{O}(mnd)$	$8nd+4$
SaturationBasis	No	$\widetilde{O}(mnr^{\omega-2}+mnd+nr^{\omega-1}d)$	$\widetilde{O}(nd)$	$\widetilde{O}(mnd)$	$8nd+2d+4$
UnimodularCompletable	No	$\widetilde{O}(nm^{\omega-1}d)$	$\widetilde{O}(nd)$	$\widetilde{O}(mnd)$	$8md+4$
KernelBasis	No	$\widetilde{O}((m+n)m^{\omega-1}d)$	$\widetilde{O}(md)$	$\widetilde{O}(m(m+n)d)$	$8md+4$

Table 3.1: Summary of the contributions. The first column states whether the Prover's algorithm is deterministic or not. The costs are given in number of arithmetic operations over the base field and the communication is in number of elements in the base field F. The last column reports the minimum size size of F needed to ensure perfect completeness and soundness with probability at least $\frac{1}{2}$.

bases. We will also often consider the *rational* row space or $\mathsf{F}(x)$-row space of $\boldsymbol{A}$, denoted by $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A})$, which is an $\mathsf{F}(x)$-vector space.

Matrices which preserve the row space under left-multiplication, that is, $\boldsymbol{U} \in \mathsf{F}[x]^{m\times m}$ such that the $\mathsf{F}[x]$-row space of $\boldsymbol{UA}$ is the same as that of $\boldsymbol{A}$, are said to be *unimodular*. They are characterized by the fact that their determinant is a nonzero constant; or equivalently that they have an inverse (with entries in $\mathsf{F}[x]$).

3.1.1 Some probability bounds.

Many of our protocols rely on the fact that when picking an element uniformly at random from a sufficiently large finite subset of the field, this element is unlikely to be a root of some given polynomial. This was stated formally in (Schwartz, 1980; Zippel, 1979; Demillo and Lipton, 1978) and is often referred to as the *DeMillo-Lipton-Schwartz-Zippel lemma*.

Specifically, it states that for any nonzero k-variate polynomial $f(x_1, \ldots, x_k)$ with coefficients in a field F, and any finite subset $\mathsf{S} \subseteq \mathsf{F}$, if an evaluation point $(\alpha_1, \ldots, \alpha_k) \in \mathsf{F}^k$ has entries chosen at random uniformly and independently from S, then the probability that $f(\alpha_1, \ldots, \alpha_k) = 0$ is at most $d/\#\mathsf{S}$, where d is the total degree of f.

The following consequence is a standard extension of the soundness proof of Freivalds' algorithm (1979).

Lemma 3.1.1. *Let $\boldsymbol{A} \in \mathsf{F}^{m\times n}$ be a matrix with at least one nonzero entry and let $S \subseteq \mathsf{F}$ be a finite subset. For a vector of scalars $\boldsymbol{w} \in S^{n\times 1}$ chosen uniformly at random, we have $\Pr[\boldsymbol{Aw} = \boldsymbol{0}] \leqslant 1/\#S$.*

Proof. Consider each of the n entries of $\boldsymbol{w}$ as an indeterminate. Because $\boldsymbol{A}$ is not zero, $\boldsymbol{Aw}$ has at least one nonzero entry, which is a nonzero polynomial in n variables with total degree 1. Then a direct application of the DeMillo-Lipton-Schwartz-Zippel lemma gives the stated result. □

The next lemma will also be frequently used when analyzing protocols: it bounds the probability of picking a "bad" evaluation point.

Lemma 3.1.2. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ with rank at least r. For any finite subset $S \subseteq \mathsf{F}$ and for a point $\alpha \in S$ chosen uniformly at random, the probability that $\mathrm{rank}(\boldsymbol{A}(\alpha)) < r$ is at most $r\deg(\boldsymbol{A})/\#S$.*

Proof. Any $r \times r$ minor of $\boldsymbol{A}$ has degree at most $r\deg(\boldsymbol{A})$, and at least one must be nonzero since $\mathrm{rank}(\boldsymbol{A}) \geqslant r$. On the other hand, $\mathrm{rank}(\boldsymbol{A}(\alpha)) < r$ if and only if α is a root of all such minors. □

3.2 Linear algebra operations

In this section, we give some verification protocols for the computation of classical linear algebra properties on polynomial matrices: singularity, rank, and determinant of a matrix, as well as system solving and matrix multiplication.

The protocols we present here all rely on the same general idea, which consists in picking a random point and evaluating the input polynomial matrix (or matrices) at that point. This allows us to achieve sub-linear communication space. Note that this technique has been used before by Kaltofen et al. (2011) to certify the same properties for integer matrices: in that setup, computations were performed modulo some prime number, while, in our context, this translates into evaluating polynomials at some element of the base field.

In several of our protocols, the Prover has to solve a linear system over the base field. For a linear system whose matrix is in $\mathsf{F}^{m\times n}$ and has rank r, this can be done in $O(mnr^{\omega-2})$ operations in F (see Jeannerod et al., 2013, Algorithm 6).

3.2.1 Singularity and nonsingularity

We start by certifying the singularity of a matrix. Here, the Verifier picks a random evaluation point and sends it to the Prover, who evaluates the input matrix at that point and sends back a nontrivial kernel vector, which the Prover will always be able to compute since a singular polynomial matrix is still singular when evaluated at any point. Then, all the Verifier needs to do is to check that the vector received is indeed a kernel vector. Note that the evaluation trick here is really what allows us to have a sub-linear — with respect to the input size — communication cost, as the answer the Prover provides to the challenge is a vector over the base field, and not over the polynomials.

Protocol 3.1: Protocol Singularity

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{n\times n}$
Certifies: $\boldsymbol{A}$ is singular

	Prover		Verifier
1.			$\alpha \xleftarrow{\$} \mathsf{S}$
		$\xleftarrow{\alpha}$	
2.	Find $\boldsymbol{v} \in \mathsf{F}^{1\times n}\setminus\{\boldsymbol{0}\}$ s.t. $\boldsymbol{v}\boldsymbol{A}(\alpha) = \boldsymbol{0}$		
		$\xrightarrow{\boldsymbol{v}}$	
3.			$\boldsymbol{v} \stackrel{?}{\neq} \boldsymbol{0}$ $\boldsymbol{v}\boldsymbol{A}(\alpha) \stackrel{?}{=} \boldsymbol{0}$

In the next theorem, and for the remainder of the section, for convenience we write $d = \max(1, \deg(\boldsymbol{A}))$.

Theorem 3.2.1. *Protocol 3.1 is a complete and probabilistically sound interactive protocol which requires $O(n)$ communication and has Verifier cost $O(n^2d)$. The probability that the Verifier incorrectly accepts is at most $nd/\#S$. If $\boldsymbol{A}$ is singular, there is an algorithm for the Prover with cost $O(n^2d + nr^{\omega-1})$.*

Proof. If $\boldsymbol{A}$ is singular, $\boldsymbol{A}(\alpha)$ must also be singular and there exists a nontrivial nullspace vector that the Verifier will accept.

If $\boldsymbol{A}$ is nonsingular, then the Prover will be able to cheat if the Verifier picked an α such that $\boldsymbol{A}(\alpha)$ is singular, which happens only with probability $nd/\#\mathsf{S}$ according to lemma 3.1.2.

Now, for the complexities: the Prover will have to evaluate $\boldsymbol{A}$ at α, which costs $O(n^2d)$ and to find a nullspace vector over the base field, which costs $O(nr^{\omega-1})$, hence the Prover cost. The Verifier computes the evaluation and a vector-matrix product over F, for a total cost of $O(n^2d)$ operations. Finally, a vector over F^n and a scalar are communicated, which yields a communication cost of $O(n)$. □

We now present a verification protocol for nonsingularity. This relies on the same evaluation-based approach, with one variation: here, we let the Prover provide the evaluation point. Indeed, if the Verifier picked a random point, they could choose an "unlucky" point for which a nonsingular matrix evaluates to a singular one, and in that case, the protocol would be incomplete as the Prover will not be able to convince the Verifier of nonsingularity. Instead, we let the Prover pick a point as they have the computational power to find a suitable point (Step 1 in Protocol NonSingularity). Once this value is committed to the Verifier, in Steps 2 to 4 we use the protocol verifying nonsingularity over a field due to Dumas and Kaltofen (2014, Theorem 3).

Theorem 3.2.2. *Protocol 3.2 is a probabilistically sound interactive protocol and is complete assuming that $\#S \geqslant nd + 1$. It requires $O(n)$ communication and has Verifier cost $O(n^2d)$. The probability that the Verifier incorrectly accepts is at most $1/\#S$. There is an algorithm for the Prover with cost $\widetilde{O}(n^\omega d)$.*

Proof. If $\boldsymbol{A}$ is nonsingular, then, as the field is large enough, there exists an α for which the rank of $\boldsymbol{A}(\alpha)$ does not drop, and as Steps 2 to 4 form a complete verification protocol, Protocol NonSingularity is complete.

If $\boldsymbol{A}$ is singular, it is not possible to find an α such that $\boldsymbol{A}(\alpha)$ is nonsingular. This means the Prover successfully cheats if they manage to convince the Verifier that $\boldsymbol{A}(\alpha)$ is nonsingular, which only happens with probability $1/\#\mathsf{S}$ (Dumas and Kaltofen, 2014, Theorem 3), hence the soundness of Protocol NonSingularity.

Now, for the complexities: the Prover needs to find a suitable α. The Prover first computes $\det(\boldsymbol{A}) \in \mathsf{F}[x]$ using the deterministic algorithm of Labahn et al. (2017, Theorem 1.1) in $\widetilde{O}(n^\omega d)$ time. Then, using fast multipoint evaluation, the determinant is evaluated at $nd+1$ points from S in time $\widetilde{O}(nd)$ (von zur Gathen and Gerhard, 2013, Corollary 10.8); since $\deg(\det(\boldsymbol{A})) \leqslant nd$, at least one evaluation will be nonzero. Computing this determinant dominates the later cost for the Prover to evaluate $\boldsymbol{A}(\alpha)$ and solve a linear system over the base field, hence a total cost of $\widetilde{O}(n^\omega d)$.

Protocol 3.2: NonSingularity

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{n \times n}$
Certifies: $\boldsymbol{A}$ is nonsingular

	Prover		Verifier
1.	Find $\alpha \in \mathsf{F}$ s.t. $\det(\boldsymbol{A}(\alpha)) \neq 0$		
		$\xrightarrow{\alpha}$	
2.			$\boldsymbol{b} \xleftarrow{\$} \mathsf{S}^{n \times 1}$
		$\xleftarrow{\boldsymbol{b}}$	
3.	Find $\boldsymbol{w} \in \mathsf{F}^{n \times 1}$ s.t. $\boldsymbol{A}(\alpha)\boldsymbol{w} = \boldsymbol{b}$		
		$\xrightarrow{\boldsymbol{w}}$	
4.			$\boldsymbol{A}(\alpha)\boldsymbol{w} \stackrel{?}{=} \boldsymbol{b}$

The Verifier needs to evaluate $\boldsymbol{A}$ at α and to perform a matrix-vector multiplication over the base field, hence a cost of $O(n^2 d)$. Finally, total communications are two vectors of size n over the base field and a scalar, hence the cost of $O(n)$. □

3.2.2 Matrix Rank

From the protocol for nonsingularity, we immediately infer one for a lower bound ρ on the rank: the Prover commits a set of row indices and a set of column indices which locate a $\rho \times \rho$ submatrix which is nonsingular, and then the protocol verifying nonsingularity is run on this submatrix.

Theorem 3.2.3. *Let r be the actual rank of $\boldsymbol{A}$. Protocol 3.3 is a probabilistically sound interactive protocol and is complete assuming $\#\mathsf{S} \geqslant \rho d + 1$ in its subprotocol. It requires $O(\rho)$ communication and has Verifier cost $O(\rho^2 d)$. If we indeed have $r \geqslant \rho$, then there is a Las Vegas randomized algorithm for the Prover with expected cost $O(mnr^{\omega-2} + mnd)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $1/\#\mathsf{S}$.*

Proof. If ρ is indeed a lower bound on the rank of $\boldsymbol{A}$, there exist two sets $I \subseteq \{1, \ldots, m\}$ and $J \subseteq \{1, \ldots, n\}$ of size ρ such that $\boldsymbol{A}_{I,J}$ is nonsingular, and since Protocol NonSingularity is complete, so is this protocol. Note that the completeness of the sub-protocol is ensured only if $\#\mathsf{S} \geqslant \rho d + 1$.

If ρ is not a lower bound on the rank of $\boldsymbol{A}$, meaning $\operatorname{rank}(\boldsymbol{A}) < \rho$, then the Prover will not be able to find suitable I and J and hence the sets provided by a cheating Prover yield a singular submatrix $\boldsymbol{A}_{I,J}$. Now, if the Prover provided sets which do not

Protocol 3.3: RankLowerBound

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\rho \in \mathbf{N}$
Certifies: $\operatorname{rank}(\boldsymbol{A}) \geqslant \rho$

Prover		Verifier
1. Find index sets I, J such that $\#I = \rho$, $\#J = \rho$, and $\boldsymbol{A}_{I,J}$ is nonsingular	$\xrightarrow{I,\ J}$	
2.		$I \overset{?}{\subseteq} \{1,\ldots,m\}$ $J \overset{?}{\subseteq} \{1,\ldots,n\}$
3.	Use Protocol NonSingularity($\boldsymbol{A}_{I,J}$)	

contain ρ elements or which contain elements outside the allowed dimension bounds, this will always be detected by the Verifier. If the Prover provided sets with enough elements, the Verifier incorrectly accepts with the same probability as in Protocol NonSingularity, which is $1/\#\mathsf{S}$.

Regarding the complexities, the Prover has to find a $\rho \times \rho$ nonsingular submatrix of an $m \times n$ degree d matrix. This can be achieved in a Las Vegas fashion, by evaluating the matrix $\boldsymbol{A}$ at a random α in time $O(mnd)$, and computing the rank profile matrix (or a rank profile revealing PLUQ decomposition) of $\boldsymbol{A}(\alpha)$, see for instance (Dumas et al., 2015). The cost of this computation is $O(mnr^{\omega-2})$, with r the actual rank of $\boldsymbol{A}$.

As $\rho \leqslant r$, running Protocol 3.2 on a $\rho \times \rho$ matrix does not dominate the complexity, hence the total Prover cost of $O(mnr^{\omega-2} + mnd)$. From theorem 3.2.2, the Verifier cost is $O(\rho^2 d)$. Finally, here two sets of ρ integers are transmitted, which with the communications in Protocol NonSingularity adds up to a communication cost of $O(\rho)$. □

Now, we give a protocol verifying an upper bound on the rank. Note that Steps 2 and 3 of Protocol 3.4 come from the protocol verifying an upper bound on the rank for matrices over a field (see Dumas and Kaltofen, 2014, Theorem 4). In this protocol, we use the notation $|\cdot|_H$ to refer to the Hamming weight: $|\boldsymbol{\gamma}|_H \leqslant \rho$ means that the vector $\boldsymbol{\gamma}$ as at most ρ nonzero entries.

Theorem 3.2.4. *Let r be the actual rank of $\boldsymbol{A}$. Then, Protocol 3.4 is a complete and probabilistically sound interactive protocol which requires $O(n)$ communication and has Verifier cost $O(mnd)$. If we indeed have $r \leqslant \rho$, then there is an algorithm for the Prover with cost bound $O(mnr^{\omega-2}+mnd)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(rd+1)/\#\mathsf{S}$.*

Protocol 3.4: RankUpperBound

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\rho \in \mathbf{N}$
Certifies: $\operatorname{rank}(\boldsymbol{A}) \leqslant \rho$

	Prover		Verifier
1.			$\alpha \xleftarrow{\$} \mathsf{S}$ $\boldsymbol{v} \xleftarrow{\$} \mathsf{S}^{n\times 1}$
		$\alpha, \boldsymbol{v}$ ⟵	
2.	Find $\boldsymbol{\gamma} \in \mathsf{F}^{n\times 1}$ such that $\boldsymbol{A}(\alpha)\boldsymbol{\gamma} = \boldsymbol{A}(\alpha)\boldsymbol{v}$ and $\lvert\boldsymbol{\gamma}\rvert_H \leqslant \rho$		
		⟶ $\boldsymbol{\gamma}$	
3.			$\lvert\boldsymbol{\gamma}\rvert_H \overset{?}{\leqslant} \rho$ $\boldsymbol{A}(\alpha)\boldsymbol{\gamma} \overset{?}{=} \boldsymbol{A}(\alpha)\boldsymbol{v}$

Proof. If ρ is indeed an upper bound on the rank of $\boldsymbol{A}$, then, whichever evaluation point the Verifier picked, ρ will be an upper bound on the rank of $\boldsymbol{A}(\alpha)$ and, as the protocol from (Dumas and Kaltofen, 2014, Theorem 4) is complete, this protocol is complete.

If ρ is not an upper bound on the rank of $\boldsymbol{A}$, there are two possibilities of failure. Either the Verifier picked an evaluation point for which the rank of $\boldsymbol{A}$ drops, which happens with probability at most $rd/\#\mathsf{S}$ by Lemma 3.1.2; or the Prover managed to cheat during the execution of Steps 2 and 3 which happens with probability at most $1/\#\mathsf{S}$ (Dumas and Kaltofen, 2014, Theorem 4). Then, the union bound gives a total probability of $(rd+1)/\#\mathsf{S}$ for the Verifier to accept a wrong answer.

The Prover has to evaluate the matrix at α for a cost of $O(mnd)$. Then to find the vector $\boldsymbol{\gamma}$, the Prover can for instance first compute a PLUQ decomposition $\boldsymbol{A}(\alpha) = \boldsymbol{P}\begin{bmatrix}\boldsymbol{L}_1\\ \boldsymbol{L}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{U}_1 & \boldsymbol{U}_2\end{bmatrix}\boldsymbol{Q}$ for a cost of $O(mnr^{\omega-2})$. Then, the Prover computes the vector $\begin{bmatrix}\boldsymbol{w}_1\\ \boldsymbol{w}_2\end{bmatrix} = \boldsymbol{Q}\boldsymbol{v}$, where $\boldsymbol{w}_1$ has size r, and computes $\boldsymbol{\gamma} = \boldsymbol{Q}^{\mathsf{T}}\begin{bmatrix}\boldsymbol{w}_1 + \boldsymbol{U}_1^{-1}\boldsymbol{U}_2\boldsymbol{w}_2\\ \boldsymbol{0}\end{bmatrix}$. This vector has Hamming weight at most r (recall that $\boldsymbol{Q}$ is a permutation matrix) and satisfies $\boldsymbol{A}(\alpha)\boldsymbol{\gamma} = \boldsymbol{P}\begin{bmatrix}\boldsymbol{L}_1\\ \boldsymbol{L}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{U}_1\boldsymbol{w}_1 + \boldsymbol{U}_2\boldsymbol{w}_2\end{bmatrix} = \boldsymbol{A}(\alpha)\boldsymbol{v}$. The Verifier has to evaluate the matrix at α and to perform two matrix-vector products over the base field, which yields a cost of $O(mnd)$. The communication cost is the one of sending a scalar and two vectors of size n over the base field, that is, $O(n)$. □

From these protocols verifying upper bounds and lower bounds on the rank, we

directly obtain one for the rank (Protocol 3.5).

Protocol 3.5: Rank

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$, $\rho \in \mathbf{N}$
Certifies: $\operatorname{rank}(\boldsymbol{A}) = \rho$

	Prover		Verifier
1.		Use RankLowerBound$(\boldsymbol{A}, \rho)$	
2.		Use RankUpperBound$(\boldsymbol{A}, \rho)$	

Corollary 3.2.5. *Let r be the actual rank of $\boldsymbol{A}$. Protocol 3.5 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant rd + 1$ in its subprotocols. It requires $O(n)$ communication and has Verifier cost $O(mnd)$. If we indeed have $\rho = r$, then there is a Las Vegas randomized algorithm for the Prover with expected cost $O(mnr^{\omega-2} + mnd)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(rd + 1)/\#S$.*

3.2.3 Determinant

We follow on with a protocol verifying the determinant of a polynomial matrix, using a similar evaluation-based approach: after the Verifier has checked that the degree of the provided determinant is suitable, a random evaluation point is sampled and the actual verification occurs on the evaluated input. The check on the degree of the provided determinant is to allow that the DeMillo-Lipton-Schwartz-Zippel Lemma applies and produces the claimed probability of the success. There are two choices available for the protocol to use over the base field: either the one from (Dumas et al., 2016, Section 2), which runs in a constant number of rounds but requires a minimum field size of n^2, or the one from (Dumas et al., 2017b, Section 4.1) which runs in n rounds but has no requirement on the field size. Whichever protocol is chosen here, this has no impact on the asymptotic complexities which are the same for both, or on the completeness as both are perfectly complete.

Theorem 3.2.6. *Protocol 3.6 is a complete and probabilistically sound interactive protocol which requires $O(n)$ communication and has Verifier cost $O(n^2 d)$. If δ is indeed the determinant of $\boldsymbol{A}$, there is an algorithm for the Prover which costs $O(n^2 d + n^\omega)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(nd + 1)/\#S$.*

Proof. Let $g = \det(\boldsymbol{A}) \in \mathsf{F}[x]$ be the actual determinant of $\boldsymbol{A}$.

If $\delta = g$, then it must be the case that $\deg(\delta) \leqslant nd$. Then, as FieldDeterminant is complete, the final check will be positive.

Protocol 3.6: Determinant

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{n\times n}, \delta \in \mathsf{F}[x]$
Certifies: $\det(\boldsymbol{A}) = \delta$

	Prover		Verifier
1.			$\deg(\delta) \overset{?}{\leqslant} nd$ $\alpha \xleftarrow{\$} \mathsf{S}$ $\beta \leftarrow \delta(\alpha)$
		$\xleftarrow{\quad\alpha\quad}$	
2.	Use FieldDeterminant$(\boldsymbol{A}(\alpha), \beta)$		

If $\delta \neq g$, there are two possibilities of failure. If the Verifier has picked an α which is a root of $\delta - g$, then $\delta(\alpha) = g(\alpha)$ and the checks from FieldDeterminant will always pass; by the DeMillo-Lipton-Schwartz-Zippel lemma, this happens with probability at most $nd/\#\mathsf{S}$. Otherwise, the Verifier has picked an α such that $\delta(\alpha) \neq g(\alpha)$ which means they will accept δ as the determinant with the probability of failure of FieldDeterminant, that is, $1/\#\mathsf{S}$. Overall, by the union bound, the probability that the Verifier accepts a wrong statement is at most $(nd+1)/\#\mathsf{S}$.

The Prover has to evaluate the matrix at α and to compute a determinant over the base field, which yields the cost of $O(n^2d + n^\omega)$; the Verifier has to evaluate $\boldsymbol{A}$ at α, hence a cost of $O(n^2d)$; and the communication cost is the one of FieldDeterminant, that is, $O(n)$. □

3.2.4 Protocols based on matrix multiplication

Finally, we propose verification protocols related to matrix multiplication. While they are once again based on evaluation techniques, unlike the above protocols the ones given in this section are non-interactive and thus have no Prover or communication cost. We first consider linear system solving; recall that when working over $\mathsf{F}[x]$, given a nonsingular matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$, this problem consists in finding a solution vector $\boldsymbol{v}$ over $\mathsf{F}[x]$ *together with* a nonzero polynomial $\delta \in \mathsf{F}[x]$ such that $\delta^{-1}\boldsymbol{v} = \boldsymbol{A}^{-1}\boldsymbol{b}$ (see for example Gupta et al., 2012).

Theorem 3.2.7. *Let d be an upper bound on the degree of $\boldsymbol{A}$, $\boldsymbol{v}$, $\boldsymbol{b}$, and δ. Then, Protocol 3.7 is a complete and probabilistically sound non-interactive protocol which has Verifier cost $O(mnd)$. The probability that the Verifier incorrectly accepts is at most $2d/\#\mathsf{S}$.*

Proof. If $\boldsymbol{A}\boldsymbol{v} = \delta\boldsymbol{b}$, then the same holds when evaluating at α, hence the completeness of this protocol.

Protocol 3.7: SystemSolve

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}, \boldsymbol{b} \in \mathsf{F}[x]^{m\times 1}, \boldsymbol{v} \in \mathsf{F}[x]^{n\times 1}, \delta \in \mathsf{F}[x]$
Certifies: $\deg(\delta) \leqslant \min(m,n)\deg(\boldsymbol{A})$ and $\boldsymbol{A}\boldsymbol{v} = \delta\boldsymbol{b}$

Prover	Verifier
1.	$\alpha \xleftarrow{\$} \mathsf{S}$
	$\boldsymbol{A}(\alpha)\boldsymbol{v}(\alpha) \stackrel{?}{=} \delta(\alpha)\boldsymbol{b}(\alpha)$

Otherwise, we have $\boldsymbol{A}\boldsymbol{v} - \delta\boldsymbol{b} = \boldsymbol{\Delta}$ for some nonzero vector $\boldsymbol{\Delta} \in \mathsf{F}[x]^{m\times 1}$, and the Verifier incorrectly accepts if and only if the Verifier picked an α such that $\boldsymbol{\Delta}(\alpha) = 0$. Using lemma 3.1.2 with the vector $\boldsymbol{\Delta}$ of rank 1 and degree at most $2d$, it follows that the Verifier picked such an α with probability at most $2d/\#\mathsf{S}$.

The dominating step in the Verifier's work is evaluating $\boldsymbol{A}$ at α, which costs $O(mnd)$ operations in F. □

Remark that when solving linear systems over $\mathsf{F}[x]$ one is often interested in the case of a nonsingular matrix $\boldsymbol{A}$ with $m = n$. In this context, one usually seeks a solution $(\boldsymbol{v}, \delta)$ with δ of minimal degree (see (Storjohann, 2003, Section 9) and (Gupta et al., 2012, Section 7)); this implies $\deg(\delta) \leqslant \deg(\det(\boldsymbol{A})) \leqslant n\deg(\boldsymbol{A})$ and $\deg(\boldsymbol{v}) = \deg(\delta\boldsymbol{A}^{-1}\boldsymbol{b}) \leqslant (n-1)\deg(\boldsymbol{A}) + \deg(\boldsymbol{b})$. In this particular case, these bounds could be checked by the Verifier at the beginning of the protocol; the probability that the Verifier incorrectly accepts becomes $(nd_{\boldsymbol{A}} + d_{\boldsymbol{b}})/\#\mathsf{S}$; and the Verifier's work costs $O(n^2 d_{\boldsymbol{A}} + nd_{\boldsymbol{b}})$ operations.

Similarly, we propose a protocol verifying matrix multiplication following an approach attributed to Freivalds (1979).

Protocol 3.8: MatMul

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}, \boldsymbol{B} \in \mathsf{F}[x]^{n\times \ell}, \boldsymbol{C} \in \mathsf{F}[x]^{m\times \ell}$
Certifies: $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$

Prover	Verifier
1.	$\deg(\boldsymbol{C}) \stackrel{?}{\leqslant} \deg(\boldsymbol{A}) + \deg(\boldsymbol{B})$
	$\alpha \xleftarrow{\$} \mathsf{S}$
	$\boldsymbol{v} \xleftarrow{\$} \mathsf{S}^{\ell\times 1}$
	$\boldsymbol{C}(\alpha)\boldsymbol{v} \stackrel{?}{=} \boldsymbol{A}(\alpha)(\boldsymbol{B}(\alpha)\boldsymbol{v})$

Theorem 3.2.8. *Let $d_{\boldsymbol{A}} = \max(1, \deg(\boldsymbol{A}))$ and similarly for $d_{\boldsymbol{B}}, d_{\boldsymbol{C}}$. Protocol 3.8 is a complete and probabilistically sound non-interactive protocol which has Verifier cost $O(mnd_{\boldsymbol{A}} + n\ell d_{\boldsymbol{B}} + m\ell d_{\boldsymbol{C}})$. The probability that the Verifier incorrectly accepts is at most $(d_{\boldsymbol{A}} + d_{\boldsymbol{B}} + 1)/\#S$.*

Proof. Let $\boldsymbol{D}$ be the actual product $\boldsymbol{D} = \boldsymbol{AB}$, and let $\boldsymbol{\Delta} = \boldsymbol{D} - \boldsymbol{C}$. Note that the final check of the Verifier is equivalent to $\boldsymbol{\Delta}(\alpha)\boldsymbol{v} \stackrel{?}{=} \boldsymbol{0}$.

If $\boldsymbol{C} = \boldsymbol{D}$, then $\boldsymbol{\Delta} = \boldsymbol{0}$ and whichever evaluation point α the Verifier picked, we have $\boldsymbol{\Delta}(\alpha) = \boldsymbol{0}$. The degree bound checked initially by the Verifier is also valid whenever $\boldsymbol{AB} = \boldsymbol{C}$, hence this protocol is complete.

Otherwise, $\boldsymbol{\Delta}$ is a nonzero matrix with degree at most $d_{\boldsymbol{A}} + d_{\boldsymbol{B}}$. There are two events that would lead to the Verifier accepting incorrectly. First, if the Verifier picked an evaluation point such that $\boldsymbol{\Delta}(\alpha) = \boldsymbol{0}$, which happens with probability at most $(d_{\boldsymbol{A}} + d_{\boldsymbol{B}})/\#\mathsf{S}$ by lemma 3.1.2 (with rank lower bound 1), then whichever verification vector $\boldsymbol{v}$ is picked afterwards, the Verifier will always accept. Otherwise, the Verifier picked an evaluation point for which $\boldsymbol{\Delta}(\alpha) \neq \boldsymbol{0}$ but they picked a unlucky verification vector $\boldsymbol{v}$, that is, $\boldsymbol{v}$ is in the right kernel of $\boldsymbol{\Delta}(\alpha)$, which happens with probability at most $1/\#\mathsf{S}$ according to lemma 3.1.1. The union bound gives the stated bound for the probability that the Verifier incorrectly accepts.

The cost for the Verifier comes from evaluating all three matrices at α and then performing three matrix-vector products over F. □

Verifying a matrix inverse is a straightforward application of the previous protocol.

Corollary 3.2.9. *For $\boldsymbol{A} \in \mathsf{F}[x]^{n\times n}$ and $\boldsymbol{B} \in \mathsf{F}[x]^{n\times n}$, there exists a non-interactive protocol which certifies that $\boldsymbol{B}$ is the inverse of $\boldsymbol{A}$ in Verifier cost $O(n^2 d)$, where $d = \max(1, \deg(\boldsymbol{A}), \deg(\boldsymbol{B}))$. If $\boldsymbol{B} \neq \boldsymbol{A}^{-1}$, the probability that the Verifier incorrectly accepts is at most $(2d + 1)/\#S$.*

3.3 Row space membership

In this section we present the main tool for verification problems that are essentially about $\mathsf{F}[x]$-modules, which is to determine whether a given row vector $\boldsymbol{v} \in \mathsf{F}[x]^{1\times n}$ is in the $\mathsf{F}[x]$-row space of a given matrix $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$.

The approach has two steps. First, FullRankRowSpaceMembership shows how to solve the problem in case $\boldsymbol{A}$ has full row rank. Then, in RowSpaceMembership, we extend this to the general setting by designing a reduction to several calls to the full row rank case.

3.3.1 Full row rank case

For this case, we propose Protocol 3.9. Before studying its properties, we emphasize that its soundness crucially depends on the fact that $\boldsymbol{A}$ has full row rank. To see why, let $\boldsymbol{A} = [x \quad -x]^\intercal$ and $\boldsymbol{v} = [1]$, and write $\boldsymbol{c} = [c_1 \quad c_2]^\intercal$ for the random vector chosen

Protocol 3.9: FullRankRowSpaceMembership

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$, $\boldsymbol{v} \in \mathsf{F}[x]^{1 \times n}$
Certifies: $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ and $\mathrm{rank}(\boldsymbol{A}) = m$

	Prover		Verifier
1.		RankLowerBound$(\boldsymbol{A}, m)$	
2.			Is convinced that $\mathrm{rank}(\boldsymbol{A}) \overset{?}{\geqslant} m$
3.			$\boldsymbol{c} \xleftarrow{\$} \mathsf{S}^{m \times 1}$
		$\boldsymbol{c}$ ⟵	
4.	$\boldsymbol{u} \in \mathsf{F}[x]^{1 \times m}$ s.t. $\boldsymbol{u}\boldsymbol{A} = \boldsymbol{v}$ $g \leftarrow \boldsymbol{u}\boldsymbol{c}$		
		g ⟶	
5.			$\deg(g) \overset{?}{\leqslant} m \deg(\boldsymbol{A}) + \deg(\boldsymbol{v})$ $\alpha \xleftarrow{\$} \mathsf{S}$
		α ⟵	
6.	$\boldsymbol{w} \leftarrow \boldsymbol{u}(\alpha) \in \mathsf{F}^{1 \times m}$		
		$\boldsymbol{w}$ ⟶	
7.			$\boldsymbol{w}\boldsymbol{A}(\alpha) \overset{?}{=} \boldsymbol{v}(\alpha)$ $\boldsymbol{w}\boldsymbol{c} \overset{?}{=} g(\alpha)$

by the Verifier. Here, $\boldsymbol{A}$ does not have full row rank and $\boldsymbol{v}$ is not in the row space of $\boldsymbol{A}$; it is however in the rational row space of $\boldsymbol{A}$. This allows a dishonest Prover to make the Verifier accept by means of forging a *rational* vector $\boldsymbol{u}$ such that $\boldsymbol{uA} = \boldsymbol{v}$ and $\boldsymbol{uc} \in \mathsf{F}[x]$: the Verifier cannot detect that $\boldsymbol{u}$ was not over $\mathsf{F}[x]$, since they only receive $\boldsymbol{uc}$ and an evaluation of $\boldsymbol{u}$. Indeed, any vector of the form $\boldsymbol{u} = [x^{-1} + f(x) \quad f(x)]$ for some $f \in \mathsf{F}(x)$ is such that $\boldsymbol{uA} = \boldsymbol{v}$. In the likely event that $c_1 + c_2 \neq 0$, the Prover can choose any polynomial $g \in \mathsf{F}[x]$ and define $f = (c_1 + c_2)^{-1}(g - c_1 x^{-1})$; then $\boldsymbol{uc} = g$ is a polynomial.

