New Schemes for Secure Distributed Matrix Multiplication: Cooperative and Analog SDMM

Okko Makkonen

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo February 24, 2022

Supervisor and advisor

Prof. Camilla Hollanti

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Okko Makkonen

Title New Schemes for Secure Distributed Matrix Multiplication: Cooperative and Analog SDMM

Degree programme Mathematics and Operations Research

Major Mathematics **Code of major** SCI3054

Supervisor and advisor Prof. Camilla Hollanti

Date February 24, 2022 **Number of pages** 78+4 **Language** English

Abstract

Secure distributed matrix multiplication (SDMM) is a way of distributing the computation of a matrix product to external servers while providing information-theoretic guarantees on the security of the data. SDMM can be used to speed up the computation of large-scale matrix products, which are used in many applications, in situations where the confidentiality of the data is important. This thesis presents the necessary background knowledge in coding theory and information theory, along with some of the existing research in SDMM. A novel framework for different SDMM constructions is then used to present a new type of cooperative SDMM, which allows for outsourcing the communication costs of SDMM. Furthermore, analog SDMM, which enables computation over real numbers, is presented. Finally, some applications of SDMM are discussed.

Keywords secure distributed matrix multiplication, coding theory, information theory, coded computation

Aalto University
School of Science

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Okko Makkonen

Työn nimi Yhteistyöhön perustuva ja analoginen turvallinen hajautettu matriisikertolasku

Koulutusohjelma Matematiikka ja systeemianalyysi

Pääaine Matematiikka **Pääaineen koodi** SCI3054

Työn valvoja ja ohjaaja Prof. Camilla Hollanti

Päivämäärä 24. helmikuuta 2022 **Sivumäärä** 78+4 **Kieli** Englanti

Tiivistelmä

Turvallinen hajautettu matriisikertolasku on tapa hajauttaa matriisitulon laskeminen ulkoisille palvelimille siten, että tiedot pysyvät informaatioteoreettisesti turvassa. Turvallisella hajautetulla matriisikertolaskulla voidaan nopeuttaa yleisesti käytettyä matriisien kertolaskua sovelluksissa, joissa tietojen salassapito on tärkeää. Tässä työssä käydään läpi välttämättömät taustatiedot koodausteoriasta ja informaatioteoriasta, sekä esitellään aikaisempaa tutkimusta aiheeseen liittyen. Uuden lineaarisen salaisen laskennan kehikon avulla rakennetaan yhteistyöhön perustuva turvallisen hajautetun matriisikertolaskun menetelmä, jonka avulla osa kommunikaatiosta voidaan ulkoistaa palvelimille. Lisäksi käsitellään analogista turvallista hajautettua matriisikertolaskua, jonka avulla laskutoimituksia voidaan suorittaa käyttämällä reaalilukuja. Lopuksi käsitellään salaisen laskennan käyttökohteita.

Avainsanat turvallinen hajautettu matriisikertolasku, koodausteoria, informaatioteoria, koodattu laskenta

Contents

1 Introduction

Matrix multiplication is a fundamental tool in many numerical computational tasks across multiple areas. At their simplest, matrices are used to represent linear transformations between two vector spaces, but they can be used to represent even more complicated things. These matrices can then be easily manipulated on a computer to perform complicated computations on huge amounts of data.

There are many operations that can be done on matrices. For example, matrices can be decomposed into products of simpler matrices, or an inverse matrix can be computed. In this thesis, we are interested in matrix multiplication, which has many applications in different areas. In applied mathematics, matrix multiplication can be used to compute the eigenvalues of a matrix using power iteration. In graph theory, the powers of the adjacency graph can be used to find walks of different lengths in a graph. In statistics, the correlation matrix of a data sample can be computed using a matrix multiplication. In machine learning, the fundamental operation of an artificial neural network is matrix multiplication.

In all of the above applications, the size of the input data can be large. The need for computing over data grows in tandem with the amount of data that can be collected, transferred, and stored. This presents a problem in the context of matrix multiplication. An $n \times n$ matrix clearly has n^2 elements, but the schoolbook algorithm for multiplying two such matrices requires $\mathcal{O}(n^3)$ operations. This means that the computational time for matrix multiplication grows faster than the size of the data.

There has been substantial research into trying to improve the efficiency of matrix multiplication so that the amounts of data can be computed using the existing computing power. Strassen introduces a new algorithm in [1] for computing the product of two $n \times n$ matrices in approximately $\mathcal{O}(n^{2.8})$ arithmetical operations. Further research has been done to reduce the complexity even lower, though these algorithms are not practical for matrices of realistic size.

If the computational power of one computational unit is not enough to perform some computation, then that computation can be distributed to multiple computing nodes. Distributed computation is not simple for all computational problems, but matrix multiplication lends itself to distribution quite easily. When the computation is distributed to multiple places, the computation can happen in parallel, which means that the result can be computed faster.

Computation can be distributed in multiple ways and to many different places. In traditional computer architecture, matrix multiplication can be distributed from the central processing unit to the graphics processing unit, which contains multiple computing cores. However, in this thesis, we are interested in distributing the computation to cloud servers that are contacted over the internet.

Distributing computation to cloud servers is characterized by two things: these servers can be busy with other tasks, and the servers cannot be trusted with confidential data. These limitations strongly influence the way we need to perform the computation. Additionally, we need to account for the amount of data that is transferred to and from the servers, since they can be geographically far away.

The first problem is that some servers can be substantially slower than others for many reasons, like having other tasks to perform at the same time. This means that the total computation time is slow, since it is determined by the slowest server. One way of fixing this problem is to design the algorithm such that not all the responses from the servers are needed, so that the fastest servers can compute the result without being slowed down by the slowest servers. Such a scheme is said to be robust against straggling servers.

Data confidentiality is important in many applications, including banking and medical information. Such data cannot be distributed to untrusted servers as the data needs to be held secret. A common way of achieving the confidentiality of data is through encryption, which is widely used in data storage and transportation. For matrix multiplication, we need to compute over the confidential data, which means that a special type of encryption would need to be used. Encryption that allows for meaningful computations over the encrypted data is said to be homomorphic encryption. There have been many efforts to make fully homomorphic encryption feasible in real applications, but at the moment, it is not practical to use encryption in distributed matrix multiplication.

These problems are fixed by *secure distributed matrix multiplication (SDMM)*, which was first presented by Chang and Tandon in [2]. SDMM uses techniques from coding theory to provide robustness to the computation and information theoretic tools to provide security.

The organization of this thesis is as follows: In Section 2, we present some useful preliminaries to coding theory, information theory, secret sharing, and cryptography. The most important concepts in these sections are Reed–Solomon codes and information-theoretic security. In Section 3, we present some examples of SDMM and the general constructions from prior research, including the secure MatDot code and the GASP code. In Section 3.5, we present a new framework for SDMM schemes that are built from linear codes, which allows for simple proofs of security. In Section 4, we present a new mode of secure distributed matrix multiplication, called cooperative SDMM, which was first introduced by the author in [3]. In Section 5, we present a new way of performing SDMM over the analog domain, *i.e.*, over real or complex numbers. This research has been presented by the author in [4]. Finally, in Section 6, we present some applications of SDMM in the context of digital and remote healthcare, referred to as eHealth.

2 Preliminaries

The *finite field* of size q is denoted by $\mathbb{F}_q$. The group of units of a finite field is denoted by $\mathbb{F}_q^\times$. The set $\{1, \dots, n\}$ is denoted by $[n]$. The set of natural numbers is $\mathbb{N} = \{0, 1, \dots\}$ and the set of positive integers is denoted by $\mathbb{Z}_+ = \{1, 2, \dots\}$.

Matrices are denoted by capital letters, while vectors and scalars are denoted by lowercase letters. Bold letters are reserved for random variables corresponding to the nonbold variables.

The ith column of a matrix G is denoted by g_i and the jth column by g^j. If $\mathcal{I}$ is an ordered set of indices, then the matrix formed by the columns of G corresponding to those indices is denoted by $G_\mathcal{I}$. Similarly, if $\mathcal{J}$ is an ordered set of indices, then the matrix formed by the rows of G corresponding to those indices is denoted by $G^\mathcal{J}$. As a special case of this, we denote by $G^{\leq m}$ the first m rows of G and $G^{>m}$ the matrix formed by removing the first m rows. Additionally, if a is a vector and $\mathcal{I}$ is some ordered set of indices, then $a_\mathcal{I}$ denotes the vector formed by choosing the entries of a with indices in $\mathcal{I}$.

2.1 Coding theory

One goal of coding theory is to send messages over unreliable channels such that the message can be correctly interpreted on the other side even if errors are introduced in transit. In distributed storage and computing, coding theory can be used to add robustness and security to computations, by adding redundancy and error-correction capability. The process of converting a message m into a codeword c is known as *encoding*, and the inverse operation is known as *decoding*. The encoding adds redundancy to the message such that it can be recovered even if errors or erasures occur during transmission. Some good general coding theory references include [5] and [6].

In coding theoretic terms, the messages are sequences of message symbols over some finite alphabet. These messages are then encoded into codewords that are sequences of code symbols over some, possibly different, finite alphabet. An important concept in coding theory is block codes, where all the messages of a fixed length k are encoded as codewords of a fixed length n, where $n \geq k$.

Example 2.1.1 (Repetition code)**.** An obvious way to add redundancy is by sending multiple copies of the message in the hope that at least half of the copies are not erroneous. Such a code is known as a *repetition code*. If a symbol m is sent using a repetition code of length n, then the symbol m is sent n times. This code is denoted by $\mathrm{Rep}(n)$.

2.1.1 Linear codes

A linear code is a special case of a block code, where the alphabet used is a finite field $\mathbb{F}_q$ of size q and the codewords form a vector space over $\mathbb{F}_q$.

Definition 2.1.2. Let $\mathcal{C} \subset \mathbb{F}_q^n$ be a vector space of dimension $k \leq n$. Then, $\mathcal{C}$ is an $[n, k]_q$-linear code, or $[n, k]$-linear code if the field $\mathbb{F}_q$ is clear from context. The

dimension k is known as the *dimension* of the code, and n is known as the *length* of the code.

Example 2.1.3 (Simple parity-check code)**.** Let $m = (m_1, m_2) \in \mathbb{F}_2^2$ represent a two-bit message. Then, the vector $c = (m_1, m_2, m_1 + m_2)$ is a codeword in a $[3, 2]_2$-linear code known as the *simple parity-check code*.

In the following, a code will always mean a linear code. The idea behind this definition is that a small error in a codeword will transform the vector to something that is not a codeword. An error in a codeword can be represented by a random vector $e \in \mathbb{F}_q^n$ that transforms the codeword $c \in \mathcal{C}$ to $r = c + e$. It is easy to detect an error, since one can check if the received vector is in the code. The dimension of a code $\mathcal{C}$ means its dimension as a $\mathbb{F}_q$ vector space. If the dimension of $\mathcal{C}$ is k, it contains q^k codewords out of the possible q^n words of length n.

The "size" of an error e is typically measured using the Hamming weight of e.

Definition 2.1.4. The number of nonzero coordinates in x is the *Hamming weight* of a vector $x \in \mathbb{F}_q^n$, and it is denoted by $\mathrm{wt}(x)$. In $\mathbb{F}_q^n$, the *Hamming distance* between two vectors x, y is $d(x, y) = \mathrm{wt}(x - y)$.

The definition makes it clear that $0 \leq \mathrm{wt}(x) \leq n$ for all $x \in \mathbb{F}_q^n$, and the Hamming distance between two vectors counts the number of places where the two vectors differ. The Hamming distance defines a metric on $\mathbb{F}_q^n$ and is an important measure of the similarity of vectors in coding theory. An error $e \in \mathbb{F}_q^n$ of weight $\mathrm{wt}(e)$ means that a codeword is changed in $\mathrm{wt}(e)$ coordinates. To detect errors of a certain weight, we need that any two codewords are not too close to each other in terms of their Hamming distance. This separation of codewords in a code is captured by the minimum distance.

Definition 2.1.5. The *minimum distance* of a linear code $\mathcal{C}$ is defined as the smallest weight of any nonzero codeword in $\mathcal{C}$. If $\mathcal{C}$ is an $[n, k]$ code with minimum distance d, then it can be denoted as an $[n, k, d]$ code.

Example 2.1.6. The minimum distance of the repetition code of length n is n, since any nonzero codeword has all elements with the same nonzero value. The minimum distance of the simple parity-check code is 2, since all codewords have an even Hamming weight by definition.

Given the Hamming distance, we can define a ball with center $y \in \mathbb{F}_q^n$ and radius $r \in \mathbb{N}$, which consists of all vectors $y' \in \mathbb{F}_q^n$ such that $d(y, y') < r$. If $\mathcal{C} \subset \mathbb{F}_q^n$ is a code with minimum distance d, then any ball of radius d centered at a codeword $c \in \mathcal{C}$ will not contain any other codeword of $\mathcal{C}$. On the other hand, balls of radius $t = \lfloor \frac{d-1}{2} \rfloor$ centered at codewords $c \in \mathcal{C}$ are the largest nonoverlapping balls that can be fit in the space. This problem has a natural connection to sphere packing.

We can form another code from an $[n, k, d]$ code $\mathcal{C}$ by removing some fixed coordinate i from each codeword. This process is known as *puncturing* and it defines a new code $\mathcal{C}^*$ of length $n-1$. The dimension and minimum distance of the punctured code depend on the structure of $\mathcal{C}$. Puncturing can be extended to removing multiple

coordinates from each codeword. This will be useful in distributed computation and storage, since the responses can be seen as being in a punctured code if some workers do not respond.

Another way of forming a new code from an existing code is by taking the dual of a code.

Definition 2.1.7. The *dual code* of a code $\mathcal{C}$ is its dual as a vector space with respect to the standard inner product, *i.e.*, the vector space

$$\mathcal{C}^\perp = \{y \in \mathbb{F}_q^n \colon y \cdot c = 0 \text{ for all } c \in \mathcal{C}\}.$$

If $\mathcal{C}$ is an $[n, k]$ code, then $\mathcal{C}^\perp$ is an $[n, n-k]$ code.

Given that the minimum distance of $\mathcal{C}$ is d, it is not trivial to state the minimum distance of $\mathcal{C}^\perp$ in general. However, for some nice codes, this is possible.

We need a way to convert a message vector in $\mathbb{F}_q^k$ to a codeword in $\mathcal{C}$ before we can use the linear code as an encoding scheme. This is done by right multiplying the message vector with the generator matrix of the code.

Definition 2.1.8. A $k \times n$ matrix G over $\mathbb{F}_q$ is a *generator matrix* for $\mathcal{C}$ if the rows of G span $\mathcal{C}$. The generator matrix of $\mathcal{C}^\perp$ is the *parity-check matrix* of $\mathcal{C}$ and is often denoted by H.

The message vector $m \in \mathbb{F}_q^k$ is mapped to a codeword by $m \mapsto mG$, which is injective, since G is full rank. The generator matrix of a code is not unique, as row operations on a generator matrix will yield another generator matrix.

Example 2.1.9. Every codeword in the binary repetition code $\mathrm{Rep}_2(3)$ has the form (m, m, m), *i.e.*, it is a multiple of the basis vector $(1, 1, 1)$. If $y \in \mathbb{F}_2^3$ is such that $y \cdot c = 0$ for each $c \in \mathrm{Rep}_2(3)$, then $y_1 + y_2 + y_3 = 0$ or $y_3 = y_1 + y_2$. Therefore, the dual code of $\mathrm{Rep}_2(3)$ is the simple parity-check code. This code has codewords of the form $(m_1, m_2, m_1 + m_2)$ for some $m_1, m_2 \in \mathbb{F}_2$. These codewords can be expressed as $m_1(1, 0, 1) + m_2(0, 1, 1)$, so the simple parity-check code has basis $\{(1, 0, 1), (0, 1, 1)\}$. Therefore, the generator matrix and the parity-check matrix of $\mathrm{Rep}_2(3)$ are

$$G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad H = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

On the other hand, these are the parity-check matrix and generator matrix of the simple parity-check code, respectively.

If $\mathcal{C}$ is a code, then $(\mathcal{C}^\perp)^\perp = \mathcal{C}$. This means that a vector $y \in \mathbb{F}_q^n$ is a codeword in $\mathcal{C}$ if and only if it is orthogonal to all $h \in \mathcal{C}^\perp$. The codewords in $\mathcal{C}^\perp$ are linear combinations of the basis vectors $h^1, \ldots, h^{n-k}$, which are the rows of the parity-check matrix H. As a result, the vector y is in $\mathcal{C}$ if and only if $y \cdot h^i = 0$ for all $i = 1, \ldots, n-k$. This can be written compactly as $yH^T = 0$. Next, we will prove a useful lemma.

Lemma 2.1.10. *Let $\mathcal{C}$ be an $[n, k]$ code with minimum distance d and parity-check matrix H. Then every $d-1$ columns of H are linearly independent, and there are d columns of H that are linearly dependent.*

Proof. Assume that H has $d-1$ linearly dependent columns, say $h_{i_1}, \ldots, h_{i_{d-1}}$. Thus, we have coefficients $\lambda_{i_1}, \ldots, \lambda_{i_{d-1}} \in \mathbb{F}_q$ that are not all zero such that

$$\sum_{j=1}^{d-1} \lambda_{i_j} h_{i_j} = 0.$$

Then, we can construct the vector $y \in \mathbb{F}_q^n$ such that $y_{i_j} = \lambda_{i_j}$ for $j = 1, \ldots, d-1$ and $y_i = 0$ otherwise. Thus, $yH^T = 0$, so $y \in \mathcal{C}$. By construction, it is clear that y has at most $d-1$ nonzero coordinates, but it is not the zero vector. This contradicts the minimum distance of $\mathcal{C}$. Hence, any $d-1$ columns of H are linearly independent.