Remark that if $\boldsymbol{A}$ has full row rank and $\boldsymbol{v}$ belongs to the rational row space of $\boldsymbol{A}$, then we have *uniqueness* of the (rational) vector $\boldsymbol{u}$ such that $\boldsymbol{uA} = \boldsymbol{v}$ and thus there is no flexibility for the Prover on the choice of $\boldsymbol{u}$. In this case, the following lemma plays a key role in the soundness of Protocol 3.9.

Lemma 3.3.1. *Let $\boldsymbol{u} \in \mathsf{F}(x)^{1\times n}$ be a rational fraction vector with $\operatorname{denom}(\boldsymbol{u}) \neq 1$ and let $S \subseteq \mathsf{F}$ be a finite subset. For a vector of scalars $\boldsymbol{c} \in S^{n\times 1}$ chosen uniformly at random, the probability that the inner product $\boldsymbol{uc}$ is a polynomial, i.e., that $\operatorname{denom}(\boldsymbol{uc}) = 1$, is at most $1/\#S$.*

Proof. Write $f = \operatorname{denom}(\boldsymbol{u})$ and $\hat{\boldsymbol{u}} = \operatorname{numer}(\boldsymbol{u})$. By the condition of the lemma we know that $\deg(f) \geqslant 1$. We see that the inner product of $\boldsymbol{u}$ and $\boldsymbol{c}$ is a polynomial if and only if the inner product of $\hat{\boldsymbol{u}}$ and $\boldsymbol{c}$ is divisible by f.

Now let h be any irreducible factor of f, and consider the inner product $\hat{\boldsymbol{u}}\boldsymbol{c}$ with $\hat{\boldsymbol{u}}$ seen as a vector over the extension field $\mathsf{F}[x]/\langle h\rangle$. Because $h \mid \operatorname{denom}(\boldsymbol{u})$, we know that $\hat{\boldsymbol{u}} \bmod h$ is not zero; otherwise the degree of the denominator f is not minimal. Then, since $\mathsf{S} \subseteq \mathsf{F} \subseteq \mathsf{F}[x]/\langle h\rangle$, the stated bound follows from lemma 3.1.1. □

Another ingredient for our full row rank space membership protocol is a subroutine the Prover may use to actually compute the solution $\boldsymbol{u}$ to the linear system, shown in algorithm 1. More precisely, this algorithm computes the numerator $\hat{\boldsymbol{u}}$ and the corresponding minimal denominator f. This algorithm will also be used in the protocol for arbitrary-rank matrices presented in the next section.

As above, to simplify the cost bounds we write $d_{\boldsymbol{A}} = \max(1, \deg(\boldsymbol{A}))$ and $d_{\boldsymbol{v}} = \max(1, \deg(\boldsymbol{v}))$.

Lemma 3.3.2. *Algorithm 1 uses $\widetilde{O}(m^{\omega-1}nd_{\boldsymbol{A}} + m^{\omega-1}d_{\boldsymbol{v}})$ operations in F. If $\boldsymbol{A}$ has rank less than m, then* `LOW_RANK` *is returned. If $\boldsymbol{A}$ has rank m and $\boldsymbol{v} \notin \operatorname{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A})$, then* `NO_SOLUTION` *is returned. Otherwise, the algorithm returns $(\hat{\boldsymbol{u}}, f)$ such that $\hat{\boldsymbol{u}}\boldsymbol{A} = f\boldsymbol{v}$ and f has minimal degree; in particular, $\deg(f) \leqslant m\deg(\boldsymbol{A})$ and $\deg(\hat{\boldsymbol{u}}) \leqslant (m-1)\deg(\boldsymbol{A}) + \deg(\boldsymbol{v})$.*

Proof. (Zhou, 2012, Chapter 11) presents a deterministic algorithm to compute the column rank profile on line 1 using $\widetilde{O}(m^{\omega-1}nd_{\boldsymbol{A}})$ field operations. This guarantees that `LOW_RANK` is returned whenever $\boldsymbol{A}$ does not have full row rank.

Now assume that $\operatorname{rank}(\boldsymbol{A}) = m$. Then $\boldsymbol{B}$ is nonsingular, and Gupta et al. (2012) showed how to solve the linear system on line 5 deterministically using $\widetilde{O}(m^{\omega}d_{\boldsymbol{A}} +$

Algorithm 1: Linear system solving with full row rank

Input: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}, \boldsymbol{v} \in \mathsf{F}[x]^{1 \times n}$
Output: Either `LOW_RANK`, or `NO_SOLUTION`, or $(\hat{\boldsymbol{u}}, f) \in (\mathsf{F}[x]^{1 \times m} \times \mathsf{F}[x])$ such that $\hat{\boldsymbol{u}}\boldsymbol{A} = f\boldsymbol{v}$ and f has minimal degree

1 $r, i_1, \ldots, i_r \leftarrow$ column rank profile of $\boldsymbol{A}$
2 **if** $r < m$ **then return** `LOW_RANK` `// below,` $r = m$
3 $\boldsymbol{B} \in \mathsf{F}[x]^{r \times r} \leftarrow$ columns $i_1, \ldots, i_r$ from $\boldsymbol{A}$
4 $\boldsymbol{y} \in \mathsf{F}[x]^{1 \times r} \leftarrow$ columns $i_1, \ldots, i_r$ from $\boldsymbol{v}$
5 Compute $(\hat{\boldsymbol{u}}, f) \in (\mathsf{F}[x]^{1 \times m} \times \mathsf{F}[x])$ such that $f^{-1}\hat{\boldsymbol{u}} = \boldsymbol{y}\boldsymbol{B}^{-1}$ and f has minimal degree, using (Gupta et al., 2012, Algorithm `RationalSystemSolve`)
6 **if** $\hat{\boldsymbol{u}}\boldsymbol{A} \neq f\boldsymbol{v}$ **then return** `NO_SOLUTION`
7 **return** $\hat{\boldsymbol{u}}$

$m^{\omega-1}d_{\boldsymbol{v}})$ operations; precisely, this cost bound is obtained from the results in (Gupta et al., 2012, Section 7) applied with $d = \max(d_{\boldsymbol{A}}, md_{\boldsymbol{v}})$. The degree bounds on $\hat{\boldsymbol{u}}$ and f follow from Cramer's rule.

Let $(\hat{\boldsymbol{u}}, f)$ be the system solution computed on line 5. If $\boldsymbol{v} \notin \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A})$, then we must have $\hat{\boldsymbol{u}}\boldsymbol{A} \neq f\boldsymbol{v}$ and thus `NO_SOLUTION` is returned. Now assume that $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A})$, that is, there exists $\boldsymbol{w} \in \mathsf{F}(x)^{1 \times m}$ such that $\boldsymbol{w}\boldsymbol{A} = \boldsymbol{v}$. Then we have in particular $\boldsymbol{w}\boldsymbol{B} = \boldsymbol{y}$. But because $\boldsymbol{B}$ is nonsingular, we have $\boldsymbol{w} = \boldsymbol{y}\boldsymbol{B}^{-1} = f^{-1}\hat{\boldsymbol{u}}$; hence $\hat{\boldsymbol{u}}\boldsymbol{A} = f\boldsymbol{v}$. □

Finally, we present the main result of this subsection.

Theorem 3.3.3. *Protocol 3.9 is a complete and probabilistically sound interactive protocol which requires $O(md_{\boldsymbol{A}} + d_{\boldsymbol{v}})$ communication and has Verifier cost $O(mnd_{\boldsymbol{A}} + nd_{\boldsymbol{v}})$. If $\mathrm{rank}(\boldsymbol{A}) = m$ and $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, there is an algorithm for the Prover with cost $\widetilde{O}(nm^{\omega-1}d_{\boldsymbol{A}} + m^{\omega-1}d_{\boldsymbol{v}})$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(3md_{\boldsymbol{A}} + d_{\boldsymbol{v}} + 1)/\#S$.*

Proof. If $\mathrm{rank}(\boldsymbol{A}) < m$, then from theorem 3.2.3, the probability that the Verifier incorrectly accepts is at most $1/\#\mathsf{S}$, less than the stated bound in the theorem. And if $\boldsymbol{v}$ is the zero vector, then the protocol easily succeeds when the Prover sends all zeros for g and $\boldsymbol{w}$; remark that $\boldsymbol{u} = \boldsymbol{0}$ is the only solution to $\boldsymbol{u}\boldsymbol{A} = \boldsymbol{v}$ when $\boldsymbol{A}$ has full row rank. So for the remainder of the proof, assume that $\boldsymbol{v}$ is nonzero and $\boldsymbol{A}$ has full row rank m.

The rank check entails $2m + 1$ field elements of communication, and the degree check by the Verifier assures that g contains at most $md_{\boldsymbol{A}} + d_{\boldsymbol{v}} + 1$ field elements, bringing the total communication in the protocol to at most $m(d_{\boldsymbol{A}} + 4) + d_{\boldsymbol{v}} + 3$ field elements.

The work of the Verifier is dominated by computing the evaluations $\boldsymbol{A}(\alpha)$ and $\boldsymbol{v}(\alpha)$ on the last step. Using Horner's method the total cost for these is $O(mnd_{\boldsymbol{A}} + nd_{\boldsymbol{v}})$, as claimed.

We now divide the proof into three cases, depending on whether $\boldsymbol{v}$ is in the polynomial row space of $\boldsymbol{A}$ (as checked by the protocol), the rational row space of $\boldsymbol{A}$, or neither.

Case 1: $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ Here we want to prove that an honest Prover and Verifier succeed with costs as stated in the theorem.

The vector $\boldsymbol{u}$ as defined in Step 4 must exist by the definition of $\mathrm{RowSp}_{\mathsf{F}[x]}$, and computing $\boldsymbol{u}$ can be completed by the Verifier according to lemma 3.3.2 in the stated cost bound.

If the computations of $\boldsymbol{u}$ and g at Step 4 and of $\boldsymbol{w}$ at Step 6 are performed correctly by the Prover, then the Verifier's checks on Step 7 will succeed for any choice of α. Note also that in this case, $\deg(g) = \deg(\boldsymbol{uc}) \leqslant \deg(\boldsymbol{u})$, and $\deg(\boldsymbol{u}) \leqslant m\deg(\boldsymbol{A}) + \deg(\boldsymbol{v})$ holds (see lemma 3.3.2), hence the degree check at Step 5.

This proves the completeness of the protocol.

Case 2: $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A}) \setminus \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ In this case, the assertion being verified is false, and we want to show probabilistic soundness.

Let $\boldsymbol{c} \in \mathsf{F}^{m\times 1}$ be the random vector chosen by the Verifier on Step 3. Since $\boldsymbol{A}$ has full row rank, there is a unique rational solution $\boldsymbol{u} \in \mathsf{F}(x)^{1\times m}$ such that $\boldsymbol{uA} = \boldsymbol{v}$, and by the assumption of this case we have $\mathrm{denom}(\boldsymbol{u}) \neq 1$; besides, lemma 3.3.2 ensures $\deg(\mathrm{denom}(\boldsymbol{u})) \leqslant md_{\boldsymbol{A}}$ and $\deg(\mathrm{numer}(\boldsymbol{u})) \leqslant (m-1)d_{\boldsymbol{A}} + d_{\boldsymbol{v}}$. Then, lemma 3.3.1 tells us that the probability that $\boldsymbol{uc}$ is a polynomial is at most $1/\#\mathsf{S}$. Let g be the polynomial sent by the Prover at Step 4. If $\boldsymbol{uc}$ is not a polynomial, then $\boldsymbol{uc} - g$ is a nonzero rational fraction with numerator degree at most

$$\max(\deg(\mathrm{numer}(\boldsymbol{u})), \deg(g) + \deg(\mathrm{denom}(\boldsymbol{u}))) \leqslant 2md_{\boldsymbol{A}} + d_{\boldsymbol{v}}. \tag{3.3.1}$$

From lemma 3.1.2, the probability that $\boldsymbol{A}(\alpha)$ does not have full row rank is at most $md_{\boldsymbol{A}}/\#\mathsf{S}$. Otherwise, the vector $\boldsymbol{w} = \boldsymbol{u}(\alpha)$ is the unique solution to $\boldsymbol{wA}(\alpha) = \boldsymbol{v}(\alpha)$, so the Prover is obliged to send this $\boldsymbol{w}$ on Step 6.

Then, if the Verifier incorrectly accepts, we must have $\boldsymbol{wc} = g(\alpha)$, which means $\boldsymbol{u}(\alpha)\boldsymbol{c} = g(\alpha)$, or in other words, α is a root of $\boldsymbol{uc} - g$. The degree bound in eq. (3.3.1) gives an upper bound on the number of such roots $\alpha \in \mathsf{F}$.

Therefore the Verifier accepts only when either $\boldsymbol{uc} \in \mathsf{F}[x]$, or $\boldsymbol{A}(\alpha)$ is singular, or α is a root of $\boldsymbol{uc} - g$, which by the union bound has probability at most $(3md_{\boldsymbol{A}} + d_{\boldsymbol{v}} + 1)/\#\mathsf{S}$, as stated.

Case 3: $\boldsymbol{v} \notin \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A})$ Again, the assertion being verified is false, and our goal is to prove probabilistic soundness. As with the last case, assume by way of contradiction that the Verifier accepts.

Consider the augmented matrix

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{v} \end{bmatrix} \in \mathsf{F}[x]^{(m+1)\times n}.$$

By the assumption of this case, $\operatorname{rank}(\tilde{\boldsymbol{A}}) = \operatorname{rank}(\boldsymbol{A}) + 1 = m + 1$. But the vector $\boldsymbol{w}$ provided at Step 7 is such that $\boldsymbol{w}\boldsymbol{A}(\alpha) = \boldsymbol{v}(\alpha)$: it corresponds to a nonzero vector $[-\boldsymbol{w} \;\; 1]$ in the left kernel of $\tilde{\boldsymbol{A}}(\alpha)$, which therefore has rank at most m.

Since all $(m+1) \times (m+1)$ minors of $\tilde{\boldsymbol{A}}$ have degree at most $md_{\boldsymbol{A}} + d_{\boldsymbol{v}}$, the proof of lemma 3.1.2 shows that the probability that $\operatorname{rank}(\tilde{\boldsymbol{A}}(\alpha)) \leqslant m$ is at most $(md_{\boldsymbol{A}} + d_{\boldsymbol{v}})/\#\mathsf{S}$. □

3.3.2 Arbitrary rank case

Now we move to the general case of a matrix $\boldsymbol{A}$ with arbitrary rank r.

The idea behind our protocol is inspired by Mulders and Storjohann (2004). We make use of the full row rank case by considering a matrix $\boldsymbol{C} \in \mathsf{F}[x]^{r\times m}$ such that $\boldsymbol{C}\boldsymbol{A}$ has full row rank. Thus $\boldsymbol{C}\boldsymbol{A}$ has the same rational row space as $\boldsymbol{A}$, and if $\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, then there is a unique rational vector $\boldsymbol{w} \in \mathsf{F}(x)^{1\times r}$ such that $\boldsymbol{w}\boldsymbol{C}\boldsymbol{A} = \boldsymbol{v}$. In particular, for $f = \operatorname{denom}(\boldsymbol{w})$ we have $f\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, and therefore if $f = 1$ the verification is already complete.

Although it might be that $\boldsymbol{w}$ has a nontrivial denominator f, this approach can still be used for verification by considering *several* such matrices $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t$ and rational vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t$ with denominators $f_1, \ldots, f_t$. Indeed, we will see that these matrices can be chosen such that the greatest common divisor of $f_1, \ldots, f_t$ is 1; as we show in the next lemma, this implies $\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$.

Lemma 3.3.4. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ and $\boldsymbol{v} \in \mathsf{F}[x]^{1\times n}$. Let $f_1, \ldots, f_t \in \mathsf{F}[x]$ be such that $f_i\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ for $1 \leqslant i \leqslant t$. If $\gcd(f_1, \ldots, f_t) = 1$, then $\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$.*

Proof. The gcd assumption implies that there exist $u_1, \ldots, u_t \in \mathsf{F}[x]$ such that $u_1f_1 + \cdots + u_tf_t = 1$. It directly follows that $\boldsymbol{v} = u_1(f_1\boldsymbol{v}) + \cdots + u_t(f_t\boldsymbol{v})$ belongs to $\operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$. □

Before giving the full protocol for row membership, we first present a subprotocol CoPrime to confirm that the greatest common divisor of a set of polynomials is 1.

Lemma 3.3.5. *Let $d = \max_i \deg(f_i)$, and suppose $\#S \geqslant 2d$. Then Protocol 3.10 is a complete and probabilistically sound interactive protocol which requires $O(d+t)$ communication and has Verifier cost $O(dt)$. If $\gcd(f_1, \ldots, f_t) = 1$, then there is a Las Vegas randomized algorithm for the Prover with expected cost bound $\widetilde{O}(dt)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(2d-1)/\#S$.*

Proof. The communication and Verifier costs are clear.

Write $g = \gcd(f_1, \ldots, f_t)$, and suppose first that $g \neq 1$. Since g divides $f_1s_1 + hs_2$, the polynomial $f_1s_1 + hs_2 - 1$ is nonzero and has degree at most $2d-1$. If the Verifier incorrectly accepts, then α must be a root of this polynomial, which justifies the probability claim.

If $g = 1$, then a well-known argument (von zur Gathen and Gerhard, 2013, Theorem 6.46) says that, for $\beta_3, \ldots, \beta_t$ chosen randomly from a subset $\mathsf{S} \subseteq \mathsf{F}$, the probability that

Protocol 3.10: CoPrime

Public: $t \geqslant 2$ polynomials $f_1, \ldots, f_t \in \mathsf{F}[x]$
Certifies: $\gcd(f_1, \ldots, f_t) = 1$

	Prover		Verifier
1.	Compute polynomials $s_1, s_2 \in \mathsf{F}[x]$ and scalars $\beta_3, \ldots, \beta_t \in \mathsf{F}$ s.t. $f_1 s_1 + h s_2 = 1$, $\deg(s_1) < \deg(h)$, and $\deg(s_2) < \deg(f_1)$, where $h = f_2 + \sum_{i=3}^{t} \beta_i f_i$	$\xrightarrow{s_1, s_2, \beta_3, \ldots, \beta_t}$	
2.			$\deg(s_1) \overset{?}{<} \max_{i \geqslant 2} \deg(f_i)$ $\deg(s_2) \overset{?}{<} \deg(f_1)$ $\alpha \xleftarrow{\$} \mathsf{S}$ $h_\alpha \leftarrow f_2(\alpha) + \sum_{i=3}^{t} \beta_i f_i(\alpha)$ $f_1(\alpha) s_1(\alpha) + h_\alpha s_2(\alpha) \overset{?}{=} 1$

$\gcd(f_1, h) \neq \gcd(f_1, \ldots, f_t)$ is at most $d/\#\mathsf{S}$. Based on the assumption that $\#\mathsf{S} \geqslant 2d$, the Prover can find such a tuple $\beta_3, \ldots, \beta_t$ after expected $O(1)$ iterations. Then computing the Bézout coefficients s_1, s_2 is done via the fast extended Euclidean algorithm on f_1 and h, which costs $\widetilde{O}(dt)$. □

Protocol 3.11 shows an interactive protocol verifying row space membership. For free (and as a necessary aspect of the protocol), the rank ρ is also verified.

The Prover first selects t matrices $\boldsymbol{C}_i \in \mathsf{F}^{r \times m}$ such that $\boldsymbol{C}_i \boldsymbol{A}$ has full row rank $r = \operatorname{rank}(\boldsymbol{A})$ and the corresponding denominators f_i of the rational solutions to $\boldsymbol{w} \boldsymbol{C}_i \boldsymbol{A} = \boldsymbol{v}$ have no common factor.

The Verifier then confirms that the gcd of all denominators is 1 using CoPrime. Using FullRankRowSpaceMembership, the Verifier also confirms that each $\boldsymbol{C}_i \boldsymbol{A}$ has full rank and each $f_i \boldsymbol{v}$ is in the row space of $\boldsymbol{C}_i \boldsymbol{A}$ and therefore in the row space of $\boldsymbol{A}$ as well; by lemma 3.3.4 this ensures that $\boldsymbol{v}$ is itself in the row space of $\boldsymbol{A}$.

To save communication costs, the matrices $\boldsymbol{C}_i$ have a certain structure:

Definition 3.3.6. *A matrix $\boldsymbol{C} \in \mathsf{F}^{m \times n}$ is a* sub-Toeplitz *matrix if $m \leqslant n$ and $\boldsymbol{C}$ consists of m rows selected out of a full $n \times n$ Toeplitz matrix.*

Note that we can always write such a matrix $\boldsymbol{C}$ as a sub-permutation matrix $\boldsymbol{S} \in$

$\{0,1\}^{m\times n}$ times the full Toeplitz matrix $\boldsymbol{T} \in \mathsf{F}^{n\times n}$, i.e., $\boldsymbol{C} = \boldsymbol{S}\boldsymbol{T}$[1]. The benefit for us is in the communication savings:

Lemma 3.3.7. *An $m \times n$ sub-Toeplitz matrix $\boldsymbol{C}$ can be sent with $O(n)$ communication.*

Proof. Writing $\boldsymbol{C} = \boldsymbol{S}\boldsymbol{T}$ as above, simply send the $2n-1$ entries of the full Toeplitz matrix $\boldsymbol{T}$ and the m row indices selected by $\boldsymbol{S}$. □

The number t of sub-Toeplitz matrices sent must be large enough, according to the field size, so that the Prover can actually find them with the required properties (see algorithm 2 below). This value t is computed by the Verifier and Prover independently as shown in Step 3, where we use the slight abuse of notation that, when F is infinite, $\log_{\#\mathsf{F}} \alpha = 0$ for any positive finite value α.

We now proceed to show how the Prover can actually find the values required on Step 4. We write $r = \operatorname{rank}(\boldsymbol{A})$; if the Prover is honest, then in fact $r = \rho$. The next lemma is inspired from (Mulders and Storjohann, 2004).

Lemma 3.3.8. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ with rank r; $\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$; $\boldsymbol{S} \in \{0,1\}^{r\times m}$ be a selection of r out of m rows; $p \in \mathsf{F}[x]$ be an irreducible polynomial; and $\boldsymbol{T} \in \mathsf{F}^{m\times m}$ be a Toeplitz matrix with entries chosen independently and uniformly at random from a finite subset S of F. Then either $\boldsymbol{S}\boldsymbol{T}\boldsymbol{A}$ always has rank strictly below r, or for any rational solution $\boldsymbol{w} \in \mathsf{F}(x)^{1\times r}$ to $\boldsymbol{w}\boldsymbol{S}\boldsymbol{T}\boldsymbol{A} = \boldsymbol{v}$, the probability that $\operatorname{rank}(\boldsymbol{S}\boldsymbol{T}\boldsymbol{A}) < r$ or that p divides $\operatorname{denom}(\boldsymbol{w})$ is at most $r/\#\mathsf{S}$.*

Proof. Let $\hat{\boldsymbol{T}}$ be a generic $m \times m$ Toeplitz matrix, defined by $2m-1$ indeterminates $z_1, \ldots, z_{2m-1}$. Because $\boldsymbol{I}_m$ is an evaluation of $\hat{\boldsymbol{T}}$, then clearly $\operatorname{rank}(\hat{\boldsymbol{T}}\boldsymbol{A}) = \operatorname{rank}(\boldsymbol{A}) = r$, and furthermore $\operatorname{rank}(\boldsymbol{S}\hat{\boldsymbol{T}}\boldsymbol{A}) = r$ if and only if $\boldsymbol{S}$ selects r linearly independent rows from $\hat{\boldsymbol{T}}\boldsymbol{A}$.

So for the remainder assume that $\operatorname{rank}(\boldsymbol{S}\hat{\boldsymbol{T}}\boldsymbol{A})$ is nonsingular over $(\mathsf{F}[x])[z_1, \ldots, z_{2m-1}]$; otherwise $\operatorname{rank}(\boldsymbol{S}\boldsymbol{T}\boldsymbol{A}) < r$ for any choice of $\boldsymbol{T}$, and we are done.

The structure of the proof is as follows. We first show the existence of a unimodular-completable matrix $\boldsymbol{U} \in \mathsf{F}[x]^{n\times r}$ such that $\boldsymbol{S}\boldsymbol{T}\boldsymbol{A}$ and $\boldsymbol{S}\boldsymbol{T}\boldsymbol{U}$ are closely related: in particular, they both have full rank r if and only if the latter has non-zero determinant, and p divides $\boldsymbol{w}$ only when this determinant is divisible by p. The proof proceeds to demonstrate these properties, as well as the fact that $\boldsymbol{S}\hat{\boldsymbol{T}}\boldsymbol{U}$ has nonzero determinant generically, and therefore with high probability for a random choice of $\boldsymbol{T}$.

Let $\boldsymbol{P} \in \{0,1\}^{n\times r}$ be a sub-permutation matrix which selects r linearly independent columns from $\boldsymbol{S}\hat{\boldsymbol{T}}\boldsymbol{A}$. Then $\operatorname{rank}(\boldsymbol{A}\boldsymbol{P}) = r$ and we consider a factorization $\boldsymbol{A}\boldsymbol{P} = \boldsymbol{U}\boldsymbol{B}$, where

- $\boldsymbol{B} \in \mathsf{F}[x]^{r\times r}$ is a row basis of $\boldsymbol{A}\boldsymbol{P}$ (and therefore $\boldsymbol{B}$ is nonsingular); and
- $\boldsymbol{U} \in \mathsf{F}[x]^{m\times r}$ can be completed to a square unimodular matrix, meaning there exists some matrix $\boldsymbol{V} \in \mathsf{F}[x]^{m\times(m-r)}$ such that $\det([\boldsymbol{U} \mid \boldsymbol{V}]) \in \mathsf{F}\backslash\{0\}$.

[1] We hope that the reader will forgive us for overloading the capital letter S: a bold $\boldsymbol{S}$ always refers to this sub-permutation matrix, while a sans-serif S refers to a subset of the field F used to select random elements.

Protocol 3.11: RowSpaceMembership

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$, $\boldsymbol{v} \in \mathsf{F}[x]^{1 \times n}$, $\rho \in \mathbf{N}$
Certifies: $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ and $\mathrm{rank}(\boldsymbol{A}) = \rho$

	Prover		Verifier
1.		RankUpperBound$(\boldsymbol{A}, \rho)$	
2.			Is convinced that $\mathrm{rank}(\boldsymbol{A}) \stackrel{?}{\leqslant} \rho$
3.			$\rho \stackrel{?}{\leqslant} \min(m, n)$ $t \leftarrow 1 +$ $\max\left(1, \lceil \log_{\#\mathsf{F}/\rho}(2\rho \deg(\boldsymbol{A})) \rceil\right)$
4.	Compute sub-Toeplitz $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t \in \mathsf{F}^{\rho \times m}$ and polynomials $f_1, \ldots, f_t \in \mathsf{F}[x]$ s.t. $\forall i, \mathrm{rank}(\boldsymbol{C}_i \boldsymbol{A}) = \rho$, and $\forall i, f_i \boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{C}_i \boldsymbol{A})$, and $\gcd(f_1, \ldots, f_t) = 1$		
		$\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t, f_1, \ldots, f_t \longrightarrow$	
5.			$\forall i, \deg(f_i) \stackrel{?}{\leqslant} \rho \deg(\boldsymbol{A})$
6.		CoPrime$(f_1, \ldots, f_t)$	
7.		**for** $i = 1, \ldots, t$ **do** FullRankRowSpaceMembership$(\boldsymbol{C}_i \boldsymbol{A}, f_i \boldsymbol{v})$	

Such a factorization always exists: if $\hat{\boldsymbol{B}} \in \mathsf{F}[x]^{r\times n}$ is any row basis of $\boldsymbol{A}$, then there is a unimodular matrix $[\boldsymbol{U} \mid \boldsymbol{V}] \in \mathsf{F}[x]^{m\times m}$ such that $[\boldsymbol{U} \mid \boldsymbol{V}][\hat{\boldsymbol{B}}^{\mathsf{T}} \mid \boldsymbol{0}]^{\mathsf{T}} = \boldsymbol{U}\hat{\boldsymbol{B}} = \boldsymbol{A}$; and then we have $\boldsymbol{UB} = \boldsymbol{AP}$ where $\boldsymbol{B} = \hat{\boldsymbol{B}}\boldsymbol{P}$. It is easily verified that $\boldsymbol{B}$ has full row rank and the same $\mathsf{F}[x]$-row space as $\boldsymbol{AP}$; that is, $\boldsymbol{B}$ is a row basis of $\boldsymbol{AP}$.

From this factorization and the fact that $\boldsymbol{B}$ is nonsingular, we know that

$$\operatorname{rank}(\boldsymbol{AP}) = \operatorname{rank}(\boldsymbol{S\hat{T}AP}) = \operatorname{rank}(\boldsymbol{S\hat{T}UB}) = \operatorname{rank}(\boldsymbol{S\hat{T}U}) = r$$

over the ring $(\mathsf{F}[x])[z_1, \ldots, z_{2m-1}]$.

Now because $[\boldsymbol{U} \mid \boldsymbol{V}]$ is unimodular, it is always nonsingular over the extension field $\mathsf{F}[x]/\langle p\rangle$, and therefore $\operatorname{rank}(\boldsymbol{U}) = r$ over $\mathsf{F}[x]/\langle p\rangle$. Since the entries of $\boldsymbol{S\hat{T}}$ do not contain x and from the rank condition above, this means that $\boldsymbol{S\hat{T}U}$ is nonsingular over $(\mathsf{F}[x]/\langle p\rangle)[z_1, \ldots, z_{2m-1}]$ for any choice of the polynomial p.

The determinant $\det(\boldsymbol{S\hat{T}U})$ is therefore a nonzero polynomial in $z_1, \ldots, z_{2m-1}$ over $\mathsf{F}[x]/\langle p\rangle$ with total degree at most r. Then, by the DeMillo-Lipton-Schwartz-Zippel lemma, the probability that $\det(\boldsymbol{STU}) \bmod p = 0$ is at most $r/\#\mathsf{S}$.

Connecting this back to $\boldsymbol{A}$, if $p \nmid \det(\boldsymbol{STU})$, then we have

$$r = \operatorname{rank}(\boldsymbol{STU}) = \operatorname{rank}(\boldsymbol{STUB}) = \operatorname{rank}(\boldsymbol{STAP}) \leqslant \operatorname{rank}(\boldsymbol{STA}) \leqslant r,$$

and hence $\boldsymbol{STA}$ has full row rank r.

Finally, we show that $\operatorname{denom}(\boldsymbol{w})$ divides $\det(\boldsymbol{STU})$; this implies $p \nmid \operatorname{denom}(\boldsymbol{w})$. Recall that $\boldsymbol{B}$ is nonsingular with the same $\mathsf{F}[x]$-row space as $\boldsymbol{AP}$; then, because $\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, we have $\boldsymbol{vP} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$, so there exists $\boldsymbol{y} \in \mathsf{F}[x]^{1\times r}$ such that $\boldsymbol{yB} = \boldsymbol{vP}$. In addition, since $\boldsymbol{STU}$ is nonsingular, there exists an *adjugate* matrix $\boldsymbol{D} \in \mathsf{F}[x]^{r\times r}$ such that $\det(\boldsymbol{STU})(\boldsymbol{STU})^{-1} = \boldsymbol{D}$. Putting these facts together, we have

$$\begin{aligned}
\boldsymbol{wSTA} &= \boldsymbol{v} \\
\boldsymbol{wSTAP} &= \boldsymbol{vP} \\
\boldsymbol{wSTUB} &= \boldsymbol{yB} \\
\boldsymbol{wSTU} &= \boldsymbol{y} \\
\boldsymbol{w}\det(\boldsymbol{STU}) &= \boldsymbol{yD}.
\end{aligned}$$

Because the right-hand side of the last equation has entries in $\mathsf{F}[x]$, then so does the left-land side, which means that $\det(\boldsymbol{STU})$ is a multiple of $\operatorname{denom}(\boldsymbol{w})$. Hence $\operatorname{denom}(\boldsymbol{w})$ is divisible by p only if $\det(\boldsymbol{STU})$ is divisible by p, which we already established occurs with probability at most $r/\#\mathsf{S}$. □

Repeatedly applying the previous lemma, involving calls to the rational linear solver of algorithm 1, leads to a Las Vegas randomized algorithm for an honest Prover.

Here we require the Prover to know a finite subset $\mathsf{S} \subseteq \mathsf{F}$. Because this set is never communicated nor part of the public information, it is not necessarily the same as any subset S used by the Verifier in other protocols. In order to match with Protocol 3.11, the Prover should choose $\mathsf{S} = \mathsf{F}$ if F is finite, and otherwise $\#\mathsf{S} \geqslant 2r^2 \deg(\boldsymbol{A})$.

Algorithm 2: Honest Prover for RowSpaceMembership

Input: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ with rank r, $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, finite $\mathsf{S} \subseteq \mathsf{F}$
Output: $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_t, f_1, \ldots, f_t$ satisfying the conditions of Step 4 from Protocol 3.11

1 $t \leftarrow 1 + \lceil \log_{\#\mathsf{S}/r}(2r \deg(\boldsymbol{A})) \rceil$
2 **repeat**
3 $\quad \boldsymbol{T}_1 \leftarrow$ random $m \times m$ Toeplitz matrix with entries from S
4 **until** $\mathrm{rank}(\boldsymbol{T}_1\boldsymbol{A}) = r$
5 $\boldsymbol{S} \in \{0,1\}^{r\times m} \leftarrow$ selection of r linearly independent rows from $\boldsymbol{T}_1\boldsymbol{A}$
6 $w_1 \leftarrow$ solution to $\boldsymbol{w}_1\boldsymbol{S}\boldsymbol{T}_1\boldsymbol{A} = \boldsymbol{v}$, using algorithm 1
7 **repeat**
8 $\quad i \leftarrow 2$
9 $\quad$ **while** $i \leqslant t$ **do**
10 $\quad\quad \boldsymbol{T}_i \leftarrow$ random $m \times m$ Toeplitz matrix with entries from S
11 $\quad\quad \boldsymbol{w}_i \leftarrow$ solution to $\boldsymbol{w}_i\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A} = \boldsymbol{v}$, using algorithm 1
12 $\quad\quad$ **if** $\boldsymbol{w}_i$ *is not* `LOW_RANK` **then** $i \leftarrow i + 1$
13 **until** $\gcd(\mathrm{denom}(\boldsymbol{w}_1), \ldots, \mathrm{denom}(\boldsymbol{w}_t)) = 1$
14 **return** $\boldsymbol{S}\boldsymbol{T}_1, \ldots, \boldsymbol{S}\boldsymbol{T}_t$ *and* $\mathrm{denom}(\boldsymbol{w}_1), \ldots, \mathrm{denom}(\boldsymbol{w}_t)$

Lemma 3.3.9. *If $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ and $\#S \geqslant 2r$ where $r = \mathrm{rank}(\boldsymbol{A})$, then algorithm 2 is a correct Las Vegas randomized algorithm with expected cost bound $\widetilde{O}(mnr^{\omega-2}d_{\boldsymbol{A}} + r^{\omega-1}d_{\boldsymbol{v}})$.*

Proof. Computing the rank and the row rank profile (giving r independent rows) on lines 4 and 5 can be done deterministically in the stated cost bound via the column rank profile algorithm from (Zhou, 2012, Section 11), just as was used in algorithm 1.

Each matrix product $\boldsymbol{T}_i\boldsymbol{A}$ can be explicitly computed in $\widetilde{O}(mnd_{\boldsymbol{A}})$ operations using $O(nd_{\boldsymbol{A}})$ Toeplitz-vector products, each done in $\widetilde{O}(m)$ operations by relying on fast polynomial multiplication (Bini and Pan, 1994, Problem 5.1).

If the algorithm returns, correctness is clear from the correctness of algorithm 1.

What remains is to prove the expected number of iterations of each nested loop.

From lemma 3.3.8, for each random Toeplitz matrix $\boldsymbol{T}_i$, the probability that $\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A}$ is nonsingular is at least $1 - r/\#\mathsf{S} \geqslant 1/2$. Therefore the expected number of iterations of the first loop is at most 2, and the expected number of iterations of the nested while loop until i reaches t is at most $2t$.

Write $f_i = \mathrm{denom}(\boldsymbol{w}_i)$. To find the expected number of iterations of the outer loop on lines 7 to 13, we need the probability that $\gcd(f_1, \ldots, f_t) = 1$ given that each $\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A}$ has full row rank r.

If $\gcd(f_1, \ldots, f_t) \neq 1$, then there is some irreducible polynomial p which divides every denominator $f_1, \ldots, f_t$. Because the $\boldsymbol{T}_i$'s are chosen independently of each other, the events "p divides f_i" are pairwise independent; thus, according to lemma 3.3.8, the probability that any given irreducible polynomial p is such a common factor is at most $(r/\#\mathsf{S})^{t-1}$.

The degree of f_1 is at most $rd_{\boldsymbol{A}}$ since $\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A}$ is $r \times n$ with degree $d_{\boldsymbol{A}}$; this also gives an upper bound on the number of distinct irreducible factors p of f_1. Taking the union bound we see that the probability of *any* factor being shared by all denominators is at most

$$\frac{r^t d_{\boldsymbol{A}}}{(\#\mathsf{S})^{t-1}},$$

which is at most $\frac{1}{2}$ from the definition of t. Therefore the expected number of iterations of the outer loop is $O(1)$.

The stated cost bound follows from lemma 3.3.2. It does not involve t because we can see that $t \in O(\log(rd_{\boldsymbol{A}}))$, which is subsumed by the soft-oh notation. □

For the sake of simplicity in presentation, and because they do not affect the asymptotic cost bound, we have omitted a few optimizations to the Prover's algorithm that would be useful in practice, namely:

- The Prover can reduce to the full column rank case by computing a column rank profile of $\boldsymbol{A}$ once at the beginning (using Zhou (2012, Chapter 11)), and then removing corresponding non-pivot columns from $\boldsymbol{A}$ and $\boldsymbol{v}$. This does not change the correctness, but means that each matrix $\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A}$ is square.
- When each $\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A}$ is square, computing $\boldsymbol{w}_i$ can be done in a simpler way than by calling algorithm 1, as follows: check that $\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A}$ is nonsingular to confirm the rank, and then use a fast linear system solver to obtain $\boldsymbol{w}_i$.
- The solution vectors $\boldsymbol{w}_i$ may be re-used in the subprotocols FullRankRowSpaceMembership confirming that each $f_i\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{S}\boldsymbol{T}_i\boldsymbol{A})$.

We conclude the section by proving RowSpaceMembership is complete, sound, and efficient. As above, write $d_{\boldsymbol{A}} = \max(1, \deg(\boldsymbol{A}))$ and $d_{\boldsymbol{v}} = \max(1, \deg(\boldsymbol{v}))$, and let $r = \mathrm{rank}(\boldsymbol{A})$.

Theorem 3.3.10. *Assuming $\#S \geqslant 2\min(m,n)d_{\boldsymbol{A}}$, then Protocol 3.11 is a complete and probabilistically sound interactive protocol which requires $O(n + md_{\boldsymbol{A}}t + d_{\boldsymbol{v}}t)$ communication and has Verifier cost $O(mnd_{\boldsymbol{A}}t + nd_{\boldsymbol{v}}t)$. If $\boldsymbol{v} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, there is a Las Vegas randomized algorithm for the Prover with expected cost $\widetilde{O}(mnr^{\omega-2}d_{\boldsymbol{A}} + r^{\omega-1}d_{\boldsymbol{v}})$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(3rd_{\boldsymbol{A}} + d_{\boldsymbol{v}} + 1)/\#S$.*

Proof. For the communication, note that sending each $\boldsymbol{C}_i$ has communication cost $O(m)$ from lemma 3.3.7. Furthermore, the Verifier does not actually compute the products $\boldsymbol{C}_i\boldsymbol{A}$, but rather uses these as a *black box* for matrix-vector products in the two subprotocols. For any scalar $\alpha \in \mathsf{F}$, the complexity of computing $\boldsymbol{C}_i\boldsymbol{A}(\alpha)$ times any vector of scalars on the left or right-hand side is $O(mnd_{\boldsymbol{A}})$.

Along with the degree conditions on each f_i and theorems 3.2.3, 3.2.4 and 3.3.3 and lemma 3.3.5, this proves the communication and Verifier cost claims.

The Prover's cost comes from lemma 3.3.9, which dominates the cost for the Prover in any of the subprotocols.

If the rank conditions being verified on Steps 1 and 7 are true, then all matrices $\boldsymbol{C}_i\boldsymbol{A}$ have full row rank equal to the rank of $\boldsymbol{A}$, that is, $\operatorname{rank}(\boldsymbol{C}_i\boldsymbol{A}) = \rho = r$. And if the statements verified on Steps 6 and 7 are true as well, then we have $\boldsymbol{v} \in \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ according to lemma 3.3.4. Therefore the soundness of this protocol depends only on the probabilistic soundness of those subprotocols.

For the remainder of the proof, we assume that $\boldsymbol{v} \notin \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ and we want to know an upper bound on the probability that the Verifier incorrectly accepts. For this, we divide into cases depending on which subprotocol incorrectly accepted:

Case 1: $\operatorname{rank}(\boldsymbol{A}) > \rho$ According to theorem 3.2.4, the probability that the Verifier incorrectly accepts in RankUpperBound on Step 1 is at most $(rd_{\boldsymbol{A}} + 1)/\#\mathsf{S}$.

Case 2: $\operatorname{rank}(\boldsymbol{A}) \leqslant \rho$ **and** $\gcd(f_1, \ldots, f_t) \neq 1$ We know that each $\deg(f_i) \leqslant rd_{\boldsymbol{A}}$, where r is the true rank of $\boldsymbol{A}$. By lemma 3.3.5, the probability that the Verifier incorrectly accepts in subprotocol CoPrime is at most $(2\max_i(\deg(f_i)) - 1)/\#\mathsf{S}$, which is at most $(2rd_{\boldsymbol{A}} - 1)/\#\mathsf{S}$.

Case 3: $\operatorname{rank}(\boldsymbol{A}) \leqslant \rho$ **and** $\gcd(f_1, \ldots, f_t) = 1$ Then, by lemma 3.3.4, there exists $i \in \{1, \ldots, t\}$ such that $f_i\boldsymbol{v} \notin \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, and thus either $\operatorname{rank}(\boldsymbol{C}_i\boldsymbol{A}) < \rho$ or $f_i\boldsymbol{v} \notin \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{C}_i\boldsymbol{A})$. That is, the statement being verified by FullRankRowSpaceMembership on the ith iteration of Step 7 is false.

Because of the degree checks on Step 5, we know that $\deg(f_i\boldsymbol{v}) \leqslant rd_{\boldsymbol{A}} + d_{\boldsymbol{v}}$. Therefore from theorem 3.3.3, the probability that the Verifier incorrectly accepts in FullRankRowSpaceMembership is at most $(4rd_{\boldsymbol{A}} + d_{\boldsymbol{v}} + 1)/\#\mathsf{S}$.

Observe that the three cases are disjoint and cover all possibilities. In every case, the probability that the Verifier incorrectly accepts is at most that in Case 3, which proves the last claim in the Theorem statement. □

We note that it is always possible to conduct the checks on Step 7 of RowSpaceMembership in parallel, so that the total number of *rounds* of communication in the protocol is $O(1)$.

A crucial factor in the communication and Verifier costs as seen in theorem 3.3.10 is the value of t, which in any case satisfies $t \in O(\log(\min(m, n)))$ due to the condition on the size of S, so this adds only a logarithmic factor to the cost. Indeed, when the set S of field elements is large enough, t can be as small as 2. For clarity, we state as a corollary a condition under which this logarithmic factor can be eliminated.

Corollary 3.3.11. *If* $\#S \geqslant 2mnd_{\boldsymbol{A}}$*, then Protocol 3.11 requires only* $O(n + md_{\boldsymbol{A}} + d_{\boldsymbol{v}})$ *communication and has Verifier cost* $O(mnd_{\boldsymbol{A}} + nd_{\boldsymbol{v}})$.

3.4 Row spaces and normal forms

In this section, we use the row space membership protocol from the previous section in order to certify the equality of the row spaces of two matrices. Along with additional non-interactive checks by the Verifier, this can also be applied to prove the correctness of certain important normal forms of polynomial matrices.

3.4.1 Row space subset and row basis

We will use RowSpaceMembership to give a protocol for the certification of *row space subset*; by this we mean the problem of deciding whether the row space of $\boldsymbol{A}$ is contained in the row space of $\boldsymbol{B}$, for two given matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

Our approach is the following: take a random vector $\boldsymbol{\lambda}$ and certify that the row space element $\boldsymbol{\lambda A}$ is in the row space of $\boldsymbol{B}$, the latter being done via row space membership (section 3.3). We will see that taking $\boldsymbol{\lambda}$ with entries in the base field is enough to ensure good probability of success.

Lemma 3.4.1. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ and $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$. Let $R \in \{\mathsf{F}[x], \mathsf{F}(x)\}$. Then the following statement holds: Assuming that*

$$\operatorname{RowSp}_R(\boldsymbol{A}) \not\subseteq \operatorname{RowSp}_R(\boldsymbol{B}),$$

then the F-vector space

$$\mathsf{V} = \left\{\boldsymbol{\lambda} \in \mathsf{F}^{1\times m} \mid \boldsymbol{\lambda A} \in \operatorname{RowSp}_R(\boldsymbol{B})\right\}$$

has dimension at most $m-1$. For $\boldsymbol{\lambda} \in \mathsf{F}^{1\times m}$ with entries chosen independently and uniformly at random from a finite subset $S \subseteq \mathsf{F}$ then $\boldsymbol{\lambda A} \in \operatorname{RowSp}_R(\boldsymbol{B})$ with probability at most $\frac{1}{\#S}$.

Proof. Suppose that the vector space V has dimension at least m. Then V is the entire space $\mathsf{F}^{1\times m}$, and every row of $\boldsymbol{A}$ is in $\operatorname{RowSp}_R(\boldsymbol{B})$; hence $\operatorname{RowSp}_R(\boldsymbol{A}) \subseteq \operatorname{RowSp}_R(\boldsymbol{B})$, a contradiction. Then the probability that the uniformly random vector $\boldsymbol{\lambda}$ belongs to the proper subspace $\mathsf{V} \subsetneq \mathsf{F}^{1\times m}$ follows from lemma 3.1.1. □

In the following, let $r_{\boldsymbol{A}}$ and $r_{\boldsymbol{B}}$ denote respectively the ranks of $\boldsymbol{A}$ and $\boldsymbol{B}$, and let $d_{\boldsymbol{A}} = \max(1, \deg(\boldsymbol{A}))$ and $d_{\boldsymbol{B}} = \max(1, \deg(\boldsymbol{B}))$.

Theorem 3.4.2. *Protocol 3.12 is a probabilistically sound interactive protocol, and is complete assuming $\#S \geqslant 2\ell d_{\boldsymbol{B}}$ in its subprotocols. It requires $O(n + (\ell d_{\boldsymbol{B}} + d_{\boldsymbol{A}})\log(\ell))$ communication and has Verifier cost*

$$O((\ell n d_{\boldsymbol{B}} + n d_{\boldsymbol{A}})\log(\ell) + mn d_{\boldsymbol{A}}).$$

If $\operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) \subseteq \operatorname{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$, there is a Las Vegas randomized algorithm for the Prover with expected cost

$$\widetilde{O}\big(\ell n r_{\boldsymbol{B}}^{\omega-2} d_{\boldsymbol{B}} + r_{\boldsymbol{B}}^{\omega-1} d_{\boldsymbol{A}} + mn d_{\boldsymbol{A}}\big).$$

Protocol 3.12: RowSpaceSubset

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell \times n}$, $\rho \in \mathbf{N}$
Certifies: $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) \subseteq \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$ and $\mathrm{rank}(\boldsymbol{B}) = \rho$

Prover		Verifier
1.		$\boldsymbol{\lambda} \xleftarrow{\$} \mathsf{S}^{1 \times m}$
2.		$\boldsymbol{v} \leftarrow \boldsymbol{\lambda}\boldsymbol{A}$
	$\xleftarrow{\boldsymbol{v}}$	
3.	RowSpaceMembership$(\boldsymbol{B}, \boldsymbol{v})$	

Otherwise, the probability that the Verifier incorrectly accepts is at most

$$\frac{4r_{\boldsymbol{B}}d_{\boldsymbol{B}} + d_{\boldsymbol{A}} + 2}{\#S}.$$

Proof. The Verifier may incorrectly accept if either $\boldsymbol{\lambda}$ is such that $\boldsymbol{\lambda}\boldsymbol{A} \in \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$, which happens with probability at most $1/\#\mathsf{S}$ by lemma 3.4.1, or the subprotocol RowSpaceMembership has incorrectly accepted. From Theorem 3.3.10, and the union bound, we obtain the claimed probability bound. □

Repeating this check in both directions proves that two matrices have the same row space.

Protocol 3.13: RowSpaceEquality

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell \times n}$, $\rho \in \mathbf{N}$
Certifies: $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$ and $\mathrm{rank}(\boldsymbol{A}) = \mathrm{rank}(\boldsymbol{B}) = \rho$

Prover		Verifier
1.	RowSpaceSubset$(\boldsymbol{A}, \boldsymbol{B})$	
2.	RowSpaceSubset$(\boldsymbol{B}, \boldsymbol{A})$	

Theorem 3.4.3. *Let $r = \max(r_{\boldsymbol{A}}, r_{\boldsymbol{B}})$ and $d = \max(d_{\boldsymbol{A}}, d_{\boldsymbol{B}})$. Protocol 3.13 is a probabilistically sound interactive protocol, and is complete assuming $\#S \geqslant 2\max(md_{\boldsymbol{A}}, \ell d_{\boldsymbol{B}})$. It requires*

$$O((m\log(m) + \ell\log(\ell))d + n) \subset \widetilde{O}(md + \ell d + n)$$

communication and has Verifier cost

$$O((m\log(m)+\ell\log(\ell))nd) \subset \widetilde{O}(mnd+\ell nd).$$

If $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$*, there is a Las Vegas randomized algorithm for the Prover with expected cost* $\widetilde{O}((m+\ell)nr^{\omega-2}d)$*. Otherwise, the probability that the Verifier incorrectly accepts is at most* $(4rd+d+2)/\#S$*.*

From RowSpaceEquality, we deduce a protocol verifying the property that $\boldsymbol{B}$ is a row basis of $\boldsymbol{A}$, that is, a matrix which has the same row space as $\boldsymbol{A}$ and which has full row rank.

Protocol 3.14: RowBasis

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$
Certifies: $\boldsymbol{B}$ is a row basis of $\boldsymbol{A}$

Prover		Verifier
1.	RowSpaceEquality$(\boldsymbol{A}, \boldsymbol{B})$	

Corollary 3.4.4. *Let* $r = \max(r_{\boldsymbol{A}}, r_{\boldsymbol{B}})$ *and* $d = \max(d_{\boldsymbol{A}}, d_{\boldsymbol{B}})$*. Then, Protocol 3.14 is a probabilistically sound interactive protocol, and is complete assuming* $\#S \geqslant 2\max(md_{\boldsymbol{A}}, \ell d_{\boldsymbol{B}})$*. It requires*

$$O((m\log(m)+\ell\log(\ell))d+n) \subset \widetilde{O}(md+\ell d+n)$$

communication and has Verifier cost

$$O((m\log(m)+\ell\log(\ell))nd) \subset \widetilde{O}(mnd+\ell nd).$$

If $\boldsymbol{B}$ *is a row basis of* $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$*, there is a Las Vegas randomized algorithm for the Prover with expected cost* $\widetilde{O}(mn\ell^{\omega-2}d)$*, with* $\ell = r_{\boldsymbol{A}}$ *in this case. Otherwise, the probability that the Verifier incorrectly accepts is at most* $(4rd+d+3)/\#S$*.*

3.4.2 Normal forms

Here, we give protocols for certifying *normal forms* of polynomial matrices, including the Hermite form (Hermite, 1851; MacDuffee, 1933) and the Popov form (Popov, 1972; Kailath, 1980). These forms are specific row bases with useful properties such as being triangular for the former or having minimal degrees for the latter, and being unique in the sense that a given matrix in $\mathsf{F}[x]^{m\times n}$ has exactly one row basis in Hermite (resp. Popov) form.

Roughly speaking, the Hermite form is a row echelon form that stays within the underlying ring.

Definition 3.4.5. *A matrix $\boldsymbol{B} = [b_{i,j}] \in \mathsf{F}[x]^{r\times n}$ with $r \leqslant n$ is in* Hermite form *if there are* pivot indices $1 \leqslant k_1 < \cdots < k_r \leqslant n$ *such that:*

(i) (Pivots are monic, hence nonzero)

b_{i,k_i} *is monic for all* $1 \leqslant i \leqslant r$*,*

(ii) (Entries right of pivots are zero)

$b_{i,j} = 0$ *for all* $i \leqslant i \leqslant r$ *and* $k_i < j \leqslant n$*,*

(iii) (Entries below pivots have smaller degree)

$\deg(b_{i',k_i}) < \deg(b_{i,k_i})$ *for all* $1 \leqslant i < i' \leqslant r$.

Each entry at row i and column k_i is called a *pivot*. Observe that these conditions guarantee $\boldsymbol{B}$ has full row rank, hence the use of the notation r for the row dimension. For a matrix $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, its Hermite form $\boldsymbol{B} \in \mathsf{F}[x]^{r\times n}$ is the unique row basis of $\boldsymbol{A}$ which is in Hermite form.

Protocol HermiteForm certifies that a matrix $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$ is the Hermite form of $\boldsymbol{A}$. It first checks that $\boldsymbol{B}$ is in Hermite form, and then it checks that $\boldsymbol{B}$ and $\boldsymbol{A}$ have the same row space using RowSpaceEquality from section 3.4.1.

Protocol 3.15: HermiteForm

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$
Certifies: $\boldsymbol{B}$ is the Hermite form of $\boldsymbol{A}$

Prover		Verifier
1.		Check that $\boldsymbol{B}$ satisfies definition 3.4.5
2.	RowSpaceEquality$(\boldsymbol{A}, \boldsymbol{B})$	

Theorem 3.4.6. *Let $r = \max(r_{\boldsymbol{A}}, r_{\boldsymbol{B}})$ and $d = \max(d_{\boldsymbol{A}}, d_{\boldsymbol{B}})$. Protocol 3.15 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant 2\max(md_{\boldsymbol{A}}, \ell d_{\boldsymbol{B}})$ in its subprotocol. It requires $O(md\log(m) + n)$ communication and has Verifier cost $O(mnd\log(m))$. If $\boldsymbol{B}$ is the Hermite form of $\boldsymbol{A}$, there is a Las Vegas randomized algorithm for the Prover with expected cost $\widetilde{O}(mnr^{\omega-2}d)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(4rd + d + 2)/\#S$.*

Proof. To check that $\boldsymbol{B}$ is in Hermite form at Step 1, the Verifier first computes the pivot indices as the index of the first nonzero on each row, then checks the degree conditions specified in definition 3.4.5. (If any row is zero, $\boldsymbol{B}$ is not in Hermite form.) This is a deterministic check with complexity only $O(\ell n)$.

As discussed previously, the fact that $\boldsymbol{B}$ is in Hermite form immediately implies that it has full row rank ℓ, and hence checking the row space equality is sufficient to confirm that $\boldsymbol{B}$ is a row basis for $\boldsymbol{A}$.

The subprotocol RowSpaceEquality dominates the complexity and is also the only possibility for the Verifier to incorrectly accept when the statement is false; hence the stated costs follow directly from theorem 3.4.3. □

While the Hermite form has an echelon shape, it is also common in polynomial matrix computations to resort to the Popov form, for which the pivot of a row is no longer the rightmost nonzero entry but rather the rightmost entry whose degree is maximal among the entries of that row. This form loses the echelon shape, but has the advantage of having smaller-degree entries than the Hermite form.

Here we consider the more general *shifted* forms (Van Barel and Bultheel, 1992; Beckermann et al., 2006), which encompass Hermite forms and Popov forms via the use of the following degree measure. For a given tuple $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbf{Z}^n$, the $\boldsymbol{s}$-degree of the row vector $\boldsymbol{v} = [v_1 \ \cdots \ v_n] \in \mathsf{F}[x]^{1\times n}$ is

$$\deg_{\boldsymbol{s}}(\boldsymbol{v}) = \max(\deg(v_1) + s_1, \ldots, \deg(v_n) + s_n).$$

We use the notation $\boldsymbol{B}_{i,*}$ to denote the ith row of the matrix $\boldsymbol{B}$.

Definition 3.4.7. *Let $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbf{Z}^n$. A matrix $\boldsymbol{B} = [b_{i,j}] \in \mathsf{F}[x]^{r\times n}$ with $r \leqslant n$ is in* $\boldsymbol{s}$-Popov form *if there are indices $1 \leqslant k_1 < \cdots < k_r \leqslant n$ such that,*

(i) (Pivots are monic and determine the row degree)
b_{i,k_i} *is monic and* $\deg(b_{i,k_i}) + s_{k_i} = \deg_{\boldsymbol{s}}(\boldsymbol{B}_{i,*})$ *for all* $1 \leqslant i \leqslant r$,

(ii) (Entries right of pivots do not reach the row degree)
$\deg(b_{i,j}) + s_j < \deg_{\boldsymbol{s}}(\boldsymbol{B}_{i,*})$ *for all* $1 \leqslant i \leqslant r$ *and* $k_i < j \leqslant n$,

(iii) (Entries above and below pivots have lower degree)
$\deg(b_{i',k_i}) < \deg(b_{i,k_i})$ $1 \leqslant i' \neq i \leqslant r$.

The usual Popov form corresponds to the uniform shift $\boldsymbol{s} = (0, \ldots, 0)$. Furthermore, one can verify that, specifying the shift as $\boldsymbol{s} = (nt, \ldots, 2t, t)$ for any given $t > \deg(\boldsymbol{B})$, then the Hermite form is the same as the $\boldsymbol{s}$-Popov form (Beckermann et al., 2006, Lem. 2.6).

For a matrix $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, there exists a unique row basis $\boldsymbol{B} \in \mathsf{F}[x]^{r\times n}$ of $\boldsymbol{A}$ which is in $\boldsymbol{s}$-Popov form (Beckermann et al., 2006, Thm. 2.7); $\boldsymbol{B}$ is called the $\boldsymbol{s}$*-Popov form of* $\boldsymbol{A}$. Generalizing Protocol 3.15 to this more general normal form yields Protocol 3.16 (although the former could be derived as a particular case of the latter for a specific shift $\boldsymbol{s}$, we preferred to write both explicitly for the sake of clarity).

The next result is identical to Theorem 3.4.6, in both statement and proof. The only difference in the protocol is determining the indices of each pivot column in order to confirm the conditions of $\boldsymbol{s}$-Popov form; this can be accomplished in linear time by first computing the $\boldsymbol{s}$-degree of the row and then finding the rightmost column which determines this shifted row degree.

Protocol 3.16: ShiftedPopovForm

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbf{Z}^n$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$
Certifies: $\boldsymbol{B}$ is the $\boldsymbol{s}$-Popov form of $\boldsymbol{A}$

	Prover	Verifier
1.		Check that $(\boldsymbol{s}, \boldsymbol{B})$ satisfies Definition 3.4.7
2.	RowSpaceEquality$(\boldsymbol{A}, \boldsymbol{B})$	

Theorem 3.4.8. *Let $r = \max(r_{\boldsymbol{A}}, r_{\boldsymbol{B}})$ and $d = \max(d_{\boldsymbol{A}}, d_{\boldsymbol{B}})$. Protocol 3.16 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant 2\max(md_{\boldsymbol{A}}, \ell d_{\boldsymbol{B}})$ in its subprotocol. It requires $O(md\log(m) + n)$ communication and has Verifier cost $O(mnd\log(m))$. If $\boldsymbol{B}$ is the $\boldsymbol{s}$-Popov form of $\boldsymbol{A}$, there is a Las Vegas randomized algorithm for the Prover with expected cost $\widetilde{O}(mnr^{\omega-2}d)$. Otherwise, the probability that the Verifier incorrectly accepts is at most $(4rd + d + 2)/\#S$.*

3.5 Saturation and kernel bases

In this section, we use the protocols described in previous sections to design protocols verifying computations related to saturations and kernels of polynomial matrices.

3.5.1 Saturation and saturated matrices

The saturation of a matrix over a principal ideal domain is a useful tool in computations; we refer to (Bourbaki, 1972, Section II.§2.4) for a general definition of saturation. It was exploited for example in (Zhou and Labahn, 2013) where a matrix is factorized as the product of a column basis times some saturation basis, and in (Neiger et al., 2018) in order to find the location of pivots in the context of the computation of normal forms. The saturation can be computed from the Hermite form, as described in (Pernet and Stein, 2010, Section 8) for integer matrices, and alternatively it can be obtained as a left kernel basis of a right kernel basis of the matrix as we prove below (lemma 3.5.3).

Definition 3.5.1. *The* saturation *of a matrix $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ is the $\mathsf{F}[x]$-module*

$$\mathrm{Saturation}(\boldsymbol{A}) = \mathsf{F}[x]^{1\times n} \cap \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A});$$

it contains $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$ and has rank $r = \mathrm{rank}(\boldsymbol{A})$. A saturation basis of $\boldsymbol{A}$ *is a matrix in $\mathsf{F}[x]^{r\times n}$ whose rows form a basis of the saturation of $\boldsymbol{A}$. A matrix is said to be* saturated *if its saturation is equal to its $\mathsf{F}[x]$-row space.*

Two matrices with the same saturation may have different $\mathsf{F}[x]$-row spaces. For example, the matrices

$$\begin{bmatrix} 1 & 1 \\ x^2 & x^2+x \\ x & x \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 1+x^2 \\ 0 & x^2 \end{bmatrix}$$

have the same saturation $\mathsf{F}[x]^{1\times 2}$, but the $\mathsf{F}[x]$-row space of the former matrix contains $[0\ x]$ which is not in the $\mathsf{F}[x]$-row space of the latter matrix. We also remark that all nonsingular matrices in $\mathsf{F}[x]^{n\times n}$ have saturation equal to $\mathsf{F}[x]^{1\times n}$.

The saturation is defined in terms of the $\mathsf{F}(x)$-row space of the matrix: two matrices have the same saturation if and only if they have the same $\mathsf{F}(x)$-row space. In particular, $\boldsymbol{A}$ is saturated if and only if any row basis of $\boldsymbol{A}$ is saturated. This yields the following characterization for matrices having full column rank.

Lemma 3.5.2. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ have full column rank. Then $\boldsymbol{A}$ is saturated if and only if $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathsf{F}[x]^{1\times n}$.*

Proof. Since $\boldsymbol{A}$ has full column rank, its row bases are nonsingular $n \times n$ matrices, or equivalently, $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A}) = \mathsf{F}(x)^{1\times n}$. Hence the saturation of $\boldsymbol{A}$ is $\mathsf{F}[x]^{1\times n}$, and the equivalence follows by definition of being saturated. □

Thus, in this case, verifying that $\boldsymbol{A}$ is saturated boils down to verifying that $\mathsf{F}[x]^{1\times n}$ is a subset of $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A})$, which can be done using RowSpaceSubset.

To obtain a similar result in the case of matrices with full row rank, we will rely on the following characterization of the saturation using kernel bases.

Lemma 3.5.3. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ have rank r, and let $\boldsymbol{K} \in \mathsf{F}[x]^{n\times(n-r)}$ be a basis for the right kernel of $\boldsymbol{A}$. Then, $\mathrm{Saturation}(\boldsymbol{A})$ is the left kernel of $\boldsymbol{K}$. In particular, the saturation bases of $\boldsymbol{A}$ are precisely the left kernel bases of $\boldsymbol{K}$.*

Proof. Each row of $\boldsymbol{A}$ is in the left kernel of $\boldsymbol{K}$, hence so is any polynomial vector $\boldsymbol{v} \in \mathsf{F}[x]^{1\times n}$ which is an $\mathsf{F}(x)$-linear combination of rows of $\boldsymbol{A}$, that is, any $\boldsymbol{v} \in \mathrm{Saturation}(\boldsymbol{A})$.

For the other direction, it is enough to prove that each row of a given left kernel basis $\boldsymbol{B} \in \mathsf{F}[x]^{r\times n}$ of $\boldsymbol{K}$ is in $\mathrm{Saturation}(\boldsymbol{A})$. Let $\hat{\boldsymbol{A}} \in \mathsf{F}[x]^{r\times n}$ be a set of r linearly independent rows of $\boldsymbol{A}$; since these rows are in the left kernel of $\boldsymbol{K}$, we have $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{B}$ for some nonsingular $\boldsymbol{U} \in \mathsf{F}[x]^{r\times r}$. Thus each row of $\boldsymbol{B} = \boldsymbol{U}^{-1}\hat{\boldsymbol{A}}$ is an $\mathsf{F}(x)$-linear combination of rows of $\boldsymbol{A}$. □

Combining this with (Zhou and Labahn, 2013, Lemma 3.3), it follows that for any saturation basis $\boldsymbol{B} \in \mathsf{F}[x]^{r\times n}$ of $\boldsymbol{A}$ and any factorization $\boldsymbol{A} = \boldsymbol{C}\boldsymbol{B}$ with $\boldsymbol{C} \in \mathsf{F}[x]^{m\times r}$, then $\boldsymbol{C}$ is a column basis of $\boldsymbol{A}$. If $\boldsymbol{A}$ has full row rank we obtain that $\boldsymbol{C}$ is nonsingular, and that $\boldsymbol{A}$ is saturated if and only if $\boldsymbol{C}$ is unimodular, or equivalently $\mathrm{ColSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathsf{F}[x]^{m\times 1}$. For the sake of completeness, we now present a concise proof of this characterization (lemma 3.5.5); we will need the following standard result which essentially says that any kernel basis is saturated (see for example (Giorgi and Neiger, 2018, Lemma 2.2) for a proof).

Fact 3.5.4. *Let $\boldsymbol{K} \in \mathsf{F}[x]^{n\times\ell}$. For any left kernel basis $\boldsymbol{B} \in \mathsf{F}[x]^{r\times n}$ of $\boldsymbol{K}$, we have $\mathrm{ColSp}_{\mathsf{F}[x]}(\boldsymbol{B}) = \mathsf{F}[x]^{r\times 1}$.*

Lemma 3.5.5. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ have full row rank. Then, $\boldsymbol{A}$ is saturated if and only if $\mathrm{ColSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathsf{F}[x]^{m\times 1}$.*

Proof. If $\boldsymbol{A}$ is saturated, it is a basis of its own saturation since it has full row rank. Then writing $\boldsymbol{K}$ for a right kernel basis of $\boldsymbol{A}$, by lemma 3.5.3, $\boldsymbol{A}$ is a left kernel basis of $\boldsymbol{K}$. Then fact 3.5.4 gives $\mathrm{ColSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathsf{F}[x]^{m\times 1}$.