Let $c \in \mathcal{C}$ be a nonzero codeword of minimal weight, *i.e.*, $\mathrm{wt}(c) = d$. Let $i_1, \ldots, i_d$ be the indices of the nonzero coordinates in c. Hence, $h_{i_1}, \ldots, h_{i_d}$ are linearly dependent. □

A message m can be retrieved from its corresponding codeword $c = mG$ by choosing some k coordinates of c and multiplying with an invertible $k \times k$ submatrix of G. Let $\mathcal{I}$ denote the indices of any k linearly independent columns of G. Such a set of indices $\mathcal{I} \subset [n]$ is known as an *information set*. The message m can then be retrieved with $c_{\mathcal{I}} G_{\mathcal{I}}^{-1}$ because $G_{\mathcal{I}}$ is an invertible submatrix of G. It is worth noting that not every coordinate of c is needed to decode the message, only some set of coordinates that includes an information set of the code. Equivalently, the vector $c_{\mathcal{I}}$ can be seen as a word in the punctured code $\mathcal{C}_{\mathcal{I}}$.

The $k \times n$ generator matrix G of a code $\mathcal{C}$ is said to be in *systematic form* if the left $k \times k$ submatrix of G is the identity matrix. The encoding of a message $m = (m_1, \ldots, m_k)$ is $c = (m_1, \ldots, m_k, c_{k+1}, \ldots, c_n)$ for some $c_{k+1}, \ldots, c_n \in \mathbb{F}_q$, *i.e.*, the codeword has the message in the beginning. The redundancy symbols $c_{k+1}, \ldots, c_n$ are known as *parity symbols*. The decoding of such an encoding is easy, since the message can be read as the first k symbols of c if no errors have occurred. Any generator matrix G can be reduced to systematic form by row-reduction.

The parity-check matrix is easy to form when the generator matrix is in systematic form. The parity-check matrix corresponding to $G = \begin{pmatrix} I_k & X \end{pmatrix}$ is $H = \begin{pmatrix} -X^T & I_{n-k} \end{pmatrix}$. This follows from the fact that

$$GH^T = \begin{pmatrix} I_k & X \end{pmatrix} \begin{pmatrix} -X \\ I_{n-k} \end{pmatrix} = -X + X = 0,$$

and the fact that H is of full rank.

When transmitting a codeword over an unreliable channel, there might be some errors or erasures introduced. An error is modeled as an error vector $e \in \mathbb{F}_q^n$, which is unknown to the receiver. Erasures mean that a symbol in the codeword is erased at a known position. As seen earlier, erasures can be corrected as long as the remaining coordinates include an information set of the code. The received symbols can be seen as part of a punctured code of the original code. Sometimes it is also important to detect that an error has happened, but not necessarily correct it. If a codeword $c \in \mathcal{C}$ is sent and the vector $r = c + e$ is received, an error can be detected if $\mathrm{wt}(e) < d$, where d is the minimum distance of $\mathcal{C}$.

In practice, detecting an error is simple because a vector $y \in \mathbb{F}_q^n$ is in the code $\mathcal{C}$ if and only if $yH^T = 0$. Therefore, if yH^T, which is known as the *syndrome*, is not zero, then there has been an error in transmission. Correcting errors is more difficult, since it might not be clear which codeword was sent. If errors are independently distributed such that an error is introduced to some coordinate with a constant probability, then the most likely codeword is the codeword that is closest to r with respect to the Hamming distance. This strategy is generally known as *maximum likelihood decoding*, since it chooses the codeword with the greatest likelihood of being received as r. The error-correcting is unambiguous if the number of errors is no more than $t = \lfloor \frac{d-1}{2} \rfloor$.

In theory, an error can be corrected just by choosing the codeword that is closest to the received vector. In practice, finding such a codeword can be difficult, so it is important to find efficient algorithms for correcting errors. Many efficient algorithms have been developed for different classes of codes.

2.1.2 MDS codes

The minimum distance of a code is an important metric for the error-correction capabilities of a code. Therefore, many bounds have been computed for the minimum distance. One of the most important bounds is the Singleton bound.

Proposition 2.1.11 (Singleton bound)**.** *Let $\mathcal{C}$ be a linear $[n, k, d]$ code over $\mathbb{F}_q$. Then*

$$d \leq n - k + 1.$$

Proof. Let c_1 and c_2 be two distinct codewords in $\mathcal{C}$. Consider the vectors c_1' and c_2' in $\mathbb{F}_q^{n-d+1}$, which are composed of the first $n - d + 1$ coordinates of c_1 and c_2, respectively. Because c_1 and c_2 differ in at least d coordinates, c_1' and c_2' differ in at least 1 coordinate. Therefore, the map sending $c \in \mathcal{C}$ to its first $n-d+1$ coordinates is injective. By comparing the cardinalities we get that

$$q^k \leq q^{n-d+1}, \tag{2.1.1}$$

because $|\mathcal{C}| = q^k$ and $|\mathbb{F}_q^{n-d+1}| = q^{n-d+1}$. Taking logarithms in base q proves the claim. □

The above proposition describes an upper bound for the minimum distance of a code with specified length n and dimension k. Codes that achieve the upper bound with equality are known as *maximum distance separable (MDS)* codes. The name means that the codewords are as separated as possible with regards to the Hamming distance.

Example 2.1.12. Consider the repetition code $\mathcal{C} = \mathrm{Rep}(n)$, which has dimension $k = 1$, length n and minimum distance n. Thus, $d = n = n - k + 1$, so the repetition code is an MDS code.

Lemma 2.1.13. *Let $\mathcal{C}$ be an $[n, k]$ code. Then, $\mathcal{C}$ is an MDS code if and only if any set $\mathcal{I} \subset [n]$ of size $|\mathcal{I}| = k$ is an information set of $\mathcal{C}$.*

Proof. "⇒" Let $\mathcal{I} \subset [n]$ be a set of k indices. Consider a nonzero vector $x \in \mathbb{F}_q^k \setminus \{0\}$ and the codeword $y = xG \in \mathcal{C}$. As the rows of G are linearly independent, we get that $y \neq 0$. If $y_{\mathcal{I}} = 0$, then y is a nonzero vector with at most $n-k$ nonzero coordinates. This is a contradiction, since the minimum distance of $\mathcal{C}$ is $n-k+1$. Hence, $y_{\mathcal{I}} \neq 0$. As a result, the $k \times k$ submatrix $G_{\mathcal{I}}$ has trivial nullspace, implying that the columns of G corresponding to the indices in $\mathcal{I}$ are linearly independent, and $\mathcal{I}$ is an information set.

"⇐" For the sake of contradiction, assume that $\mathcal{C}$ is not MDS. Therefore, there exists a nonzero codeword $c = mG$ of weight $d < n-k+1$, where $m \in \mathbb{F}_q^k \setminus \{0\}$ and G is the generator matrix of $\mathcal{C}$. Because c contains at least k zeros, there is now a set $\mathcal{I} \subset [n]$ of size k such that $c_{\mathcal{I}} = 0$. As a result of the columns of $G_{\mathcal{I}}$ not being linearly independent, $\mathcal{I}$ is not an information set. This contradicts the assumption, so $\mathcal{C}$ is an MDS code. □

The above lemma shows that MDS codes are useful in erasure correction, since any set of at least k coordinates from a codeword is enough to decode the message. This will later be useful in the setting of distributed computing and secret sharing.

Proposition 2.1.14. *The dual code of an MDS code is an MDS code.*

Proof. Let $\mathcal{C}$ be an $[n,k]$ MDS code with generator matrix G. Thus, G is the parity-check matrix of $\mathcal{C}^\perp$. Any k columns of G are linearly independent, according to Lemma 2.1.13. On the other hand, there cannot be $k+1$ linearly independent columns, since the columns are in $\mathbb{F}_q^k$. As a result, the minimum distance of $\mathcal{C}^\perp$, according to Lemma 2.1.10, is $k+1$, which equals the Singleton bound $n-(n-k)+1$. Hence, $\mathcal{C}^\perp$ is MDS. □

Definition 2.1.15. A $k \times n$ matrix is said to have the *MDS property* if it generates an MDS code. This is equivalent to saying that any maximal submatrix is invertible.

2.1.3 Reed–Solomon codes

A particularly interesting class of MDS codes are those that are based on polynomial oversampling. The following lemma is a useful tool for interpolating polynomials from samples.

Lemma 2.1.16 (Lagrange interpolation)**.** *Let $(x_1, y_1), \ldots, (x_k, y_k)$ be k points such that the x_i's are distinct. Then there exists a unique polynomial $p(x)$ of degree $k-1$ such that $p(x_i) = y_i$ for all $i \in [k]$.*

Proof. The Lagrange basis polynomial is defined as

$$\ell_i(x) = \prod_{\substack{j \in [k] \\ j \neq i}} \frac{x - x_j}{x_i - x_j} = \ell_i^{(0)} + \ell_i^{(1)} x + \cdots + \ell_i^{(k-1)} x^{k-1}.$$

Thus, $\ell_i(x_i) = 1$ and $\ell_i(x_j) = 0$ for all $j \in [k]$, $j \neq i$. Hence,

$$p(x) = \sum_{i \in [k]} y_i \ell_i(x)$$

is a degree $k-1$ polynomial such that $p(x_i) = y_i$ by using the above property of the Lagrange basis polynomials. Let $r(x)$ be another polynomial of degree $k-1$ with the property that $r(x_i) = y_i$. Thus, $p(x) - r(x)$ is a polynomial of degree $k-1$ with k roots. As a result, $p(x) = r(x)$, because any nonzero polynomial of degree n over a field has no more than n roots. Therefore, the interpolation polynomial is unique. □

Sampling a polynomial of degree $k-1$ at more than k points will give redundancy in the interpolation phase, since not all points are needed to construct the original polynomial. As seen in the definition below, Reed–Solomon codes (and their generalizations) are based on polynomial oversampling.

Definition 2.1.17. Let $\alpha \in \mathbb{F}_q^n$ be a vector of distinct locators and $\nu \in (\mathbb{F}_q^\times)^n$ be a vector of nonzero column multipliers. The *generalized Reed–Solomon (GRS) code* of length n and dimension k is

$$\mathrm{GRS}_k(\alpha, \nu) = \{(\nu_1 p(\alpha_1), \ldots, \nu_n p(\alpha_n)) \mid p(x) \in \mathbb{F}_q[x],\ \deg p(x) < k\}\,.$$

We will often be interested in the case of $\nu = \overline{1}$; *i.e.*, the column multipliers are all 1. As a shorthand, we define $\mathrm{RS}_k(\alpha) = \mathrm{GRS}_k(\alpha, \overline{1})$ to be the *Reed–Solomon code*.

Let $p(x) \in \mathbb{F}_q[x]$, $\deg p(x) < k$, be the message. The message can then be encoded naturally by evaluating the polynomial $p(x)$ at the locators α and multiplying the results with the column multipliers ν. Hence, the generator matrix of an $[n, k]$ generalized Reed–Solomon code is

$$G = \begin{pmatrix} 1 & \ldots & 1 \\ \alpha_1 & & \alpha_n \\ \alpha_1^2 & \ldots & \alpha_n^2 \\ \vdots & & \vdots \\ \alpha_1^{k-1} & \ldots & \alpha_n^{k-1} \end{pmatrix} \cdot \mathrm{diag}(\nu). \tag{2.1.2}$$

The matrix on the left is the transpose of a Vandermonde matrix.

Definition 2.1.18. An $n \times k$ matrix of the form

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ \vdots & & \vdots & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

is known as a *Vandermonde matrix*. The $\alpha_1, \ldots, \alpha_n$ are known as locators.

Lemma 2.1.19. *An $n \times k$ Vandermonde matrix has a trivial right kernel, when $k \leq n$ and the locators are distinct.*

Proof. Let $p \in \mathbb{F}_q^k$ be any vector. Then we can write

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ \vdots & & \vdots & & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} p_1 \\ \vdots \\ p_k \end{pmatrix} = \begin{pmatrix} p(\alpha_1) \\ \vdots \\ p(\alpha_n) \end{pmatrix},$$

where $p(x) \in \mathbb{F}_q[x]$ is the $k-1$ degree polynomial with coefficients equal to p's coordinates. If p is in the right kernel of the Vandermonde matrix, then $p(\alpha_i) = 0$ for $i \in [n]$. Because the evaluation points $\alpha_1, \ldots, \alpha_n$ are distinct, $p(x)$ is a degree $k-1$ polynomial with $n > k-1$ roots. As a result, $p(x)$ must be the zero polynomial, implying that $p = 0$. □

The result of Lemma 2.1.19 is that the generalized Reed–Solomon is k-dimensional because the generator matrix G is of full rank. As discussed earlier, a degree $k-1$ polynomial can be uniquely retrieved from any set of k evaluations, which implies that any $\mathcal{I} \subset [n]$ is an information set of a GRS code. According to Lemma 2.1.13, an $[n, k]$ GRS code is MDS with a minimum distance $n - k + 1$.

Another method of encoding a message $m \in \mathbb{F}_q^k$ is to first find the unique degree $k-1$ polynomial that goes through the points $(\alpha_i, \nu_i^{-1} m_i)$ for $i = 1, \ldots, k$ and evaluate it at the points $\alpha_1, \ldots, \alpha_n$ before multiplying by the column multipliers. Thus, the encoding of a message m starts with m and includes $n-k$ parity symbols. Finding such an interpolation polynomial can be done using Lagrange interpolation. Therefore, the generator matrix of a GRS code is

$$G = \begin{pmatrix} 1 & 0 & \ldots & 0 & \ell_1(\alpha_{k+1}) & \ldots & \ell_1(\alpha_n) \\ 0 & 1 & \ldots & 0 & \ell_2(\alpha_{k+1}) & \ldots & \ell_2(\alpha_n) \\ \vdots & & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & 1 & \ell_k(\alpha_{k+1}) & \ldots & \ell_k(\alpha_n) \end{pmatrix} \cdot \operatorname{diag}(\nu). \tag{2.1.3}$$

If $\nu = \overline{1}$, then this generator matrix is in systematic form. In fact, the generator matrix in (2.1.3) is the reduced row echelon form of the generator matrix in (2.1.2). We can form the parity-check matrix

$$H = \begin{pmatrix} -\ell_1(\alpha_{k+1}) & -\ell_2(\alpha_{k+1}) & \ldots & -\ell_k(\alpha_{k+1}) & 1 & 0 & \ldots & 0 \\ -\ell_1(\alpha_{k+2}) & -\ell_2(\alpha_{k+2}) & \ldots & -\ell_k(\alpha_{k+2}) & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \vdots \\ -\ell_1(\alpha_n) & -\ell_2(\alpha_n) & \ldots & -\ell_k(\alpha_n) & 0 & \ldots & 0 & 1 \end{pmatrix}.$$

2.1.4 Star products of codes

In addition to the dual of a code or a punctured code, new codes can be built from existing codes by taking the star product of two codes.

Definition 2.1.20. Let $x, y \in \mathbb{F}_q^n$ be vectors. Then the *star product* of x and y is defined as the element-wise product

$$x \star y = (x_1 y_1, \ldots, x_n y_n) \in \mathbb{F}_q^n.$$

In the literature this is known also as the Schur product or the Hadamard product of vectors. Furthermore, if $\mathcal{C}$ and $\mathcal{D}$ are length n codes, then the *star product* of $\mathcal{C}$ and $\mathcal{D}$ is

$$\mathcal{C} \star \mathcal{D} = \langle c \star d \mid c \in \mathcal{C}, d \in \mathcal{D} \rangle.$$

In other words, the star product code is the vector space generated by the star products of the codewords in $\mathcal{C}$ and $\mathcal{D}$.

In general, the star product of two codes can be difficult to describe. For generalized Reed–Solomon codes, the star product can be formulated.

Proposition 2.1.21. *Let α be a vector of distinct locators in $\mathbb{F}_q^n$ and $\nu, \omega \in (\mathbb{F}_q^\times)^n$ be vectors of column multipliers. Then*

$$\mathrm{GRS}_k(\alpha, \nu) \star \mathrm{GRS}_\ell(\alpha, \omega) = \mathrm{GRS}_{\min(k+\ell-1,n)}(\alpha, \nu \star \omega).$$

Proof. By the definition of star product, we get that the star product is generated by elements of the form

$$(\nu_1\omega_1 p(\alpha_1)r(\alpha_1), \ldots, \nu_n\omega_n p(\alpha_n)r(\alpha_n)),$$

where $p(x), r(x) \in \mathbb{F}_q[x]$, $\deg p(x) < k$ and $\deg r(x) < \ell$. Now, $p(x)r(x)$ is a polynomial of degree at most $k + \ell - 2$, implying that the left side is included in the right side if $k + \ell - 1 \leq n$. On the other hand, if $k + \ell - 1 > n$, then $\mathrm{GRS}_n(\alpha, \nu \star \omega) = \mathbb{F}_q^n$, so the inclusion is obvious.

For the other direction, we get that $\mathrm{GRS}_{k+\ell-1}(\alpha, \nu \star \omega)$ is generated by elements of the form

$$(\nu_1\omega_1\alpha_1^i, \ldots, \nu_n\omega_n\alpha_n^i)$$

for $i = 0, \ldots, k+\ell-2$, because any polynomial can be reached by linear combinations of the monomials x^i. Now, $x^i = x^{j_1}x^{j_2}$, where $j_1 < k$ and $j_2 < \ell$. The evaluations of x^{j_1} are in $\mathrm{GRS}_k(\alpha, \nu)$ and evaluations of x^{j_2} are in $\mathrm{GRS}_\ell(\alpha, \omega)$. Therefore, the right side is included in the left side, which shows the equality. □

2.2 Information theory

Information theory considers random variables and the uncertainty an observer has about the value of a random variable. A reduction in the uncertainty of a random variable means that some new information about the variable has been obtained. The measure of this uncertainty is called entropy, and it is one of the most fundamental tools in information theory. This approach was first taken by Shannon in his seminal paper [7], which is why information-theoretic entropy is sometimes called Shannon entropy. In addition to studying basic information theory, we define the tools needed to prove a strong notion of information-theoretic security. A good general reference on information theory is [8].

All the random variables are assumed to be random variables on a probability space $(\Omega, \mathcal{F}, \mathrm{P})$. A random variable is the measurable function $\mathbf{X}\colon \Omega \to \mathcal{X}$, where $\mathcal{X}$ is the image of $\mathbf{X}$. A random variable is said to be *discrete* if the image $\mathcal{X}$ is countable. A random variable with a density function is said to be *continuous*.

2.2.1 Information theory for discrete random variables

Let us write $p(\cdot)$ to represent the probability mass function of a discrete random variable. We will use this notation freely, *i.e.*, if $\mathbf{X}$ and $\mathbf{Y}$ are random variables, then $p(x \mid y) = \mathrm{P}[\mathbf{X} = x \mid \mathbf{Y} = y]$ and so on. We will now use the probability mass function to define the entropy of a random variable.

Definition 2.2.1. Let $\mathbf{X}$ be a discrete random variable. Then the *entropy* of $\mathbf{X}$ is

$$H(\mathbf{X}) = -\sum_{x \in \mathcal{X}} p(x) \log p(x),$$

when defined. For convenience, we take "$0 \cdot \log(0) = 0$".

The logarithm is usually taken using base 2, which means that the unit of entropy is bits. To avoid the unnecessary cases where $p(x) = 0$, it is enough to restrict the image $\mathcal{X}$ to the support of $\mathbf{X}$.

Example 2.2.2. Let $\mathbf{X}$ represent the value of a fair dice. Then $\mathbf{X}$ has 6 outputs, all with probability $\frac{1}{6}$. The entropy of $\mathbf{X}$ is

$$H(\mathbf{X}) = -6 \cdot \frac{1}{6} \log_2 \left(\frac{1}{6}\right) = \log_2(6) \approx 2.58 \text{ bits}.$$

From the definition it is easy to see that the entropy does not depend on the values of $\mathbf{X}$, but only on the distribution of those values. As a result, for any injective function f defined on $\mathcal{X}$, $H(f(\mathbf{X})) = H(\mathbf{X})$. Another way of defining the entropy is by $H(\mathbf{X}) = -\mathbb{E}[\log p]$, where the expectation is taken over the probability distribution defined by p. These two approaches are the same, but it can be easier to use tools from probability theory when formulating using the expected value. The next lemma states an important lower bound for entropy.

Lemma 2.2.3. *Let $\mathbf{X}$ be a random variable. Then $H(\mathbf{X}) \geq 0$, with equality if and only if $\mathbf{X}$ is constant almost surely.*

Proof. For all $x \in \mathcal{X}$, $p(x) \log p(x) \leq 0$, since either $p(x) = 0$ or $0 < p(x) \leq 1$, which implies $\log p(x) \leq 0$. Therefore,

$$H(\mathbf{X}) = -\sum_{x \in \mathcal{X}} \underbrace{p(x) \log p(x)}_{\leq 0} \geq 0.$$

Assume that $\mathbf{X}$ is not constant almost surely. Thus, there exists $x' \in \mathcal{X}$ such that $0 < p(x') < 1$. Thus,

$$H(\mathbf{X}) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \geq -p(x') \log p(x') > 0.$$

On the other hand, assume that $\mathbf{X}$ is constant almost surely. Thus, there exists $x' \in \mathcal{X}$ such that $p(x') = 1$ and $p(x) = 0$ for all $x \in \mathcal{X} \setminus \{x'\}$. Therefore,

$$H(\mathbf{X}) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) = -p(x') \log p(x') = 1 \cdot \log 1 = 0.$$

□

The entropy of multiple random variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$ can be computed using their joint distribution, *i.e.*, the entropy is the entropy of the random vector $(\mathbf{X}_1, \ldots, \mathbf{X}_n)$. The next definition gives the conditional entropy of a random variable given some other random variable.

Definition 2.2.4. Let $\mathbf{X}$ and $\mathbf{Y}$ be discrete random variables. Then the *conditional entropy* of $\mathbf{X}$ given $\mathbf{Y}$ is

$$H(\mathbf{X} \mid \mathbf{Y}) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x \mid y).$$

It is worth noting that $H(\mathbf{X} \mid \mathbf{Y})$ and $H(\mathbf{X} \mid \mathbf{Y} = y)$ are not generally the same. The conditional entropy is the expected value of $H(\mathbf{X} \mid \mathbf{Y} = y)$ over all y. The following example continues the example from above.

Example 2.2.5. Let $\mathbf{X}$ represent the value of a fair dice and $\mathbf{Y}$ denote the parity of $\mathbf{X}$. Then

$$H(\mathbf{X} \mid \mathbf{Y}) = -6 \cdot \frac{1}{6} \log_2 \left(\frac{1}{3} \right) = \log_2(3) \approx 1.58 \text{ bits},$$

since $p(x \mid y) = \frac{p(x)p(y|x)}{p(y)} = \frac{1/6}{1/2} = \frac{1}{3}$, when $y = x \bmod 2$.

The conditional entropy means the uncertainty of a random variable given some other random variable. It is intuitive that this uncertainty is smaller than if we were not conditioning on the other random variable. If the conditional entropy is zero, then there is no uncertainty in the value of $\mathbf{X}$ when given the value of $\mathbf{Y}$. Hence, $\mathbf{X}$ is a random variable that is completely determined by $\mathbf{Y}$. The following lemma formalizes this notion.

Lemma 2.2.6. *Let $\mathbf{X}$, $\mathbf{Y}$ be discrete random variables. Then*

(i) $0 \leq H(\mathbf{X} \mid \mathbf{Y})$*, with equality if and only if* $\mathbf{X} = f(\mathbf{Y})$ *for some* f *almost surely, and*

(ii) $H(\mathbf{X} \mid \mathbf{Y}) \leq H(\mathbf{X})$*, with equality if and only if* $\mathbf{X}$ *and* $\mathbf{Y}$ *are independent.*

Proof.

(i) For all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, we get that $p(x, y) \log p(x \mid y) \leq 0$, because either $p(x, y) = 0$ or $p(x \mid y) \leq 1$, which implies $\log p(x \mid y) \leq 0$. Hence,

$$H(\mathbf{X} \mid \mathbf{Y}) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \underbrace{p(x, y) \log p(x \mid y)}_{\leq 0} \geq 0.$$

If $H(\mathbf{X}, \mathbf{Y}) = 0$, then $p(x, y) \log p(x \mid y) = 0$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Therefore, either $p(x, y) = 0$ or $p(x \mid y) = 1$. Consider all the pairs (x, y) with positive probability. We must have that $p(x \mid y) = 1$, *i.e.*, x is uniquely determined given y. All y with positive probability must then have a unique x associated to it. Hence, we may define a map $f \colon \mathcal{Y} \to \mathcal{X}$ for almost all $y \in \mathcal{Y}$. Therefore, $\mathbf{X} = f(\mathbf{Y})$.

(ii) Using the definition of conditional entropy, we get that

$$\begin{aligned}
H(\mathbf{X} \mid \mathbf{Y}) &= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x)p(y \mid x)}{p(y)} \\
&= -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x)p(y \mid x)\log p(x) + \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(y)}{p(y \mid x)} \\
&\leq -\sum_{x\in\mathcal{X}}\left(\sum_{y\in\mathcal{Y}} p(y \mid x)\right) p(x)\log p(x) \\
&\quad + \log e \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x)p(y \mid x)\left(\frac{p(y)}{p(y \mid x)} - 1\right) \\
&= H(\mathbf{X}) + \log e \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} (p(x)p(y) - p(x,y)) \\
&= H(\mathbf{X}).
\end{aligned}$$

The inequality follows from the elementary inequality $\log x \leq \log e(x-1)$. Equality is achieved if and only if $\frac{p(y)}{p(y|x)} - 1 = 0$ almost everywhere. Hence, $p(y) = p(y \mid x)$ for almost all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Therefore, equality is achieved if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent.

□

Proposition 2.2.7. *Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be discrete random variables. Then*

$$H(\mathbf{X}_1, \ldots, \mathbf{X}_n) = \sum_{i=1}^{n} H(\mathbf{X}_i \mid \mathbf{X}_1, \ldots, \mathbf{X}_{i-1})$$

Proof. Let us show the claim for $n = 2$. Consider

$$\begin{aligned}
H(\mathbf{X}_1, \mathbf{X}_2) &= -\sum_{x_1\in\mathcal{X}_1}\sum_{x_2\in\mathcal{X}_2} p(x_1,x_2)\log p(x_1,x_2) \\
&= -\sum_{x_1\in\mathcal{X}_1}\sum_{x_2\in\mathcal{X}_2} p(x_1,x_2)\log p(x_1)p(x_2 \mid x_1) \\
&= -\sum_{x_1\in\mathcal{X}_1}\sum_{x_2\in\mathcal{X}_2} p(x_1,x_2)\log p(x_1) - \sum_{x_1\in\mathcal{X}_1}\sum_{x_2\in\mathcal{X}_2} p(x_1,x_2)\log p(x_2 \mid x_1) \\
&= -\sum_{x_1\in\mathcal{X}_1}\Big(\sum_{x_2\in\mathcal{X}_2} p(x_2 \mid x_1)\Big)p(x_1)\log p(x_1) + H(\mathbf{X}_2 \mid \mathbf{X}_1) \\
&= H(\mathbf{X}_1) + H(\mathbf{X}_2 \mid \mathbf{X}_1).
\end{aligned}$$

Now, assume that the claim holds for $n = k$. Thus,

$$\begin{aligned}
H(\mathbf{X}_1, \ldots, \mathbf{X}_{k+1}) &= H(\mathbf{X}_1, \ldots, \mathbf{X}_k) + H(\mathbf{X}_{k+1} \mid \mathbf{X}_1, \ldots, \mathbf{X}_k) \\
&= \sum_{i=1}^{k} H(\mathbf{X}_i \mid \mathbf{X}_1, \ldots, \mathbf{X}_{i-1}) + H(\mathbf{X}_{k+1} \mid \mathbf{X}_1, \ldots, \mathbf{X}_k) \\
&= \sum_{i=1}^{k+1} H(\mathbf{X}_i \mid \mathbf{X}_1, \ldots, \mathbf{X}_{i-1}),
\end{aligned}$$

by the $n = 2$ case and the induction hypothesis. Hence, by induction, the claim holds for all $n \geq 2$. □

As a corollary to the previous proposition and Lemma 2.2.6(ii), we get the union bound for entropies.

Corollary 2.2.8. *Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be discrete random variables. Then*

$$H(\mathbf{X}_1, \ldots, \mathbf{X}_n) \leq \sum_{i=1}^{n} H(\mathbf{X}_i).$$

In addition to considering independent variables, we may consider random variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ such that $\mathbf{X}$ is conditionally independent of $\mathbf{Z}$ given $\mathbf{Y}$. Hence, $H(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}) = H(\mathbf{X} \mid \mathbf{Y})$. One important concept is the mutual information between two random variables, which describes how much the knowledge of one random variable reduces the uncertainty of another random variable.

Definition 2.2.9. Let $\mathbf{X}, \mathbf{Y}$ be random variables. Then the *mutual information* between $\mathbf{X}$ and $\mathbf{Y}$ is defined as

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}).$$

Example 2.2.10. Let $\mathbf{X}$ be a random variable of a fair dice and $\mathbf{Y}$ the parity of that dice. Then the mutual information between $\mathbf{X}$ and $\mathbf{Y}$ is

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}) = 1 \text{ bit},$$

as demonstrated in Examples 2.2.2 and 2.2.5.

Proposition 2.2.11. *Let $\mathbf{X}$ and $\mathbf{Y}$ be random variables. Then*

(i) $I(\mathbf{X}; \mathbf{Y}) = I(\mathbf{Y}; \mathbf{X})$,

(ii) $I(\mathbf{X}; \mathbf{Y}) \leq H(\mathbf{X})$, *with equality if and only if* $\mathbf{X} = f(\mathbf{Y})$ *for some* f *almost surely, and*

(iii) $I(\mathbf{X}; \mathbf{Y}) \geq 0$ *with equality if and only if* $\mathbf{X}$ *and* $\mathbf{Y}$ *are independent.*

Proof. (i) Using Proposition 2.2.7 we get that

$$H(\mathbf{X}) + H(\mathbf{Y} \mid \mathbf{X}) = H(\mathbf{X}, \mathbf{Y}) = H(\mathbf{Y}, \mathbf{X}) = H(\mathbf{Y}) + H(\mathbf{X} \mid \mathbf{Y}).$$

Therefore, using the definition of mutual information we get

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X}) = I(\mathbf{Y}; \mathbf{X}).$$

(ii) Using Lemma 2.2.6(i), we obtain that $I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}) \leq H(\mathbf{X})$, with equality if and only if $\mathbf{X} = f(\mathbf{Y})$ for some f almost surely.

(iii) Using Lemma 2.2.6(ii), we get that $I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y}) \geq 0$, with equality if and only if $\mathbf{X}$ and $\mathbf{Y}$ are independent.

□

Mutual information describes how much information $\mathbf{Y}$ leaks about $\mathbf{X}$. According to the previous proposition, this equals the amount of information $\mathbf{X}$ leaks about $\mathbf{Y}$, since mutual information is symmetric. The amount of information leaked is zero if and only if the random variables are independent, which is expected, since independent random variables do not reveal anything about each other. If $\mathbf{X}$ is a random variable, then $I(\mathbf{X};\mathbf{X}) = H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{X}) = H(\mathbf{X})$. This means that the entropy of a random variable describes its self-information. According to the above proposition, the amount of information leaked about $\mathbf{X}$ is no more than the information contained in $\mathbf{X}$.

The next proposition describes how much information $f(\mathbf{Y})$ reveals about $\mathbf{X}$. In this case, $f(\mathbf{Y})$ is a processed version of $\mathbf{Y}$, and the inequality states that no more information can be gained from the knowledge of $\mathbf{Y}$ by processing it in some way. Therefore, mutual information describes a fundamental limit and is not dependent on how cleverly the data is processed.

Proposition 2.2.12 (Data processing inequality)**.** *Let $\mathbf{X} \to \mathbf{Y} \to \mathbf{Z}$ be a Markov chain. Then*

$$I(\mathbf{X};\mathbf{Z}) \leq I(\mathbf{X};\mathbf{Y}).$$

In particular, if f is some function on $\mathcal{Y}$, then

$$I(\mathbf{X}; f(\mathbf{Y})) \leq I(\mathbf{X};\mathbf{Y}).$$

Proof. We can express the entropy of $(\mathbf{X},\mathbf{Y},\mathbf{Z})$ in two ways using Proposition 2.2.7

$$\begin{aligned} H(\mathbf{X},\mathbf{Y},\mathbf{Z}) &= H(\mathbf{X}) + H(\mathbf{Z} \mid \mathbf{X}) + H(\mathbf{Y} \mid \mathbf{X},\mathbf{Z}) \\ &= H(\mathbf{Z}) + H(\mathbf{Y} \mid \mathbf{Z}) + H(\mathbf{X} \mid \mathbf{Y}). \end{aligned}$$

The last term on the second line comes from the fact that $\mathbf{X}$ is conditionally independent of $\mathbf{Z}$ given $\mathbf{Y}$, since $\mathbf{X} \to \mathbf{Y} \to \mathbf{Z}$ is a Markov chain. Then we can write the mutual information as

$$\begin{aligned} I(\mathbf{X};\mathbf{Z}) &= H(\mathbf{Z}) - H(\mathbf{Z} \mid \mathbf{X}) \\ &= \underbrace{H(\mathbf{X}) - H(\mathbf{X} \mid \mathbf{Y})}_{I(\mathbf{X};\mathbf{Y})} - \underbrace{(H(\mathbf{X}) + H(\mathbf{Z} \mid \mathbf{X}) + H(\mathbf{Y} \mid \mathbf{X},\mathbf{Z}))}_{H(\mathbf{X},\mathbf{Y},\mathbf{Z})} \\ &\quad + \underbrace{(H(\mathbf{Z}) + H(\mathbf{Y} \mid \mathbf{Z}) + H(\mathbf{X} \mid \mathbf{Y}))}_{H(\mathbf{X},\mathbf{Y},\mathbf{Z})} + \underbrace{H(\mathbf{Y} \mid \mathbf{X},\mathbf{Z}) - H(\mathbf{Y} \mid \mathbf{Z})}_{\leq 0} \\ &\leq I(\mathbf{X};\mathbf{Y}), \end{aligned}$$

where the inequality follows from Lemma 2.2.6. The second part of the claim follows from the fact that $\mathbf{X} \to \mathbf{Y} \to f(\mathbf{Y})$ is a Markov chain, since $f(\mathbf{Y})$ is conditionally independent of $\mathbf{X}$ given $\mathbf{Y}$. □

A useful result when working with entropy and mutual information is that there exists an upper bound for the entropy of random variables with a finite image.

Proposition 2.2.13. *Let* $\mathbf{X}$ *be a discrete random variable such that* $\mathcal{X}$ *is finite. Then*

$$H(\mathbf{X}) \leq \log|\mathcal{X}|,$$

with equality if and only if $\mathbf{X}$ *is uniformly distributed over* $\mathcal{X}$*.*

Proof. Consider

$$\begin{aligned} H(\mathbf{X}) - \log|\mathcal{X}| &= \sum_{x\in\mathcal{X}} p(x) \log \frac{1}{p(x)} - \sum_{x\in\mathcal{X}} p(x) \log|\mathcal{X}| \\ &= \sum_{x\in\mathcal{X}} p(x) \log \frac{1}{p(x)|\mathcal{X}|} \\ &\leq \log e \sum_{x\in\mathcal{X}} p(x) \left(\frac{1}{p(x)|\mathcal{X}|} - 1\right) \\ &= \log e \sum_{x\in\mathcal{X}} \left(\frac{1}{|\mathcal{X}|} - p(x)\right) \\ &= 0, \end{aligned}$$

where the inequality follows from $\log x \leq \log e(x-1)$. Equality is achieved if and only if $p(x)|\mathcal{X}| = 1$ for $x \in \mathcal{X}$, *i.e.*, if $\mathbf{X}$ is uniformly distributed. □

We will often use the above proposition when $\mathbf{X}$ is a random variable over $\mathbb{F}_q^n$. Thus, $H(\mathbf{X}) \leq n \log q$.

2.2.2 Information theory for continuous random variables

So far we have discussed discrete random variables, but it is possible to define all of the concepts to continuous random variables taking values in $\mathbb{R}$ or $\mathbb{C}$. A real random variable is said to be *continuous* if it has a probability density function. Sometimes this property is known as *absolutely continuous*, since the probability measure defined by the real random variable $\mathbf{X}$ is absolutely continuous with the Lebesgue measure. The probability density function of $\mathbf{X}$ is denoted by $p(\cdot)$.