Conversely, assume $\mathrm{ColSp}_{\mathsf{F}[x]}(\boldsymbol{A}) = \mathsf{F}[x]^{m\times 1}$. Since the row space of $\boldsymbol{A}$ is a submodule of its saturation, we have $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{B}$ where $\boldsymbol{B} \in \mathsf{F}[x]^{m\times n}$ is a saturation basis of $\boldsymbol{A}$ and $\boldsymbol{U} \in \mathsf{F}[x]^{m\times m}$ is nonsingular. By assumption, we have $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{I}_m$ for some $\boldsymbol{V} \in \mathsf{F}[x]^{n\times m}$, hence $\boldsymbol{U}(\boldsymbol{B}\boldsymbol{V}) = \boldsymbol{I}_m$. Because these are all polynomial matrices, this means that $\boldsymbol{U}$ is unimodular, and $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{B}$ implies that $\boldsymbol{A}$ is saturated. □

We are now ready to state Protocol 3.17 for the certification that a matrix is saturated, assuming it has either full row rank or full column rank. The latter restriction is satisfied in all the applications we have in mind, including the two we present below (section 3.5.2): unimodular completability and kernel basis certification. We note that, if one accepts a communication cost similar to the size of the public matrix $\boldsymbol{A}$, then removing this assumption is easily done by making use of a row basis of $\boldsymbol{A}$.

Protocol 3.17: Saturated

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$
Certifies: $\boldsymbol{A}$ is saturated and $\boldsymbol{A}$ has full rank

	Prover	**Verifier**

1.	**if** $m \leqslant n$: // full row rank
	  RowSpaceSubset$(\boldsymbol{I}_m, \boldsymbol{A}^\intercal)$
	else: // full column rank
	  RowSpaceSubset$(\boldsymbol{I}_n, \boldsymbol{A})$	

Theorem 3.5.6. *Let $d = \max(1, \deg(\boldsymbol{A}))$, $\mu = \max(m, n)$, and $\nu = \min(m, n)$. Protocol 3.17 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant 2\mu d$ in its subprotocol. It requires $O(\mu d \log \mu)$ communication and has Verifier cost $O(mnd \log \mu)$. Assuming that $\boldsymbol{A}$ has full rank and is saturated, there is a Las Vegas randomized algorithm for the Prover with expected cost $\widetilde{O}(\mu\nu^{\omega-1}d)$, and otherwise the probability that the Verifier incorrectly accepts is at most $(4\nu d + 2)/\#S$.*

Proof. This directly follows from lemmas 3.5.2 and 3.5.5 and theorem 3.4.2. Remark that in both cases $m \leqslant n$ and $m > n$, the protocol RowSpaceSubset is applied with public matrices $\boldsymbol{I}_\nu$ and a $\mu \times \nu$ matrix of rank at most ν and degree at most d. □

Concerning the certification of a saturation basis of $\boldsymbol{A}$, our protocol will rely on the following characterization.

Lemma 3.5.7. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$. Then, a matrix $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$ is a saturation basis of $\boldsymbol{A}$ if and only if the following conditions are satisfied:*

(i) $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A}) \subseteq \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B})$,

(ii) $\mathrm{rank}(\boldsymbol{B}) = \ell$ *and* $\mathrm{rank}(\boldsymbol{A}) \geqslant \ell$,

(iii) $\boldsymbol{B}$ *is saturated.*

Proof. If $\boldsymbol{B}$ is a saturation basis of $\boldsymbol{A}$, then by definition $\boldsymbol{B}$ is saturated; $\boldsymbol{B}$ has full row rank with $\ell = \mathrm{rank}(\boldsymbol{B}) = \mathrm{rank}(\boldsymbol{A})$; and $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B}) = \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A})$.

Conversely, assume that the three items hold. The first two items together imply $\ell = \mathrm{rank}(\boldsymbol{B}) = \mathrm{rank}(\boldsymbol{A})$, and hence $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A}) = \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B})$. This means $\mathrm{Saturation}(\boldsymbol{A}) = \mathrm{Saturation}(\boldsymbol{B})$, and the latter saturation is equal to $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$ since $\boldsymbol{B}$ is saturated by the third item. Hence $\boldsymbol{B}$ has full row rank and $\mathsf{F}[x]$-row space equal to the saturation of $\boldsymbol{A}$. □

Protocol 3.18: SaturationBasis

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times n}$
Certifies: $\boldsymbol{B}$ is a saturation basis of $\boldsymbol{A}$

	Prover		Verifier
1.			Check $\ell \overset{?}{\leqslant} \min(m,n)$
2.		RankLowerBound$(\boldsymbol{A}, \ell)$	
3.			$\alpha \xleftarrow{\$} \mathsf{S}$ $\boldsymbol{\lambda} \xleftarrow{\$} \mathsf{S}^{1\times n}$ $\boldsymbol{v} \leftarrow \boldsymbol{\lambda}\boldsymbol{A}(\alpha) \in \mathsf{F}^{1\times n}$
		$\xleftarrow{\alpha, \boldsymbol{v}}$	
4.	Find $\boldsymbol{\gamma} \in \mathsf{F}^{1\times n}$ such that $\boldsymbol{\gamma}\boldsymbol{B}(\alpha) = \boldsymbol{v}$.		
		$\xrightarrow{\boldsymbol{\gamma}}$	
5.			$\boldsymbol{\gamma}\boldsymbol{B}(\alpha) \overset{?}{=} \boldsymbol{v}$
6.		Saturated$(\boldsymbol{B})$	

Theorem 3.5.8. *Protocol 3.18 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant \max(\ell d_{\boldsymbol{A}}+1, 2nd_{\boldsymbol{B}})$ in its subprotocols. It requires $O(nd_{\boldsymbol{B}}\log(n))$*

communication and has Verifier cost $O(mnd_{\boldsymbol{A}} + \ell n d_{\boldsymbol{B}} \log(n))$. *If* $\boldsymbol{B}$ *is a saturation basis of* $\boldsymbol{A}$*, then there is a Las Vegas randomized algorithm for the Prover with expected cost*

$$\widetilde{O}\big(mn\ell^{\omega-2} + mnd_{\boldsymbol{A}} + n\ell^{\omega-1}d_{\boldsymbol{B}}\big);$$

otherwise the probability that the Verifier incorrectly accepts is at most $(4\ell d_{\boldsymbol{B}} + 2)/\#S$.

Proof. The check on Step 1 has no arithmetic cost, but ensures that ℓ is less than or equal to m and n. Steps 3 to 5 certify that $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A}) \subseteq \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B})$. Note that this only communicates $O(n)$ field elements, and that the Verifier's cost at these steps amounts to the evaluation of $\boldsymbol{A}$ and $\boldsymbol{B}$ at α as well as two scalar vector-matrix products, hence a total of $O(mnd_{\boldsymbol{A}} + \ell n d_{\boldsymbol{B}})$ operations. Then the Verifier and Prover's costs follow from theorems 3.2.3 and 3.5.6.

Note that $\mathrm{rank}(\boldsymbol{A}) \geqslant \ell$ and $\mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{A}) \subseteq \mathrm{RowSp}_{\mathsf{F}[x]}(\boldsymbol{B})$ imply that $\mathrm{rank}(\boldsymbol{B}) = \ell$, so that the precondition for the Saturated protocol on Step 6 is valid unless one of the previous checks failed.

For the probability bound, we consider each of the checks in Steps 2, 5 and 6. By theorem 3.2.3, the Verifier incorrectly accepts an $\boldsymbol{A}$ with rank $< \ell$ with probability at most $1/\#\mathsf{S}$. By theorem 3.5.6, the Verifier incorrectly accepts an unsaturated $\boldsymbol{B}$ with probability at most $(4\ell d_{\boldsymbol{B}} + 2)/\#\mathsf{S}$.

For Step 5, assume now that $\mathrm{rank}(\boldsymbol{A}) \geqslant \ell$ and $\boldsymbol{B}$ is saturated, and in particular $\mathrm{rank}(\boldsymbol{B}) = \ell$, but that $\mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{A}) \not\subseteq \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B})$. By lemma 3.4.1, then for a random $\boldsymbol{\lambda} \in \mathsf{S}^{1\times m}$ we have $\boldsymbol{\lambda A} \in \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B})$ with probability at most $\frac{1}{\#\mathsf{S}}$. Consider now that $\boldsymbol{\lambda A} \notin \mathrm{RowSp}_{\mathsf{F}(x)}(\boldsymbol{B})$. Let $\boldsymbol{P}$ be an $n \times n$ permutation matrix such that $\boldsymbol{BP} = [\boldsymbol{B}_0 \mid \boldsymbol{B}_1]$ with $\boldsymbol{B}_0 \in \mathsf{F}[x]^{\ell\times\ell}$ and full rank. Let $\boldsymbol{u} \in \mathsf{F}(x)^{1\times\ell}$ be the unique vector such that $\boldsymbol{\lambda A}P = [\boldsymbol{u}\boldsymbol{B}_0 \mid \hat{\boldsymbol{a}}]$, for some $\hat{\boldsymbol{a}} \in \mathsf{F}(x)^{1\times(n-\ell)}$. Then there is some index i of $\hat{\boldsymbol{a}}$ that differs from index i of $\boldsymbol{u}\boldsymbol{B}_1$. By Cramer's rule the entries of $\boldsymbol{u}$ can be written with common denominator $\det(B)$ and numerators of degree at most $(\ell-1)d_{\boldsymbol{B}}$. Hence the entries of $\boldsymbol{u}\boldsymbol{B}_1$ have denominators and numerators of degree at most $\ell d_{\boldsymbol{B}}$ each. Hence $\boldsymbol{\lambda A}(\alpha)$ and $\boldsymbol{u}(\alpha)\boldsymbol{B}(\alpha)$ agree at index i for at most $2\ell d_{\boldsymbol{B}}$ choices of $\alpha \in \mathsf{F}$, and we conclude that the Verifier incorrectly accepts in this case with probability at most $2\ell d_{\boldsymbol{B}}/\#\mathsf{S}$.

Summing up, the worst case is the check at Step 6, where the Verifier incorrectly accepts with probability at most $(4\ell d_{\boldsymbol{B}} + 2)/\#\mathsf{S}$. □

3.5.2 Kernel bases and unimodular completability

Here, we derive two protocols which follow from the ones concerning the saturation. The second protocol is for the certification of kernel bases, while the first protocol is about matrices that can be completed into unimodular matrices.

The fast computation of such completions was studied by Zhou and Labahn (2014). We say that $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$ is *unimodular completable* if $m < n$ and there exists a matrix $\boldsymbol{B} \in \mathsf{F}[x]^{(n-m)\times n}$ such that $\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix}$ is unimodular. Note that if $\boldsymbol{A}$ does not have full

row rank, then it is not unimodular completable. Otherwise, Zhou and Labahn (2014, Lemma 2.10) showed that $\boldsymbol{A}$ is unimodular completable if and only if $\boldsymbol{A}$ has unimodular column bases; by lemma 3.5.5, this holds if and only if $\boldsymbol{A}$ is saturated. This readily leads us to Protocol 3.19.

Protocol 3.19: UnimodularCompletable

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$
Certifies: $\boldsymbol{A}$ is unimodular completable

	Prover		Verifier
1.			$m \overset{?}{<} n$
2.		Saturated($\boldsymbol{A}$)	

Theorem 3.5.9. *Protocol 3.19 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant 2md$ in its subprotocols. It requires $O(nd\log(n))$ communication and has Verifier cost $O(mnd\log(n))$. If $\boldsymbol{A}$ is unimodular completable, then there is a Las Vegas randomized algorithm for the Prover with expected cost $\widetilde{O}(nm^{\omega-1}d)$; otherwise the probability that the Verifier incorrectly accepts is at most $(4md+2)/\#S$.*

Proof. The costs follow from theorems 3.2.3 and 3.5.6, noting that the protocol aborts early if $m \geqslant n$, and therefore m is an upper bound on the rank in the Saturated subprotocol. The probability of the Verifier incorrectly accepting here is the same as in Saturated from theorem 3.5.6. □

Finally, Protocol 3.20 for the certification of kernel bases will follow from the characterization in the next lemma.

Lemma 3.5.10. *Let $\boldsymbol{A} \in \mathsf{F}[x]^{m \times n}$ and let $\boldsymbol{B} \in \mathsf{F}[x]^{\ell \times m}$. Then, $\boldsymbol{B}$ is a left kernel basis of $\boldsymbol{A}$ if and only if*

(i) $\operatorname{rank}(\boldsymbol{B}) = \ell$ *and* $\operatorname{rank}(\boldsymbol{A}) \geqslant m - \ell$,

(ii) $\boldsymbol{B}\boldsymbol{A} = \boldsymbol{0}$,

(iii) $\boldsymbol{B}$ *is saturated.*

Proof. If $\boldsymbol{B}$ is a left kernel basis of $\boldsymbol{A}$, then we have $\operatorname{rank}(\boldsymbol{B}) = \ell = m - \operatorname{rank}(\boldsymbol{A})$ as well as $\boldsymbol{B}\boldsymbol{A} = \boldsymbol{0}$; the third item follows from fact 3.5.4 and lemma 3.5.5.

Now assume that the three items hold. Consider some left kernel basis $\boldsymbol{K}$ of $\boldsymbol{A}$. Then, $\operatorname{rank}(\boldsymbol{K}) = m - \operatorname{rank}(\boldsymbol{A}) \leqslant \ell$ by the first item, while the second item implies that the row space of $\boldsymbol{B}$ is contained in the row space of $\boldsymbol{K}$, hence $\ell = \operatorname{rank}(\boldsymbol{B}) \leqslant \operatorname{rank}(\boldsymbol{K})$; therefore $\operatorname{rank}(\boldsymbol{K}) = \ell$. As a result, $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{K}$ for some nonsingular $\boldsymbol{U} \in \mathsf{F}[x]^{\ell \times \ell}$. Item

(iii) implies $\mathrm{ColSp}_{\mathsf{F}[x]}(\boldsymbol{B}) = \mathsf{F}[x]^{\ell\times 1}$ according to lemma 3.5.5, hence $\boldsymbol{I}_\ell = \boldsymbol{BV} = \boldsymbol{UKV}$ for some $\boldsymbol{V} \in \mathsf{F}[x]^{m\times\ell}$. Then, $\boldsymbol{U}$ must be unimodular, and thus $\boldsymbol{B} = \boldsymbol{UK}$ is a left kernel basis of $\boldsymbol{A}$. □

Protocol 3.20: KernelBasis

Public: $\boldsymbol{A} \in \mathsf{F}[x]^{m\times n}$, $\boldsymbol{B} \in \mathsf{F}[x]^{\ell\times m}$
Certifies: $\boldsymbol{B}$ is a left kernel basis of $\boldsymbol{A}$

	Prover		Verifier
1.			$\ell \overset{?}{\leqslant} m$
2.		RankLowerBound$(\boldsymbol{A}, m-\ell)$	
3.		MatMul$(\boldsymbol{AB}, \boldsymbol{0})$	
4.		Saturated$(\boldsymbol{B})$	

Theorem 3.5.11. *Protocol 3.20 is a probabilistically sound interactive protocol and is complete assuming $\#S \geqslant \max((m-\ell)d_{\boldsymbol{A}} + 1, 2md_{\boldsymbol{B}})$ in its subprotocols. It requires $O(md_{\boldsymbol{B}}\log(m))$ communication and has Verifier cost*

$$O(\ell md_{\boldsymbol{B}}\log(m) + mnd_{\boldsymbol{A}}).$$

If $\boldsymbol{B}$ is a left kernel basis of $\boldsymbol{A}$, then there is a Las Vegas randomized algorithm for the Prover with expected cost

$$\widetilde{O}\big(m\ell^{\omega-1}d_{\boldsymbol{B}} + mn(m-\ell)^{\omega-2}d_{\boldsymbol{A}}\big);$$

otherwise the probability that the Verifier incorrectly accepts is at most

$$\frac{\max(d_{\boldsymbol{A}} + d_{\boldsymbol{B}} + 1, 4\ell d_{\boldsymbol{B}} + 2)}{\#S}.$$

Proof. The costs follow from lemma 3.5.10 and theorems 3.2.3, 3.2.8 and 3.5.6. As before, the worst case for the Verifier is that only one of the three checked statements is wrong, and the resulting maximum of probabilities comes either from Step 2 or Step 4. □

Conclusion and perspectives

We have developed interactive protocols verifying a variety of problems concerning polynomial matrices. For rank, determinant, system solving, and matrix multiplication

(section 3.2), these amount to evaluating at some random point(s) and reducing to field-based verifications. For row bases, saturation, normal forms, and kernel basis computations (sections 3.4 and 3.5), the verifications essentially reduce to testing row space membership of a single vector (section 3.3) and testing that ranks are the expected ones.

Our protocols are efficient. The volume of data exchanged in communications is roughly the size of a single row of the matrix. The time complexity for the Verifier is linear (or nearly-linear) in the size of the object being checked, and the time for the Prover is roughly the same as it would take to perform the computation being verified.

Still, there is some room for improvement in these costs. It would be nice to remove the logarithmic factors in the complexities of most later protocols for the Verifier time and communication cost; these come from the number of repetitions t required in the RowSpaceMembership protocol.

Our protocols also require to work over sufficiently large fields, to ensure soundness of the randomized verification. For smaller fields, a classic workaround is to resort to a field extension, increasing the arithmetic and communication cost by a logarithmic factor. An alternative is to increase the dimension in the challenges and responses, e.g. verifying a block of vectors instead of a single vector of field elements. A further study on whether this approach is applicable and competitive here is required.

Another possibility for improvement in our complexities would be to have the same costs where d is the *average* matrix-vector degree, rather than the maximum degree. Such complexity refinements have appeared for related computational algorithms, frequently by "partial linearization" of the rows or columns with highest degree (Gupta et al., 2012, Section 6), and it would be interesting to see if similar techniques could work here. This would be especially helpful in more efficiently verifying an unbalanced shifted Popov form, and the Hermite form in particular, of a nonsingular matrix.

The protocols presented here do not assume that the Prover has computed the result to be verified. This is however likely to be the case in many instances of verified computing, and it would then be relevant to identify which intermediate results in a Prover's computation of the solution (such as the rank profile matrix, the weak Popov form, etc), could be reused in a certificate for verifying this solution. Though more constraining on the Prover's choice of an algorithm, such information would help reducing the leading constant in the arithmetic cost of its computation.

While we have presented protocols for a variety of basic problems on polynomial matrices, there are still more for which we do not know yet whether any efficient verification exists. These include:

- high-order terms in expansion of the inverse (see the high-order lifting algorithm of Storjohann (2003));
- univariate relations, generalizing Hermite-Padé approximation (Beckermann and Labahn, 2000; Neiger and Vu, 2017);
- Smith form (see (Storjohann, 2003) for the fastest known algorithm).

We also do not know in all cases how to prove the *negation* of our statements — for

example, that a vector is *not* in the row space of a polynomial matrix. It seems that some similar techniques to those we have used may work, but we have not investigated the question deeply.

Perhaps the most interesting direction for future work would be to adapt our protocols to the case of Euclidean lattices, i.e., integer matrices and vectors. It seems that most of our protocols in section 3.2 should translate when we replace evaluation at a point α with reduction modulo a sufficiently-large prime p, but the analysis in terms of bit complexity rather than field operations will likely be more delicate. Another seeming hurdle is in our central protocols in section 3.3 on deciding row membership: while the general ideas of these protocols *might* translate to integer lattices, the proof techniques we have used are particular for polynomials.

Chapter 4

Secure multiparty matrix multiplication based on Strassen-Winograd algorithm

Contents

Technical summary and overview of this chapter

Model We consider the product of square $N \times N$ matrices, further denoted as A and B. Our protocol supports up to N players, and each player knows one row (or one block of consecutive rows) of A, the corresponding row (or block of rows) of B and learns the corresponding row (or block of rows) of $C = A \times B$. Our protocol is proven secure against non-colluding, semi-honest adversaries as defined in Section 1.1.3. We require communication between players to be performed over secure channels, meaning that, for any communication between two players, only these two players will know the piece of information that were communicated. The main tool used to secure our protocol is semi-homomorphic cryptography (such as defined in Section 1.2.4) supporting homomorphic additions. We focus on improving the communication volume of multiparty matrix product. Theoretical communication costs given in this chapter are based on the number of field elements exchanged between players, while practical communication costs are a direct measure of the total communication volume in (Giga)Bytes.

State of the art – main competition We primarily compare our work to the protocol of Dumas et al. (2017a), which is the state of the art for multiparty matrix multiplication with the same data repartition model. Their protocol, called `YTP-SS`, is based on textbook matrix multiplication and consists of sequential secure multiparty dot products. For input matrices of size N, they achieve a communication cost of $O(N^3)$ field elements. Their protocol, however, has stronger security than ours, as it supports colluding malicious adversaries. They manage to achieve such security by splitting input elements in k shares for each dot product performed, and repeating each dot product k times. As this security measure requires many exchanges between players, we will propose a relaxed version of `YTP-SS`, stripped of these additional security measures. This simplified protocol, which we call `MP-PDP`, has the same security than ours and will thus be a fairer comparison than `YTP-SS`.

Results The main contribution of this chapter is a new recursive multiparty matrix multiplication protocol based on Strassen-Winograd algorithm. By taking full advantage of the recursive structure of the algorithm, we manage to achieve a better communication cost than the state of the art, $O(N^{2.81})$ for input matrices of size N. We show that, with this improvement, our protocol has cheaper communication cost than `MP-PDP` for matrices of size $N = 96$ and larger. We implemented our solution and practical experiments confirm that our protocol has less communications than `MP-PDP`. To achieve this result, we designed a data layout which, associated with the new sub-protocols we propose, ensures that no information is leaked during the execution. Furthermore, we also propose a variant of our protocol based on proxy re-encryption techniques that allows us to improve the communication cost of our solution even more. Finally, we compare estimated runtimes of our protocol and a fully homomorphic version of Strassen-Winograd algorithm based on a simplified setting

and we show that our proposition compares favourably to the former is this model.

The article associated with this chapter is (Dumas et al., 2019a).

Outline

We start by presenting Strassen-Winograd algorithm and the competitor YTP-SS protocols in in Section 4.1. There, we also define the dedicated data layout and the cryptographic tools we will use. Next, in Section 4.2, we first describe our building block protocols, with their security analysis. Second, we present in this Section a new cubic-time matrix multiplication algorithm on ciphered entries to be used as a base case. Section 4.3 describes the complete novel sub-cubic MPC Strassen-Winograd algorithm and details its theoretical communication cost. In Section 4.4, we propose practical comparisons between our C++ and competitor implementations. Finally, in Section 4.5, we give an improved variant of our protocol using proxy re-encryption techniques and compare estimated runtimes of our protocols and fully-homomorphic cryptography-based ones.

Introduction

Secure multiparty computations (MPC) allows n players to compute together the output of some function, using private inputs without revealing them. This is useful, e.g., for a distributed evaluation of trust, as defined in (Jøsang, 2007; Dumas and Hossayni, 2013). In this context, players compute a confidence level by combining their mutual degrees of trust. This aggregation of trust amongst players can be represented as a matrix product $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$, where each player knows one row of the matrix containing their partial trust towards their neighbours and the network has to compute a distributed matrix exponentiation, which reduces to several matrix multiplications. In this chapter, we thus focus on this particular layout of data, and on multiparty matrix multiplication of dimension $N \times N$ with N players.

As was described in Section 1.2, several tools exist to design MPC protocols, like Shamir's secret sharing scheme (Shamir, 1979), homomorphic encryption (Goethals et al., 2005), oblivious transfer (Dagdelen and Venturi, 2015) or using a Trusted Third Party (Du and Zhan, 2002). Then, several MPC implementations are available[1]. Some of them are for two parties only and most of the others are generic and transform programs into circuits or use oblivious transfer (Demmler et al., 2015; Rindal and Rosulek, 2016; Damgård et al., 2017; Jarecki, 2018; Mishra et al., 2018/663). For instance the symmetric system solving phase of the LINREG-MPC software is reported in (Gascón et al., 2017) to take about 45 minutes for $n = 200$, while, in (Dumas et al., 2017a), a secure multiparty specific algorithm, YTP-SS, developed for matrix multiplication, requires about a hundred seconds to perform an $n = 200$ matrix multiplication. These timings, however, do not take into account communications, but for multiparty matrix multiplication, the number of communications and the number of operations

[1] https://github.com/rdragos/awesome-mpc

should be within the same order of magnitude. Our goal is thus to improve on existing algorithms, primarily in terms of this number of communications (we do not minimize the number of messages, as in (Ishai et al., 2018), but instead consider the overall volume). Our idea is to use an algorithm with a lower time and communication complexity for matrix multiplication. Strassen's algorithm (Strassen, 1969) was the first sub-cubic time algorithm, with an exponent $\log_2 7 \approx 2.81$ leading to a complexity of $O(n^{2.81})$. As Strassen is one of the simplest matrix multiplication algorithm, and one of the only practically efficient ones, we construct an MPC protocol based Winograd's variant of this algorithm which carries the aforementioned complexity over the communication volume.[2] (Aho et al., 1974, Ex.6.5).

We need to preserve input privacy throughout the computation, which can be achieved by the use of homomorphic cryptography. Such ciphers allow to perform arithmetic operations over ciphertexts, which will help to maintain privacy in our setting. There exists two major families of homomorphic cryptosystems, fully-homomorphic and semi-homomorphic ones. While the former are usually considered more expensive than the latter, their ability to use all arithmetic operations over ciphertexts make them far less constrained than the others (see Section 1.2.4 for a few more details and references on the topic). Table 4.1 shows a comparison of basic cryptographic (encryption, decryption) and arithmetic (addition, multiplication) operations between semi-homomorphic cryptosystems and two fully homomorphic libraries. Naccache-Stern and Paillier cryptosystems are our own C++ implementations, based on Givaro (Givaro group, 2020), SEAL is the Microsoft SEAL fully homomorphic encryption library (Microsoft Research, 2019) based on Brakerski/Fan-Vercauteren scheme (Fan and Vercauteren, 2012; Brakerski, 2012) and HElib is the IBM Helib fully homomorphic encryption library (hel, 2019), based on Brakerski-Gentry-Vaikuntanathan scheme (Brakerski et al., 2012). Timings were measured on a workstation with an Intel i5-7300U @2.60 GHz, 16GB of RAM, with parameters set to have a security level of $\lambda = 130$. Accordingly to recommendations given in the libraries documentation, SEAL timings were made for a plaintext space of 20 bits and HElib timings for a plaintext space of 64 bits. While a few fully homomorphic operations are faster (SEAL decryption) or competitive (SEAL encryption vs. Paillier) with semi-homomorphic cryptosystems, the former are still faster in most situations.

We will hence use partial homomorphic encryption scheme (Cramer et al., 2015) as they allow to perform the operations we need, namely:

1. $D_{sk}(E_{pk}(m_1) \times E_{pk}(m_2)) = m_1 + m_2$ (*Additive homomorphism*)
2. $D_{sk}(E_{pk}(m_1)^{m_2}) = m_1 \times m_2$ (*Cipher/clear multiplicative homomorphism*)

Several cryptosystems do satisfy these, e.g., the ones designed by Naccache-Stern or Paillier (Naccache and Stern, 1998; Paillier, 1999). The former is usually costlier than the latter. However, as the former allow parties to agree on a common message block size, which solves the issue of defining a consistent message space among them, we

[2]The best value known to date, due to (Le Gall, 2014), of approximately 2.3728639. However, only a few sub-cubic time algorithms are competitive in practice and used in software (Dumas et al., 2008; Boyer and Dumas, 2016; Kaporin, 1999) (see also (Karstadt and Schwartz) and references therein), among which Strassen's algorithm and its variants stand out as a very effective one in practice.

Operation	Message size	Naccache-Stern	Paillier	HElib	SEAL
Encryption	20	0.07	19.3	–	19.7
	64	0.14	20.1	94.6	–
Decryption	20	15.1	19.2	–	7.2
	64	61.6	19.4	46.5	–
Addition	20	0.001	0.01	–	0.3
	64	0.001	0.01	6.2	–
Multiplication	20	0.02	0.17	–	78.8
	64	0.05	0.50	29	–

Table 4.1: Timings in ms for basic operations on libraries and cryptosystems at security level $\lambda = 130$

choose here to use the Naccache-Stern cryptosystem.

Finally, Strassen-Winograd algorithm involves numerous additions and subtractions on parts of the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices that are held by different players. Security concerns require then that these entries should be encrypted from the start, contrarily to (Dumas et al., 2017a). As a consequence, the classical matrix multiplication can no longer be used as stated in the latter reference, even for the base case of the recursive algorithm. We therefore propose an alternative base case. Its arithmetic cost is higher, but it involves an equivalent amount of communication. We shall show that this choice combined with our multiparty recursive Strassen-Winograd algorithm compares favourably to existing implementations in communication cost for matrices of dimensions larger than $N = 96$.

As Strassen-Winograd algorithm trades multiplications for additions, and as homomorphic additions are cheaper than multiplications, this algorithm is a very good candidate for a multiparty protocol using homomorphic techniques.

Hypotheses. In this chapter, we will only consider the case of *semi-honest* (also called *honest-but-curious*) adversaries. As a reminder, such adversaries, represented as probabilistic polynomial time machines, try to gather as many information as possible during the execution of the protocol, and can locally run any computation based on this information in order to deduce some private input. However, they strictly follow protocol specifications. We also consider that *communications are performed over secure channels*: this means transferred data is resistant to eavesdropping and that only the recipient will learn anything from communicated data.

4.1 Preliminaries

4.1.1 Strassen-Winograd algorithm

$\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ by splitting the input matrices in four quadrants of equal dimensions: $\boldsymbol{A} = \left[\begin{smallmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{smallmatrix}\right]$ and $\boldsymbol{B} = \left[\begin{smallmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{smallmatrix}\right]$. Each recursive call consists in 22 block operations:

- 8 additions:
$$\begin{array}{llll} \boldsymbol{S}_1 \leftarrow \boldsymbol{A}_{21} + \boldsymbol{A}_{22} & \boldsymbol{S}_2 \leftarrow \boldsymbol{S}_1 - \boldsymbol{A}_{11} & \boldsymbol{S}_3 \leftarrow \boldsymbol{A}_{11} - \boldsymbol{A}_{21} & \boldsymbol{S}_4 \leftarrow \boldsymbol{A}_{12} - \boldsymbol{S}_2 \\ \boldsymbol{T}_1 \leftarrow \boldsymbol{B}_{12} - \boldsymbol{B}_{11} & \boldsymbol{T}_2 \leftarrow \boldsymbol{B}_{22} - \boldsymbol{T}_1 & \boldsymbol{T}_3 \leftarrow \boldsymbol{B}_{22} - \boldsymbol{B}_{12} & \boldsymbol{T}_4 \leftarrow \boldsymbol{T}_2 - \boldsymbol{B}_{21} \end{array}$$
- 7 recursive multiplications:
$$\begin{array}{llll} \boldsymbol{R}_1 \leftarrow \boldsymbol{A}_{11} \times \boldsymbol{B}_{11} & \boldsymbol{R}_2 \leftarrow \boldsymbol{A}_{12} \times \boldsymbol{B}_{21} & \boldsymbol{R}_3 \leftarrow \boldsymbol{S}_4 \times \boldsymbol{B}_{22} & \boldsymbol{R}_4 \leftarrow \boldsymbol{A}_{22} \times \boldsymbol{T}_4 \\ \boldsymbol{R}_5 \leftarrow \boldsymbol{S}_1 \times \boldsymbol{T}_1 & \boldsymbol{R}_6 \leftarrow \boldsymbol{S}_2 \times \boldsymbol{T}_2 & \boldsymbol{R}_7 \leftarrow \boldsymbol{S}_3 \times \boldsymbol{T}_3 & \end{array}$$
- 7 final additions:
$$\begin{array}{llll} \boldsymbol{U}_1 \leftarrow \boldsymbol{R}_1 + \boldsymbol{R}_2 & \boldsymbol{U}_2 \leftarrow \boldsymbol{R}_1 + \boldsymbol{R}_6 & \boldsymbol{U}_3 \leftarrow \boldsymbol{U}_2 + \boldsymbol{R}_7 & \boldsymbol{U}_4 \leftarrow \boldsymbol{U}_2 + \boldsymbol{R}_5 \\ \boldsymbol{U}_5 \leftarrow \boldsymbol{U}_4 + \boldsymbol{R}_3 & \boldsymbol{U}_6 \leftarrow \boldsymbol{U}_3 - \boldsymbol{R}_4 & \boldsymbol{U}_7 \leftarrow \boldsymbol{U}_3 + \boldsymbol{R}_5 & \end{array}$$
- The result is the matrix: $\boldsymbol{C} = \left[\begin{smallmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_5 \\ \boldsymbol{U}_6 & \boldsymbol{U}_7 \end{smallmatrix}\right]$.

Although the recursion could be run down to products of 1×1 matrices, it is commonly stopped at a fixed dimension threshold, where a classical cubic time algorithm is then used, in order to reduce the overhead of recursion on small dimension instances. For the sake of simplicity, we consider henceforth that the initial input matrices are of dimension $N \times N$, with $N = b2^{\ell}$, so that up to ℓ recursive calls can be made without having to deal with padding with zeroes nor with peeling thin rows or columns.

4.1.2 Data layout and encryption

We consider the setting where the two input matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ have dimension $N \times N$ and each of the N players stores one row of $\boldsymbol{A}$ and the corresponding row of $\boldsymbol{B}$ and learns the corresponding row of $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$. In this setting, the YTP-SS Algorithm (Dumas et al., 2017a, Algorithm 15) can compute $\boldsymbol{C}$ by encrypting the rows of $\boldsymbol{A}$ only and then relying on homomorphic multiplications of encrypted coefficients of $\boldsymbol{A}$ by plain coefficients of $\boldsymbol{B}$.