Definition 2.2.14. The *differential entropy* of a continuous real random variable $\mathbf{X}$ with density function $p(\cdot)$ is defined as

$$h(\mathbf{X}) = -\int_{\mathbb{R}} p(x) \log p(x)\, \mathrm{d}x,$$

when it exists.

To avoid the special cases of $p(x) = 0$, we may restrict the integral to the support of $\mathbf{X}$. In contrast to the entropy of discrete random variables, which is always nonnegative, differential entropy can be negative. Similarly, the entropy is not preserved under arbitrary injective functions, but it is preserved under measure preserving functions, such as translations. Hence, $h(\mathbf{X} + a) = h(\mathbf{X})$ for all $a \in \mathbb{R}$.

Example 2.2.15. Let $\mathbf{X}$ be an n-dimensional real random vector with a normal distribution, *i.e.*, $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$, where $\mu \in \mathbb{R}^n$ is the mean and $\Sigma \in \mathbb{R}^{n\times n}$ is the nonsingular covariance matrix of $\mathbf{X}$. The density function of $\mathbf{X}$ is then

$$p(x) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(\Sigma)^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}.$$

Then the entropy of $\mathbf{X}$ can be computed as

$$\begin{aligned}
&h(\mathbf{X}) \\
&= -\frac{1}{\ln 2} \int_{\mathbb{R}^n} p(x) \ln p(x) \, \mathrm{d}x \\
&= \frac{1}{2 \ln 2} \left(\int_{\mathbb{R}^n} p(x)(x-\mu)^T \Sigma^{-1}(x-\mu) \, \mathrm{d}x + \int_{\mathbb{R}^n} p(x) \ln((2\pi)^n \det(\Sigma)) \, \mathrm{d}x \right) \\
&= \frac{1}{2 \ln 2} \left(\int_{\mathbb{R}^n} p(x) \sum_{i,j\in[n]} (x_i - \mu_i)(\Sigma^{-1})_{ij}(x_j - \mu_j) \, \mathrm{d}x + \ln((2\pi)^n \det(\Sigma)) \int_{\mathbb{R}^n} p(x) \, \mathrm{d}x \right) \\
&= \frac{1}{2 \ln 2} \left(\sum_{i,j\in[n]} (\Sigma^{-1})_{ij} \int_{\mathbb{R}^n} p(x)(x_i - \mu_i)(x_j - \mu_j) \, \mathrm{d}x + \ln((2\pi)^n \det(\Sigma)) \right) \\
&= \frac{1}{2 \ln 2} \left(\sum_{i,j\in[n]} (\Sigma^{-1})_{ij} \Sigma_{ji} + \ln((2\pi)^n \det(\Sigma)) \right) \\
&= \frac{1}{2 \ln 2} \left(\sum_{i=1}^{n} I_{ii} + \ln((2\pi)^n \det(\Sigma)) \right) \\
&= \frac{1}{2 \ln 2} \Big(n + \ln((2\pi)^n \det(\Sigma)) \Big) \\
&= \frac{1}{2 \ln 2} \ln((2\pi e)^n \det(\Sigma)) \\
&= \frac{1}{2} \log((2\pi e)^n \det(\Sigma)) \text{ bits.}
\end{aligned}$$

If two continuous random variables have a joint distribution, then we may define the conditional entropy similarly to the discrete random variables.

Definition 2.2.16. Let $\mathbf{X}$ and $\mathbf{Y}$ be continuous real random variables with a joint density function. Then the *conditional entropy* of $\mathbf{X}$ given $\mathbf{Y}$ is

$$h(\mathbf{X} \mid \mathbf{Y}) = -\int_{\mathbb{R}^2} p(x, y) \log p(x \mid y) \, \mathrm{d}(x, y).$$

Even though many results from discrete random variables do not carry over to continuous random variables, there are still many results that work. The next proposition is similar to Lemma 2.2.6(ii). The proof works in the same way, though care needs to be taken to justify convergence and switching the order of the double integrals. The most useful tool for these is Fubini's theorem, which states that a double integral can be switched if the integrand is integrable.

Proposition 2.2.17. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be continuous real random variables with a joint density function, and let* $\mathbf{X}$ *have finite entropy. Then*

$$h(\mathbf{X} \mid \mathbf{Y}) \leq h(\mathbf{X}),$$

with equality if and only if $\mathbf{X}$ *and* $\mathbf{Y}$ *are independent.*

Proof. Using the definition of conditional entropy, we get

$$\begin{aligned}
h(\mathbf{X} \mid \mathbf{Y}) &= -\int_{\mathbb{R}^2} p(x,y) \log \frac{p(x)p(y \mid x)}{p(y)} \,\mathrm{d}(x,y) \\
&= -\int_{\mathbb{R}^2} p(x,y) \log p(x) \,\mathrm{d}(x,y) + \int_{\mathbb{R}^2} p(x,y) \log \frac{p(y)}{p(y \mid x)} \,\mathrm{d}(x,y) \\
&\leq -\int_{\mathbb{R}} \int_{\mathbb{R}} p(x)p(y \mid x) \log p(x) \,\mathrm{d}y \,\mathrm{d}x \\
&\quad + \log e \int_{\mathbb{R}^2} p(x)p(y \mid x) \left(\frac{p(y)}{p(y \mid x)} - 1 \right) \mathrm{d}(x,y) \\
&= -\int_{\mathbb{R}} \underbrace{\int_{\mathbb{R}} p(y \mid x) \,\mathrm{d}y}_{=1} \; p(x) \log p(x) \,\mathrm{d}x + \log e \int_{\mathbb{R}^2} p(x)p(y) - p(x,y) \,\mathrm{d}(x,y) \\
&= -\int_{\mathbb{R}} p(x) \log p(x) \,\mathrm{d}x + \log e \Big(\underbrace{\int_{\mathbb{R}} p(x) \,\mathrm{d}x}_{=1} \underbrace{\int_{\mathbb{R}} p(y) \,\mathrm{d}y}_{=1} - \underbrace{\int_{\mathbb{R}^2} p(x,y) \,\mathrm{d}(x,y)}_{=1} \Big) \\
&= h(\mathbf{X}).
\end{aligned}$$

The inequality follows from $\log x \leq \log e(x-1)$, with equality if and only if $p(y) = p(y \mid x)$, *i.e.*, if $\mathbf{X}$ and $\mathbf{Y}$ are independent. □

Mutual information can now be defined for continuous random variables. The definition presented here is a bit different from Definition 2.2.9, but the result is the same.

Definition 2.2.18. Let $\mathbf{X}$ and $\mathbf{Y}$ be continuous real random variables with a joint density function. Then the *mutual information* between $\mathbf{X}$ and $\mathbf{Y}$ is

$$I(\mathbf{X}; \mathbf{Y}) = \int_{\mathbb{R}^2} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \,\mathrm{d}(x,y),$$

when defined.

From the definition it is clear that mutual information is symmetric. Furthermore, we get the following lemma.

Lemma 2.2.19. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be continuous real random variables with a joint density function. Then*

$$I(\mathbf{X}; \mathbf{Y}) = h(\mathbf{X}) - h(\mathbf{X} \mid \mathbf{Y}).$$

Proof. The lemma follows directly from the definition using the following equalities:

$$\begin{aligned} I(\mathbf{X};\mathbf{Y}) &= \int_{\mathbb{R}^2} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \,\mathrm{d}(x,y) \\ &= \int_{\mathbb{R}^2} p(x,y) \log \frac{p(x,y)}{p(y)} \,\mathrm{d}(x,y) - \int_{\mathbb{R}^2} p(x,y) \log p(x) \,\mathrm{d}(x,y) \\ &= -h(\mathbf{X} \mid \mathbf{Y}) + \int_{\mathbb{R}} \int_{\mathbb{R}} p(y \mid x) \,\mathrm{d}y \; p(x) \log p(x) \,\mathrm{d}x \\ &= h(\mathbf{X}) - h(\mathbf{X} \mid \mathbf{Y}). \end{aligned}$$

Splitting the double integral is justified by Fubini's theorem with a similar argument as in Proposition 2.2.17. □

Even though we cannot bound the differential entropy from below, we can still bound the mutual information from below. For discrete random variables, we were able to say that the entropy of a random variable equals the self-information of the variable. This is not the case for continuous random variables, since a continuous random variable $\mathbf{X}$ does not have a joint density with itself.

Proposition 2.2.20. *Let $\mathbf{X}$ and $\mathbf{Y}$ be continuous random variables with joint density. Then $I(\mathbf{X};\mathbf{Y}) \geq 0$.*

Proof. The proposition follows directly from $I(\mathbf{X};\mathbf{Y}) = h(\mathbf{X}) - h(\mathbf{X} \mid \mathbf{Y})$ and Proposition 2.2.17. □

The maximal entropy for finite random variables is achieved by a uniform distribution, as seen in Proposition 2.2.13. For real random variables, such a distribution does not exist, so entropy is maximized by a different distribution. The following proof has been presented in [8, Theorem 8.6.5] and [9, Theorem 7.4.1].

Proposition 2.2.21. *Let $\mathbf{X}$ be an n-dimensional real random vector with a joint density function, mean μ, and nonsingular covariance Σ. Then*

$$h(\mathbf{X}) \leq \frac{1}{2} \log((2\pi e)^n \det(\Sigma)),$$

with equality if and only if $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$, where $\mathcal{N}$ denotes a normal distribution with a given mean and covariance matrix.

Proof. Let $p(\cdot)$ be the density function of $\mathbf{X}$, and let $\mathbf{Y} \sim \mathcal{N}(\mu, \Sigma)$ with density function $r(\cdot)$. The random vectors $\mathbf{X}$ and $\mathbf{Y}$ have the same covariance by construction, so we get

$$\begin{aligned} \Sigma_{ij} &= \int_{\mathbb{R}^n} p(x)(x_i - \mu_i)(x_j - \mu_j) \,\mathrm{d}x \\ &= \int_{\mathbb{R}^n} r(x)(x_i - \mu_i)(x_j - \mu_j) \,\mathrm{d}x. \end{aligned}$$

Now, the integral of any quadratic form in $x - \mu$ is the same if integrated with $p(\cdot)$ or $r(\cdot)$. We can use the above because $\log r(x)$ is a quadratic form in $x - \mu$, as is seen easily from the definition of $r(x)$. Then we get that

$$\begin{aligned} h(\mathbf{X}) - h(\mathbf{Y}) &= -\int_{\mathbb{R}^n} p(x) \log p(x) \, \mathrm{d}x + \int_{\mathbb{R}^n} r(x) \log r(x) \, \mathrm{d}x \\ &= -\int_{\mathbb{R}^n} p(x) \log p(x) \, \mathrm{d}x + \int_{\mathbb{R}^n} p(x) \log r(x) \, \mathrm{d}x \\ &= \int_{\mathbb{R}^n} p(x) \log \frac{r(x)}{p(x)} \, \mathrm{d}x \\ &\leq \log e \int_{\mathbb{R}^n} r(x) - p(x) \, \mathrm{d}x \\ &= 0. \end{aligned}$$

Equality is achieved if $\frac{r(x)}{p(x)} = 1$ is constant almost everywhere, *i.e.*, $\mathbf{X}$ has the same distribution as $\mathbf{Y}$. Therefore, the upper bound follows from the computation in Example 2.2.15, with equality if and only if $\mathbf{X} \sim \mathcal{N}(\mu, \Sigma)$. □

In addition to real random variables, we can also have complex valued random variables. The entropy of a continuous random variable is defined as the joint entropy of the real and imaginary parts, given that they have a joint density function. Therefore, if $\mathbf{Z}$ is a complex random variable, then

$$h(\mathbf{Z}) = h(\mathrm{Re}(\mathbf{Z}), \mathrm{Im}(\mathbf{Z})).$$

All of the above definitions for real random variables are then naturally extended to complex random variables. The imaginary unit is denoted by $\imath$.

The mean of a complex random vector is defined in the same way that the mean of a real random vector is, namely, $\mu = \mathbb{E}[\mathbf{Z}]$. The covariance of a complex random vector is defined as $\Sigma = \mathbb{E}[(\mathbf{Z} - \mu)(\mathbf{Z} - \mu)^*]$, where $(\,\cdot\,)^*$ denotes the conjugate transpose of a matrix. In addition to these, an additional *pseudo-covariance* can be defined as $\widetilde{\Sigma} = \mathbb{E}[(\mathbf{Z} - \mu)(\mathbf{Z} - \mu)^T]$. Clearly, the pseudo-covariance of a real random vector equals its covariance. A complex random vector with zero pseudo-covariance is said to be *proper*.

By setting $\mathbf{X} = \mathrm{Re}(\mathbf{Z})$ and $\mathbf{Y} = \mathrm{Im}(\mathbf{Z})$, the covariance matrix and pseudo-covariance matrix can be related to the covariance matrices of the real and imaginary parts. Then we can write

$$\begin{aligned} \Sigma &= \Sigma_{XX} + \Sigma_{YY} + \imath(\Sigma_{YX} - \Sigma_{XY}), \\ \tilde{\Sigma} &= \Sigma_{XX} - \Sigma_{YY} + \imath(\Sigma_{YX} + \Sigma_{XY}), \end{aligned}$$

where Σ_{XX} and Σ_{YY} are the covariance matrices of $\mathbf{X}$ and $\mathbf{Y}$, respectively, and Σ_{YX} and Σ_{XY} are the cross-covariance matrices of $\mathbf{X}$ and $\mathbf{Y}$. These blocks have the properties $\Sigma_{XX} = \Sigma_{XX}^T$, $\Sigma_{YY} = \Sigma_{YY}^T$, and $\Sigma_{YX} = \Sigma_{XY}^T$. A complex random variable is proper if and only if $\Sigma_{XX} = \Sigma_{YY}$ and $\Sigma_{YX} = -\Sigma_{XY}$.

A complex random variable $\mathbf{Z}$ is said to be *circularly symmetric* if $\mathbf{Z} \sim e^{\imath\theta}\mathbf{Z}$ for all $\theta \in \mathbb{R}$. It is clear that a circularly symmetric random variable has zero mean and zero pseudo-covariance, which means that it is proper.

A complex random vector is said to be *complex normally distributed* if the real random vector consisting of the real and imaginary parts of $\mathbf{Z}$ is normally distributed, and it is denoted by $\mathbf{Z} \sim \mathcal{CN}(\mu, \Sigma, \widetilde{\Sigma})$.

The maximal entropy for complex valued random vectors is similar to real valued random vectors in Proposition 2.2.21. However, the additional requirement is that the pseudo-covariance of an entropy maximizing distribution is zero. This means that no real valued random vector can maximize entropy when complex valued random vectors are also considered. The maximal entropy theorem can be proven like in Proposition 2.2.21, where one has to use the density function for a circular complex normal distribution. The following proof highlights the necessity for the properness of an entropy maximizing distribution. The one-dimensional version was presented in [10, Theorem 2].

Theorem 2.2.22. *Let $\mathbf{Z}$ be an n-dimensional complex random vector with a joint density function, mean μ, and nonsingular covariance Σ. Then*

$$h(\mathbf{Z}) \leq \log((\pi e)^n \det(\Sigma)),$$

with equality if and only if $\mathbf{X} \sim \mathcal{CN}(\mu, \Sigma, 0)$, that is, equality is reached only with a proper complex normal distribution.

Proof. Let $\mathbf{W} = (\mathbf{X}, \mathbf{Y})^T = (\mathrm{Re}(\mathbf{Z}), \mathrm{Im}(\mathbf{Z}))^T$ be the $2n$-dimensional real random vector that contains the real and imaginary parts of $\mathbf{Z}$. By definition and Proposition 2.2.21, we get

$$h(\mathbf{Z}) = h(\mathbf{W}) \leq \frac{1}{2} \log((2\pi e)^{2n} \det(\Sigma_W)).$$

The covariance matrix of $\mathbf{W}$ is nonsingular and can be written as

$$\Sigma_W = \begin{pmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{pmatrix}.$$

Now,

$$\begin{aligned}
&\det(\Sigma_W)^2 \\
&= \det \begin{pmatrix} \Sigma_{XX} & -\imath\Sigma_{XY} \\ \imath\Sigma_{YX} & \Sigma_{YY} \end{pmatrix} \det \begin{pmatrix} \Sigma_{YY} & \imath\Sigma_{YX} \\ -\imath\Sigma_{XY} & \Sigma_{XX} \end{pmatrix} \\
&\leq \left(\frac{1}{2^{2n}} \det \begin{pmatrix} \Sigma_{XX} + \Sigma_{YY} & \imath(\Sigma_{YX} - \Sigma_{XY}) \\ \imath(\Sigma_{YX} - \Sigma_{XY}) & \Sigma_{XX} + \Sigma_{YY} \end{pmatrix} \right)^2 \\
&= \left(\frac{1}{2^{2n}} \det(\Sigma_{XX} + \Sigma_{YY} + \imath(\Sigma_{YX} - \Sigma_{XY})) \det(\Sigma_{XX} + \Sigma_{YY} - \imath(\Sigma_{YX} - \Sigma_{XY})) \right)^2 \\
&= \left(\frac{1}{2^{2n}} \det(\Sigma)^2 \right)^2.
\end{aligned}$$

The first determinant in the first line follows by multiplying the first n rows of the matrix by $-\imath$, and the first n columns by $\imath$. These produce a factor of $(-\imath)^n \imath^n = 1$, and the upper left block gets multiplied by $(-\imath) \cdot \imath = 1$. The second determinant on

the first line follows by permuting the rows and columns of the previous determinant. The matrices on the first line are positive semi-definite, since

$$\begin{pmatrix} u^* & v^* \end{pmatrix} \begin{pmatrix} \Sigma_{XX} & -\imath\Sigma_{XY} \\ \imath\Sigma_{YX} & \Sigma_{YY} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} (-\imath u)^* & v^* \end{pmatrix} \begin{pmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{pmatrix} \begin{pmatrix} -\imath u \\ v \end{pmatrix} \geq 0,$$

since Σ_W is positive semi-definite. A similar argument works for the other matrix. Hence, we can use the inequality from Lemma A.5. Equality is reached if and only if $\Sigma_{XX} = \Sigma_{YY}$ and $\Sigma_{YX} = -\Sigma_{XY}$, *i.e.*, if and only if $\mathbf{Z}$ is proper. The equality on line three follows from Lemma A.2. On the third line we have the determinants of Σ and Σ^T, whose product is $\det(\Sigma)^2$. Taking square roots of the above yields

$$\det(\Sigma_W) \leq \frac{\det(\Sigma)^2}{2^{2n}}.$$

Hence,

$$h(\mathbf{Z}) \leq \frac{1}{2} \log((2\pi e)^{2n} \frac{1}{2^{2n}} \det(\Sigma)^2) = \log((\pi e)^n \det(\Sigma)),$$

with equality if and only if $\mathbf{Z}$ is proper and complex normally distributed. □

2.2.3 Information-theoretic security

Information theory can be used to characterize the security of an encryption scheme. If $\mathbf{X}$ is the message that needs to be kept secret and $\mathbf{Y}$ is the encryption of $\mathbf{X}$, then the encryption scheme is information-theoretically secure if the encryption reveals no information about the plaintext.

Definition 2.2.23. If $\mathbf{X}$ is the plaintext and $\mathbf{Y}$ is the ciphertext of some encryption scheme, then the scheme is said to be *information-theoretically secure* if

$$I(\mathbf{X}; \mathbf{Y}) = 0.$$

The above definition is also known as *perfect security*.

Example 2.2.24 (One-time pad)**.** Let $x \in \mathbb{F}_q^n$ denote some arbitrary data. Then choose $y \in \mathbb{F}_q^n$ uniformly at random. The value $x + y$ is an encryption of x, since x can be recovered by computing $(x + y) - y$, when the secret key y is known. The drawback of such a scheme is that the key y has the same size as the data x. The security of this scheme is shown in the next proposition.

It was shown in Shannon's seminal paper [11] that the only way to achieve information-theoretic security in an encryption system is by having the same properties as the one-time pad, *i.e.*, the key has to be the same size as the data.

Proposition 2.2.25. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be independent random variables over some finite additive group* G *and* $\mathbf{Y}$ *be uniformly distributed. Then*

$$I(\mathbf{X} + \mathbf{Y}; \mathbf{X}) = 0.$$

Proof. By definition of mutual information, we get that

$$I(\mathbf{X}+\mathbf{Y};\mathbf{X}) = H(\mathbf{X}+\mathbf{Y}) - H(\mathbf{X}+\mathbf{Y} \mid \mathbf{X}).$$

Let us analyze both terms separately. The probability mass function of $\mathbf{Z} = \mathbf{X}+\mathbf{Y}$ is

$$p(z) = \sum_{\substack{x,y\in G \\ x+y=z}} p(x,y) = \sum_{x\in G} p(x)p(z-x) = \frac{1}{|G|},$$

which follows from the independence of $\mathbf{X}$ and $\mathbf{Y}$ and the fact that $\mathbf{Y}$ is uniformly distributed. As a result, $\mathbf{X}+\mathbf{Y}$ is uniformly distributed. Hence, $H(\mathbf{X}+\mathbf{Y}) = \log|G|$ by Proposition 2.2.13.

For the second term we get

$$\begin{aligned} H(\mathbf{X}+\mathbf{Y} \mid \mathbf{X}) &= -\sum_{x\in G}\sum_{y\in G} p(x+y,x)\log p(x+y \mid x) \\ &= -\sum_{x\in G}\sum_{y\in G} p(x)p(y)\log p(y) \\ &= -\sum_{y\in G} p(y)\log p(y) \\ &= H(\mathbf{Y}). \end{aligned}$$

As $\mathbf{Y}$ is uniformly distributed, we get that

$$I(\mathbf{X}+\mathbf{Y};\mathbf{X}) = \log|G| - \log|G| = 0.$$

□

According to the above proposition, we are able to hide data by adding uniform noise to it. We will use this technique later to show the security of multiple different schemes.

Corollary 2.2.26. *Let* $\mathbf{X}$ *and* $\mathbf{Y}$ *be as in the previous proposition, and* f *some function. Then*

$$I(f(\mathbf{X})+\mathbf{Y};\mathbf{X}) = 0.$$

The result follows directly from Propositions 2.2.12 and 2.2.25.

2.3 Secret sharing

Secret sharing is a way of splitting a secret, usually denoted by a finite field element, into multiple shares, which are distributed to N parties. The parties can recover the original secret by combining their shares. Secret sharing was first introduced by Shamir in his seminal paper [12]. Secret sharing schemes are built such that the parties in some specified access structure are able to recover the secret, while parties that are not in the access structure will not gain any information about the secret. The simplest kind of access structure is that of threshold secret sharing.

Definition 2.3.1. A (K, N)-*threshold secret sharing scheme* is defined such that each of the N participants has a share $\tilde{a}_i$ such that any set of K parties can decode the secret value, but no information about the secret is gained by any set of at most $K-1$ parties.

In particular, we are interested in linear secret sharing schemes, where the encoding and decoding functions are linear. From the information theoretic definition of perfect secrecy, it is clear that a secret sharing scheme will need to use randomness to hide the secrets.

Example 2.3.2. Let $a \in \mathbb{F}_q$ be a secret value. Consider the $(2,3)$-threshold secret sharing scheme, which works as follows. Let $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{F}_q^\times$ be distinct nonzero values. Then the share of party i will be

$$\tilde{a}_i = a + r\alpha_i,$$

where r is chosen uniformly at random from $\mathbb{F}_q$. Any two parties, i and j, can now decode the secret by computing

$$\frac{\tilde{a}_i\alpha_j - \tilde{a}_j\alpha_i}{\alpha_j - \alpha_i} = \frac{(\alpha_j - \alpha_i)a}{\alpha_j - \alpha_i} = a,$$

since any two evaluation points are distinct. As r is uniformly distributed, then $r\alpha_i$ is uniformly distributed as α is nonzero. Therefore,

$$I(\mathbf{a}; \tilde{\mathbf{a}}_i) = I(\mathbf{a}; \mathbf{a} + \mathbf{r}\alpha) = 0$$

by Proposition 2.2.25. Therefore, any single party will gain no information about a, while any two parties will be able to recover a fully.

The previous example uses the $[3,2]$ GRS code to encode the secret and the randomness to the shares. In general, such a scheme can be constructed for any $[N,K]$ GRS code, and such a scheme is usually referred to as Shamir's secret sharing.

Lemma 2.3.3 (Shamir's secret sharing)**.** *Let $\mathcal{C}$ be an $[N,K]$ GRS code with nonzero locators $\alpha_1, \ldots, \alpha_N$. Then there is a (K,N)-threshold secret sharing scheme based on $\mathcal{C}$.*

Proof. Let

$$G = \begin{pmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_N \\ \vdots & & \vdots \\ \alpha_1^{K-1} & \ldots & \alpha_N^{K-1} \end{pmatrix}$$

be a generator matrix for $\mathcal{C}$. If $m \in \mathbb{F}_q$ is the secret and $r_1, \ldots, r_{K-1} \in \mathbb{F}_q$ are independent and uniformly distributed, then $(\tilde{a}_1, \ldots, \tilde{a}_N) = (a, r_1, \ldots, r_{K-1})G$ are the shares for the parties $1, \ldots, N$. Because $\mathcal{C}$ is an MDS code, any set $\mathcal{I} \subset [N]$ of size K is an information set, implying that the submatrix $G_\mathcal{I}$ is invertible. As a

result, any K parties can decode the secret by computing the dot product between the first column of $(G_{\mathcal{I}})^{-1}$ and $\tilde{a}_{\mathcal{I}}$.

Now, let $\mathcal{X} \subset [N]$ be a set of size $K-1$. Then the shares associated with these parties can be represented as

$$\tilde{a}_{\mathcal{X}} = a(1,\dots,1) + (r_1,\dots,r_{K-1})G_{\mathcal{X}}^{\geq 1},$$

where $G_{\mathcal{X}}^{\geq 1}$ consists of the lower $K-1$ rows of $G_{\mathcal{X}}$. If $\mathcal{X} = \{i_1,\dots,i_{K-1}\}$, then

$$G_{\mathcal{X}}^{\geq 1} = \begin{pmatrix} \alpha_{i_1} & \dots & \alpha_{i_{K-1}} \\ \vdots & & \vdots \\ \alpha_{i_1}^{K-1} & \dots & \alpha_{i_{K-1}}^{K-1} \end{pmatrix} = \operatorname{diag}(\alpha_{i_1},\dots,\alpha_{i_{K-1}}) \cdot \begin{pmatrix} 1 & \dots & 1 \\ \alpha_{i_1} & \dots & \alpha_{i_{K-1}} \\ \vdots & & \vdots \\ \alpha_{i_1}^{K-2} & \dots & \alpha_{i_{K-1}}^{K-2} \end{pmatrix}.$$

Now, $G_{\mathcal{X}}^{\geq 1}$ is a product of a diagonal matrix with nonzero entries and a $(K-1)\times(K-1)$ Vandermonde matrix on distinct points, which means that $G_{\mathcal{X}}^{\geq 1}$ is invertible. Hence, $(r_1,\dots,r_k)G_{\mathcal{X}}^{\geq 1}$ is uniformly distributed, since an invertible mapping transforms a uniform distribution to a uniform distribution. Therefore, by Corollary 2.2.26, the scheme is information-theoretically secure. □

Similar schemes can be constructed for other MDS codes. The important property is the MDS property of the generator matrix, since it allows for both recovering the secret by inverting a submatrix of the generator matrix and proving the security as in the previous proof.

2.4 Cryptography

Cryptography is about providing confidentiality, authenticity, and integrity to data that is travelling through an insecure channel. As opposed to information-theoretic measures of security used in secret sharing, cryptography protects against computationally bounded adversaries. The study of computational complexity of algorithms is an important part of cryptography. Good references on cryptography include [13, 14].

2.4.1 Algorithms and computational complexity

An *algorithm* is a sequence of computation steps in some model of computation. Such models of computation can be, for example, Turing machines, λ-calculus, or general recursive functions. Even though these models of computation differ in their description of what computation means, they are still equivalent in their capabilities. Therefore, we do not need to care about the exact model of computation used, and we can describe algorithms using any reasonable description of the computation steps.

An algorithm takes as input a string from finite alphabet Σ, *i.e.*, a finite sequence of symbols, and returns some result, which is also a string over the alphabet. An algorithm can be either deterministic or probabilistic, which means that it is allowed

to draw values from some probability distributions. The time complexity of an algorithm can be measured as the number of computation steps needed to compute the result. We are often not interested in the number of steps needed for some specific input, but rather the number of steps as a function of the length of the input. Here, the model of computation becomes important once again, since the number of computation steps is dependent on the model of computation. Therefore, we will only discuss the complexity classes of algorithms.

Cryptography aims to provide security against arbitrary adversaries, which are assumed to be algorithms with a time complexity bounded from above. This corresponds to assuming that the adversary is computationally bounded. We do not place any further restrictions on the strategy a potential adversary can have. Therefore, all security definitions are formulated such that no "efficient" adversary can "break" the scheme with "high" probability.

One way of defining the terms "efficient" and "high probability" would be to give some explicit bounds on the maximal number of steps an algorithm can take and the probability that it succeeds. The problem is that choosing such bounds would be quite arbitrary and would depend on the exact model of computation. A better approach is to consider a *security parameter*, denoted by n, which can measure the difficulty of breaking the scheme. Often the security parameter describes the length of a key in an encryption scheme, but this is not necessarily always the case. Now, we can define *efficient* algorithms as algorithms, that run in polynomial time with respect to the length of their input. We give each adversary the security parameter encoded in unary as 1^n, *i.e.*, the length n string of all ones. Furthermore, we allow the adversaries to be probabilistic algorithms. This class of algorithms is known as *probabilistic polynomial-time (PPT)* algorithms. Choosing the class to be polynomial time algorithms is particularly useful, since polynomials are closed under sum, product, and composition. Furthermore, as all reasonable models of computation are equivalent, there exist polynomial time reductions from any model to any other model. Hence, this definition is independent of the chosen model of computation.

The cryptographic primitives we will be using are based on some secret knowledge, which is usually chosen at random from some probability distribution. Therefore, an adversary has a small chance of guessing the secret and, hence, breaking the cryptographic primitive, even if it does not actually do any other processing. Thus, we allow the adversary to have a small probability of breaking the scheme, where the probability is taken over the randomness used in the cryptographic system and the randomness of the probabilistic adversary. As a complement to the polynomial-time algorithms, a small probability is something that is negligible in the security parameter.

Definition 2.4.1. A function $\mu\colon \mathbb{N} \to \mathbb{R}_+$ is said to be *negligible* if for every positive polynomial $p(n)$ there exists $N \in \mathbb{N}$ such that $\mu(n) \leq \frac{1}{p(n)}$ for all $n \geq N$.

The definition of negligible is convenient since the class of negligible functions is closed under sum and product. Furthermore, multiplying a negligible function by a polynomial is still a negligible function, which means that an efficient adversary

repeating a strategy that works with a negligible probability will have a negligible probability of success, since the adversary is allowed to take only polynomially many steps.

2.4.2 Pseudorandomness

We define the security of encryption algorithms by comparing the behavior of some actual realization of such an algorithm to some idealized implementation. If the difference between the behaviors is negligible for all efficient adversaries, then we say that the scheme is secure. This is because it is computationally indistinguishable from the ideal version. We say that the indexed collection $\mathbf{X} = \{\mathbf{X}_n\}_{n\in\mathbb{N}}$ of random variables is a *probability ensemble*.

Definition 2.4.2. The probability ensembles $\mathbf{X} = \{\mathbf{X}_n\}_{n\in\mathbb{N}}$ and $\mathbf{Y} = \{\mathbf{Y}_n\}_{n\in\mathbb{N}}$ are said to be *computationally indistinguishable* if

$$|\mathrm{P}[\mathcal{A}(1^n, \mathbf{X}_n) = 1] - \mathrm{P}[\mathcal{A}(1^n, \mathbf{Y}_n) = 1]|$$

is negligible for all PPT adversaries $\mathcal{A}$. The probability is taken over the probability ensembles and the internal randomness of the adversary $\mathcal{A}$.

The algorithm $\mathcal{A}$ tries to determine if the value it received is from the distribution $\mathbf{X}$ or $\mathbf{Y}$ and outputs either 0 or 1 to indicate its choice. If $\mathbf{X}$ and $\mathbf{Y}$ are computationally indistinguishable, then the behaviour of $\mathcal{A}$ will be almost identical in both cases. It is easy to see that if $\mathbf{X}$ and $\mathbf{Y}$ are computationally indistinguishable, then the random variables $\mathbf{X}_n$ and $\mathbf{Y}_n$ have to be of polynomial length in n.

As seen in the context of secret sharing, it is convenient to use randomness to hide information. A similar technique is used in cryptography, but pseudorandomness is used instead.

Definition 2.4.3. Let $\ell(n)$ be a positive polynomial. An algorithm G is said to be a *pseudorandom generator (PRG)* with length expansion $\ell(n)$ if

- G is efficiently computable and deterministic,
- $|G(x)| = \ell(|x|) > |x|$ for all $x \in \Sigma^*$, *i.e.*, G is length expanding, and
- $\{\mathbf{U}_{\ell(n)}\}_{n\in\mathbb{N}}$ and $\{G(\mathbf{U}_n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.

Here $\mathbf{U}_n$ is the uniform distribution on Σ^n.

A PRG G can produce $\ell(n) > n$ symbols of randomness from n input symbols. The image of G has at most $q^n < q^{\ell(n)}$ elements, *i.e.*, it does not fill its codomain. However, an efficient adversary is still not able to distinguish the output of G from a random element in the codomain with a nonnegligible probability. The existence of a PRG is still an open problem. However, it is believed that they do exist, and most existing implementations of encryption algorithms rely on pseudorandomness.

2.4.3 Symmetric encryption

One of the most important applications of the primitive constructions such as the PRG is the symmetric encryption scheme. The name comes from the fact that the encryption and decryption algorithms use the same key.

Definition 2.4.4. A *symmetric encryption* scheme consists of PPT algorithms $(\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ such that

- the key generation algorithm gen takes the security parameter 1^n and gives the key $k = \mathsf{gen}(1^n)$,
- the encryption algorithm enc takes the key k and a message m and gives the ciphertext $c = \mathsf{enc}(k, m)$, and
- the decryption algorithm dec takes the key k and a ciphertext c and returns $m = \mathsf{dec}(k, c)$.

The above definition gives the syntax of a symmetric encryption scheme, but does not define security in any way. There are multiple different ways of defining security in different circumstances. In addition to assuming that the adversary is computationally bounded, we may also assume other capabilities of the adversary. For example, in some circumstances, the adversary may be able to craft some or all of the messages that are used in the encryption system to deduce the secret key. In the following, we assume that the adversary is only passively observing the messages, but is not allowed to interact further. Such an adversary is known as an *eavesdropper*.

Definition 2.4.5. A symmetric encryption scheme $(\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ is said to have *indistinguishable encryptions under eavesdropping attack (IND-EAV secure)* if for all sequences of messages $\{m_n\}_{n\in\mathbb{N}}$ and all PPT adversaries $\mathcal{A}$

$$|\mathrm{P}[\mathcal{A}(1^n, \mathsf{enc}(\mathbf{K}, m_n)) = 1] - \mathrm{P}[\mathcal{A}(1^n, \mathsf{enc}(\mathbf{K}, 0^{|m_n|})) = 1]|$$

is negligible in n, where $\mathbf{K} = \mathsf{gen}(1^n)$ and $0^{|m_n|}$ is the all-zero message of the same length as m_n.

This definition guarantees that the encryption of a particular message is indistinguishable from the encryption of an arbitrary message, say the all zero message, for all efficient adversaries. This mimics the definition of information-theoretic security, since given a ciphertext, all messages are equally likely to have produced that ciphertext. This definition works only in the context of eavesdroppers. If the adversary is allowed to listen to multiple messages encrypted with the same key or alter the sent messages, then some other definition of security is required.

We can finally construct an encryption scheme. The following construction is similar to the one-time pad encryption, which is known to be information-theoretically secure, but instead of using pure randomness, it uses pseudorandomness from a PRG. We consider some finite field $\mathbb{F}_q$, instead of the usual binary alphabet.