However, Strassen's algorithm, considered here, requires adding and subtracting submatrices of $\boldsymbol{B}$ of distinct row index sets (e.g. $\boldsymbol{T}_3 \leftarrow \boldsymbol{B}_{22} - \boldsymbol{B}_{12}$). These operations on non-ciphered rows of $\boldsymbol{B}$ would automatically leak information. We therefore impose that the rows of both operands $\boldsymbol{A}$ and $\boldsymbol{B}$, of the result $\boldsymbol{C}$ and of any intermediate matrix are encrypted by the public key of a player who is not the one hosting the row. We therefore introduce the notion of location and key sequences for a matrix, to identify the roles of the players in this data layout:

Definition 4.1.1. *An $n \times n$ matrix $\boldsymbol{A}$ of* ciphered values *has location sequence $L = (l_1, l_2, \ldots, l_n)$ and key sequence $K = (k_1, k_2, \ldots, k_n)$ if player P_{l_i} stores row i of $\boldsymbol{A}$, that was encrypted with the public key pk_{k_i} of player P_{k_i} for all $1 \leqslant i \leqslant n$.*

Example 1. *For $n = 3$, consider the location sequence $L = (2, 3, 1)$ and key sequence $K = (3, 1, 2)$. This means that player P_2 stores row 1 of $\boldsymbol{A}$ encrypted with the public key of player P_3; player P_3 stores row 2 of $\boldsymbol{A}$ encrypted with the public key of player P_1 and finally player P_1 stores row 3 of $\boldsymbol{A}$ encrypted with the public key of player P_2.*

In the matrix multiplication algorithms presented in the later sections, the location and key sequences of operand $\boldsymbol{A}$ and $\boldsymbol{C}$ will always be identical. On the other hand the location and key sequences of $\boldsymbol{B}$ may equal those of $\boldsymbol{A}$ (in the first recursive call), or differ, but then they must have an empty intersection with those of $\boldsymbol{A}$.

A recursive step in Strassen-Winograd algorithm splits the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ into four quadrants of equal dimensions. Hence their key and location sequences are split into two sub-sequences: for $\boldsymbol{X} \in \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$, $L_{\boldsymbol{X}} = (L_{\boldsymbol{X}_U}, L_{\boldsymbol{X}_L})$ and $K_{\boldsymbol{X}} = (K_{\boldsymbol{X}_U}, K_{\boldsymbol{X}_L})$ such that $(L_{\boldsymbol{X}_U}, K_{\boldsymbol{X}_U})$ are the location and key sequences for the upper half of $\boldsymbol{X}$ and $(L_{\boldsymbol{X}_L}, K_{\boldsymbol{X}_L})$ are the location and key sequences for the lower half of $\boldsymbol{X}$.

Figure 4.1 summarizes the notations we use on the input/output operands in Strassen-Winograd algorithm.

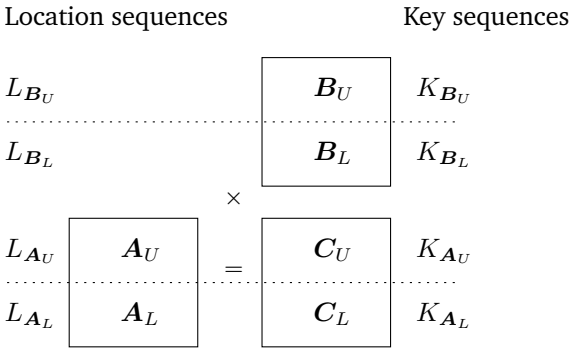

Figure 4.1: Recursive splitting of the location and key sequences of the input and output operands in Strassen-Winograd algorithm.

More formally, we present in Definition 4.1.2 the two distinct data layouts used in our algorithms: one for the recursive levels of Strassen-Winograd, and one for its base case.

Definition 4.1.2. *Let $N \in \mathbf{N}$, $n \leqslant N$ and $\boldsymbol{A}$ and $\boldsymbol{B}$ two $n \times n$ matrices with location and key sequences $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}) \in (\{1..N\}^n)^2$ and $(L_{\boldsymbol{B}}, K_{\boldsymbol{B}}) \in (\{1..N\}^n)^2$.*

1. *$(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}, L_{\boldsymbol{B}}, K_{\boldsymbol{B}})$ is a valid data layout if*
   a) *$\forall i \in \{1..n\}$, $L_{\boldsymbol{A}[i]} \neq K_{\boldsymbol{A}[i]}$ and $L_{\boldsymbol{B}[i]} \neq K_{\boldsymbol{B}[i]}$.*
   b) *$\forall i, j \in \{1..n\}$ with $i \neq j$, $L_{\boldsymbol{A}[i]} \neq L_{\boldsymbol{A}[j]}$ and $L_{\boldsymbol{B}[i]} \neq L_{\boldsymbol{B}[j]}$*
   c) *$\forall i, j \in \{1..n\}$ with $i \neq j$, $K_{\boldsymbol{A}[i]} \neq K_{\boldsymbol{A}[j]}$ and $K_{\boldsymbol{B}[i]} \neq K_{\boldsymbol{B}[j]}$*
2. *$(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}, L_{\boldsymbol{B}}, K_{\boldsymbol{B}})$ is a base case or a 0-recursive data layout if it is a valid data layout and $(L_{\boldsymbol{A}} \cup K_{\boldsymbol{A}}) \cap (L_{\boldsymbol{B}} \cup K_{\boldsymbol{B}}) = \varnothing$.*
3. *$(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}, L_{\boldsymbol{B}}, K_{\boldsymbol{B}})$ is a ℓ-recursive data layout if it is a valid data layout and*
   a) *$(L_{\boldsymbol{A}_U} \cup K_{\boldsymbol{A}_U}) \cap (L_{\boldsymbol{A}_L} \cup K_{\boldsymbol{A}_L}) = \varnothing = (L_{\boldsymbol{B}_U} \cup K_{\boldsymbol{B}_U}) \cap (L_{\boldsymbol{B}_L} \cup K_{\boldsymbol{B}_L})$*

b) $(L_{\boldsymbol{A}_U}, K_{\boldsymbol{A}_U}, L_{\boldsymbol{B}_L}, K_{\boldsymbol{B}_L})$ *and* $(L_{\boldsymbol{A}_L}, K_{\boldsymbol{A}_L}, L_{\boldsymbol{B}_U}, K_{\boldsymbol{B}_U})$ *are both* $(\ell-1)$*-recursive data layouts*

Lemma 4.1.3. *For* $N = b2^\ell$*, the following values for the location and key sequences form an* ℓ*-recursive data layout according to Definition 4.1.2:*

$$\begin{cases} k_i &= i & \text{for } 0 \leqslant i < N \\ l_{ib+j} &= ib + (j+1 \mod b) & \text{for } 0 \leqslant i < N/b, \text{and } 0 \leqslant j < b \end{cases} \tag{4.1.1}$$

For instance, for a product of dimension 12, with base case dimension $b = 3$, this gives; $L_{\boldsymbol{A}} = L_{\boldsymbol{B}} = L_{\boldsymbol{C}} = (1,2,0,4,5,3,7,8,6,11,9,10)$ and $K_{\boldsymbol{A}} = K_{\boldsymbol{B}} = K_{\boldsymbol{C}} = (0,1,2,3,4,5,6,7,8,9,10,11)$.

4.1.3 Homomorphic encryption

Naccache-Stern cryptosystem.

In the following, we use Naccache-Stern (Naccache and Stern, 1998) partially homomorphic cryptosystem, with security parameter 1^λ, set up as follows:

Setup(1^λ) : Select $2k$ small primes $p_1, \ldots, p_{2k}$; compute $u = \prod_{i=1}^{k} p_i$ and $v = \prod_{i=k+1}^{2k} p_i$; let $\sigma = u{\cdot}v$; uniformly select two large prime numbers a and b of size $\lambda/2$; find f_1 and f_2 such that $p = f_1{\cdot}a{\cdot}u + 1$ and $q = f_2{\cdot}b{\cdot}v + 1$ are primes; let $m = p{\cdot}q$ and randomly choose g of order $aubv$ in $\mathbf{Z}_m^*$. The private key is $SK = (p_1, \ldots, p_{2k}, p, q)$, the public key is $PK = (\sigma, g, m)$.

Encrypt$_{PK}(x)$: for $x \in \mathbf{Z}_\sigma$, randomly choose $r \in \mathbf{Z}_m$ and encrypt x as $c = E_{PK}(x) \equiv r^\sigma \cdot g^x \mod m$.

Decrypt$_{SK}(c)$: let $\phi = (p-1)(q-1)$, $c_i \equiv c^{\phi/p_i} \mod m$ and recover, by exhaustive search (p_i is small), $x_i \mod p_i$ such that $x_i = \log_{g^{\phi/p_i}}(c_i) \mod m$. Finally reconstruct x with the Chinese remaindering, $x \equiv CRT(\{x_i, p_i\}) \mod \sigma$.

For the rest of this chapter, cleartexts will be elements of $\mathbf{Z}_\sigma$ while elements are elements of $\mathbf{Z}_m$. As σ is shared by all players, the cleartext space is the same for all the players and we will hence denote this common message space by $\mathcal{M}$. However, there is a distinct modulus $\mathbf{Z}_m$ for each player, otherwise they would have to share their private keys. Consequently, each player has their own ciphertext space, which we will denote by $\mathcal{C}_P$. A plain text matrix then has coefficients in $\mathcal{M}$ but in a layout where each row is encrypted using a different key pk_i, its encryption is no longer a matrix but a sequence of rows over distinct rings $\mathbf{Z}_m[pk_i]$. We will abusively refer to this ciphered data as the ciphered matrix. Finally, for a key sequence K and a matrix $\boldsymbol{A}$ over $\mathcal{M}$, the ciphered matrix obtained by encrypting row i of $\boldsymbol{A}$ by $K[i]$ is denoted by $\{\boldsymbol{A}\}_K$. Row i of $\{\boldsymbol{A}\}_K$ is over $\mathcal{C}_{K[i]}$.

4.1.4 Multiparty protocols security

Here, we recall some widely used notations and results for the security of multiparty protocols.

Definition 4.1.4 ((Goldreich, 2004)). *Let f be a n-ary functionality, where $f_i(x_1, ..., x_n)$ denotes the i^{th} element of $f(x_1, ..., x_n)$. For $I = \{i_1, ..., i_t\} \subset [n] = \{1, ..., n\}$, we denote by $f_I(x_1, ..., x_m)$ the subsequence $f_{i_1}(x_1, ..., x_n), ..., f_{i_t}(x_1, ..., x_n)$. We let $x_I = (x_{i_1}, ..., x_{i_t})$. Let Π be a n-party protocol for computing f. The view of the i^{th} party during an execution of Π on $\overline{x} = (x_1, ..., x_n)$ is denoted $\mathsf{view}_i^{\Pi}(\overline{x})$, and for I, we let $\mathsf{view}_I^{\Pi}(\overline{x}) = (I, \mathsf{view}_{i_1}^{\Pi}(\overline{x}), ..., \mathsf{view}_{i_t}^{\Pi}(\overline{x}))$. We say that Π securely computes f if there exist a probabilistic polynomial time algorithm, such that, with S a simulator for Π, for every $I \subset [n]$: $\{S_I((x_I), f_I(\overline{x}))\}_{\overline{x}} \overset{C}{\equiv} \{\mathsf{view}_I^{\Pi}(\overline{x}), \mathsf{output}^{\Pi}(\overline{x})\}_{\overline{x}}$, where $\overset{C}{\equiv}$ denotes the computational indistinguishability between two ensembles.*

Definition 4.1.5. *Let $f_1, ..., f_{p(n)}$ be functionalities, and let Π be a protocol. We say that the protocol Π is executed in the $f_1, ..., f_{p(n)}$-hybrid mode if Π uses ideal calls to a trusted party to compute $f_1, ..., f_{p(n)}$.*

Theorem 4.1.6 (from (Lindell, 2017)). *Let p(n) be a polynomial, let $f_1, ..., f_{p(n)}$ be functionalities, and let $\pi_1, ..., \pi_{p(n)}$ be protocols such that each π_i securely computes f_i in the presence of semi-honest adversaries. Let g be a functionality, and let Π be a protocol that securely computes g in the $f_1, ..., f_{p(n)}$-hybrid model. Then, the protocol $\Pi^{\pi_1,...,\pi_{p(n)}}$ securely computes g in presence of semi-honest adversaries.*

We will also need a function, which, given a small input is able to securely and deterministically produce a stream of uniformly generated random values. We will achieve this by using classical **mask generation functions**, as defined in (Kaliski and Staddon, 1998, Section 10.2): a function which takes two parameters, a seed s and a length l and returns a random string of length l. We will then split the output string in as many fragments as needed, and use each of these fragments as a mask. Such function achieve an output indistinguishable property: if the seed is unknown, it is impossible to distinguish between the output of a mask generation function and a truly random string. Such secure functions exist, see for instance the one given in (Kaliski and Staddon, 1998) and in what follows, we will denote by MGF any function that have the aforementioned security properties. Finally, for the rest of this chapter, our functionalities follow the input/output specification described in the protocols they realize.

4.1.5 Relaxing an existing algorithm: YTP-SS

The matrix multiplication algorithm using the secure dot-product protocol YTP-SS (Dumas et al., 2017a, Algorithm 15) is secure against semi-honest adversaries over insecure communication channels. In order to analyse the difference with our proposition, MP-SW, we extract here the core of the former protocol, i.e., without

the securization of the channel (that is we remove the protection of the players private elements by random values, and the final communications to derandomize the results). The resulting simplification is called MP-PDP and its costs are given in Theorem 4.1.7. More details can be found in (Dumas et al., 2017a, Algorithm 15)

Theorem 4.1.7. *For n players, (Dumas et al., 2017a, Algorithm 15), without the channel securization, requires $2(n-1)$ communications. When used to compute a classical matrix product, it requires $n^3 + n(n-1)$ operations overall.*

4.2 Toolbox

4.2.1 Initialization Phase

Before the actual computation, the involved parties need to agree on the location and key sequences they will use, generate their key pairs, share their associated public keys, cipher their input data and communicate it where needed. Parties know their identifier, which is the index of the row they own, and use Equation (4.1.1) to compute the location/key sequences. SW-Setup shows how the input data is initially ciphered and dispatched: each party, identified as $P_i, i \in \{1..N\}$ starts with the i-th row of $\boldsymbol{A}$ and $\boldsymbol{B}$, and, after generating its own key pair, ciphers its row according to the key sequence.

Finally, the protocol sends the ciphered row to the party hosting this row, designated by the location sequence. For input matrices of size N, SW-Setup requires $2N^2$ communications.

4.2.2 Multiparty copy

In the various subroutines that compose our algorithm, we will often need to copy and recipher a vector from one Party to another following location and key sequences. This is done by masking and decryption, as shown in MP-Copy. For a given ciphered element x hosted by Bob and encrypted for Dan, to its new location at Alice and encrypted for Charlie. A schematic version of this protocol focusing on data exchanges is given in Figure 4.2. Here, Dan is in charge of performing the decryption and the re-encryption of the element. To prevent Dan from learning the value of x, Bob masks it additively with a random value. Bob therefore needs to clear out this random mask on the value re-encrypted by Dan, with Charlie's key, before sending it to Alice. This protocol uses a total of 3 communications.

4.2.3 Classical Matrix Multiplication base case

We describe in this section an algorithm to perform classical matrix multiplications in the data and encryption layout of Definition 4.1.2. It consists in n^2 scalar products in which, products of elements $a_{i,k}$ of $\boldsymbol{A}$ by elements $b_{k,j}$ of $\boldsymbol{B}$ are performed using

Protocol 4.1: SW-Setup

Input: Two $N \times N$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ over $\mathcal{M}$, where $N = b2^{\ell}$, such that party P_i knows the i-th row of $\boldsymbol{A}$ and the i-th row $\boldsymbol{B}$ for all $i \in \{1..N\}$. A location and a key sequence $L \in \{1..N\}^N$ and $K \in \{1..N\}^N$ such that (L, K, L, K) form an ℓ-recursive data layout, following Definition 4.1.2. All parties know a security parameter λ.
Output: For all $i \in \{1..N\}$, party $P_{L[i]}$ learns vectors $\{a_{i,*}\}_{K[i]}$ and $\{b_{i,*}\}_{K[i]}$ and learns the public key of every other party.
Goal: Generate key pairs for each party, cipher and distribute input matrices according to their respective location and key sequences.

1. **Key generation:** for all $i \in \{1..N\}$, each party P_i locally executes $\texttt{NaccacheSternSetup}(1^{\lambda})$ to generate a pair of keys (pk_i, sk_i).
2. **Broadcast keys:** for all $i \in \{1..N\}$, party P_i broadcasts its public key pk_i.
3. **Cipher inputs:** for all $i \in \{1..N\}$, for all $j \in [n]$, party P_i locally performs $\texttt{NaccacheSternEncrypt}(pk_{K[i]}, a_{ij})$ and stores the result as a new vector $\{a_{i,*}\}_{K[i]}$. It does the exact same operation with $b_{i,*}$ to get $\{b_{i,*}\}_{K[i]}$.
4. **Distribute rows:**
   a) **Rows of $\boldsymbol{A}$:** for all $i \in \{1..N\}$, party P_i sends $\{a_{i,*}\}_{K[i]}$ to party $P_{L[i]}$.
   b) **Rows of $\boldsymbol{B}$:** for all $i \in \{1..N\}$, party P_i sends $\{b_{i,*}\}_{K[i]}$ to party $P_{L[i]}$.

Protocol 4.2: MP-Copy

Input: Four parties, Alice, Bob, Charlie and Dan. Bob knows a ciphered element $\{x\}_D \in \mathcal{C}_D$ (for $x \in \mathcal{M}$), ciphered using Dan's public key.
Output: Alice learns the element $\{x\}_C$, ciphered using Charlie's public key.
Goal: Recipher from Dan to Charlie and transfer from Bob to Alice.

1. **Add masking**
   a) **Random:** Bob samples uniformly at random $r \in \mathcal{M}$
   b) **Mask:** Bob locally computes $\alpha = \{x\}_D \cdot g^r = \{x + r\}_D \in \mathcal{C}_D$
   c) **Communication:** Bob sends α to Dan.
2. **Recipher:**
   a) **Decipher:** Dan computes $\beta = \texttt{NaccacheSternDecrypt}(sk_D, \alpha) = x + r \in \mathcal{M}$.
   b) **Cipher:** Dan computes $\gamma = \texttt{NaccacheSternEncrypt}(pk_C, \beta) \in \mathcal{C}_C$.
   c) **Communication:** Dan sends γ to Bob.
3. **Remove masking:**
   a) **Unmask:** Bob locally computes $\delta = \gamma \cdot g^{-r} = \{x\}_C \in \mathcal{C}_C$
   b) **Communication:** Bob sends δ to Alice.

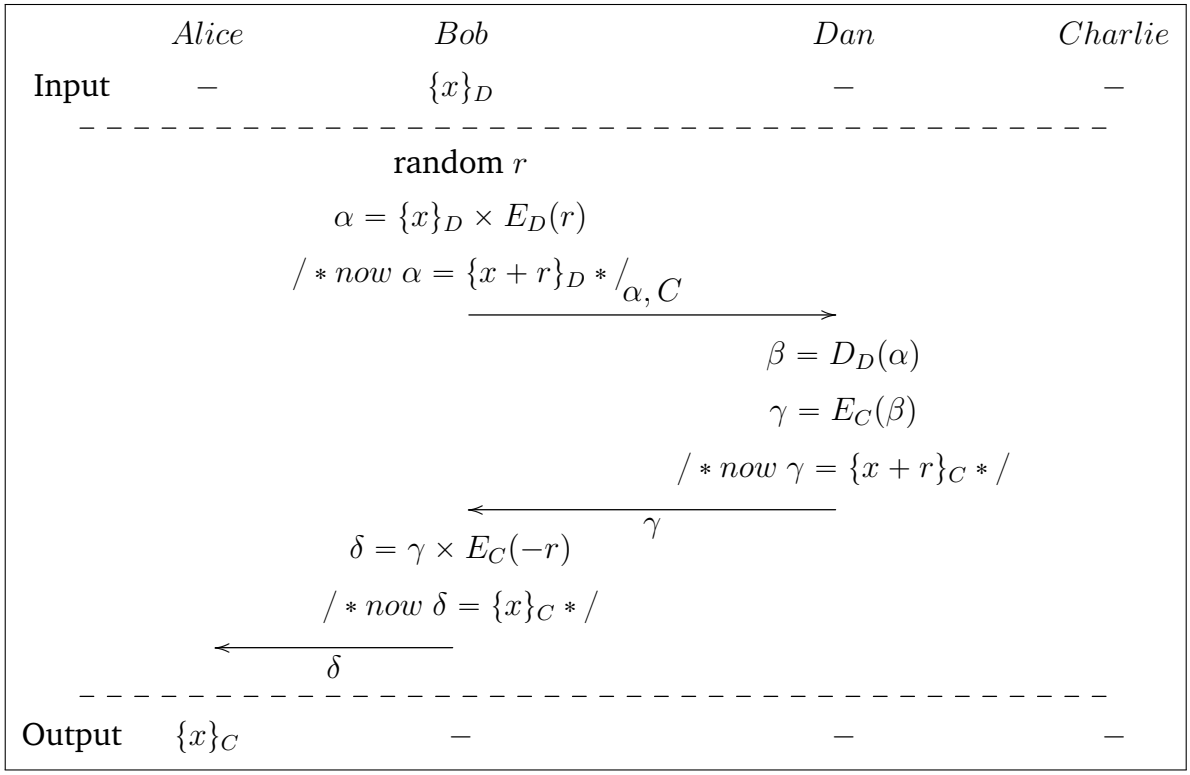

Figure 4.2: MP-Copy : Multiparty copy

the homomorphic multiplication between a ciphertext and a plaintext: $\{a_{i,k}\}_{PK}^{b_{k,j}} = \{a_{i,k}b_{k,j}\}_{PK}$, where PK is the public key that has been used to cipher the element. Therefore, the coefficient $b_{k,j}$ should first be deciphered, and to avoid leaking information, it should also be masked beforehand by some random value.

MaskAndDecrypt takes care of masking and deciphering a whole column of $\boldsymbol{B}$. There, player Charlie is the only one able to decrypt the masked value $\beta_{k,j} = \{b_{k,j} + t_{k,j}\}_C$. For this we require a stream of uniformly random values $t_{k,j}$, that can be sent. To reduce communications, we here instead use a mask generating function (MGF) that generates this stream from a small seed. Then only the seed need to be communicated to remove the mask. All players have of course to agree beforehand on a choice for this mask generating function.

PointwiseProducts shows how player Alice can then recover the ciphertext of one product $\{a_{i,k}b_{k,j}\}_D$. Alice sends her value $\{a_{i,k}\}_D$ to player Charlie who then performs the exponentiation, corresponding to a multiplication on the plaintexts, and sends it back to Alice. Meanwhile Alice has received the seed and generated the masking values $t_{k,j}$ to clean out the product. Finally each coefficient $\{c_{i,j}\}_D$ of the result is computed during a reduction step where player Alice simply multiplies together all corresponding point-wise products.

Overall, BaseCase schedules these three operations. In the calls to Protocols MaskAndDecrypt and PointwiseProducts, Alice is incarnated by Player $P_{L_A[i]}$, Bob by $P_{L_B[k]}$, Char-

Protocol 4.3: MaskAndDecrypt

Input: Two parties, further denoted as Bob and Charlie. They both know their own private key, public keys of all the parties involved, the security parameter $\lambda \in \mathbf{N}$ and the modulus $m \in \mathbf{N}$. Moreover, Bob knows a seed $s_k \in \mathbf{N}$ and a ciphered vector of size n, $\{b_{k,*}\}_C$, whose elements $(b_{k,j}) \in \mathcal{M}^n$ have been ciphered using Charlie's public key.
Output: Charlie learns the additively masked plaintext of Bob's input vector.
Goal: Perform the additive masking of Bob's input vector, and let Charlie learn it.

1. **Mask Bob's input:**
   a) **Generate randoms:** Bob performs MGF(s_k, bitsize(σ)$\times n$) and splits the output in n shares of size bitsize(σ), denoted as $t_{k,j}$ for $j \in \{1..n\}$.
   b) **Mask vector:** for $j \in \{1..n\}$, Bob computes $\beta_{k,j} = \{b_{k,j}\}_C \cdot g^{t_{k,j}} \in \mathcal{C}_C$.
   c) **Communication:** for $j \in \{1..n\}$, Bob sends $\beta_{k,j}$ to Charlie.
2. **Finalise:**
   a) **Decipher:** for $j \in \{1..n\}$, Charlie performs `NaccacheSternDecrypt`(sk_C, $\beta_{k,j}$) and stores the results in $u_{k,j} = b_{k,j} + t_{k,j} \in \mathcal{M}$.

Protocol 4.4: PointwiseProducts

Input: Four parties, further denoted as Alice, Bob, Charlie and Dan. Alice knows a ciphered $\{a_{i,k}\}_D \in \mathcal{C}_D$ for given i and k, ciphered using Dan's public key. Bob knows a seed $s_k \in \mathbf{N}$ and Charlie knows a masked vector $(u_{k,*}) \in \mathcal{M}^n$ (each coefficient is masked by a random value).
Output: Alice learns all the ciphertexts $\{a_{i,k}b_{k,j}\}_D$ for $j \in \{1..n\}$.
Goal: Compute the point-wise products for naive matrix product on a given row.

1. **Communication:** Alice sends $\{a_{i,k}\}_D$ to Charlie
2. **Multiplication:** for $j \in \{1..n\}$, Charlie computes $\delta_{i,k,j} = \{a_{i,k}\}_D^{u_{k,j}}$, $\delta_{i,k,j} \in \mathcal{C}_D$
3. **Communication:** for $j \in \{1..n\}$, Charlie sends $\delta_{i,k,j}$ to Alice.
4. **Send seed:** Bob sends s_k to Alice
5. **Generate and remove masks:** Alice performs MGF(s_k, bitsize(σ)$\times n$) and splits the output in n shares of size σ, denoted as $t_{k,j}$ for $j \in \{1..n\}$.

   For $j \in \{1..n\}$, Alice computes:

$$\epsilon_{i,k,j} = \delta_{i,k,j} / \left(\{a_{i,k}\}_D^{t_{k,j}}\right) = \{a_{i,k}(b_{k,j} + t_{k,j}) - a_{i,k}t_{k,j}\}_D \in \mathcal{C}_D.$$

lie by $P_{K_B[k]}$ and Dan by $P_{K_A[i]}$.

Protocol 4.5: BaseCase

Input: two $n \times n$ matrices $\{\boldsymbol{A}\}_{K_{\boldsymbol{A}}}$ and $\{\boldsymbol{B}\}_{K_{\boldsymbol{B}}}$ distributed and ciphered according to a base-case data layout $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}, L_{\boldsymbol{B}}, K_{\boldsymbol{B}}) \in (\{1..N\}^n)^4$ among parties $(P_1, \ldots, P_N)$ as in Definition 4.1.2.
Output: Matrix $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ is distributed and ciphered among parties $(P_1, \ldots, P_N)$ according to the location and key sequences $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}})$.
Goal: Compute $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ distributed and ciphered in the same way as $\boldsymbol{A}$ is.

1. **Computation**:

   For all $k \in \{1..n\}$

   a) **Choose a seed:** Party $P_{L_{\boldsymbol{B}[k]}}$ samples uniformly at random a seed $s_k \in \mathbf{N}$ according to the security parameter λ.
   b) Parties $P_{L_{\boldsymbol{B}[k]}}$ and $P_{K_{\boldsymbol{B}[k]}}$ run MaskAndDecrypt on vector $\{b_{k,*}\}_{K_{\boldsymbol{B}[k]}}$
   c) For all $i \in \{1..n\}$

      Parties $P_{L_{\boldsymbol{A}[i]}}$, $P_{L_{\boldsymbol{B}[k]}}$, $P_{K_{\boldsymbol{B}[k]}}$ and $P_{K_{\boldsymbol{A}[i]}}$ run PointwiseProducts where Parties $P_{L_{\boldsymbol{A}[i]}}$ learn $\epsilon_{i,k,j} = \{a_{i,k}b_{k,j}\}_{K_A[i]}$ for all $j \in \{1..n\}$.

2. **Reduction**: for all $i \in \{1..n\}$ Party $P_{L_{\boldsymbol{A}[i]}}$ computes $\{c_{i,j}\}_{K_{\boldsymbol{A}[i]}} \leftarrow \prod_{k=1}^{n} \epsilon_{i,k,j}$

Theorem 4.2.1. *BaseCase correctly computes the product $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$ in the specified layout. It requires a communication of $n^3 + 3n^2 + n$ modular integers.*

Proof. Correctness stems first from the fact that $c_{i,j} = \sum_{k=1}^{n} a_{i,k}b_{k,j}$ is obtained "in the exponents" by the homomorphic properties (4). Second the masks applied in MaskAndDecrypt are all removed in PointwiseProducts. Now, the communication cost in number of ring element is n for MaskAndDecrypt and $n + 1$ for PointwiseProducts. MaskAndDecrypt and PointwiseProducts also send one seed, which, for simplicity, we consider smaller than a modular integer. Overall this yields a communication cost lower than $n(n + 1) + n^2(n + 2) = n^3 + 3n^2 + n$ modular integers for BaseCase. □

As an illustration,we give in Figure 4.3 an illustration of the scheduling of BaseCase in a scenario with 4 players.

4.2.4 Security Analysis

From the formalization of the different protocols we can state the security of the overall base case for matrix multiplication in the following Theorem 4.2.2.

Theorem 4.2.2. *If players share a 0-data-layout, BaseCase is secure against one semi-honest adversary.*

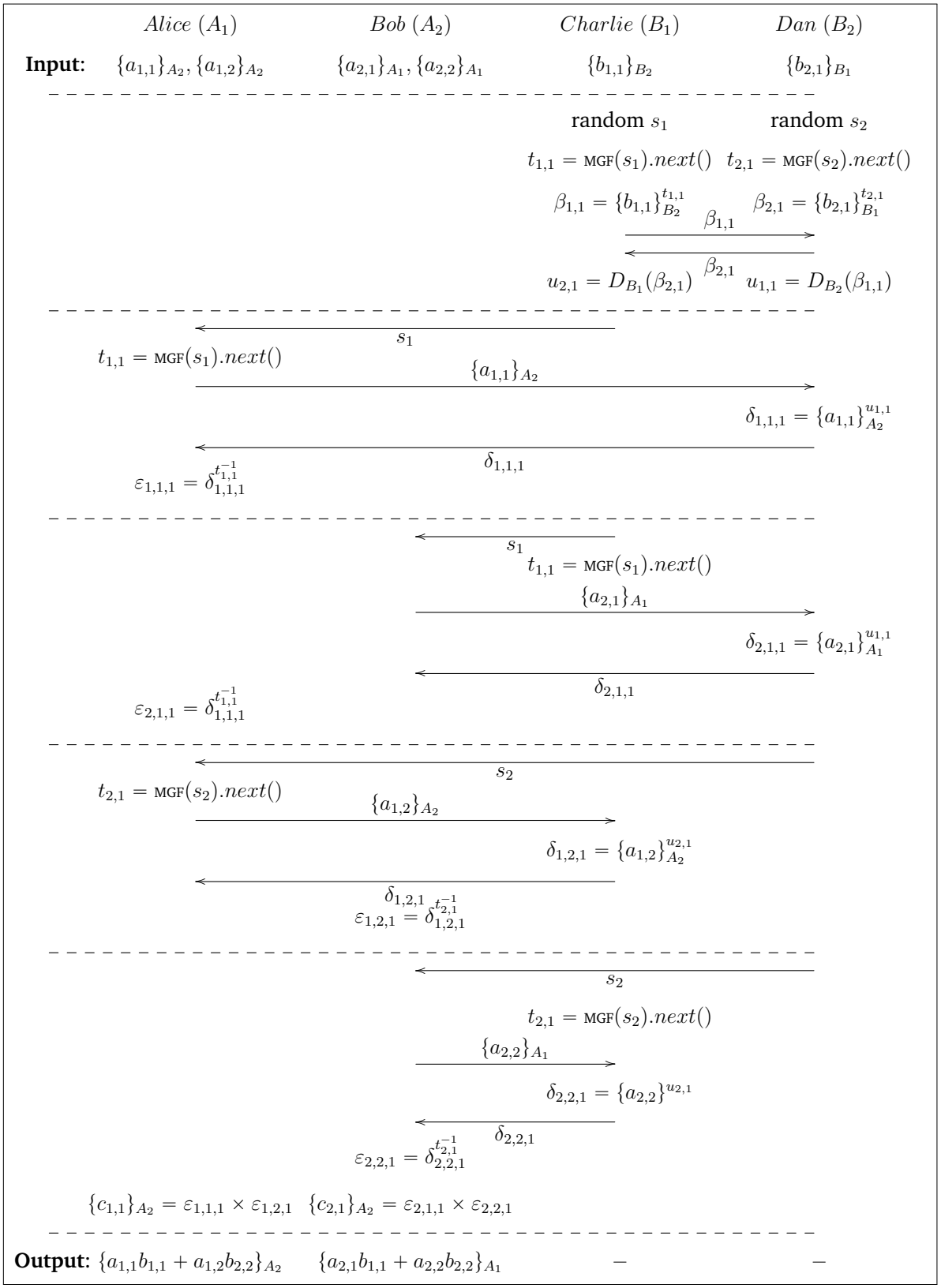

Figure 4.3: BaseCase protocol execution with 4 players

Proof. We start by proving that both subprotocols MaskAndDecrypt ($M\&D$) and PointwiseProducts (PWP) are secure against one semi-honest adversary.