Construction 2.4.6. Let G be a PRG with length expansion $\ell(n)$. Define gen such that $\mathsf{gen}(1^n)$ gives a key that is uniformly chosen from $\mathbb{F}_q^n$. The encryption of a message $m \in F_q^{\ell(n)}$ is done by computing

$$c = \mathsf{enc}(k, m) = m + G(k).$$

The decryption is done simply by subtraction

$$m = \mathsf{dec}(k, c) = c - G(k).$$

Proposition 2.4.7. *The encryption scheme described in Construction 2.4.6 is IND-EAV secure.*

Proof. Let m_n be a message of length $\ell(n)$ and $\mathbf{K} = \mathbf{U}_n$. If $\mathcal{A}$ is an arbitrary efficient adversary, then

$$\begin{aligned}
&\Big|\mathrm{P}[\mathcal{A}(1^n, \mathsf{enc}(\mathbf{K}, m_n)) = 1] - \mathrm{P}[\mathcal{A}(1^n, \mathsf{enc}(\mathbf{K}, 0^{\ell(n)})) = 1]\Big| \\
&= \Big|\mathrm{P}[\mathcal{A}(1^n, m_n + G(\mathbf{U}_n)) = 1] - \mathrm{P}[\mathcal{A}(1^n, G(\mathbf{U}_n)) = 1]\Big| \\
&= \Big|\mathrm{P}[\mathcal{A}(1^n, m_n + G(\mathbf{U}_n)) = 1] - \mathrm{P}[\mathcal{A}(1^n, m_n + \mathbf{U}_{\ell(n)}) = 1] \\
&\quad + \mathrm{P}[\mathcal{A}(1^n, m_n + \mathbf{U}_{\ell(n)}) = 1] - \mathrm{P}[\mathcal{A}(1^n, G(\mathbf{U}_n)) = 1]\Big| \\
&\leq \Big|\mathrm{P}[\mathcal{B}(1^n, G(\mathbf{U}_n)) = 1] - \mathrm{P}[\mathcal{B}(1^n, \mathbf{U}_{\ell(n)}) = 1]\Big| \\
&\quad + \Big|\mathrm{P}[\mathcal{A}(1^n, \mathbf{U}_{\ell(n)}) = 1] - \mathrm{P}[\mathcal{A}(1^n, G(\mathbf{U}_n)) = 1]\Big|,
\end{aligned}$$

where $\mathcal{B}$ is the efficient adversary defined by $\mathcal{B}(1^n, y) = \mathcal{A}(1^n, m_n + y)$. The result follows from the triangle inequality and the fact that $\mathbf{U}_{\ell(n)} \sim m_n + \mathbf{U}_{\ell(n)}$. The two terms in the sum are both negligible by the definition of the PRG G. Hence, the symmetric encryption scheme is IND-EAV secure. □

3 Secure distributed matrix multiplication

Matrix multiplication is an important operation that can be used to compute various computations in engineering, data science, and medicine, for example. Various methods for speeding up the computation have been developed over the years. One way of making the computation faster is by breaking the problem into smaller pieces and distributing those pieces to multiple computing nodes. The work can then be done in parallel. Matrix multiplication is easily parallelized by splitting the matrices into smaller submatrices and computing the products of the submatrices.

Distributing work to multiple worker nodes has advantages, including the fact that more work can be done in parallel. The disadvantage is that the time it takes to finish the computation is determined by the slowest worker node. There can be multiple factors in determining how long the computation is going to take at each worker node, which means that eliminating slow worker nodes before the computation happens is not possible. The phenomenon of a few workers significantly increasing the computation time is known as the *straggler effect*, and the slow worker nodes are known as *stragglers*. To mitigate the straggler effect, coded computation can be used similarly to the secret sharing schemes in Section 2.3. Using tools from coding theory, we can construct distributed computation schemes such that any K out of N responses is enough to decode the answer.

Another thing to consider is the fact that the data contained in the matrices can be sensitive and should be hidden from the worker nodes. One way to achieve this is by using tools from information theory, which provide security against arbitrarily powerful adversaries, as long as the underlying assumptions are met. The most important assumption is that the number of *colluding nodes* is small enough. A set of colluding nodes share the data they receive to infer information about the secret matrices. Information theory can be used to guarantee that any set of at most X colluding worker nodes will gain no information about the secret matrices.

Using these tools, we will define *secure distributed matrix multiplication*, or *SDMM*. Furthermore, we will define a new framework for different types of SDMM constructions known as linear SDMM. SDMM was first considered by Chang and Tandon in [2]. Since then, different constructions have been studied, such as the MatDot codes in [15], the GASP code in [16, 17, 18], the cross subspace alignment code in [19, 20], the A3S code in [21, 22, 23], and the codes by Yang *et al.* in [24], Mital *et al.* in [25] and Yu *et al.* in [26]. Futhermore, different modes of SDMM have been presented, including batch matrix multiplication in [19, 26, 27, 28], Lagrange coded polynomial evaluation in [29], private SDMM in [26, 27, 30, 31], and matrix-vector multiplication in [32, 33]. The theoretical limits for the capacity of SDMM have been studied in [2, 19, 22].

3.1 Illustrating examples

Before stating the general model of computation for SDMM, we will give a few illustrative examples for some fixed parameters. The following example is based on the construction presented in [15].

Example 3.1.1 (Secure MatDot code)**.** Let $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$. Let us partition the matrices into 3 pieces and protect against 2 colluding workers. The matrices are partitioned into block matrices according to

$$A = \begin{pmatrix} A_1 & A_2 & A_3 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ B_2 \\ B_3 \end{pmatrix}.$$

The resulting submatrices are $A_j \in \mathbb{F}_q^{t\times \frac{s}{3}}$, $B_{j'} \in \mathbb{F}_q^{\frac{s}{3}\times r}$. The product is then

$$AB = A_1B_1 + A_2B_2 + A_3B_3.$$

Let $R_1, R_2 \in \mathbb{F}_q^{t\times \frac{s}{3}}$ and $S_1, S_2 \in \mathbb{F}_q^{\frac{s}{3}\times r}$ be matrices chosen uniformly at random and independently of A and B. Define the polynomials

$$\begin{aligned} f(x) &= A_1 + A_2x + A_3x^2 + R_1x^3 + R_2x^4, \\ g(x) &= B_1x^2 + B_2x + B_3 + S_1x^3 + S_2x^4. \end{aligned}$$

These polynomials are polynomials of a single scalar variable with matrix coefficients. Another way of looking at it is that there is a matrix of polynomials with scalar coefficients.

Let $\alpha_1, \ldots, \alpha_N \in \mathbb{F}_q^\times$ be distinct nonzero points. Evaluate the polynomials $f(x)$ and $g(x)$ at these points to get the encoded matrices

$$\widetilde{A}_i = f(\alpha_i), \quad \widetilde{B}_i = g(\alpha_i).$$

These encoded matrices can be sent to each worker node through a secure communication channel. The worker nodes compute the matrix products $\widetilde{C}_i = \widetilde{A}_i\widetilde{B}_i$ and return these back to the user. The user receives evaluations of the polynomial $h(x) = f(x)g(x)$ from each worker node. Using the definition of $f(x)$ and $g(x)$ we can write out the coefficients of $h(x)$ as

$$\begin{aligned} h(x) = {} & A_1B_3 + (A_1B_2 + A_2B_3)x + (A_1B_1 + A_2B_2 + A_3B_3)x^2 \\ & + (A_1S_1 + A_2B_1 + A_3B_2 + R_1B_3)x^3 \\ & + (A_1S_2 + A_2S_1 + A_3B_1 + R_1B_2 + R_2B_3)x^4 \\ & + (A_2S_2 + A_3S_1 + R_1B_1 + R_2B_2)x^5 + (A_3S_2 + R_1S_1 + R_2B_1)x^6 \\ & + (R_1S_2 + R_2S_1)x^7 + R_2S_2x^8. \end{aligned}$$

Now we notice that the coefficient of the term x^2 is exactly the product AB, which we wish to recover. Using polynomial interpolation as described in Lemma 2.1.16, we can interpolate using 9 evaluations of $h(x)$ and get the coefficient of x^2. Therefore, we need a response from $K = 9$ worker nodes. It is worth noting that we do not care which K workers respond to us. The *rate* of the scheme is $\frac{1}{9}$, which describes that we need to download 9 times as much data as the product AB.

The matrix partitioning used in the previous example is known as *inner product partitioning*, or *IPP*. Another way of partitioning the matrices is by splitting A vertically and B horizontally, which is known as *outer product partitioning*, or *OPP*. The following example is based on the construction presented in [2].

Example 3.1.2 (Chang–Tandon code)**.** Let $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$. Let us partition the matrices into 3 pieces and protect against 2 colluding workers. The matrices are partitioned into block matrices according to

$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & B_2 & B_3 \end{pmatrix}.$$

The resulting submatrices are $A_j \in \mathbb{F}_q^{\frac{t}{3}\times s}$, $B_{j'} \in \mathbb{F}_q^{s\times \frac{r}{3}}$. The product can be expressed as

$$AB = \begin{pmatrix} A_1B_1 & A_1B_2 & A_1B_3 \\ A_2B_1 & A_2B_2 & A_2B_3 \\ A_3B_1 & A_3B_2 & A_3B_3 \end{pmatrix}.$$

We need to recover each of the submatrices to construct the product AB. Let $R_1, R_2 \in \mathbb{F}_q^{\frac{t}{3}\times s}$ and $S_1, S_2 \in \mathbb{F}_q^{s\times \frac{r}{3}}$ be matrices chosen uniformly at random independently of each other and independently of A and B. Define the polynomials

$$\begin{aligned} f(x) &= A_1 + A_2x + A_3x^2 + R_1x^3 + R_2x^4, \\ g(x) &= B_1 + B_2x^5 + B_3x^{10} + S_1x^{15} + S_2x^{20}. \end{aligned}$$

Let $\alpha_1, \ldots, \alpha_N \in \mathbb{F}_q^\times$ be distinct nonzero points. Evaluate the polynomials $f(x)$ and $g(x)$ at these points to get the encoded matrices

$$\widetilde{A}_i = f(\alpha_i), \quad \widetilde{B}_i = g(\alpha_i).$$

These encoded matrices can be sent to each worker node through a secure communication channel. The worker nodes compute the matrix products $\widetilde{C}_i = \widetilde{A}_i\widetilde{B}_i$ and return these back to the user. The user receives evaluations of the polynomial $h(x) = f(x)g(x)$ from each worker. Using the definition of $f(x)$ and $g(x)$ we can write out the coefficients of $h(x)$ as

$$\begin{aligned} h(x) = {} & A_1B_1 + A_2B_1x + A_3B_1x^2 + R_1B_1x^3 + R_2B_1x^4 + A_1B_2x^5 + A_2B_2x^6 \\ & + A_3B_2x^7 + R_1B_2x^8 + R_2B_2x^9 + A_1B_3x^{10} + A_2B_3x^{11} + A_3B_3x^{12} \\ & + R_1B_3x^{13} + R_2B_3x^{14} + A_1S_1x^{15} + A_2S_1x^{16} + A_3S_1x^{17} + R_1S_1x^{18} \\ & + R_2S_1x^{19} + A_1S_2x^{20} + A_2S_2x^{21} + A_3S_2x^{22} + R_1S_2x^{23} + R_2S_2x^{24}. \end{aligned}$$

Now we notice that the coefficients of the terms $1, x, x^2, x^5, x^6, x^7, x^{10}, x^{11}, x^{12}$ are exactly the submatrices we wish to recover. Using polynomial interpolation, we can interpolate using 25 evaluations of $h(x)$, since $h(x)$ has degree 24. Therefore, we need a response from $K = 25$ workers to get the result. We do not care which K workers respond. The rate of this scheme is $\frac{9}{25}$.

An even better rate can be achieved by choosing the polynomials $f(x)$ and $g(x)$ in a more clever way. The following example is a special case of the GASP code presented in [16].

Example 3.1.3 (GASP code)**.** Let $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$. Let us partition the matrices into 3 pieces and protect against 2 colluding workers. The matrices are partitioned into block matrices according to

$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & B_2 & B_3 \end{pmatrix}.$$

The resulting submatrices are $A_j \in \mathbb{F}_q^{\frac{t}{3}\times s}$, $B_{j'} \in \mathbb{F}_q^{s\times \frac{r}{3}}$. The product can be expressed as

$$AB = \begin{pmatrix} A_1B_1 & A_1B_2 & A_1B_3 \\ A_2B_1 & A_2B_2 & A_2B_3 \\ A_3B_1 & A_3B_2 & A_3B_3 \end{pmatrix}.$$

We need to recover each of the submatrices to construct the product AB. Let $R_1, R_2 \in \mathbb{F}_q^{\frac{t}{3}\times s}$ and $S_1, S_2 \in \mathbb{F}_q^{s\times \frac{r}{3}}$ be matrices chosen uniformly at random independently of each other and independently of A and B. Define the polynomials

$$\begin{aligned} f(x) &= A_1 + A_2x + A_3x^2 + R_1x^9 + R_2x^{12}, \\ g(x) &= B_1 + B_2x^3 + B_3x^6 + S_1x^9 + S_2x^{10}. \end{aligned}$$

The exponents are chosen carefully so that the total number of worker nodes needed is as low as possible.

Let $\alpha_1, \ldots, \alpha_N \in \mathbb{F}_q^\times$ be distinct nonzero points. Evaluate the polynomials $f(x)$ and $g(x)$ at these points to get the encoded matrices

$$\widetilde{A}_i = f(\alpha_i), \quad \widetilde{B}_i = g(\alpha_i).$$

These encoded matrices can be sent to each worker node through a secure communication channel. The worker nodes compute the matrix products $\widetilde{C}_i = \widetilde{A}_i\widetilde{B}_i$ and return these back to the user. The user receives evaluations of the polynomial $h(x) = f(x)g(x)$ from each worker. Using the definition of $f(x)$ and $g(x)$ we can write out the coefficients of $h(x)$ as

$$\begin{aligned} h(x) = {} & A_1B_1 + A_2B_1x + A_3B_1x^2 + A_1B_2x^3 + A_2B_2x^4 \\ & + A_2B_3x^5 + A_1B_3x^6 + A_2B_3x^7 + A_3B_3x^8 \\ & + (A_1S_1 + R_1B_1)x^9 + (A_1S_2 + A_2S_1)x^{10} \\ & + (A_2S_2 + A_3S_1)x^{11} + (A_3S_2 + R_1B_2 + R_2B_1)x^{12} \\ & + (R_1B_3 + R_2B_2)x^{15} + (R_1S_1 + R_2B_3)x^{18} \\ & + R_1S_2x^{19} + R_1S_2x^{21} + R_2S_2x^{22}. \end{aligned}$$

Now we notice that the coefficients of the first 9 terms are exactly the submatrices we wish to recover. Using polynomial interpolation, we can interpolate using 18 evaluations of $h(x)$, since h has 18 nonzero coefficients, provided that the corresponding linear equations are solvable. This is always possible if the field is large enough. Therefore, we need a response from 18 workers. Here the rate is $\frac{1}{2} > \frac{9}{25}$.

3.2 General model of computation

Examples 3.1.1, 3.1.2 and 3.1.3 show two different ways of partitioning the matrices to 3 pieces and protecting against 2 colluding workers. All of the examples follow the same general model of computation. The scheme takes as input two matrices: $A \in \mathbb{F}_q^{t \times s}$ and $B \in \mathbb{F}_q^{s \times r}$. The parameters are the number of worker nodes N, the number of colluding nodes X, and the matrix partitioning parameters p, m, n such that $p \mid s$, $m \mid t$ and $n \mid r$. The general model for SDMM schemes is as follows.

- Matrices A and B are partitioned to block matrices such that

$$A = \begin{pmatrix} A_{11} & \dots & A_{1p} \\ \vdots & & \vdots \\ A_{m1} & \dots & A_{mp} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & \dots & B_{1n} \\ \vdots & & \vdots \\ B_{p1} & \dots & B_{pn} \end{pmatrix},$$

  where the submatrices are $\frac{t}{m} \times \frac{s}{p}$ and $\frac{s}{p} \times \frac{r}{n}$, respectively. The product AB can be represented as

$$C = \begin{pmatrix} C_{11} & \dots & C_{1n} \\ \vdots & & \vdots \\ C_{m1} & \dots & C_{mn} \end{pmatrix},$$

  where $C_{ij} = \sum_{k=1}^{p} A_{ik} B_{kj}$. We denote by A_j the submatrices of A in some order, and similarly, $B_{j'}$ for the submatrices of B.

- Matrices R_k, $S_{k'}$ are drawn uniformly at random from matrices of the same size as A_j and $B_{j'}$, respectively, for $k, k' \in [X]$.

- Polynomials $f(x)$ and $g(x)$ are formed such that the coefficients are the matrices A_j and R_k for $f(x)$ and $B_{j'}$ and $S_{k'}$ for $g(x)$.

- The polynomials $f(x)$ and $g(x)$ are evaluated at points $\alpha_1, \dots, \alpha_N$, and the encoded matrices $\widetilde{A}_i = f(\alpha_i)$ and $\widetilde{B}_i = g(\alpha_i)$ are sent to worker node i through a secure communication channel. These evaluations are matrices because $f(x)$ and $g(x)$ have matrices as coefficients.

- Worker node i computes $\widetilde{C}_i = \widetilde{A}_i \widetilde{B}_i$ and returns it to the user.

- The user receives evaluations of the polynomial $h(x) = f(x)g(x)$ at the points $\alpha_1, \dots, \alpha_N$ and decodes the coefficients using polynomial interpolation. From the coefficients, the user can easily compute the final answer AB.

If the number of colluding nodes X is greater than 1, then the scheme is said to protect against X-collusion. If $X = 1$, then the SDMM scheme protects against honest-but-curious nodes, but not against any collusion between the workers. Many of the SDMM schemes can also be applied in the case of $X = 0$, which is known as *distributed matrix multiplication*. The security of an SDMM scheme is defined by the following criterion.

We use information-theoretic security to characterize the security of SDMM schemes. Let $\mathbf{A}$ and $\mathbf{B}$ be the random variables that correspond to the matrices

A and B, respectively, with some a priori distributions. The encodings are also associated with the random variables $\widetilde{\mathbf{A}}_i$ and $\widetilde{\mathbf{B}}_i$. For the collection of random variables $\{\widetilde{\mathbf{A}}_i\}_{i\in\mathcal{X}}$, we denote $\widetilde{\mathbf{A}}_{\mathcal{X}}$.