Security of MaskAndDecrypt

MaskAndDecrypt is a 2-party protocol such that: $\text{output}^{M\&D}((\{\boldsymbol{b_1}\}_{P_2}, s_1), -) = (-, \boldsymbol{u_1})$, with $\boldsymbol{b_1} = b_{k,*}$, $s_1 = s_k$, and $u_1 = \{u_{k,j}\}_{j\in\{1..n\}}$. The proof is then divided in two parts: one for each corruption case. We labeled P_1 the player providing the seeds as input.

P_1 is corrupted: the view of P_1 is: $\text{view}_{P_1}^{M\&D} = (\boldsymbol{t_1}, \boldsymbol{\beta_1})$. From the inputs of P_1, the simulator is able to perfectly simulate the view of P_1.

P_2 is corrupted: the view of P_2 is: $\text{view}_{P_2}^{M\&D} = (\boldsymbol{\beta_1})$. From the output $\boldsymbol{u_1}$ of P_2, the simulator S_2 ciphers each of its elements with the key of P_2. From the IND-CPA security, the simulated view is computationally indistinguishable from the real one.

Security of PointwiseProducts

PointwiseProducts is a 4-party protocol. However, the 4^{th} player does not have any input nor output: only its public key is used. In the same vein, P_2 only sends s_2 and does not interact otherwise. Its view is empty, so that its simulator is trivial. Therefore, the proof is only divided in two parts. The output of the protocol is: $\text{output}^{PWP}(\{a_1\}_{P_4}, s_2, \boldsymbol{u_k}, -) = (\boldsymbol{\epsilon}, -, -, -)$ with $a_1 = a_{i,k}$, $\boldsymbol{u_k} = u_{k,*}$ and $\boldsymbol{\epsilon} = \{\epsilon_{i,k,j}\}_{j\in\{1..n\}}$.

P_1 is corrupted: the view of P_1 is: $\text{view}_{P_1}^{PWP} = (s_2, \boldsymbol{t_1}, \boldsymbol{\delta_1})$. The simulator S_1 picks $s_2' \xleftarrow{\$} \mathbf{N}$ and computes $\boldsymbol{t_1'}$ as in the protocol. Then, from the output $\boldsymbol{\epsilon}$ and the input $\{a_1\}_{P_4}$, it computes $\boldsymbol{\delta'} = \boldsymbol{\epsilon} * \{a_1\}_{P_4}^{\boldsymbol{t_1'}}$ component wise. Since the $\boldsymbol{\delta}$ values are ciphered with the key of P_4, and that s_1 is a random value, both views are indistinguishable.

P_3 is corrupted: we have $\text{view}_{P_3}^{PWP} = (\{a_1\}_{P_4}, \boldsymbol{\delta_1})$. S_3: $a_1' \xleftarrow{\$} \mathcal{M}$, then the value is ciphered with the public key of P_4 to obtain $\{a_1'\}_{P_4}$. Next, it computes $\boldsymbol{\delta_1'}$ as in protocol using the simulated value $\{a_1'\}_{P_4}$. This simulation is computationally indistinguishable from the real view thanks to the IND-CPA security of the cryptosystem.

We denote by $F^{M\&D}$ (respectively F^{PWP}) the ideal functionalities associated to MaskAndDecrypt (resp. PointwiseProducts). We will now prove that if players shares a 0-data-layout, BaseCase is secure against one semi-honest adversary in the $(F^{M\&D}, F^{PWP})$-hybrid model.

Security of BaseCase

BaseCase is am N-party protocol, where the view depends on which group the player belongs. Since players share a 0-data-layout, there are four distinct possibilities: $\{L_A, K_A, L_B, K_B\}$. The cases where $P_{K_A[i]}$ or $P_{L_B[i]}$ is corrupted are trivial, since their respective view are empty in the $(F^{M\&D}, F^{PWP})$-hybrid model.

$P_{L_A[i]}$ **is corrupted:** the view of $P_{L_A[i]}$ is: $\text{view}^{\text{BaseCase}}_{P_{L_A[i]}} = (\{\boldsymbol{\epsilon}\}P_{K_A[i]})$ where ϵ_i is the output of a call to F^{PWP}. The simulator S_i executes: for each $k \in \{1..N\}$: from BaseCase output in the ideal world, it picks $N-1$ random shares in $\mathcal{M}$ (denoted $\epsilon'_i, i \in \{1..N-1\}$), and ciphers them using $Pk_{K_A}[i]$. Then, it chooses the last share ϵ'_n such that: $\epsilon'_n * \prod_{k=1}^{n-1} \epsilon'_i$. If ϵ'_n belongs to $\mathcal{C}_{K_{\boldsymbol{A}[i]}}$, then it outputs each component of $\boldsymbol{\epsilon}'$, otherwise it redoes the process from the beginning for the k^{th} step. The definition of the data layout ensures that $P_{L_A[i]} \neq P_{K_A[i]}$, so that ϵ_i and ϵ'_i are indistinguishable as long as the encryption scheme is IND-CPA. Moreover, since the choice of each share is consistent with the output of the protocol (i.e., their product is equal to the output), the adversary is not able to computationally distinguish between the real and the simulated execution.

$P_{L_B[i]}$ **is corrupted:** the view of $P_{L_B[i]}$ is: $\{\text{view}^{\text{BaseCase}}_{P_{L_B[i]}} = (\boldsymbol{u})\}$. The output of the protocol is empty for this player. The simulator picks n random values from $\mathcal{M}$, andoutputs each of them to form $\boldsymbol{u}'$. In the real world, each u_i is masked by a random value (unknown by $P_{L_B[i]}$ since $P_{L_B[i]} \neq P_{K_B[i]}$), so that u_i and u'_i are then perfectly indistinguishable.

Finally, we apply the composition Theorem 4.1.6: since we have proven the security of BaseCase in the $(F^{M\&D}, F^{PWP})$-hybrid model, and that PointwiseProducts and MaskAndDecrypt are secure, and that each call to both of these protocols are sequentially made, we conclude that BaseCase is secure against one semi-honest adversary. Moreover, the 0-data layout ensures that the seed sharing does not leak information. □

4.3 Multiparty Strassen-Winograd

4.3.1 Operation schedule in MP-SW

The 22 operations in a recursive step of Strassen-Winograd's algorithm is composed by 15 matrix additions and 7 recursive calls. The matrix additions are performed using component-wise homomorphic additions, denoted by Hom-Mat-Add: each player performs locally a simple homomorphic addition of the rows of the two input operands that she stores. Homomorphic subtraction, denoted by Hom-Mat-Sub, works similarly. However, this requires that the two operands share the same key and location sequences. To ensure this, some matrices will be copied from one key-location sequence to another, using a multiparty matrix copy, denoted by MP-Mat-Copy. The location sequences of the input an output are non-intersecting (and therefore so are the related key sequences). These operations are achieved by n^2 instances of MP-Copy as shown in MP-Mat-Copy.

Theorem 4.3.1. *Assuming an ℓ-data layout, Protocol MP-Mat-Copy is secure against one semi-honest adversary.*

Protocol 4.6: MP-Mat-Copy

Input: An $n \times n$ matrix $\{\boldsymbol{A}\}_{K_{\boldsymbol{A}}}$ distributed and ciphered according to a location and a key sequence $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}) \in (\{1..N\}^n)^2$ among parties $(P_1, \ldots, P_N)$ following Definition 4.1.2 and a location-key sequence (L', K').
Output: A copy $\{\boldsymbol{A}\}_{K'}$ is distributed and ciphered among parties $(P_1, \ldots, P_N)$ according to the location and key sequences (L', K').

For all $i, j \in \{1..n\}^2$

Parties $P_{L'[i]}, P_{L[i]}, P_{K'[i]}$ and $P_{K[i]}$ run MP-Copy to copy $\{a_{i,j}\}_{K[i]}$ to $\{a_{i,j}\}_{K'[i]}$

We only give a sketch of the proof, since its very similar to the one for the MaskAndDecrypt protocol within the proof of Theorem 4.2.2.

Proof. First, we prove that MP-Copy is secure against one semi-honest adversary: from the data layout or the added randomness, each players only see ciphers or additively masked values so that it does not lean anything from the execution. Then, we prove the security in an hybrid model where calls to MP-Copy are replaced by an equivalent ideal functionality. Since the output is ciphered accordingly to the data layout, a simulation by ciphering random values is computationally indistinguishable from the real execution. Finally, by sequentially composing calls to the MP-Copy protocol, we apply the sequential composition theorem to conclude. □

We propose in MP-SW a scheduling of these operations and data movement ensuring that all additions can be made homomorphically, that the key and location sequences for all seven recursive calls satisfy the requirements for a base-case data-layout (Definition 4.1.2) and finally that the output matrix also follows the location and key sequences of the first operand. The last three columns in MP-SW indicate the location sequences of the input and output operands for each operation.

Figure 4.4 also presents the data exchange between groups of players in one recursive level of MP-SW.

Note that the initial problem requires that both operands $\boldsymbol{A}$ and $\boldsymbol{B}$ share the same key and location sequences (so that matrix squaring is possible). However, the base case protocol (BaseCase) requires that these sequences are non-intersecting. In order to satisfy these two constraints the recursive Strassen-Winograd algorithm is presented with a location and key sequence for $\boldsymbol{A}$ ($L_{\boldsymbol{A}}$ and $K_{\boldsymbol{A}}$) and a location and key sequence for $\boldsymbol{B}$ ($L_{\boldsymbol{B}}$ and $K_{\boldsymbol{B}}$). The algorithm does not require that they are non intersecting, but ensures that from the first recursive call, they will always be, so as to fit with the requirement of the base case, BaseCase.

Lemma 4.3.2. *The total communication cost of a recursive level of MP-SW following the schedule defined MP-SW, Step 2 is* $18\left(\frac{n}{2}\right)^2$ *communications.*

Protocol 4.7: MP-SW

Input: two $n \times n$ matrices $\{\boldsymbol{A}\}_{K_{\boldsymbol{A}}}$ and $\{\boldsymbol{B}\}_{K_{\boldsymbol{B}}}$, distributed and ciphered according to an ℓ-recursive data layout $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}, L_{\boldsymbol{B}}, K_{\boldsymbol{B}}) \in (\{1..N\}^n)^4$ among parties $(P_1, \ldots, P_N)$ following Definition 4.1.2, where $n = b2^\ell$.

Output: $\{\boldsymbol{C}\}_{K_{\boldsymbol{A}}} = \{\boldsymbol{A} \times \boldsymbol{B}\}_{K_{\boldsymbol{A}}}$, distributed and ciphered among parties $(P_1, \ldots, P_N)$ according to the location and key sequences $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}})$.

1. If $\ell = 0$: Parties in $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}})$ and $(L_{\boldsymbol{B}}, K_{\boldsymbol{B}})$ run `BaseCase` on $\{\boldsymbol{A}\}_{K_{\boldsymbol{A}}}$ and $\{\boldsymbol{B}\}_{K_{\boldsymbol{B}}}$
2. Else

					In1 loc.	In2 loc.	Out loc.
$\{\boldsymbol{S}_1\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{A}_{21}\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{A}_{22}\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{A}'_{11}\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	MP-Mat-Copy	$(\{\boldsymbol{A}_{11}\}_{K_{\boldsymbol{A}_U}},$	$(L_{\boldsymbol{A}_L}, K_{\boldsymbol{A}_L}))$	$L_{\boldsymbol{A}_U}$		$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{S}_2\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{S}_1\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{A}'_{11}\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{S}_3\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{A}'_{11}\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{A}_{21}\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{S}'_2\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	MP-Mat-Copy	$(\{\boldsymbol{S}_2\}_{K_{\boldsymbol{A}_L}},$	$(L_{\boldsymbol{A}_U}, K_{\boldsymbol{A}_U}))$	$L_{\boldsymbol{A}_L}$		$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{S}_4\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{A}_{12}\}_{K_{\boldsymbol{A}_U}},$	$\{\boldsymbol{S}'_2\}_{K_{\boldsymbol{A}_U}})$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{T}_1\}_{K_{\boldsymbol{B}_U}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{B}_{12}\}_{K_{\boldsymbol{B}_U}},$	$\{\boldsymbol{B}_{11}\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$
$\{\boldsymbol{B}'_{22}\}_{K_{\boldsymbol{B}_U}}$	$\leftarrow$	MP-Mat-Copy	$(\{\boldsymbol{B}_{22}\}_{K_{\boldsymbol{B}_L}},$	$(L_{\boldsymbol{B}_U}, K_{\boldsymbol{B}_U}))$	$L_{\boldsymbol{B}_L}$		$L_{\boldsymbol{B}_U}$
$\{\boldsymbol{T}_2\}_{K_{\boldsymbol{B}_U}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{B}'_{22}\}_{K_{\boldsymbol{B}_U}},$	$\{\boldsymbol{T}_1\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$
$\{\boldsymbol{T}_3\}_{K_{\boldsymbol{B}_U}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{B}'_{22}\}_{K_{\boldsymbol{B}_U}},$	$\{\boldsymbol{B}_{12}\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$
$\{\boldsymbol{B}'_{21}\}_{K_{\boldsymbol{B}_U}}$	$\leftarrow$	MP-Mat-Copy	$(\{\boldsymbol{B}_{21}\}_{K_{\boldsymbol{B}_L}},$	$(L_{\boldsymbol{B}_U}, K_{\boldsymbol{B}_U}))$	$L_{\boldsymbol{B}_L}$		$L_{\boldsymbol{B}_U}$
$\{\boldsymbol{T}_4\}_{K_{\boldsymbol{B}_U}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{T}_2\}_{K_{\boldsymbol{B}_U}},$	$\{\boldsymbol{B}'_{21}\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{B}_U}$
$\{\boldsymbol{R}_1\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{A}'_{11}\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{B}_{11}\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{R}_2\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{A}_{12}\}_{K_{\boldsymbol{A}_U}},$	$\{\boldsymbol{B}_{21}\}_{K_{\boldsymbol{B}_L}})$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{B}_L}$	$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{R}_3\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{S}_4\}_{K_{\boldsymbol{A}_U}},$	$\{\boldsymbol{B}_{22}\}_{K_{\boldsymbol{B}_L}})$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{B}_L}$	$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{R}_4\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{A}_{22}\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{T}_4\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{R}_5\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{S}_1\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{T}_1\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{R}_6\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{S}_2\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{T}_2\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{R}_7\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	MP-SW	$(\{\boldsymbol{S}_3\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{T}_3\}_{K_{\boldsymbol{B}_U}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{B}_U}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{R}'_1\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	MP-Mat-Copy	$(\{\boldsymbol{R}_1\}_{K_{\boldsymbol{A}_L}},$	$(L_{\boldsymbol{A}_U}, K_{\boldsymbol{A}_U}))$	$L_{\boldsymbol{A}_L}$		$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{U}_1\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{R}'_1\}_{K_{\boldsymbol{A}_U}},$	$\{\boldsymbol{R}_2\}_{K_{\boldsymbol{A}_U}})$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{U}_2\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{R}_1\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{R}_6\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{U}_3\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{U}_2\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{R}_7\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{U}_4\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{U}_2\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{R}_5\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{U}'_4\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	MP-Mat-Copy	$(\{\boldsymbol{U}_4\}_{K_{\boldsymbol{A}_L}},$	$(L_{\boldsymbol{A}_U}, K_{\boldsymbol{A}_U}))$	$L_{\boldsymbol{A}_L}$		$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{U}_5\}_{K_{\boldsymbol{A}_U}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{U}'_4\}_{K_{\boldsymbol{A}_U}},$	$\{\boldsymbol{R}_3\}_{K_{\boldsymbol{A}_U}})$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{A}_U}$	$L_{\boldsymbol{A}_U}$
$\{\boldsymbol{U}_6\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Sub	$(\{\boldsymbol{U}_3\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{R}_4\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$
$\{\boldsymbol{U}_7\}_{K_{\boldsymbol{A}_L}}$	$\leftarrow$	Hom-Mat-Add	$(\{\boldsymbol{U}_3\}_{K_{\boldsymbol{A}_L}},$	$\{\boldsymbol{R}_5\}_{K_{\boldsymbol{A}_L}})$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$	$L_{\boldsymbol{A}_L}$

3. **End result** $\{\boldsymbol{C}\}_{K_{\boldsymbol{A}}} \leftarrow \begin{bmatrix} \{\boldsymbol{U}_1\}_{K_{\boldsymbol{A}_U}} & \{\boldsymbol{U}_5\}_{K_{\boldsymbol{A}_U}} \\ \{\boldsymbol{U}_6\}_{K_{\boldsymbol{A}_L}} & \{\boldsymbol{U}_7\}_{K_{\boldsymbol{A}_L}} \end{bmatrix}$

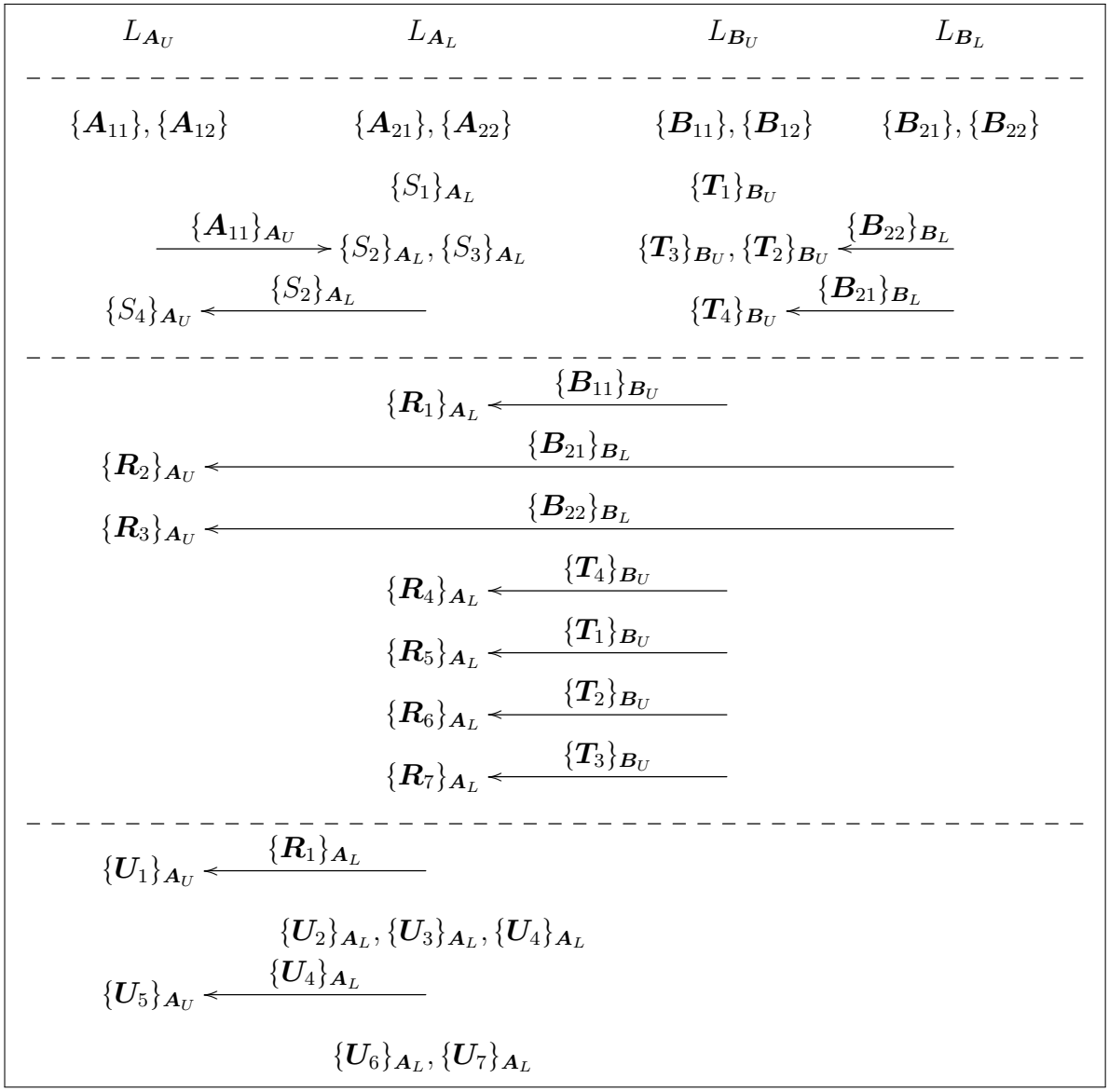

Figure 4.4: Protocol for Strassen-Winograd algorithm. Each column represents one of the four sub-group of players where the submatrices $\boldsymbol{A}_U, \boldsymbol{A}_L, \boldsymbol{B}_U, \boldsymbol{B}_L$ are stored.

Proof. The only communication are that of the 6 calls to MP-Mat-Copy, each accounting for $3(n/2)^2$ communication. □

Finally, our main security result is that of the following Theorem 4.3.3.

Theorem 4.3.3. *Assuming an ℓ-data layout, MP-SW is secure against one semi-honest adversary.*

Proof. First, we prove that MP-SW is secure in the $F^{\mathsf{BaseCase}}, F^{Copy}, F^{\mathsf{MP\text{-}SW}}_{N/2}$-hybrid model, where F^{BaseCase}, F^{Copy} and $F^{\mathsf{MP\text{-}SW}}_{N/2}$ respectively denotes the ideal functionality associated to BaseCase, MP-Mat-Copy, and MP-SW with $N/2$ players. In this model, calls to MP-Mat-Copy are replaced by ideals calls to F^{Copy}. In the same vein, if $N \leqslant T$, the MP-SW calls are replaced by F^{BaseCase}, or by $F^{\mathsf{MP\text{-}SW}}_{N/2}$ otherwise. We need to prove that for any corrupted player, its real view is indistinguishable from the simulated one. From the inputs described as in MP-SW (implicit in the following), the outputs for the player $P_{L_{\boldsymbol{A}_X[i]}}$ are the rows of the following matrices, ciphered with $Pk_{K_{\boldsymbol{A}_X[i]}}$, with $X \in \{U, L\}$. $\mathsf{output}^{\mathsf{MP\text{-}SW}}_{\boldsymbol{A}_U}(\boldsymbol{U_1}, \boldsymbol{U_5})$ and $\mathsf{output}^{\mathsf{MP\text{-}SW}}_{A_L}(\boldsymbol{U_6}, \boldsymbol{U_7})$. Using the same notations, we obtains the following views:
$\mathsf{view}^{\mathsf{MP\text{-}SW}}_{L_{\boldsymbol{A}_U}} = (\boldsymbol{S'_2}, \boldsymbol{S_4}, \boldsymbol{R'_1}, \boldsymbol{R_2}, \boldsymbol{R_3}, \boldsymbol{U'_4})$,
$\mathsf{view}^{\mathsf{MP\text{-}SW}}_{L_{\boldsymbol{A}_L}} = (\boldsymbol{S_1}, \boldsymbol{A'_{11}}, \boldsymbol{S_2}, \boldsymbol{S_3}, \boldsymbol{R_1}, \boldsymbol{R_4}, \boldsymbol{R_5}, \boldsymbol{R_6}, \boldsymbol{R_7}, \boldsymbol{U_2}, \boldsymbol{U_3}, \boldsymbol{U_4})$,
$\mathsf{view}^{\mathsf{MP\text{-}SW}}_{L_{\boldsymbol{B}_U}} = (\boldsymbol{T_1}, \boldsymbol{B'_{22}}, \boldsymbol{T_2}, \boldsymbol{T_3}, \boldsymbol{T_4})$, $\mathsf{view}^{\mathsf{MP\text{-}SW}}_{L_{\boldsymbol{B}_L}} = (-)$ We construct a generic simulator, where differences depending on the corrupted player are explicitly detailed. The simulator $S_{i \in \{1..N\}}$ takes two random matrices in α and β both in $\mathcal{M}^{(N \times N)}$. Then, it replaces the rows for the corrupted player with its actual inputs (i.e., the rows of A and B owned by the corrupted player). The remaining coefficients are ciphered accordingly to the data layout. The first part of the protocol (i.e. the computation of S_i and T_i, $i \in \{1..4\}$) is simulated using the inputs and ideal calls to F^{Copy}. This simulates the views for the $L_{\boldsymbol{B}}$ cases. Then, there are two cases.

$P_{L_{\boldsymbol{A}_U[i]}}$ is corrupted: From the output, the simulator $S_{L_{\boldsymbol{A}_U[i]}}$ takes $N/2$ random values from $\mathcal{C}_{\boldsymbol{A}_U[i]}$ to obtain the simulation of the row of $\boldsymbol{U_4}$. Then, it computes the row of $\boldsymbol{R_3} = \mathsf{Hom\text{-}Mat\text{-}Sub}(\boldsymbol{U_5}, \boldsymbol{U_4})$. Similarly, it take the row $\boldsymbol{R'_1}$ at random, and computes $\boldsymbol{R_1} = \mathsf{Hom\text{-}Mat\text{-}Sub}(\boldsymbol{U_1}, \boldsymbol{R'_1})$.

$P_{L_{\boldsymbol{A}_L[i]}}$ is corrupted: $S_{L_{\boldsymbol{A}_L[i]}}$ samples $3N/2$ random values from $\mathcal{C}_{\boldsymbol{A}_L[i]}$ to simulate the row of $\boldsymbol{U_2}$, $\boldsymbol{R_1}$ and $\boldsymbol{R_7}$. Next, it computes: $\boldsymbol{R_6} = \mathsf{Hom\text{-}Mat\text{-}Sub}(\boldsymbol{U_2}, \boldsymbol{R_1})$, $\boldsymbol{U_3} = \mathsf{Hom\text{-}Mat\text{-}Add}(\boldsymbol{U_2}, \boldsymbol{R_7})$, $\boldsymbol{R_4} = \mathsf{Hom\text{-}Mat\text{-}Sub}(\boldsymbol{U_3}, \boldsymbol{U_6})$, $\boldsymbol{R_5} = \mathsf{Hom\text{-}Mat\text{-}Sub}(\boldsymbol{U_7}, \boldsymbol{U_3}$ and $\boldsymbol{U_4} = \mathsf{Hom\text{-}Mat\text{-}Add}(\boldsymbol{U_2}, \boldsymbol{R_5})$.

We now prove that the simulated view is indistinguishable from the real one. The proof relies on a sequence of hybrid games, where each transition is based on indistinguishability.

H_0: The first game represents the view of a real protocol execution in the $(F^{Copy}, F^{\mathsf{BaseCase}}, F^{\mathsf{MP\text{-}SW}}_{N/2})$-hybrid model.

H_1: for each call to F^{Copy}, we replace the output of the functionality by random numbers, accordingly ciphered with the data layout. i.e.: $\forall j \in \{1..N\}, r_i \xleftarrow{\$} \mathcal{M}$ and $\{r_j\}_{K_X[j]}$ with $X \in \{\boldsymbol{A}_L, \boldsymbol{B}_U\}$. As only one player is corrupted, and $(L_{\boldsymbol{A}}, K_{\boldsymbol{A}}, L_{\boldsymbol{B}}, K_{\boldsymbol{B}})$ is

a l-recursive data layout which verifies $(L_{X_U} \cup K_{X_U}) \cap (L_{X_L} \cup K_{X_L}) = \varnothing, X \in \{\boldsymbol{A}, \boldsymbol{B}\}$, then the player obtains ciphers which it cannot decipher. Then, the IND-CPA security of the cryptosystem ensures that H_0 and H_1 are indistinguishable.

H_2: In this game, we replace the output obtained from: F^{BaseCase} if $N \leqslant T$; or $F^{\text{MP-SW}}_{N/2}$ otherwise; by the previously detailed simulation for the R_i, $i \in \{1..N\}$. From the data layout, the corrupted player ($P_{L_{\boldsymbol{A}_U[i]}}$ or $P_{L_{\boldsymbol{A}_L[i]}}$) in the real case gets undecipherable values that cannot be guessed from the inputs of the adversary (which knows one row of each matrix in the worst case) so that the simulation is computationally indistinguishable. Then, $H_1 \overset{C}{\equiv} H'_2$.

H_3: In this game, we replace the $\boldsymbol{U_i}$ of the real view with the simulated ones $\boldsymbol{U_i}$, $i \in \{2..4\}$. Each of the simulated values is directly computed from the output, so that as long as the adversary is not able to distinguish ciphers, the simulation is computationally indistinguishable from the real execution.

H_3 represents the simulated view for N players. We have then proven that MP-SW is secure against one semi-honest adversary in the $(F^{Copy}, F^{\text{BaseCase}}, F^{\text{MP-SW}}_{N/2})$-model.

Second, we prove that if we assume a l-data layout between the players, MP-SW is secure against one semi-honest adversary under a sequential composition of the sub-protocols MP-Mat-Copy and BaseCase. By induction, we suppose that MP-SW is secure with $N/2$ players, and we show that MP-SW for N players.

$Base\ Case : N \leqslant T$. In this case, MP-SW calls are replaced by calls to BaseCase. By construction, the data layout is now 0-recursive. Then, the corrupted player cannot act as more than one player in the execution, so that the security of the protocol against one-semi honest is enough.

$Induction : N > T$. In this case, each call to MP-SW is assumed secure from the induction hypothesis. Then, each of these calls can be sequentially realized.

Then, since all sub-protocols calls can be realized sequentially, and since we have proven that MP-SW is secure in the $F^{\text{BaseCase}}, F^{Copy}, F^{\text{MP-SW}}_{N/2}$-hybrid model, the sequential composition theorem ensures that the protocol obtained by composition is also secure. Henceforth, by induction, we have proven that from $F^{\text{MP-SW}}_{N/2}$, we are able to construct a secure execution of MP-SW. In conclusion, MP-SW is secure against one semi-honest adversary. □

4.3.2 Finalisation step

Finally, there remains to decipher and distribute each row of $\{\boldsymbol{C}\}_{K_{\boldsymbol{A}}}$ to the party who has to learn it. By setting the key sequence to $K_A = (1, 2, 3 \dots)$ as in Lemma 4.1.3, this player is able to perform the decryption himself. This finalisation step is formally described in SW-Finalise and uses N^2 communications.

Protocol 4.8: SW-Finalise

Input: An $N \times N$ matrix $\{\boldsymbol{C}\}_{K_{\boldsymbol{C}}}$ distributed and ciphered according to the location and key sequences $(L_{\boldsymbol{C}}, K_{\boldsymbol{C}}) \in (\{1..N\}^N)^2$ among parties $P_1, \ldots, P_N$, following Definition 4.1.1.
Output: Each party $P_{K_{\boldsymbol{C}[i]}}$ learns the plaintext of the i-th row of $\boldsymbol{C}$.

1. **Exchange rows:** For all $i \in \{1..N\}$, party $P_{L_{\boldsymbol{C}[i]}}$ send row i of $\boldsymbol{C}$ to party $P_{K_{\boldsymbol{C}[i]}}$.
2. **Decipher vector:** For all $i \in \{1..N\}$, for all $j \in \{1..N\}$, party $P_{K_{\boldsymbol{C}[i]}}$ runs $\texttt{NaccacheSternDecrypt}(sk_{K_{\boldsymbol{C}[i]}}, (\{c_{i,j}\}_{K_{\boldsymbol{C}[i]}}))$ and stores the output values in a vector $\boldsymbol{c}_{K_{\boldsymbol{C}[i]}} \in \mathcal{M}^N$.

4.3.3 Cost and security analysis

From Lemma 4.3.2 and Theorem 4.2.1, the recurrence relation for communication complexity of MP-SW writes:

$$\begin{cases} C(n) &= 7C\left(\frac{n}{2}\right) + 18\left(\frac{n}{2}\right)^2 \text{ for } n > b \\ C(b) &= b^3 + 3b^2 + b \text{ for the base case} \end{cases} \tag{4.3.1}$$

The threshold at which the recursive algorithm should switch to the base case algorithm is set by finding at which dimension b does the base case algorithm start to perform worse than one recursive level. In terms of communication cost, this means the following equation: $7((\frac{b}{2})^3 + 3(\frac{b}{2})^2 + \frac{b}{2}) + 18(\frac{b}{2})^2 = b^3 + 3b^2 + b$ which comes from injecting the base case cost of Theorem 4.2.1 into the recurrence formula. It gives a threshold of $b = 56$.

Theorem 4.3.4. *For $N = b2^\ell$ parties $(P_i)_{i\in\{1..N\}}$ and two matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{M}^{N\times N}$, such that party P_i knows the i-th row of $\boldsymbol{A}$ and the i-th row $\boldsymbol{B}$ for all $i \in \{1..N\}$, the execution in sequence of algorithms (SW-Setup; MP-SW; SW-Finalise), using the ℓ-recursive data layout of Equation (4.1.1), correctly computes $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B} \in \mathcal{M}$ with $O(7^\ell b^3)$ communications in $O(\ell)$ rounds and is secure against one semi-honest adversary. When b is constant, then $\ell = O(\log_2(N))$, and the communication bound is $O(N^{\log_2(7)})$.*

Proof. Correctness of MP-SW is given by Theorem 4.2.1 for the basecase and that of Strassen-Winograd algorithm (Section 4.1.1). Then SW-Setup is just the set up of the keys and initial encipherings, while SW-Finalise is the associated decipherings. Then, the communication bound stems from Theorem 4.2.1 and Equation (4.3.1), with $3N^2$ communications for SW-Setup and SW-Finalise. The non-recursive parts of each recursive level of MP-SW require a constant number of rounds, and so does the execution of BaseCase, leading to a total of $O(\ell)$ rounds. For the security. again SW-Setup is just the communication of public keys and self-ciphered values, while SW-Finalise is also the communication of ciphered values to their legitimate locations.

Finally, Theorem 4.3.3 asserts the security of MP-SW and the sequential execution of (SW-Setup; MP-SW; SW-Finalise) that of the whole process. □

We now compare the cost of MP-SW with the cost of MP-PDP, $C_{\text{MP-PDP}}(n) = n^3 + n(n-1)$. We also recall that the initialization step SW-Setup costs $C_{\text{init}} = 2n^2$ and the finalisation step SW-Finalise costs $C_{\text{final}} = n^2$. The crossover point where our full algorithm improves over MP-PDP in communication cost is obtained by solving the equation: $C(n) + 3n^2 \leqslant n^3 + n(n-1)$ which yields $n > 94$, with one recursive call. This means that for any instance of dimension larger than 96, the proposed MP-SW algorithm has a better communication cost than MP-PDP.

4.4 Experiments

We implemented the algorithms under study to demonstrate their behavior in practice and compared them to the state of the art implementations of other solutions. In the following SPD$\mathbf{Z}_{2^k}$ refers to a run of a textbook matrix multiplication algorithm performed with the general purpose library SPD$\mathbf{Z}_{2^k}$ (Cramer et al.)[3], YTP-SS refers to n^2 applications of (Dumas et al., 2017a, Algorithm 15); MP-PDP refers the relaxation and improvement of this algorithm to the current setting; MP-SW refers to our implementation of MP-SW using BaseCase as a basecase with threshold set to $n = 56$. The Naccache-Stern cryptosystem is set with public keys of size 2048 bits and message space of 224 bits (using 14 primes of 16 bits).

Please note that, while MP-PDP and MP-SW share the same security model, YTP-SS and SPD$\mathbf{Z}_{2^k}$ achieve better security: malicious adversaries over insecure channels. Also, SPD$\mathbf{Z}_{2^k}$ uses a different approach based on oblivious transfer and secret sharing. However, as they were the only state of the art implementations available, we still chose to include them in our comparisons.

Figure 4.5 presents the volume of communication performed by these four variants. Communication-wise, for $n = 100$ players, MP-SW is 4% cheaper than MP-PDP (271 vs. 261 MB), but becomes 24% cheaper for $n = 400$ (15.3 vs. 11.7 GB) and up to 27.8% for $n = 528$ (35.2 vs. 25.4 GB). Note that the cross-over point of $n = 96$ between MP-PDP and MP-SW is confirmed experimentally. For SPD$\mathbf{Z}_{2^k}$, computations were performed for small matrices only because of computational power requirements: on a workstation with 16 GB of RAM and an Intel i5-7300U @2.60GHz, computations stalled for any matrices larger than 37×37.

To reach this communication improvement, the price to pay is that of some computational slowdown, as shown in Table 4.2

Finally, as there is same order of magnitude for the computational cost and the communication cost, communications should be largely dominant. Therefore, the improvement in communication volume is the one that matters.

[3] https://github.com/bristolcrypto/SPDZ-2

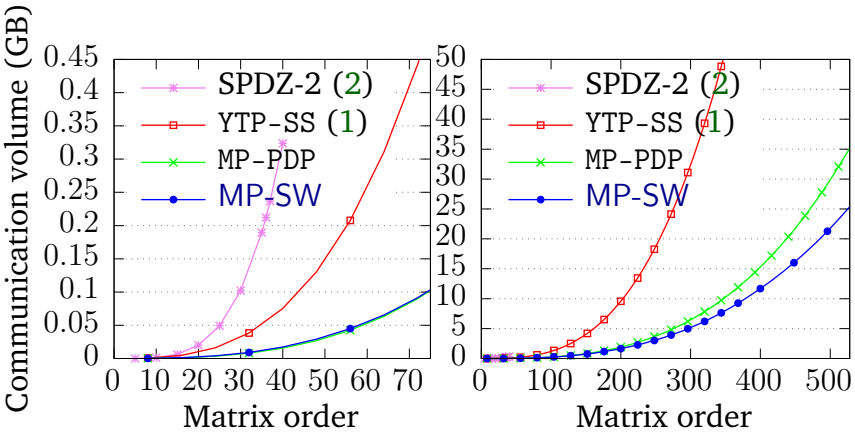

Figure 4.5: Comparing communication volume for multiparty matrix multiplications

Table 4.2: Computation time (in s) per player of Multiparty Strassen-Winograd MP-SW compared to MP-PDP on an Intel Xeon E7-8860 2.2Ghz.

Key size	1024			2048		
n	16	32	64	16	32	64
MP-PDP	0.58	2.68	11.01	4.54	18.05	69.80
MP-SW	2.87	6.19	13.27	23.63	49.22	196.24

4.5 Variant of MP-SW using proxy re-encryption

In this section we show how the use of proxy re-encryption schemes can improve the communication cost of MP-SW. Proxy re-encryption, abriged proxy in the rest of this document, are cryptosystems which allow to convert a ciphertext for one key into the same ciphertext but for another key. Formally introduced by Blaze et al. in (Blaze et al., 1998), the initial definition of a proxy involved the use of a trusted third party which is entrusted with a so-called *proxy key* allowing it to perform the re-encryption. While the original proxy scheme (Blaze et al., 1998, Section 3) was based on ElGamal encryption, techniques evolved as new needs arose, and modern proxy protocols with more properties (for instance the unidirectionality of the proxy function or the resistance to collusion) were developed (see for instance (Ateniese et al., 2006), (Green and Ateniese, 2007), (Ivan and Dodis, 2003), (Hohenberger et al., 2007)). However, these new approaches are usually based on expensive cryptographic tools, as pairings.

The main tool we use for all the recursive phases of MP-SW, MP-Copy is actually a proxy scheme in disguise: (reusing the naming conventions of MP-Copy), its goal is to recipher an element owned by Bob and protected with Dan's key to Charlie's key and send it to Alice. In order to save some communications, we propose to use proxy re-encryption instead of the masking and reciphering technique proposed in MP-Copy.

Given our security model, we can use one of the less expensive proxy solutions avail-

able and we will hence use the aforementioned Blaze et al. proxy scheme. However, the native ElGamal cryptosystem used in this scheme requires that ciphered elements remain small in order to be deciphered. As we extensively use random masking techniques, particularly in our BaseCase protocol, ensuring this requirement is not possible. While picking small masks is an option, thus limiting the growth of ciphered elements, it would however compromise the security of our protocol, as players would be able to tackle discrete logarithm computations on these new small ciphertexts.

We instead propose to rely on a variant by Castagnos and Laguillaumie (Castagnos and Laguillaumie, 2015). This cipher is a variant of the homomorphically additive ElGamal, which circumvents the main drawback of the vanilla additive ElGamal (having to solve a discrete logarithm to perform a decryption) by taking the plaintext in a subgroup where the discrete logarithm problem is less expensive to solve. This variant also allows unbounded additive homomorphic operations.

Castagnos-Laguillaumie cryptosystem sets up as follows:

Setup$(1^\lambda, 1^\mu)$: let μ be an integer, and B, n, p, s four integers such that s in an λ-bit integer, p is a μ-bit integer, p and s are coprime, $n = p \times s$ and B is an upper bound for s. Let G be a cyclic group of order n and generator g, $F \subset G$ a group of order p and generator f. Pick $x \xleftarrow{\$} \{0, \dots, Bp-1\}$ and compute $h = g^x$. The public key is $PK = (B, p, g, h, f)$ and the secret key is $SK = x$.

Encrypt$_{PK}(m)$: for $m \in \mathbf{Z}_p$, pick $r \xleftarrow{\$} \{0, \dots, Bp-1\}$ and compute $c_1 = g^r, c_2 = f^m h^r$. The ciphered message is $c = (c_1, c_2)$.

Decrypt$_{SK}(c)$: Compute $M = c_2/c_1^x$, which gives $M = f^m$. Use an algorithm

$\texttt{Solve}(p, g, f, G, F, M)$

to compute the discrete logarithm f^m and recover the message m.

This cryptosystem supports the same homomorphic operations as Naccache-Stern and thus can be used in MP-SW. The interested reader may refer to (Castagnos and Laguillaumie, 2015, Section 3) for details on G and F groups and on the `Solve` algorithm. In Blaze et al. scheme, proxy keys are a quotient of players' secret keys. While it is possible to adapt Castagnos-Laguillaumie scheme to fit this specification, it does require to restrain the choice of secret keys to invertible elements, thus modifying the security of the scheme. Instead, we propose to create a proxy key from a difference of players' secret keys: given two players, A with secret key $sk_A$and B with secret key sk_B, the key enabling recipher from A to B, $\Pi_{A\to B}$ is $sk_B - sk_A$.

This allows to use an unmodified version of Castagnos-Laguillaumie: with the same notations as above, a ciphertext using A's key would be $(c_1, c_2) = (g_A^r, f_A^m g_A^{rx_A})$. Then, reciphering this cipher for B requires to perform: $c_2 \leftarrow c_2 \times c_1^{\Pi_{A\to B}} = f_A^m g_A^{rx_A} \times g_A^{r(x_B - x_A)} = f_A^m g_A^{rx_B}$. As this multiparty use of Castagnos-Laguillaumie would require to remember which player first ciphered an element in order to know which subgroup and generator to use for discrete logarithm computation, we propose that all players share the same subgroup and generator, respectively denoted as F and f.

4.5.1 Description of the new protocol

In the original Blaze et al. proxy scheme, proxy entities were trusted third parties. This setting is not desirable here, as it would mean extra communications in order to send and recover data from the proxy, and as we chose to not rely on trusted entities anyway. The data repartition we have in Strassen-Winograd, alongside the regularity of the algorithm allows us to predict the list of players for which a given player will need to perform copy operations for throughout the protocol. This means we can ensure that there will always be a player able to take the role of a proxy without compromising the security, and we thus propose that players play the role of proxies instead of invoking trusted third parties. We then need a new protocol which allows players to craft their proxy keys in a secure way: players cannot learn any information on the secret keys of the others' secret keys. We consider three players Alice, Bob and Charlie, where Charlie needs to learn the proxy key $\Pi_{A\to B}$, Bob first ciphers his private key with Charlie's public key and sends the result to Alice. Using homomorphic properties of Castagnos-Laguillaumie cryptosystem, she computes a ciphered $sk_B - sk_A$ and sends it to Charlie, who then deciphers it and only learns the proxy key. ProxyKeyGen gives a formal description of this proxy key generation.

Protocol 4.9: ProxyKeyGen

Input: Three parties, further denoted as Alice, Bob and Charlie. They both know their own private key and the public keys of all the parties involved.
Output: Charlie learns the proxy key allowing to transform a cipher for Alice into a cipher for Bob.
Goal: Perform the difference between Bob's secret key and Alice's secret key.

1. **Cipher Bob's secret key:** Bob performs `CastagnosLaguillaumieEncrypt`(pk_C, sk_B) and stores the result in $\{sk_B\}_{sk_C}$.
2. **Communication:** Bob sends $\{sk_B\}_{sk_C}$ to Alice.
3. **Cipher Alice's secret key:** Alice performs `CastagnosLaguillaumieEncrypt`(pk_C, -sk_A) and stores the result in $\{sk_A\}_{sk_C}$.
4. **Proxy key generation:** Alice performs $\{sk_B\}_{sk_C} \times \{-sk_A\}_{sk_C}$.
5. **Communication:** Alice sends $\{sk_B - sk_A\}_{sk_C}$ to Charlie.
6. **Decipher:** Charlie performs `CastagnosLaguillaumieDecrypt`(sk_C, $\{sk_B - sk_A\}_{sk_C}$) and learns $sk_B - sk_A$.

Theorem 4.5.1. *ProxyKeyGen is secure against one semi-honest adversary.*

Proof. Players Alice and Bob only see ciphered elements so they do not learn anything from the execution. As the output of the protocol is a difference of values which are

unknown to Charlie, the player receiving them, it is computationally indistinguishable from a simulation in which inputs are ciphered random values. □

According to the definition of the location and key sequences and the schedule of MP-SW, a given player $P_{L[i]}$ will always own elements ciphered with the key of $P_{K[i]}$, hence always having to recipher elements from this player to another. The "destination" player changes at each recursive level, meaning that for an instance of MP-SW with k recursive levels, our player $P_{L[i]}$ will have to get k different proxy keys. More specifically, at a given recursive level r, where input matrices have size $s = n/(2^{r-1})$, $P_{L[i]}$ has to relocate the elements they own for $P_{L[1+(i-1+s/2 \mod s)]}$, meaning they have to recipher it for $P_{K[1+(i-1+s/2 \mod s)]}$. This allows to predict which proxy keys will be needed throughout the whole protocol, and to generate them during the setup phase.

From this information, the setup phase is now modified and the new setup is proposed in SW-Setup-Proxy.

Protocol 4.10: SW-Setup-Proxy

Input: Two $N \times N$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ over $\mathcal{M}$, where $N = b2^{\ell}$, such that party P_i knows the i-th row of $\boldsymbol{A}$ and the i-th row $\boldsymbol{B}$ for all $i \in \{1..N\}$. A location and a key sequence $L \in \{1..N\}^N$ and $K \in \{1..N\}^N$ such that (L, K, L, K) form an ℓ-recursive data layout, following Definition 4.1.2. All parties know a security parameter λ, an integer μ and a group F with generator f.
Output: For all $i \in \{1..N\}$, party $P_{L[i]}$ learns vectors $\{a_{i,*}\}_{K[i]}$ and $\{b_{i,*}\}_{K[i]}$, learns the public key of every other party and the required proxy keys.
Goal: Generate key pairs for each party, cipher and distribute input matrices according to their respective location and key sequences.

1. **Key generation:** for all $i \in \{1..N\}$, each party P_i locally executes `CastganosLaguillaumieSetup`$(1^{\lambda}, 1^{\mu})$ to generate a pair of keys (pk_i, sk_i). The group generator for the group in which the message will be mapped to has to be f.
2. **Broadcast keys:** for all $i \in \{1..N\}$, party P_i broadcasts its public key pk_i.
3. **Proxy keys generation:** for all $i \in \{1..N\}$, for all $j \in \{1..\ell\}$, run ProxyKey-Gen $(P_{K[i]}, P_{K[1+(i-1+N/2^j \mod N/2^{j+1})]}, P_{L[i]})$.
4. **Cipher inputs:** for all $i \in \{1..N\}$, for all $j \in [n]$, party P_i locally performs `CastagnosLaguillaumieEncrypt`$(pk_{K[i]}, a_{ij})$ and stores the result as a new vector $\{a_{i,*}\}_{K[i]}$. It does the exact same operation with $b_{i,*}$ to get $\{b_{i,*}\}_{K[i]}$.
5. **Distribute rows:**
   a) **Rows of $\boldsymbol{A}$:** for all $i \in \{1..N\}$, party P_i sends $\{a_{i,*}\}_{K[i]}$ to party $P_{L[i]}$.
   b) **Rows of $\boldsymbol{B}$:** for all $i \in \{1..N\}$, party P_i sends $\{b_{i,*}\}_{K[i]}$ to party $P_{L[i]}$.

The call to MP-Copy is simply replaced by a call to the proxy re-encryption function, and the reciphered value is forwarded to the appropriate player, as shown in MP-Copy-Proxy. This new copy protocol, MP-Copy-Proxy is a simple application of the secure

proxy scheme of Blaze et al. and a communication over a secure channel. It is thus secure by security of Blaze et al. scheme.

Protocol 4.11: MP-Copy-Proxy

Input: Four parties, Alice, Bob, Charlie and Dan. Bob knows a ciphered element $\{x\}_D \in \mathcal{C}_D$ (for $x \in \mathcal{M}$), ciphered using Dan's public key and a proxy key $\Pi_{D\to C}$.
Output: Alice learns the element $\{x\}_C$, ciphered using Charlie's public key.
Goal: Recipher from Dan to Charlie and transfer from Bob to Alice.

1. **Recipher:** Bob applies $\texttt{ProxyRecipher}(\Pi_{D\to C}, \{x\}_D)$ and gets $\{x\}_C$.
2. **Communication:** Bob sends $\{x\}_C$ to Alice.

The other steps in MP-SW remain unchanged.

4.5.2 Communication cost analysis

We now perform the same analysis as the one from Section 4.3.3 for the proxy-based version of MP-SW.

Lemma 4.5.2. *The total communication cost of a recursive level of the proxy-based version of MP-SW following the schedule defined MP-SW, Step 2 is* $6\left(\frac{n}{2}\right)^2$ *communications.*

Proof. The only communication are that of the 6 calls to MP-Copy-Proxy, each accounting for $(n/2)^2$ communication.

□

This gives us a new recurrence relation for the communication complexity:

$$\begin{cases} C(n) &= 7C\left(\frac{n}{2}\right) + 6\left(\frac{n}{2}\right)^2 \text{ for } n > b \\ C(b) &= b^3 + 3b^2 + b \text{ for the base case} \end{cases} \tag{4.5.1}$$

Using this new recurrence relation and the base cast cost, we can compute the new recursion threshold, which gives us $b = 32$.

We then compare the cost of proxy-based MP-SW to MP-PDP, with $C_{\text{MP-PDP}}(n) = n^3 + n(n-1)$. The new initialisation step SW-Setup-Proxy costs $C_{\text{init}} = 2n^2 + 2n\ell$, with ℓ the number of recursive levels, and the finalisation step SW-Finalise costs $C_{\text{final}} = n^2$. The crossover point where this variant improves over MP-PDP in communication cost is obtained by solving the equation: $C(n) + 3n^2 + 2n \leqslant n^3 + n(n-1)$ which yields $n > 71$, with one recursive call. This means that for any instance of dimension larger than 71, the proposed MP-SW algorithm has a better communication cost than MP-PDP, which also makes it better than the proxyless MP-SW.

4.5.3 Comparisons between fully and semi homomorphic solutions

In the introduction of this chapter, we justified setting aside fully homomorphic cryptography based on the sheer cost of their arithmetic operations. However, the protocols we developed need to perform a large amount of additional cryptographic operations, which would not be required with fully homomorphic tools. Hence, we need to verify that this argument still holds. One bottleneck of fully homomorphic cryptography is the multiplicative depth of the algorithm: if it is too high, costly bootstrapping operations have to be performed in order to ensure proper decryption. As matrix multiplication algorithms have a multiplicative depth of 1, they seem to be quite favourable algorithms for such cryptographic solutions, which further calls for a refined analysis on the cost of fully homomorphic cryptography in our setting. For Strassen-Winograd, ciphered elements are only used in product operations when computing dot products in the base case step, and there are no accumulations, as the result of multiplications performed for dot products are not reused for other multiplications. Thus, there is no need to perform bootstrapping operations with fully homomorphic cryptosystems with Strassen-Winograd algorithm.

Thus, we propose to estimate the total runtime of a run of Strassen-Winograd for SEAL and HElib. We model this run in a slightly different setting than the one considered in MP-SW. We will indeed assume there exists a trusted third party, to which players will send their ciphered inputs, and from which players will receive their output. In this model, the protocol only consists in running Strassen-Winograd algorithm homomorphically, which amounts for the exact same amount of arithmetic operations as the textbook version, and the initial and final encryptions and decryptions. This will allow us to evaluate the practical cost of the fully homomorphic libraries under consideration.

Our estimations are based on an input matrix size of $N = 224$, with a threshold of 56, as given in Section 4.3.3. This means two recursive levels will be performed before the base case. Table 4.4 gives an estimation of the execution time per operation, and of the total execution for MP-SW (proxyless vanilla version and the improved proxy-based version) and for HElib and SEAL. We computed the amount of times each operation has to be performed to complete an execution of Strassen-Winograd on both models and used the timings per operations that were given in Table 4.1.

In order to get an estimation of the runtime of the proxy variant of MP-SW, we had to get timings for elementary cryptographic and arithmetic operations in Castagnos-Laguillaumie. We chose to base these timings on Paillier cryptosystem, as Castagnos-Laguillaumie behaves the same, mostly using exponentiations modulo n^2, where n is the public modulus obtained from the key generation phase of the cryptosystem. Table 4.3 presents a comparison between the modular operations used to perform an encryption, a decryption, an homomorphic addition and a multiplications in both cryptosystems. We chose to display only the most expensive operation: Paillier, for instance, also requires to perform a multiplication modulo n in the encryption step, but its cost is negligible compared to the cost of exponentiations modulo n^2. The ratio column shows the factor to apply to the timings measured for Paillier in Table 4.1 to

	Paillier	Castagnos-Laguillaumie	Ratio
Encryption	2 exponentiations	3 exponentiations	1.5
Decryption	1 exponentiation	1 small discrete logarithm	1
Addition	1 multiplication	2 multiplications	2
Multiplication	1 exponentiation	2 exponentiations	2

Table 4.3: Comparison of cryptographic and arithmetic operations in Paillier and Castagnos-Laguillaumie

get the estimated timings for the same operation in Castagnos-Laguillaumie.

A complete run of Strassen-Winograd requires to perform N^2 encryptions, N decryptions for the input/output data, $7^3 \times (\frac{N}{4})^3 + 15 \times (\frac{N}{2})^2 + 7 \times 15 \times (\frac{N}{4})^2$ additions and $7^2 \times (\frac{N}{4})^3$ multiplications (with 2 recursive levels, using the formulae from Section 4.3.3. MP-SW requires even more operations: $7^2 \times (\frac{N}{4})^2$ encryptions and decryptions from the base case (for the masking operations) and $6 \times (\frac{N}{2})^2 + 6 \times 7 \times (\frac{N}{4})^2$ extra calls to MP-Copy (with 3 encryptions, 1 decryption and 2 additions per call). The proxy variant of MP-SW modifies this operation count once again: the cost of copy is reduced to one application of `ProxyRecipher` per call, which is equivalent to performing two modular exponentiations. Performing Decryptions and Additions is no longer needed, but there is an extra (though negligible) cost in the call to SW-Setup-Proxy, with four calls per player as our example runs over two recursive levels.

Table 4.1 shows that decryption is by far the most expensive operation for Naccache-Stern. In MP-SW, the amount of decryptions is equivalent to the amount of multiplications, $O(N^{2.8})$, which is not the case for fully homomorphic encryption (where it is $O(N^2)$). Table 4.4 shows that our solution remains cheaper: MP-SW would take roughly 7 hours to perform over messages of 64-bit at $\lambda = 130$ against 88 hours for HElib. At 20-bit, MP-SW would roughly take 2 hours against the 190 hours of SEAL. Our proxy variant of MP-SW runs roughly 3 times slower than the vanilla version for 20-bit messages, because of the field size required for Castagnos-Laguillaumie. However, runtimes are equivalent with 64-bits messages, and given the rapid growth of Naccache-Stern operation cost, especially the discrete logarithms required for deciphering elements, the proxy variant should perform better than the vanilla one on larger message size.

While considering these timings, it is important to remember that we compare elementary arithmetic operations on both sides. An interesting extension on this estimation would be to consider linear algebra optimizations (vector and matrix products) for semi- and fully-homomorphic solutions.

Finally, in term of communication, it is worth noting that fully homomorphic ciphertexts are much more heavy than Naccache-Stern's: at $\lambda = 112$, roughly 264kb for SEAL, 360kb for HElib against only 5.14kb for Naccache-Stern. For matrices of order 224, sending and recovering the matrices to the trusted third party would then cost $3 \times 224^2 \times 264 = 5.1$GB for SEAL and 6.9GB for HElib against the 2.4GB of MP-SW.

Operation			Timings			
Nature	Number	Msg bit size	MP-SW		HElib	SEAL
			Vanilla	Proxy		
Enc	100352	20	7	2910.2	–	197.6
		64	14.1	3030.6	9493.3	–
Dec	50176	20	757.7	963.4	–	361.3
		64	3090.8	973.4	2333.2	–
Add	9122624	20	9.1	182.5	–	2736.8
		64	9.1	182.5	56560.3	–
Mul	8605184	20	172.1	2925.8	–	678088.5
		64	430.3	8605.2	249550.3	–
EncBaseCase	153664	20	10.8	4456.3	–	–
		64	21.5	4640.7	–	–
DecBaseCase	153664	20	2320.3	2950.3	–	–
		64	9435	2981.1	–	–
EncCopyVanilla	620928	20	43.5	–	–	–
		64	87	–	–	–
2ExpCopyProxy	206976	20	–	3994.6	–	–
		64	–	4160.2	–	–
DecCopy	206976	20	3125.4	–	–	–
		64	12748	–	–	–
AddCopy	413952	20	0.4	–	–	–
		64	0.4	–	–	–
ProxySetupEnc	1792	20	–	52	–	–
		64	–	54.1	–	–
ProxySetupAdd	896	20	–	0.02	–	–
		64	–	0.02	–	–
ProxySetupDec	896	20	–	17.2	–	–
		64	–	17.4	–	–
Total	–	20	6446.3	18452.3	–	681384.2
		64	25836.2	24645.2	317937.1	–

Table 4.4: Timings comparison in seconds for matrices of order 224, threshold 56 and $\lambda = 130$. Number is the number of times the relevant operation is performed during an execution of the protocol.

Conclusion

We have presented in this chapter a novel secure multiparty matrix multiplication where each player owns one row of the different matrices. For this we use Strassen-Winograd algorithm and reduce for the first time the total communication volume from $O(N^3)$ to $O(N^{\log_2(7)})$. The improvement in communication cost over state of the art algorithms takes effect for dimensions as small as 96, verified in practice, or 72 with the proxy-based variant.

In this matrix multiplication protocol, it was necessary to protect both multiplicands, as Strassen-Winograd algorithm requires to perform pre-additions between elements of one input matrix at a time. This new constraint required to have a secure routine to move data and change its encryption key: this was initially done with encryption/decryption and random masking sequences, which was later improved with proxy re-encryption techniques.

The version of Strassen-Winograd we presented here is secure against semi-honest adversaries. However, as many of its building blocks have a stronger security level anyway, it would be interesting to see if it is possible to increase the security of the whole MP-SW protocol and how it would impact its performance.

Even if this chapter is a about improving the communication cost while preserving security, several arithmetic cost improvements could be envisioned. A possibility would be to replace the Naccache-Stern by a faster cryptosystem. The difficulty is to be able to combine the masking schemes with the homomorphic encryption.

This chapter proposed a new protocol for matrix multiplication. Matrix multiplication is a tool used in many applications and computations, as linear system solving, $\boldsymbol{Ax} = \boldsymbol{b}$. Developing an efficient multiparty protocol for this problem with the same data layout (each player knows one row of $\boldsymbol{A}$, one coefficient of $\boldsymbol{b}$ and learns one coefficient of $\boldsymbol{x}$) has interesting real world applications: multiparty polynomial interpolation, for instance, where each player knows one interpolation point. We explored one idea for this application, based on building a protocol with entropic security for the LU decomposition of matrix. In this protocol, players would build a shared $\boldsymbol{U}$ matrix and keep their row of $\boldsymbol{L}$ secret, thus protecting their input values. This protocol would then straightforwardly allow for system solving, as it would just be a run of the LU protocol on matrix $\boldsymbol{A}$ augmented with $\boldsymbol{b}$. However, with this idea, players input are in the worst case only protected by one coefficient – which is especially dangerous for polynomial interpolation as the input matrix is Vandermonde, meaning that leaking information about one coefficient leaks information about the entire row. It would then be worth exploring other directions, as for instance designing a protocol for secure triangular system solving which could be used to securely compute the LU decomposition.

Conclusion

In this thesis, we focused on two aspects of outsourced computation in exact linear algebra: output verification, and data confidentiality. We proposed new protocols to verify rank-related invariants: the matrix rank, the row and column rank profiles and the rank profile matrix. This lead us to design a verification protocol for the generic rank profile-ness of a matrix, which we then used to develop new protocols for problem based on triangular decompositions: the determinant and the signature. We then designed verification protocols for the classical properties of polynomial matrices, either specific to modules or carrying to vector spaces. Our protocols for modules, are, to the best of our knowledge, the first verification protocols for matrix properties on such algebraic structures. Finally, we proposed a new multi-party protocol for matrix multiplication based on Strassen-Winograd algorithm, improving the state of the art for the important communication volume cost of such protocols. It is also, as far as we are aware of, the first multi-party algorithm with a recursive structure.

These results are either maiden contributions (verification protocols for modules, multi-party protocols with a recursive structure) or crucial improvements (rank-related protocols) to the field of algorithm-based outsourced computation for exact linear algebra. Short-term perspectives and improvements to our results have been given in the conclusions to the different chapters. The two major ones are the design of a Smith form verification protocol for polynomial matrices and a secure multi-party protocol for the LU decomposition. They also allow us to envision new significant, longer-term perspectives that we propose in what follows.

Verification protocols for numerical linear algebra. The verification protocols we proposed in this thesis are based on verification checks in which the output of the check only depends on the Prover's honesty: if a check is false, then the Prover cheated. This restrictive definition of the Verifier's checks makes it harder to design protocols for numerical linear algebra: defining equality checks in this field is a complex task because of the rounding errors in computations, which might cause a check to fail with an honest Prover. To design such protocols, we could focus on new definitions of completeness, based for instance on stability analysis.

Error correction. In this thesis, we focused on verifying outsourced computations: the goal was to design protocols able to detect whether a given result was correct. An extension to this work would be to also fix the errors in a wrong result in order to recover the actual output to the outsourced problem. While coding theory seems to be the perfect candidate for this task, the redundancy in encoding is not efficient

communication-wise, and error-correcting codes are designed to deal with uniform distribution of errors, which does not fit our model. It is possible to design such protocols, by for instance using the redundancy in algebraic relations between the public input and the output. There have been advances on designing error-correction protocols for outsourced computations in exact linear algebra in the past few years, with for instance Roche (2018) on matrix multiplication (and inverse) or Dumas et al. (2019b) for LU decomposition of generic rank profile matrices. Some difficult problems remain: a straightforward improvement to these works would be to find an error-correction protocol for the LU decomposition with pivoting, leading to the correction of the LUP or the PLUQ decomposition. Designing such protocols for the properties we focused on in this thesis, as the rank profile matrix computation would be a natural extension to our work.

A slightly different focus could be to incorporate Monte-Carlo probabilistic algorithms for the Prover side of our verification protocols. These algorithms can silently fail, meaning they can return a wrong result without notice. Our verification protocols already are Monte-Carlo for the Verifier: a dishonest Prover can sometimes not be detected (this is the probabilistic soundness), but an honest Prover will never be rejected (perfect completeness). Because of the nature of our verification checks, which assume that failures are due to the Prover, we naturally excluded such algorithms when designing our protocols. Being able to fix the errors that might come from the use of Monte-Carlo algorithms would allow us to actually use these algorithms, leading to protocols with faster computation time for the Prover. As the verification certificate from (Dumas et al., 2016, Section 4) allows the use of Monte-Carlo algorithms for the Prover, their approach should also be investigated.

Verify the negation of a statement. Our verification protocols frequently prove affirmative statements, for instance that there exists a triangular matrix such T such that $AT = B$ or that a given vector is in the row space of a polynomial matrix. Designing protocols that are able to verify negative statements would be a major improvement to the family of algorithm-based verification protocols. These are often quite harder to prove than the affirmative ones, as they are more generic: the Prover can no longer provide a simple single answer (such as the constraint projection of the triangular matrix) to the Verifier. A very ambitious question would be to investigate how to get a verification protocol for a the negation of a statement from the protocol for the affirmative statement. A starting point could be to develop several negative statements verification protocols on specific problems, –such as the ones from this thesis – to investigate how they relate to their affirmative statements counterparts and whether these relations have similarities. Farkas certificate of inconsistency (Farkas, 1902) could be a useful tool for the verification of such negative statements.

Existence of fundamental building blocks for verification protocols. The verification protocols we proposed in this thesis heavily rely on a few sub-protocols: the step-by-step data exchange of TriangularEquivalence is a central tool for the protocols

in Chapter 2 while RowSpaceMembership is heavily used by the protocols from Chapter 3. These two sub-protocols can be seen as building block for verification protocols: central elements into which problems could be reduced. Matrix multiplication (with Freivalds' check) and linear system solve verification also appear to be such building blocks, as they are heavily used, not only by some of our protocols but also by many of the others that were given in Chapter 1. We then wonder the scope of these four building blocks: which verification protocols could be reduced to one of them, and amongst the protocols that we cannot reduce to our building blocks, could we find new building blocks to cover such protocols ? We would need to develop more algorithm-based verification protocols in exact linear algebra, in order to look for these building blocks. These fundamental building blocks are also extremely useful for the practicality of verification protocols: they would allow us to take one step further towards real-world applications, with the development of a library for outsourced computation verification containing highly efficient implementation of these fundamental protocols.

New data layout for MP-SW. The multiparty protocol version of Strassen-Winograd algorithm we proposed in Chapter 4 requires a very specific data layout in which players knows one full row of both input matrices. This layout is very well suited to the problem, as its high regularity allowed us to take full advantage of the algorithm's structure. A data layout often used in multi-party computation setting is the additive sharing of the input (Cramer et al., 2015, Chapter 9) (Dagdelen and Venturi, 2015): each party knows one share of the input and the full input can be obtained by adding all the shares owned by the parties. Designing a multi-party version of Strassen-Winograd for this layout would allow us to open our work to a wider array of practical applications. However, this would require some heavy work on the foundations of our protocol: with this layout, it is no longer possible to perform local homomorphic operations during the recursive steps, and our other subroutines would need to be adapted as well. Our main result was to obtain the same communication volume complexity as the textbook Strassen-Winograd's algorithm computational complexity ($O(n^{2.8})$). However, reaching the same cost with the additive setting seems much more difficult, as performing arithmetic operations would require to communicate data between more than two players, thus increasing the overall communication volume.

Bibliography

[Citing pages are listed after each reference.]

IBM HElib (release 1.3), August 2019. URL github.com/homenc/HElib. [Page 116.]

Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Pearson, 1974. ISBN 0201000296, 9780201000290. doi:10.1002/zamm.19790590233. [Page 116.]

William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT '01, pages 119–135, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42070-3. doi:10.1007/3-540-44987-6_8. [Page 31.]

Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, 2018. [Page 31.]