Definition 3.2.1. An SDMM scheme is said to be *information-theoretically secure against X-collusion* if

$$I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}}) = 0$$

for all $\mathcal{X} \subset [N]$, $|\mathcal{X}| \leq X$.

The set $\mathcal{X}$ denotes any set of colluding nodes. This definition guarantees that the colluding worker nodes do not gain any information about the matrices A or B. The shares they have are indistinguishable from random data, which makes it impossible to deduce anything about A or B.

There are multiple different metrics that can be used to compare and evaluate different SDMM schemes. The *recovery threshold*, denoted by K, is the minimal number of responses that are needed to decode the answer. The recovery threshold is also denoted by R_c in the literature. A low recovery threshold compared to the number of worker nodes protects well against straggling workers.

The worker nodes are often thought to be distributed such that the communication cost between the different parties is an important consideration. The *upload cost* is the total size of data that needs to be uploaded from the user to the workers. The *download cost* is the total size of data that needs to be downloaded from the workers to the user. Notice that downloading is only necessary from K workers, since the results from them are enough to decode the answer. The sizes of the matrices are usually measured in finite field symbols. The matrices could be compressed before sending, which means that the entropy of the matrices would be a natural way of measuring the communication costs. However, the compression would require additional computation from the user, so this is not usually done, as the goal is to save on computational cost.

The computational complexity of the worker nodes and the user is another interesting measure, though measuring it is not straightforward since much of the complexity comes from the underlying matrix multiplication.

3.3 Secure MatDot code

In this section, we generalize the scheme in Example 3.1.1. The following construction was first presented in [34] in the context of distributed matrix multiplication. The secure MatDot scheme was first presented in [15].

Construction 3.3.1 (Secure MatDot code)**.** Let p be the partitioning parameter and let $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$ be matrices. Partition A and B to block matrices such that

$$A = \begin{pmatrix} A_1 & \dots & A_p \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ \vdots \\ B_p \end{pmatrix},$$

where $A_j \in \mathbb{F}_q^{t\times\frac{s}{p}}$ and $B_{j'} \in \mathbb{F}_q^{\frac{s}{p}\times r}$, for $j, j' \in [p]$. The product of A and B is

$$AB = \sum_{j=1}^{p} A_j B_j,$$

that is, the product is a dot product of the partitioned matrices.

Let $R_k \in \mathbb{F}_q^{t\times\frac{s}{p}}$ and $S_{k'} \in \mathbb{F}_q^{\frac{s}{p}\times r}$, for $k, k' \in [X]$, be chosen uniformly at random and independently of each other. Now, let us form the polynomials

$$f(x) = \sum_{j=1}^{p} A_j x^{j-1} + \sum_{k=1}^{X} R_k x^{p+k-1}$$
$$g(x) = \sum_{j'=1}^{p} B_{j'} x^{p-j'} + \sum_{k'=1}^{X} S_{k'} x^{p+k'-1}.$$

Choose evaluation points $\alpha = (\alpha_1, \ldots, \alpha_N) \in (\mathbb{F}_q^{\times})^N$, where N is the number of worker nodes. The points should be distinct and nonzero. Evaluate the polynomials $f(x)$ and $g(x)$ at the evaluation points. The encoded matrices

$$\widetilde{A}_i = f(\alpha_i), \quad \widetilde{B}_i = g(\alpha_i)$$

are sent to worker node i through a secure communication channel. Worker i computes the matrix product $\widetilde{C}_i = \widetilde{A}_i \widetilde{B}_i \in \mathbb{F}_q^{t\times r}$ and sends that back to the user.

The user receives the products $\widetilde{C}_i = \widetilde{A}_i \widetilde{B}_i$ from the fastest nodes. Because the encoded matrices $\widetilde{A}_i$ and $\widetilde{B}_i$ are evaluations of a polynomial at a point α_i, their product is an evaluation of $h(x) = f(x)g(x)$ at the same point. Using the definition of $f(x)$ and $g(x)$ we can write the polynomial $h(x)$ explicitly as

$$\begin{aligned} h(x) &= f(x)g(x) \\ &= \sum_{j=1}^{p}\sum_{j'=1}^{p} A_j B_{j'} x^{p+j-j'-1} + \sum_{j=1}^{p}\sum_{k'=1}^{X} A_j S_{k'} x^{p+j+k'-2} \\ &\quad + \sum_{k=1}^{X}\sum_{i'=1}^{p} R_k B_{i'} x^{2p+k-i'-1} + \sum_{k=1}^{X}\sum_{k'=1}^{X} R_k S_{k'} x^{2p+k+k'-2}. \end{aligned}$$

The terms in the last three sums have degree at least p. The only time the exponent in the first term is $p-1$ is when $j = j'$, which means that the coefficient of the term x^{p-1} is

$$\sum_{j=1}^{p} A_j B_j = AB.$$

The degree of $h(x)$ is $\deg h(x) = 2p + 2X - 2$. As a result, the user needs $2p + 2X - 1$ responses from the workers to interpolate the polynomial. The user then performs polynomial interpolation to obtain the coefficients of $h(x)$, where AB is the coefficient of x^{p-1}.

Let $\mathcal{I} \subset [N]$ be the fastest $2p + 2X - 1$ workers. The user then computes the Lagrange basis polynomials $\ell_i(x)$, for $i \in \mathcal{I}$, on the points $\alpha_{\mathcal{I}}$. After that the result can be obtained by computing

$$AB = \sum_{i \in \mathcal{I}} \ell_i^{(p-1)} \widetilde{C}_i,$$

where $\ell_i^{(p-1)}$ is the coefficient of x^{p-1} in the polynomial $\ell_i(x)$. This gives the coefficient of x^{p-1} according to Lemma 2.1.16.

The number of worker nodes needs to be at least $K = 2p + 2X - 1$, which is the recovery threshold for the secure MatDot code. The upload cost is $N(\frac{ts}{p} + \frac{sr}{p})$, while the download cost is Ktr. Any K responses can be used to recover the answer, so adding additional workers increases the straggler tolerance of the scheme. As the evaluation points need to be distinct and nonzero, we need the field size to be large enough, in particular, $q > N$.

The security of the scheme will be proven later in Corollary 3.5.6. This construction is preferable because of its low recovery threshold and simplicity. The drawback of using the inner product partitioning is that the encoded products $\widetilde{C}_i$ are the same size as the product AB. Therefore, the download cost is higher than the download cost of a scheme with an outer product partition and the same recovery threshold.

3.4 GASP code

As presented earlier, another way of partitioning the matrices is by using outer product partitioning, which means that the result is formed by arranging the products of the submatrices. This method was first presented in [2], where the polynomials were chosen such that each submatrix product has its own term. As seen in Example 3.1.2 this method is wasteful, since only certain terms are needed, specifically the terms which contain $A_j B_{j'}$ for some $(j, j') \in [m] \times [n]$. The authors of [2] proposed the concept of aligned secret sharing, in which some of the unnecessary terms are aligned in the polynomial $h(x)$. This has since been researched in [16].

As can be seen by comparing Examples 3.1.2 and 3.1.3, the recovery threshold can be made smaller by choosing the exponents in the encoding polynomials cleverly. It is still highly nontrivial to choose the exponents to make the system solvable. The system of linear equations in Example 3.1.2 is always solvable, since it can be represented as a Vandermonde system, which is not possible for the system in Example 3.1.3, since the construction relies on the fact that there are "gaps" in the product polynomial $h(x)$.

In general, the encoding polynomials can be represented as

$$f(x) = \sum_{j=1}^{m} A_j x^{\varphi_j} + \sum_{k=1}^{X} R_k x^{\varphi_{m+k}},$$

$$g(x) = \sum_{j'=1}^{n} B_{j'} x^{\gamma_{j'}} + \sum_{k'=1}^{X} S_{k'} x^{\gamma_{n+k'}},$$

	γ_1	$\cdots$	γ_n	γ_{n+1}	$\cdots$	γ_{n+X}
φ_1	$\varphi_1+\gamma_1$	$\cdots$	$\varphi_1+\gamma_n$	$\varphi_1+\gamma_{n+1}$	$\cdots$	$\varphi_1+\gamma_{n+X}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
φ_m	$\varphi_m+\gamma_1$	$\cdots$	$\varphi_m+\gamma_n$	$\varphi_m+\gamma_{n+1}$	$\cdots$	$\varphi_m+\gamma_{n+X}$
φ_{m+1}	$\varphi_{m+1}+\gamma_1$	$\cdots$	$\varphi_{m+1}+\gamma_n$	$\varphi_{m+1}+\gamma_{n+1}$	$\cdots$	$\varphi_{m+1}+\gamma_{n+X}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
φ_{m+X}	$\varphi_{m+X}+\gamma_1$	$\cdots$	$\varphi_{m+X}+\gamma_n$	$\varphi_{m+X}+\gamma_{n+1}$	$\cdots$	$\varphi_{m+X}+\gamma_{n+X}$

Table 1: General degree table for arbitrary φ and γ.

for some $\varphi \in \mathbb{N}^{m+X}$ and $\gamma \in \mathbb{N}^{n+X}$ [1]. The product of these two polynomials has the form

$$h(x) = f(x)g(x) = \sum_{\eta \in \mathcal{H}} C_\eta x^\eta,$$

where $\mathcal{H} = \{\varphi_j + \gamma_{j'} \mid j \in [m+X], j' \in [n+X]\}$ is the set of sums of the exponents in the polynomials $f(x)$ and $g(x)$, and C_η are some matrices that come from the product.

A way to reason about the way to choose the exponents is by using the *degree table*, which was first presented in [16] and further studied in [17]. The degree table is the $(m+X) \times (n+X)$ array formed by the *outer sum* of φ and γ, *i.e.*, the entry at (j, j') is $\varphi_j + \gamma_{j'}$. Such a table is represented in Table 1. The $m \times n$ region in the upper left corner, colored in red, represents the useful terms, which contain the products of the useful submatrices. To be able to correctly decode the answer, these terms need to be distinct from the other elements in the table. The rest of the table contains additional terms containing a mix of the random parts and the parts from the input matrices. The exponents $\varphi_{m+1}, \ldots, \varphi_{m+X}$ have to be distinct to guarantee that enough randomness is added to the scheme. Similarly, also $\gamma_{n+1}, \ldots, \gamma_{n+X}$ need to be distinct. Hence, the blue and green regions need to contain distinct numbers, though there can be repeats in different colored regions. The goal is to minimize the total number of unique terms in the degree table, since it corresponds to the number of nonzero coefficients in the polynomial $h(x)$.

The polynomial interpolation problem that needs to be solved to solve for the coefficients is determined by the distinct terms in the degree table, *i.e.*, the set $\mathcal{H}$. If $\mathcal{H} = \{\eta_1, \ldots, \eta_{|\mathcal{H}|}\}$, then the system can be represented as

$$\begin{pmatrix} \alpha_1^{\eta_1} & \cdots & \alpha_N^{\eta_1} \\ \alpha_1^{\eta_2} & \cdots & \alpha_N^{\eta_2} \\ \vdots & & \vdots \\ \alpha_1^{\eta_{|\mathcal{H}|}} & \cdots & \alpha_N^{\eta_{|\mathcal{H}|}} \end{pmatrix}.$$

The choice of the evaluation points affects the solvability of the system. It is clear that $N \geq |\mathcal{H}|$ is the minimal requirement. If $\mathcal{H} = \{0, \ldots, |\mathcal{H}| - 1\}$, then the system

[1] In [16] φ and γ were denoted by α and β, respectively. We adopt this new notation, since α clashes with our notation for evaluation points.

is a Vandermonde system, which is solvable as long as the evaluation points are unique, according to Lemma 2.1.19.

By studying the sums of sets of integers, we can analyze the bounds of a degree table. Let $A, B \subset \mathbb{Z}$ be finite sets of integers. We can define

$$A + B = \{a + b \mid a \in A, b \in B\}.$$

The following lemma sets a useful bound on the size of the sum.

Lemma 3.4.1. *Let $A, B \subset \mathbb{Z}$ be finite sets of integers. Then $|A+B| \geq |A|+|B|-1$.*

Proof. As $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ are finite sets of integers, we can order the elements such that $a_1 < \cdots < a_m$ and $b_1 < \cdots < b_n$. Thus, we get the following distinct elements

$$a_1 + b_1 < a_1 + b_2 < \cdots < a_1 + b_n < a_2 + b_n < \cdots < a_m + b_m.$$

This chain contains $m + n - 1$ distinct elements, which proves the claim. □

With the above lemma, we can show a useful lower bound on the number of distinct elements in a degree table.

Lemma 3.4.2. *Let $\varphi \in \mathbb{N}^{m+X}$ and $\gamma \in \mathbb{N}^{n+X}$ be vectors of exponents, and $\mathcal{H}$ be the set of distinct terms in the degree table defined by φ and γ such that the correctness criterion is fulfilled. Then, $|\mathcal{H}| \geq mn + \max\{m, n\} + 2X - 1$.*

Proof. The upper left region must contain mn distinct elements according to the correctness criterion. The sum set of the sets $\{\varphi_{m+1}, \ldots, \varphi_{m+X}\}$ and $\{\gamma_1, \ldots, \gamma_{n+X}\}$ makes up the right half of the degree table and it contains at least $X + (n + X) - 1$ distinct elements, according to Lemma 3.4.1. Hence, $|\mathcal{H}| \geq mn + n + 2X - 1$. Symmetrically, we get $|\mathcal{H}| \geq mn + m + 2X - 1$, which proves the claim. □

Another way to choose the exponents using the help of the degree table was presented in [16, 17] is called *gap additive secure polynomial code*, or *GASP*.

Construction 3.4.3 (GASP code)**.** The exponents are chosen using an additional parameter $r \in [\min\{m, X\}]$ such that

$$\begin{aligned}\varphi = (&0, 1, \ldots, m-1, mn, mn+1, \ldots, mn+r-1,\\ &mn+m, mn+m+1, \ldots, mn+m+r-1, \ldots),\\ \gamma = (&0, m, \ldots, m(n-1), mn, mn+1, \ldots, mn+X-1),\end{aligned}$$

where φ has a length of $m+X$ and γ has a length of $n+X$. It is clear that the terms in the upper left region are distinct from the other terms, since $\varphi_j+\gamma_{j'} = (j-1)+m(j'-1)$ for $j \in [m]$ and $j' \in [n]$, and other terms are larger than or equal to mn. The total number of terms is highly nontrivial to calculate, as shown in [17]. The parameter r can be optimized to find the lowest possible recovery threshold $K = |\mathcal{H}|$. The way the exponents are chosen leaves "gaps" in the degree table, *i.e.*, the set $\mathcal{H}$ does not generally consist of consecutive integers. Therefore, showing that the system

	Upload cost	Download cost	Recovery threshold (K)
Secure MatDot	$N(\frac{ts}{m^2} + \frac{sr}{m^2})$	Ktr	$2m^2 + 2X - 1$
Chang–Tandon code	$N(\frac{ts}{m} + \frac{sr}{m})$	$K\frac{tr}{m^2}$	$(m + X)(m + X)$
GASP_{big} code	$N(\frac{ts}{m} + \frac{sr}{m})$	$K\frac{tr}{m^2}$	$\leq 2m^2 + 2X - 1$
GASP code	$N(\frac{ts}{m} + \frac{sr}{m})$	$K\frac{tr}{m^2}$	$\geq m^2 + m + 2X - 1$

Table 2: Comparison of SDMM schemes with $m = n$ and $p = m^2$.

of linear equations is solvable is not immediately clear. In [16] it was shown that a suitable choice of evaluation points can always be made, provided that the field is large enough. This results in a recovery threshold $K \geq mn + \max\{m, n\} + 2X - 1$, an upload cost $N(\frac{ts}{m} + \frac{sr}{n})$, and a download cost $K\frac{tr}{mn}$.

The GASP code uses clever exponent selection to minimize the total number of nonzero coefficients in $h(x)$. However, this comes at a drawback when choosing the evaluation points. The proof in [16] shows that the probability of finding a suitable choice is nonzero for a large enough field extension of $\mathbb{F}_q$. One way of choosing such points would be to uniformly pick random evaluation points and check if they are suitable. As was said in [16] this reduces down to checking some polynomial equalities, which can in principle be done efficiently.

The following construction is a special case of the GASP code, which can be solved as a Vandermonde system. The scheme has a higher download cost, but the evaluation points are easy to choose.

Construction 3.4.4 (GASP_{big} code)**.** Choosing the exponents in the encoding polynomials as

$$\varphi = (0, 1, \ldots, m-1, mn, mn+1, \ldots, mn+X-1),$$
$$\gamma = (0, m, \ldots, m(n-1), mn, mn+1, \ldots, mn+X-1)$$

gives $\mathcal{H} \subset \{0, 1, \ldots, 2mn + 2X - 2\}$. The system can therefore be solved as a Vandermonde system, even if some exponents are missing from $\mathcal{H}$. This gives a recovery threshold of $K = 2mn + 2X - 1$, upload cost of $N(\frac{ts}{m} + \frac{sr}{n})$, and download cost of $K\frac{tr}{mn}$.

The restriction that the number in the upper left corner of the degree table need to be distinct from all other numbers in the table is something that can be loosened somewhat. It is sufficient that the submatrix products $A_j B_{j'}$ are solvable from the resulting coefficients of $h(x)$. The research on optimizing these various parameters and exponents is still open.

Comparing SDMM schemes with different partitioning of the input matrices is difficult, since there is no obvious way to compare the different parameters. One way of comparing them would be to restrict the amount of upload the scheme is allowed to make. This would lead to setting $m = n = p$, so that each worker node is sent a pth fraction of the full matrices. However, this does not fix the amount of work the worker nodes need to perform. Workers in an outer product partitioning

scheme must compute the product of $\frac{t}{m} \times s$ and $s \times \frac{r}{n}$ matrices, whereas workers in an inner product partitioning scheme must compute the product of $t \times \frac{s}{p}$ and $\frac{s}{p} \times r$ matrices. Using the schoolbook algorithm, the computational complexity of these is $\mathcal{O}(\frac{tsr}{mn})$ and $\mathcal{O}(\frac{tsr}{p})$, respectively. Therefore, it would be natural to set $p = mn$ to equalize the computational complexity between the different schemes. The various SDMM schemes presented in Constructions 3.3.1, 3.4.4, 3.4.3, and Example 3.1.2 are compared in Table 2, with $m = n$ and $p = m^2$.