Artak Amirbekyan and Vladimir Estivill-Castro. A new efficient privacy-preserving scalar product protocol. In *Proceedings of the Sixth Australasian Conference on Data Mining and Analytics - Volume 70*, AusDM '07, pages 209–214, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc. ISBN 978-1-920682-51-4. doi:10.5555/1378245.1378274. URL https://dl.acm.org/doi/10.5555/1378245.1378274. [Page 34.]

Sanjeev Arora and Shmuel Safra. Approximating clique is np- complete. In *FOCS 1992*, 1992. [Page 18.]

Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45:501–555, 1998. ISSN 0004-5411. doi:10.1145/278298.278306. [Page 18.]

C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29:208–210, 2006. ISSN 0018-9448. doi:10.1109/TIT.1983.1056651. [Page 30.]

Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information Systems and Security*, 9:1–30, 2006. ISSN 1094-9224. doi:10.1145/1127345.1127346. [Page 137.]

László Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 421–429, New York, NY, USA, 1985. Association for Computing Machinery. ISBN 0-89791-151-2. doi:10.1145/22145.22192. [Page 17.]

László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36: 254–276, 1988. ISSN 0022-0000. doi:10.1016/0022-0000(88)90028-1. [Page 18.]

László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, New York, NY, USA, 1991. Association for Computing Machinery. ISBN 0897913973. doi:10.1145/103418.103428. [Page 18.]

Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology — CRYPTO '91*, pages 420–432, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-46766-3. [Page 31.]

Bernhard Beckermann and George Labahn. A uniform approach for the fast computation of matrix-type Padé approximants. *SIAM Journal of Matrix Analysis and Applications*, 15:804–823, 1994. ISSN 0895-4798. doi:10.1137/S0895479892230031. [Page 73.]

Bernhard Beckermann and George Labahn. Fraction-free computation of matrix rational interpolants and matrix gcds. *SIAM Journal of Matrix Analysis and Applications*, 22:114–144, 2000. ISSN 0895-4798. doi:10.1137/S0895479897326912. [Page 110.]

Bernhard Beckermann, George Labahn, and Gilles Villard. Normal forms for general polynomial matrices. *Journal of Symbolic Computation*, 41:708–737, 2006. ISSN 0747-7171. doi:10.1016/j.jsc.2006.02.001. [Page 102.]

Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT'11, pages 169–188, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-20464-7. doi:10.5555/2008684.2008699. [Page 32.]

Dario A. Bini and Victor Pan. *Polynomial and Matrix Computations (Vol. 1) Fundamental Algorithms*. Progress in Theoretical Computer Science. Birkhäuser Basel, 1994. ISBN 978-3-7643-3786-5. doi:10.1007/978-1-4612-0265-3. [Page 95.]

G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Monval, NJ, USA, 1979. AFIPS Press. doi:10.1109/AFIPS.1979.98. [Page 30.]

Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Proceedings of EUROCRYPT'98 in Advances in Cryptology*, pages 127–144, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-69795-4. doi:10.1007/BFb0054122. [Page 137.]

Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security: 13th International Conference*, FC 2009, pages 325–343, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-03549-4. doi:10.1007/978-3-642-03549-4_20. [Page 13.]

Alin Bostan and Éric Schost. Polynomial evaluation and interpolation on special sets of points. *Journal of Complexity*, 21:420–446, 2005. ISSN 0885-064X. doi:10.1016/j.jco.2004.09.009. [Page 73.]

N. Bourbaki. *Commutative Algebra*. Elements of Mathematics. Addison-Wesley, 1972. ISBN 978-3-540-64239-8. [Page 103.]

Brice Boyer and Jean-Guillaume Dumas. Matrix multiplication over word-size modular rings using approximate formulas. *ACM Transactions on Mathematical Software*, 42, 2016. ISSN 0098-3500. doi:10.1145/2829947. [Page 116.]

Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology*, CRYPTO 2012, page 868–886, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 9783642320088. doi:10.1007/978-3-642-32009-5_50. [Page 116.]

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311151. doi:10.1145/2090236.2090262. [Page 116.]

Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 234–238, London, UK, UK, 1987. Springer-Verlag. ISBN 0-387-18047-8. doi:10.1007/3-540-47721-7_17. [Page 31.]

James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation*, 31:163–179, 1977. ISSN 0025-5718. doi:10.1090/S0025-5718-1977-0428694-0. [Page 65.]

David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28:693–701, 1991. ISSN 0001-5903. doi:10.1007/BF01178683. [Page 74.]

Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from ddh. In *Topics in Cryptology — CT-RSA 2015*, pages 487–505, Cham, 2015. Springer International Publishing. ISBN 978-3-319-16715-2. doi:10.1007/978-3-319-16715-2_26. [Page 138.]

Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *Journal of the ACM*, 60:1–25, 2013. ISSN 0004-5411. doi:10.1145/2528404. [Page 28.]

Seung Geol Choi, Kyung-Wook Hwang, Jonathan Katz, Tal Malkin, and Dan Rubenstein. Secure multi-party computation of boolean circuits with applications to privacy in on-line marketplaces. In *Topics in Cryptology – The cryptographers' track at the RSA conference*, CT-RSA 2012, pages 416–432, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-27954-6. doi:10.1007/978-3-642-27954-6_26. [Page 33.]

Donc Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9:251 – 280, 1990. ISSN 0747-7171. doi:10.1016/S0747-7171(08)80013-2. [Page 15.]

Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 90–112, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 978-1-4503-1115-1. doi:10.1145/2090236.2090245. [Page 22.]

Ana Costache and Nigel P. Smart. Which ring based somewhat homomorphic encryption scheme is best? In *Topics in Cryptology – The cryptographers' track at the RSA conference*, CT-RSA 2016, pages 325–340, Cham, 2016. Springer International Publishing. ISBN 978-3-319-29485-8. doi:10.1007/978-3-319-29485-8_19. [Page 31.]

Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur. Geppetto: Versatile verifiable computation. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, SP '15, page 253–270, USA, 2015. IEEE Computer Society. ISBN 9781467369497. doi:10.1109/SP.2015.23. [Page 23.]

Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPDZ$_{2^k}$: Efficient mpc mod 2^k for dishonest majority. In *Proceedings of the 38nd*

Annual Cryptology Conference on Advances in Cryptology, CRYPTO 2018, pages 769–798, Berlin, Heidelberg. Springer-Verlag. doi:10.1007/978-3-319-96881-0_26. [Pages 136 and 137.]

Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. ISBN 1107043050, 9781107043053. doi:10.1017/CBO9781107337756. [Pages 116 and 149.]

Özgür Dagdelen and Daniele Venturi. A multi-party protocol for privacy-preserving cooperative linear systems of equations. In *Revised Selected Papers of the First International Conference on Cryptography and Information Security in the Balkans - Volume 9024*, BalkanCryptSec 2014, pages 161–172, New York, NY, USA, 2015. Springer-Verlag New York, Inc. ISBN 978-3-319-21355-2. doi:10.1007/978-3-319-21356-9_11. [Pages 35, 115, and 149.]

Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *Proceedings of the 10th Theory of Cryptography Conference on Theory of Cryptography*, TCC'13, pages 621–641, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 978-3-642-36593-5. doi:10.1007/978-3-642-36594-2_35. [Page 32.]

Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. In *Proceedings of the 2013 European Symposium on Research in Computer Security*, ESORICS 2013, pages 1–18, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40203-6. doi:10.1007/978-3-642-40203-6_1. [Page 32.]

Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *Proceedings of the 2017 International Conference on Advances in Cryptology*, CRYPTO 2017, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-63688-7_6. [Page 115.]

Ivan Bjerre Damgård. On sigma-protocols, 2010. URL `https://www.cs.au.dk/~ivan/Sigma.pdf`. [Page 19.]

Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7:193 – 195, 1978. ISSN 0020-0190. doi:10.1016/0020-0190(78)90067-4. [Pages 43, 62, and 76.]

Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *Proceedings of the 2015 NDSS Symposium*, NDSS 2015, 2015. doi:10.14722/ndss.2015.23113. [Page 115.]

Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983. ISSN 0018-9448. doi:10.1109/TIT.1983.1056650. [Page 29.]

Shlomi Dolev, Niv Gilboa, and Marina Kopeetsky. Computing multi-party trust privately: In o(n) time units sending one (possibly large) message at a time. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1460–1465, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 978-1-60558-639-7. doi:10.1145/1774088.1774401. [Page 34.]

Wenliang Du and Mikhail J. Atallah. Privacy-preserving cooperative statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, ACSAC '01, pages 102–110, 2001a. ISBN 0769514057. doi:10.1109/ACSAC.2001.991526. [Page 34.]

Wenliang Du and Mikhail J. Atallah. Privacy-preserving cooperative scientific computations. In *Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, CSFW '01, pages 273–273, Washington, DC, USA, 2001b. IEEE Computer Society. doi:10.1109/CSFW.2001.930152. [Page 35.]

Wenliang Du and Zhijun Zhan. A practical approach to solve secure multi-party computation problems. In *Proceedings of the 2002 Workshop on New Security Paradigms*, NSPW '02, page 127–135, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113598X. doi:10.1145/844102.844125. [Page 115.]

Wenliang Du, Shigang Chen, and Yunghsiang S. Han. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 4th SIAM International Conference on Data Mining*, pages 222–233, 2004. ISBN 978-0-89871-568-2. doi:10.1137/1.9781611972740.21. [Page 34.]

Jean-Guillaume Dumas. Proof-of-work certificates that can be efficiently computed in the cloud (invited talk). In *Computer Algebra in Scientific Computing*, pages 1–17, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99639-4. [Page 73.]

Jean-Guillaume Dumas and Hicham Hossayni. Matrix powers algorithms for trust evaluation in public-key infrastructures. In *Proceedings of the 2012 International Workshop on Security and Trust Management*, STM 2012, pages 129–144, 2013. ISBN 978-3-642-38004-4. doi:10.1007/978-3-642-38004-4_9. [Page 115.]

Jean-Guillaume Dumas and Erich Kaltofen. Essentially optimal interactive certificates in linear algebra. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 146–153, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325011. doi:10.1145/2608628.2608644. [Pages 15, 19, 24, 25, 28, 45, 52, 65, 78, 80, and 81.]

Jean-Guillaume Dumas, Pascal Giorgi, and Clément Pernet. Dense linear algebra over word-size prime fields: The FFLAS and FFPACK packages. *ACM Transactions on Mathematical Software*, 35, 2008. ISSN 0098-3500. doi:10.1145/1391989.1391992. [Page 116.]

Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Simultaneous computation of the row and column rank profiles. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ISSAC '13, pages 181–188, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 978-1-4503-2059-7. doi:10.1145/2465506.2465517. [Pages 39 and 42.]

Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Computing the rank profile matrix. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '15, pages 149–156, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 978-1-4503-3435-8. doi:10.1145/2755996.2756682. [Page 80.]

Jean-Guillaume Dumas, Erich Kaltofen, Emmanuel Thomé, and Gilles Villard. Linear time interactive certificates for the minimal polynomial and the determinant of a sparse matrix. In *Proceedings of the 41st International Symposium on Symbolic and Algebraic Computation*, ISSAC '16, pages 199–206, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4380-0. doi:10.1145/2930889.2930908. [Pages 15, 25, 27, 28, 64, 65, 82, and 148.]

Jean-Guillaume Dumas, Pascal Lafourcade, Jean-Baptiste Orfila, and Maxime Puys. Dual protocols for private multi-party matrix multiplication and trust computations. *Computers and Security*, 71:51–70, 2017a. ISSN 0167-4048. doi:10.1016/j.cose.2017.04.013. [Pages 34, 114, 115, 117, 118, 121, 122, 136, and 137.]

Jean-Guillaume Dumas, David Lucas, and Clément Pernet. Certificates for triangular equivalence and rank profiles. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '17, pages 133–140, New York, NY, USA, 2017b. Association for Computing Machinery. doi:10.1145/3087604.3087609. [Pages 15, 38, and 82.]

Jean-Guillaume Dumas, Clément Pernet, and Ziad Sultan. Fast computation of the rank profile matrix and the generalized Bruhat decomposition. *Journal of Symbolic Computation*, 83:187–210, 2017c. ISSN 0747-7171. doi:10.1016/j.jsc.2016.11.011. [Pages 38, 39, 41, 56, and 58.]

Jean-Guillaume Dumas, Pascal Lafourcade, Julio Lopez Fenner, David Lucas, Jean-Baptiste Orfila, Clément Pernet, and Maxime Puys. Secure multiparty matrix multiplication based on strassen-winograd algorithm. In *Advances in Information and Computer Security*, IWSEC'19, pages 67–88, Cham, 2019a. Springer International Publishing. ISBN 978-3-030-26834-3. doi:10.1007/978-3-030-26834-3_5. [Page 115.]

Jean-Guillaume Dumas, Joris van der Hoeven, Clément Pernet, and Daniel S. Roche. Lu factorization with errors. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '19, page 131–138, New York,

NY, USA, 2019b. Association for Computing Machinery. ISBN 9781450360845. doi:10.1145/3326229.3326244. [Page 148.]

Jean-Guillaume Dumas, Erich Kaltofen, David Lucas, and Clément Pernet. Elimination-based certificates for triangular equivalence and rank profiles. *Journal of Symbolic Computation*, 98:246 – 269, 2020. ISSN 0747-7171. doi:10.1016/j.jsc.2019.07.013. Special Issue on Symbolic and Algebraic Computation: ISSAC 2017. [Page 38.]

Wayne Eberly. A new interactive certificate for matrix rank. Technical report, University of Calgary, 2015. URL http://prism.ucalgary.ca/bitstream/1880/50543/1/2015-1078-11.pdf. [Pages 25, 28, and 52.]

Kaoutar Elkhiyaoui, Melek Önen, Monir Azraoui, and Refik Molva. Efficient techniques for publicly verifiable delegation of computation. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '16, pages 119–128, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4233-9. doi:10.1145/2897845.2897910. [Page 21.]

Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28:637–647, 1985. ISSN 0001-0782. doi:10.1145/3812.3818. [Page 31.]

Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144. [Page 116.]

Julius Farkas. Theorie der einfachen ungleichungen. *Journal fuer die reine und angewandte Mathematik*, 1902:1 – 27, 1902. doi:10.1515/crll.1902.124.1. [Page 148.]

Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag. ISBN 0-387-18047-8. doi:10.1007/3-540-47721-7_12. [Page 21.]

Rūsiņš Freivalds. Fast probabilistic algorithms. In *Mathematical Foundations of Computer Science 1979*, pages 57–69, Berlin, Heidelberg, 1979. Springer Berlin Heidelberg. ISBN 978-3-540-35088-0. [Pages 24, 28, 76, and 84.]

Martin Furer, Oded Goldreich, Yishay Mansour, Micheal Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems, 1989. [Page 18.]

Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Technologies*, 2017. doi:10.1515/popets-2017-0053. [Page 115.]

Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology – EUROCRYPT 2013*, pages 626–645, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38348-9. doi:10.1007/978-3-642-38348-9_37. [Page 23.]

Mark. Giesbrecht, Austin Lobo, and B. David Saunders. Certifying inconsistency of sparse linear systems. In *Proceedings of the 23rd International Symposium on Symbolic and Algebraic Computation*, ISSAC '98, pages 113–119, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1-58113-002-3. doi:10.1145/281508.281591. [Page 18.]

Pascal Giorgi and Vincent Neiger. Certification of minimal approximant bases. In *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, ISSAC '18, page 167–174, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355506. doi:10.1145/3208976.3208991. [Pages 73 and 104.]

Pascal Giorgi, Claude-Pierre Jeannerod, and Gilles Villard. On the complexity of polynomial matrix computations. In *Proceedings of the 2003 International Symposium on Symbolic and Algebraic Computation*, ISSAC '03, pages 135–142, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1-58113-641-2. doi:10.1145/860854.860889. [Page 73.]

The Givaro group. Givaro 4.1.1, January 2020. URL `https://github.com/linbox-team/givaro`. [Page 116.]

Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the 7th International Conference on Information Security and Cryptology*, ICISC 2004, pages 104–120, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32083-8. doi:10.1007/978-3-642-20465-4_11. [Pages 34 and 115.]

Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004. ISBN 0521830842, 9780521830843. doi:10.5555/1804390. URL `https://dl.acm.org/doi/book/10.5555/1804390`. [Page 121.]

Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. Association for Computing Machinery. ISBN 0-89791-221-7. doi:10.1145/28395.28420. [Pages 29 and 33.]

Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 59–68, New York, NY, USA, 1986. Association for Computing Machinery. ISBN 0897911938. doi:10.1145/12130.12137. [Page 18.]

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, pages 291–304, New York, NY, USA, 1985. Association for Computing Machinery. ISBN 0-89791-151-2. doi:10.1145/22145.22178. [Page 17.]

Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 978-1-60558-047-0. doi:10.1145/1374376.1374396. [Page 22.]

Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *Applied Cryptography and Network Security*, pages 288–306, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72738-5. doi:10.1007/978-3-540-72738-5_19. [Page 137.]

Somit Gupta, Soumojit Sarkar, Arne Storjohann, and Johnny Valeriote. Triangular x-basis decompositions and derandomization of linear algebra algorithms over $K[x]$. *Journal of Symbolic Computation*, 47:422–453, 2012. ISSN 0747-7171. doi:10.1016/j.jsc.2011.09.006. [Pages 83, 84, 87, 88, and 110.]

Charles Hermite. Sur l'introduction des variables continues dans la théorie des nombres. *Journal für die reine und angewandte Mathematik*, 41:191–216, 1851. ISSN 0075-4102. doi:10.1515/crll.1851.41.191. [Page 100.]

Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In *Theory of Cryptography*, pages 233–252, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-70936-7. [Page 137.]

Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israel Symposium on the Theory of Computing Systems (ISTCS '97)*, ISTCS '97, pages 174–174, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-8037-7. doi:10.5555/523986.857981. URL https://dl.acm.org/doi/10.5555/523986.857981. [Page 31.]

Yuval Ishai, Manika Mittal, and Rafail Ostrovsky. On the message complexity of secure multiparty computation. In *Proceedings of the International Workshop on Public Key Cryptography*, PKC 2018, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-76578-5_24. [Page 116.]

Anca Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *in Proceedings of the Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2003. [Page 137.]

Stanislaw Jarecki. Efficient covert two-party computation. In *Proceedings of the International Workshop on Public Key Cryptography*, PKC 2018, pages 644–674, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-319-76578-5_22. [Page 115.]

Claude-Pierre Jeannerod, Clément Pernet, and Arne Storjohann. Rank-profile revealing gaussian elimination and the cup matrix decomposition. *Journal of Symbolic Computation*, 56:46 – 68, 2013. ISSN 0747-7171. doi:10.1016/j.jsc.2013.04.004. [Pages 41 and 77.]

Audun Jøsang. Probabilistic logic under uncertainty. In *Proceedings of the Thirteenth Australasian Symposium on Theory of Computing - Volume 65*, CATS '07, page 101–110, AUS, 2007. Australian Computer Society, Inc. ISBN 1920682465. doi:10.5555/1273694.1273707. URL https://dl.acm.org/doi/10.5555/1273694.1273707. [Page 115.]

Thomas Kailath. *Linear Systems*. Prentice-Hall, 1980. ISBN 0135369614, 978-0135369616. [Page 100.]

B. Kaliski and J. Staddon. RSA Cryptography Specifications. RFC 2437, 1998. URL https://www.rfc-editor.org/rfc/rfc2437.txt. [Page 121.]

Erich L. Kaltofen, Michael Nehring, and B. David Saunders. Quadratic-time certificates in linear algebra. In *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, ISSAC '11, page 171–176, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306751. doi:10.1145/1993886.1993915. [Pages 14, 19, 24, 25, 26, 28, 38, 64, 65, 72, and 77.]

Erich L. Kaltofen, Bin Li, Zhengfeng Yang, and Lihong Zhi. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. *Journal of Symbolic Computation*, 47:1 – 15, 2012. ISSN 0747-7171. doi:10.1016/j.jsc.2011.08.002. [Page 19.]

I Kaporin. A practical algorithm for faster matrix multiplication. In *Numerical Linear Algebra with Applications*, 1999. doi:10.1002/(SICI)1099-1506(199912)6:8<687::AID-NLA177>3.0.CO;2-I. [Page 116.]

Elaye Karstadt and Oded Schwartz. Matrix multiplication, a little faster. In *SPAA '17*. doi:10.1145/3087556.3087579. [Page 116.]

Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 830–842, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4139-4. doi:10.1145/2976749.2978357. [Page 32.]

Tracy Kimbrel and Rakesh K. Sinha. A probabilistic algorithm for verifying matrix products using o(n2) time and log2n + o(1) random bits. *Information Processing Letters*, 45:107 – 110, 1993. ISSN 0020-0190. doi:10.1016/0020-0190(93)90224-W. [Page 24.]

Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 818–829, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4139-4. doi:10.1145/2976749.2978381. [Page 31.]

Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. Billion-gate secure computation with malicious adversaries. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 14–14, Berkeley, CA, USA, 2012. USENIX Association. doi:10.5555/2362793.2362807. URL https://dl.acm.org/doi/10.5555/2362793.2362807. [Page 33.]

George Labahn, Vincent Neiger, and Wei Zhou. Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. *Journal of Complexity*, 42:44–71, 2017. ISSN 0885-064X. doi:10.1016/j.jco.2017.03.003. [Page 78.]

François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 978-1-4503-2501-1. doi:10.1145/2608628.2608664. [Pages 15 and 116.]

The LinBox group. LinBox 1.6.3, July 2019. URL http://linalg.org. [Page 65.]

Yehuda Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pages 277–346. Springer International Publishing, Cham, 2017. ISBN 978-3-319-57048-8. doi:10.1007/978-3-319-57048-8_6. [Page 121.]

David Lucas, Vincent Neiger, Clement Pernet, Daniel Barry Roche, and Johan Rosenkilde. Verification Protocols with Sub-Linear Communication for Polynomial Matrix Operations. *MICA special edition of Journal of Symbolic Computation*, 2019. URL https://hal-unilim.archives-ouvertes.fr/hal-01829139. In press. [Page 72.]

Carsten Lund, Lance Fortnow, H Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39:859–868, 1992. ISSN 0004-5411. doi:10.1145/146585.146605. [Page 22.]

Cyrus Colton MacDuffee. *The Theory of Matrices*. Springer-Verlag Berlin Heidelberg, 1933. ISBN 978-3-642-99234-6. doi:10.1007/978-3-642-99234-6. [Page 100.]

Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security*

Symposium - Volume 13, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association. doi:10.5555/1251375.1251395. URL https://dl.acm.org/doi/abs/10.5555/1251375.1251395. [Page 33.]

WA. Microsoft Research, Redmond. Microsoft SEAL (release 3.4), October 2019. URL https://github.com/Microsoft/SEAL. [Page 116.]

Maurice Mignotte. How to share a secret. In *Proceedings of the 1982 Conference on Cryptography*, pages 371–375, Berlin, Heidelberg, 1983. Springer-Verlag. ISBN 3-540-11993-0. doi:10.5555/1756088.1756122. [Page 30.]

Pradeep Kumar Mishra, Deevashwer Rathee, Dung Hoang Duong, and Masaya Yasuda. Fast secure matrix multiplications over ring-based homomorphic encryption. Cryptology ePrint, 2018/663. URL https://eprint.iacr.org/2018/663. [Page 115.]

Payman Mohassel. Efficient and secure delegation of linear algebra. Cryptology ePrint Archive, Report 2011/605, 2011. https://eprint.iacr.org/2011/605. [Page 31.]

Thom Mulders and Arne Storjohann. Certified dense linear system solving. *Journal of Symbolic Computation*, 37:485–510, 2004. ISSN 0747-7171. doi:10.1016/j.jsc.2003.07.004. [Pages 18, 90, and 92.]

Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7:101–104, 1987. ISSN 1439-6912. doi:10.1007/BF02579205. [Page 60.]

David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS '98, page 59–66, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130074. doi:10.1145/288090.288106. [Pages 116 and 120.]

Vincent Neiger and Thi Xuan Vu. Computing canonical bases of modules of univariate relations. In *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC'17, pages 357–364, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 978-1-4503-5064-8. doi:10.1145/3087604.3087656. [Pages 73 and 110.]

Vincent Neiger, Johan Rosenkilde, and Grigory Solomatov. Computing Popov and Hermite forms of rectangular polynomial matrices. In *Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation*, ISSAC '18, page 295–302, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355506. doi:10.1145/3208976.3208988. [Pages 73 and 103.]

Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology*, CRYPTO 2012, pages 681–700, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-32008-8. doi:10.1007/978-3-642-32009-5_40. [Page 33.]

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT '99*, EUROCRYPT '99, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48910-8. doi:10.1007/3-540-48910-X_16. [Page 116.]

Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. *Communications of the ACM*, 59:103–112, 2016. ISSN 0001-0782. doi:10.1145/2856449. [Page 22.]

Clément Pernet and Arne Storjohann. Faster algorithms for the characteristic polynomial. In *Proceedings of the 32nd International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 307–314, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 978-1-59593-743-8. doi:10.1145/1277548.1277590. [Page 64.]

Clément Pernet and William Stein. Fast computation of Hermite normal forms of random integer matrices. *Journal of Number Theory*, 130:1675–1683, 2010. ISSN 0022-314X. doi:10.1016/j.jnt.2010.01.017. [Page 103.]

Vasile M. Popov. Invariant description of linear, time-invariant controllable systems. *SIAM Journal on Control*, 10:252–264, 1972. ISSN 0036-1402. doi:10.1137/0310020. [Pages 73 and 100.]

Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. https://eprint.iacr.org/2005/187. [Page 31.]

Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 49–62, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4132-5. doi:10.1145/2897518.2897652. [Page 20.]

Peter Rindal and Mike Rosulek. Faster malicious 2-party secure computation with online/offline dual execution. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, page 297–314, USA, 2016. USENIX Association. ISBN 9781931971324. doi:10.5555/3241094.3241118. URL https://dl.acm.org/doi/10.5555/3241094.3241118. [Page 115.]

Daniel S. Roche. Error correction in fast matrix multiplication and inverse. In *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, ISSAC '18, page 343–350, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355506. doi:10.1145/3208976.3209001. [Page 148.]

Thomas Schneider and Michael Zohner. GMW vs. Yao? efficient secure two-party computation with low depth circuits. In *Financial Cryptography and Data Security - 17th International Conference*, FC 2013, pages 275–292, Berlin, Heidelberg,

2013. Springer Berlin Heidelberg. ISBN 978-3-642-39884-1. doi:10.1007/978-3-642-39884-1_23. [Page 33.]

Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27:701–717, 1980. ISSN 0004-5411. doi:10.1145/322217.322225. [Pages 43, 62, and 76.]

Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979. ISSN 0001-0782. doi:10.1145/359168.359176. [Pages 30 and 115.]

Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39:869–877, 1992. ISSN 0004-5411. doi:10.1145/146585.146609. [Page 18.]

Arne Storjohann. High-order lifting and integrality certification. *Journal of Symbolic Computation*, 36:613–648, 2003. ISSN 0747-7171. doi:10.1016/S0747-7171(03)00097-X. [Pages 84 and 110.]

Arne Storjohann. Integer matrix rank certification. In *Proceedings of the 34th International Symposium on Symbolic and Algebraic Computation*, ISSAC '09, page 333–340, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605586090. doi:10.1145/1576702.1576748. [Page 18.]

Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13: 354–356, 1969. ISSN 0945-3245. doi:10.1007/BF02165411. [Page 116.]

Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology – CRYPTO 2013*, CRYPTO '13, pages 71–89, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40084-1. doi:10.1007/978-3-642-40084-1_5. [Page 22.]

The FFLAS-FFPACK group. FFLAS-FFPACK: Finite Field Linear Algebra Subroutines / Package, v2.4.4, June 2019. URL `http://linbox-team.github.io/fflas-ffpack`. [Page 64.]

Juan Ramon Troncoso-Pastoriza, Pedro Comesana, and Fernando Perez-Gonzalez. Secure direct and iterative protocols for solving systems of linear equations. 2009. [Page 35.]

Marc Van Barel and Adhemar Bultheel. A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numerical Algorithms*, 3:451–462, 1992. ISSN 1017-1398. doi:10.1007/BF02141952. [Page 102.]

Gilles Villard. Computing popov and hermite forms of polynomial matrices. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation*, ISSAC '96, pages 250–258, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917960. doi:10.1145/236869.237082. [Page 73.]

Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, third edition, 2013. ISBN 0521826462. doi:10.1017/CBO9781139856065. [Pages 78 and 90.]

Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them. *Communications of the ACM*, 58:74–84, 2015. ISSN 0001-0782. doi:10.1145/2641562. [Page 23.]

Ningning Wu, Jing Zhang, and li Ning. Discovering multivariate linear relationship securely. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 436– 437, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9290-6. doi:10.1109/IAW.2005.1495989. [Page 35.]

Stefan Wueller, Malte Breuer, Ulrike Meyer, and Susanne Wetzel. Privacy-preserving trade chain detection. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, LNCS '18, pages 373–388, Cham, 2018. Springer International Publishing. ISBN 978-3-030-00305-0. doi:10.1007/978-3-030-00305-0_26. [Page 13.]

Xuan Yang, Bin Kang, and Zhaoping Yu. Privacy-preserving cooperative linear system of equations protocol and its application. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-1-4244-2107-7. doi:10.1109/WiCom.2008.1131. [Page 35.]

Andrew C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 160–164, Washington, DC, USA, 1982. IEEE Computer Society. doi:10.1109/SFCS.1982.88. [Page 27.]

Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society. ISBN 0-8186-0740-8. doi:10.1109/SFCS.1986.25. [Page 32.]

W. Zhou. *Fast Order Basis and Kernel Basis Computation and Related Problems*. PhD thesis, University of Waterloo, 2012. URL `http://hdl.handle.net/10012/7326`. [Pages 87, 95, and 96.]

Wei Zhou and George Labahn. Computing column bases of polynomial matrices. In *Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation*, ISSAC'13, pages 379–386, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 978-1-4503-2059-7. doi:10.1145/2465506.2465947. [Pages 103 and 104.]

Wei Zhou and George Labahn. Unimodular completion of polynomial matrices. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC'14, pages 413–420, New York, NY, USA, 2014. Association for Computing

Machinery. ISBN 978-1-4503-2501-1. doi:10.1145/2608628.2608640. [Pages 107 and 108.]

Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, EUROSAM '79, page 216–226, Berlin, Heidelberg, 1979. Springer-Verlag. ISBN 3540095195. [Pages 43, 62, and 76.]

Abstract

Exact linear algebra is an essential tool for scientific computation, which is used in a wide array of applications such as experimental mathematics, number theory, cryptography or formal proofs. High performance computing infrastructures, used to tackle scientific computation, vastly evolved over the past 60 years: while initially performed over local, privately owned machines, the rise of the Internet in the 1990s ushered in the era of outsourced computation, which culminated with the recent development of cloud computing. Nowadays, many applications are run on machines which are no longer owned but instead rent by clients. This new model gave birth to questions related to trust in computation: as clients no longer own and control computing resources, how can they ensure the confidentiality, the security or even the validity of their computations? In this thesis, we focus on two key aspects for the security of outsourced computation for exact linear algebra: output correctness verification and data confidentiality in multiparty computations. We propose new protocols allowing clients to efficiently verify rank-related invariants and use them to improve the verification of classical linear algebra properties, matrix determinant and signature. We also introduce the first verification protocols for classical properties of the modules of polynomial vectors and their related matrices. Finally, we propose the first recursive secure multiparty computation protocol for matrix multiplication, based on Strassen-Winograd algorithm that allow users owning some rows of the matrices to compute some rows of the product. Neither the user's initial data, nor the intermediate values, even during the recurrence, are ever revealed to other users. This novel protocol allows us to improve the state of the art for the communication volume of such protocols.

Résumé

L'algèbre linéaire exacte est un outil essentiel du calcul scientifique et trouve de nombreuses applications en mathématiques expérimentales, en théorie des nombres, en cryptographie ou encore pour les preuves formelles. Les infrastructures de calcul à haute performance utilisées pour le calcul scientifique ont connu de nombreuses évolutions au cours des 60 dernières années. Les calculs, initialement exécutés sur des machines locales possédées par les utilisateurs, sont désormais réalisés sur des machines externes, évolution permise par le rapide développement d'Internet au cours des années 1990. Ce phénomène a atteint son apogée avec l'arrivée récente du cloud computing. Désormais, un grand nombre de calculs sont effectués sur des machines non plus possédées, mais louées par des clients. Ce nouveau modèle a donné naissance à de nombreuses questions autour de la confiance à accorder à de tels calculs. Alors que les clients n'ont plus le contrôle des infrastructures qu'ils utilisent, comment peuvent-ils s'assurer de la confidentialité, la sécurité ou encore la validité de leurs calculs ? Dans cette thèse, nous considérons deux aspects fondamentaux de la sécurité des calculs externalisés en algèbre linéaire exacte : la vérification de la justesse des résultats, et la confidentialité des données pour les calculs multipartites. Nous proposons de nouveaux protocoles permettant aux clients de vérifier efficacement les invariants liés au rang et nous les utilisons pour améliorer la vérification de propriétés classiques de l'algèbre linéaire, le déterminant de matrices et la signature. Nous introduisons également les premiers protocoles de vérification pour les propriétés classiques des modules de vecteurs de polynômes et leurs matrices associées. Enfin, nous proposons le premier protocole multipartite sécurisé récursif pour la multiplication de matrices basé sur l'algorithme de Strassen-Winograd. Notre protocole permet à des utilisateurs possédant certains lignes des matrices d'entrée de calculer les mêmes lignes de la matrice de sortie. Ni les données initiales, ni les valeurs intermédiaires du calcul ne sont révelées aux autres participants. Ce nouveau protocole nous permet d'améliorer l'état de l'art pour le volume de données communiquées lors de l'exécution de tels protocoles.