3.5 Linear SDMM

The previous constructions were based on polynomial interpolation, which closely resembles Reed–Solomon codes. It is possible to describe the SDMM schemes using a more abstract coding theoretic perspective. We shall call these schemes *linear SDMM schemes*, as they are built from linear codes. This also makes it simple to prove that the schemes presented before are secure.

Definition 3.5.1. Linear SDMM schemes can be constructed in general with the following formula. Given two matrices $A \in \mathbb{F}_q^{t \times s}$ and $B \in \mathbb{F}_q^{s \times r}$, the partitioning of A and B is done in some way, such that the pieces of A and B can be arranged as the vectors

$$(A_1, \ldots, A_m), \quad (B_1, \ldots, B_n)$$

of length m and n, respectively. Notice that the elements in these vectors are matrices over $\mathbb{F}_q$. Now, $2X$ matrices are drawn uniformly at random. These matrices are denoted by $R_1, \ldots, R_X$ and $S_1, \ldots, S_X$. The matrices R_k are the same size as the submatrices A_j, and the matrices $S_{k'}$ are the same size as the submatrices $B_{j'}$. These random matrices are appended to the partition vectors to get the following vectors of matrices

$$(A_1, \ldots, A_m, R_1, \ldots, R_X), \quad (B_1, \ldots, B_n, S_1, \ldots, S_X)$$

of length $m+X$ and $n+X$, respectively. Notice the similarity to linear secret sharing schemes from Section 2.3.

These messages are encoded using $[N, m+X]_q$ and $[N, n+X]_q$ codes $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively. Here, N denotes the number of worker nodes. Let F and G be suitable generator matrices for $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively. The associated codewords are then

$$\widetilde{A} = (\widetilde{A}_1, \ldots, \widetilde{A}_N) = (A_1, \ldots, A_m, R_1, \ldots, R_X)F$$
$$\widetilde{B} = (\widetilde{B}_1, \ldots, \widetilde{B}_N) = (B_1, \ldots, B_n, S_1, \ldots, S_X)G.$$

These vectors contain matrices as elements, which means that the tools from coding theory do not immediately apply. However, multiplying the message with the generator matrix is well-defined since it denotes linear combinations of the submatrices in the message. Therefore, the vector of matrices $\widetilde{A}$ can be interpreted as a matrix of codewords from $\mathcal{C}_A$. By abuse of notation, we shall denote that $\widetilde{A} \in \mathcal{C}_A$ and $\widetilde{B} \in \mathcal{C}_B$. Each worker is sent one component of both vectors, *i.e.*, worker i receives matrices $\widetilde{A}_i$ and $\widetilde{B}_i$. The worker then multiplies these matrices together and returns the result

to the user. In coding theoretic terms, this can be interpreted as a star product of the vectors $\widetilde{A}$ and $\widetilde{B}$. Hence, we may write

$$\widetilde{A} \star \widetilde{B} \in \mathcal{C}_A \star \mathcal{C}_B.$$

Let H be a suitable generator matrix for $\mathcal{C}_A \star \mathcal{C}_B$ and $\mathcal{I} \subset [N]$ the indices of the fastest responses. The generator matrix H should not be confused with a parity-check matrix. The result can be obtained by computing $(\widetilde{A} \star \widetilde{B})_{\mathcal{I}} H_{\mathcal{I}}^{-1}$, provided that $\mathcal{I}$ is an information set of $\mathcal{C}_A \star \mathcal{C}_B$.

The choice of the codes and generator matrices is important as it defines the correctness and security of the scheme.

It is worth noting that if $\mathcal{C}_A \star \mathcal{C}_B$ is an MDS code, only a certain number of fastest responses are required before decoding can be done. This is possible since any set that is large enough is an information set. This general construction does not yet provide any valid scheme, since the correctness of the result relies on how the generator matrices F, G and H are chosen. The following constructions define the SDMM schemes presented earlier using the linear SDMM framework. Using these constructions, we will prove the security of the schemes.

Construction 3.5.2 (Secure MatDot)**.** Consider the matrices $A \in \mathbb{F}_q^{t \times s}$ and $B \in \mathbb{F}_q^{s \times r}$. In the secure MatDot code, the matrices are partitioned to p pieces. The partitions look like

$$A = \begin{pmatrix} A_1 & \dots & A_p \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ \vdots \\ B_p \end{pmatrix},$$

where $A_j \in \mathbb{F}_q^{t \times \frac{s}{p}}$ and $B_{j'} \in \mathbb{F}_q^{\frac{s}{p} \times r}$. Let $\alpha_1, \dots, \alpha_N \in \mathbb{F}_q^\times$ be distinct nonzero points and $\alpha = (\alpha_1, \dots, \alpha_N)$. Define the codes as $\mathcal{C}_A = \mathrm{RS}_{p+X}(\alpha)$ and $\mathcal{C}_B = \mathrm{RS}_{p+X}(\alpha)$, with generator matrices

$$F = \begin{pmatrix} 1 & \dots & 1 \\ \alpha_1 & \dots & \alpha_N \\ \vdots & \ddots & \vdots \\ \alpha_1^{p-1} & \dots & \alpha_N^{p-1} \\ \alpha_1^{p} & \dots & \alpha_N^{p} \\ \vdots & \ddots & \vdots \\ \alpha_1^{p+X-1} & \dots & \alpha_N^{p+X-1} \end{pmatrix}, \quad G = \begin{pmatrix} \alpha_1^{p-1} & \dots & \alpha_N^{p-1} \\ \vdots & \ddots & \vdots \\ \alpha_1 & \dots & \alpha_N \\ 1 & \dots & 1 \\ \alpha_1^{p} & \dots & \alpha_N^{p} \\ \vdots & \ddots & \vdots \\ \alpha_1^{p+X-1} & \dots & \alpha_N^{p+X-1} \end{pmatrix}.$$

Notice that both of these are Vandermonde matrices, except that the upper part of G is reversed. Thus, the encodings are the same as in Construction 3.3.1. If $N \geq 2p + 2X - 1$, then the star product of $\mathcal{C}_A$ and $\mathcal{C}_B$ is $\mathcal{C}_A \star \mathcal{C}_B = \mathrm{RS}_{2p+2X-1}(\alpha)$ by Proposition 2.1.21. The product AB can be recovered from the encoded products $\widetilde{C}_i$ by decoding with the standard Vandermonde generator matrix and selecting the pth entry of the resulting vector, as seen in Construction 3.3.1.

The following construction is a special case of the GASP code, but we do not consider the "gaps" in the product polynomial $h(x)$. This makes the choice of evaluation points simpler than in the general GASP code.

Construction 3.5.3 (GASP_{big})**.** Consider the matrices $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$, which are partitioned to m and n pieces, respectively. The partitions are

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_m \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & \dots & B_n \end{pmatrix},$$

where $A_j \in \mathbb{F}_q^{\frac{t}{m}\times s}$ and $B_{j'} \in \mathbb{F}_q^{s\times \frac{r}{n}}$. Let $\alpha_1, \dots, \alpha_N \in \mathbb{F}_q^\times$ be distinct nonzero points and $\alpha = (\alpha_1, \dots, \alpha_N)$. Then, define the codes $\mathcal{C}_A$ and $\mathcal{C}_B$ using the following generator matrices

$$F = \begin{pmatrix} 1 & \dots & 1 \\ \alpha_1 & \dots & \alpha_N \\ \vdots & \ddots & \vdots \\ \alpha_1^{m-1} & \dots & \alpha_N^{m-1} \\ \alpha_1^{mn} & \dots & \alpha_N^{mn} \\ \vdots & \ddots & \vdots \\ \alpha_1^{mn+X-1} & \dots & \alpha_N^{mn+X-1} \end{pmatrix}, \quad G = \begin{pmatrix} 1 & \dots & 1 \\ \alpha_1^{m} & \dots & \alpha_N^{m} \\ \vdots & \ddots & \vdots \\ \alpha_1^{m(n-1)} & \dots & \alpha_N^{m(n-1)} \\ \alpha_1^{mn} & \dots & \alpha_N^{mn} \\ \vdots & \ddots & \vdots \\ \alpha_1^{mn+X-1} & \dots & \alpha_N^{mn+X-1} \end{pmatrix}.$$

The encodings are then the same as in Construction 3.4.4. Therefore, the encoded products are from the code $\mathcal{C}_A \star \mathcal{C}_B \subset \text{RS}_{2mn+2X-1}(\alpha)$. As seen in Construction 3.4.4, the submatrices can be recovered as the first mn elements of the result decoded with the standard Vandermonde generator matrix.

Construction 3.5.4 (GASP)**.** Now consider GASP codes. The encoding is done using a polynomial evaluation code, where the terms of the polynomials are determined by optimizing the degree table. Let φ and γ be the vectors that contain the exponents of the encoding polynomials. The generator matrices are

$$F = \begin{pmatrix} \alpha_1^{\varphi_1} & \dots & \alpha_N^{\varphi_1} \\ \vdots & \ddots & \vdots \\ \alpha_1^{\varphi_{m+X}} & \dots & \alpha_N^{\varphi_{m+X}} \end{pmatrix}, \quad G = \begin{pmatrix} \alpha_1^{\gamma_1} & \dots & \alpha_N^{\gamma_1} \\ \vdots & \ddots & \vdots \\ \alpha_1^{\gamma_{n+X}} & \dots & \alpha_N^{\gamma_{n+X}} \end{pmatrix}.$$

Thus, $\mathcal{C}_A$ is an $[N, m+X]$ code and $\mathcal{C}_B$ is an $[N, n+X]$ code. The next relevant code is $\mathcal{C}_A \star \mathcal{C}_B$, which is characterized by its generator matrix

$$H = \begin{pmatrix} \alpha_1^{\eta_1} & \dots & \alpha_N^{\eta_1} \\ \vdots & \ddots & \vdots \\ \alpha_1^{\eta_{m+X}} & \dots & \alpha_N^{\eta_{m+X}} \end{pmatrix},$$

where $\mathcal{H} = \{\eta_1, \dots, \eta_{|\mathcal{H}|}\}$ is the set of distinct elements in the degree table, as described in Section 3.4. Choosing the evaluation points such that H has invertible maximal submatrices is not trivial, but it was shown in [16] that such a choice is possible as long as the field is large enough.

The following theorem will highlight the usefulness of the linear SDMM framework, since the security of the schemes can be proven by checking the properties of the generator matrices. A matrix has the MDS property if and only if it generates an MDS code, *i.e.*, all of its maximal submatrices are invertible.

Theorem 3.5.5. *Let F and G be the generator matrices used in the encoding of some linear SDMM scheme. Then the scheme is secure against X-collusion if $F^{>m}$ and $G^{>n}$ have the MDS property.*

Proof. Let $\mathcal{X} \subset [N]$, $|\mathcal{X}| = X$, be a set of X colluding nodes. Writing the generator matrix F as

$$F = \begin{pmatrix} F^{\leq m} \\ F^{>m} \end{pmatrix}$$

allows us to write the shares the colluding nodes have about the encoded matrix $\widetilde{A}$ as

$$\widetilde{A}_{\mathcal{X}} = (A_1, \ldots, A_m)F_{\mathcal{X}}^{\leq m} + (R_1, \ldots, R_X)F_{\mathcal{X}}^{>m}.$$

If $F^{>m}$ has the MDS property, then any $X \times X$ submatrix of $F^{>m}$ is invertible. As $(R_1, \ldots, R_X)$ is uniformly distributed, we get that $(R_1, \ldots, R_X)F_{\mathcal{X}}^{>m}$ is also uniformly distributed. According to Corollary 2.2.26

$$I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}}) = 0.$$

The idea is that the confidential data of A is hidden by adding uniformly random noise. A similar argument works for the matrix B. Finally, we get that

$$\begin{aligned} I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}}) &= I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}) + I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{B}}_{\mathcal{X}} \mid \widetilde{\mathbf{A}}_{\mathcal{X}}) \\ &= H(\widetilde{\mathbf{A}}_{\mathcal{X}}) - H(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}, \mathbf{B}) + H(\widetilde{\mathbf{B}}_{\mathcal{X}} \mid \widetilde{\mathbf{A}}_{\mathcal{X}}) - H(\widetilde{\mathbf{B}}_{\mathcal{X}} \mid \widetilde{\mathbf{A}}_{\mathcal{X}}, \mathbf{A}, \mathbf{B}) \\ &\leq H(\widetilde{\mathbf{A}}_{\mathcal{X}}) - H(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}) + H(\widetilde{\mathbf{B}}_{\mathcal{X}}) - H(\widetilde{\mathbf{B}}_{\mathcal{X}} \mid \mathbf{B}) \\ &= I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}}) + I(\mathbf{B}; \widetilde{\mathbf{B}}_{\mathcal{X}}) \\ &= 0. \end{aligned}$$

The inequality follows from $\widetilde{\mathbf{A}}_{\mathcal{X}}$ is conditionally independent of $\mathbf{B}$ given $\mathbf{A}$, conditioning lowers the entropy, and $\widetilde{\mathbf{B}}_{\mathcal{X}}$ is conditionally independent of $\widetilde{\mathbf{A}}_{\mathcal{X}}$ and $\mathbf{A}$ given $\mathbf{B}$. As the mutual information is bounded from below by 0, we get that

$$I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}}) = 0.$$

If $\mathcal{X} \subset [X]$ has fewer than X worker nodes, it is obvious that the information leaked to them is bounded from above by the information leaked to any X nodes. Hence, the linear SDMM scheme leaks no information about A and B to any colluding set of at most X worker nodes.

□

The security of the SDMM schemes presented earlier follows as simple corollaries of the above theorem.

Corollary 3.5.6. *The secure MatDot scheme is information-theoretically secure against X-collusion.*

Proof. The MatDot scheme is a linear SDMM scheme with $m = n = p$. According to Construction 3.5.2 the generator matrices F and G are defined such that $F^{>p}$ and $G^{>p}$ can be written as

$$F^{>p} = G^{>p} = \operatorname{diag}(\alpha)^p V,$$

where α is the vector of evaluation points and V is the $X \times N$ Vandermonde matrix on α. As V has the MDS property and $\operatorname{diag}(\alpha)^p$ is invertible, the matrices $F^{>p}$ and $G^{>p}$ have the MDS property. The result then follows from Theorem 3.5.5. □

Corollary 3.5.7. *The* $\mathrm{GASP}_{\mathrm{big}}$ *scheme is information-theoretically secure against X-collusion.*

Proof. The $\mathrm{GASP}_{\mathrm{big}}$ scheme is a linear SDMM scheme with generator matrices F and G such that

$$F^{>m} = G^{>n} = \operatorname{diag}(\alpha)^{mn} V,$$

where α is the vector of nonzero evaluation points and V is the $X \times N$ Vandermonde matrix on α. The matrices $F^{>p}$ and $G^{>p}$ have the MDS property because V has the MDS property and $\operatorname{diag}(\alpha)^p$ is invertible. The result then follows from Theorem 3.5.5. □

The security of the general GASP scheme is not as easy to prove, since it requires the evaluation points α to be chosen carefully. It was shown in [16] that this is possible, provided that the field is large enough.

3.6 Error correction in SDMM

Errors can occur in a computation at any time for multiple different reasons, which means it is important to mitigate the effect these errors can have. There are many different ways of correcting errors in the computation and communication phases of an SDMM scheme. For example, coding theory is used to provide reliable communication over an unreliable channel. However, the result of a distributed computation scheme is heavily affected if the worker nodes are not reliable. Worker nodes that intentionally give incorrect results are known as *Byzantine workers*. In our SDMM scheme, we want to provide robustness against a certain number of Byzantine worker nodes, which means that even if some workers return incorrect results, we can still decode the correct product AB.

An obvious idea is to use coding theoretic tools to provide robustness to the computation. The linear SDMM framework presented earlier allows for this since the responses can be seen as be seen as codewords in the star product code $\mathcal{C}_A \star \mathcal{C}_B$. To be more specific, the responses contain one codeword for each matrix entry. Each of these codewords can then be error-corrected using some error correction algorithm on $\mathcal{C}_A \star \mathcal{C}_B$. For example, the star product code in the secure MatDot code with partitioning parameter p is $\mathrm{RS}_K(\alpha)$, where $K = 2p + 2X - 1$. There are efficient algorithms that can correct up to $\lfloor \frac{N-K}{2} \rfloor$ errors in any codeword. Therefore, we need

2 additional worker nodes for each Byzantine node. As we only need one additional node for every straggling worker node, it is more expensive to account for Byzantine workers than straggling workers.

Using the linear SDMM framework, we can interpret the responses as codewords in some code for each matrix entry. For example, if $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$ are the input matrices, then the responses of the secure MatDot scheme are tr codewords in $\mathrm{RS}_K(\alpha)$. Correspondingly, the responses in the GASP code are $\frac{tr}{mn}$ codewords in some code defined by the degree table. If worker i returns an incorrect result while all other workers return correct results, then each codeword in the response will have at most one error and it will always be at position i, since worker node i can only affect that coordinate. Similarly, in the case that there are multiple Byzantine workers, the errors will be in the same positions in each of the codewords. We can use the structure of this to further increase the number of errors that can be corrected.

When sending data over an unreliable channel, errors can occur as *single errors*, where a single symbol is changed, or as *burst errors*, where multiple consecutive symbols are changed. Single errors can be corrected using regular error correcting codes. The problem with burst errors is that there might be too many errors in a single codeword to correct that codeword, even if the total number of errors is relatively small. However, the structure of the error pattern can be utilized by using *interleaved codes*. With interleaved codes, multiple codewords are sent such that first the first symbol of each codeword is sent, then the second symbol of each codeword, *etc.* The receiver can then rearrange the symbols in the same way. If a burst error happens, multiple codewords are affected, and each codeword only contains a few errors, which can be corrected. This is because the errors are always in the same symbols in the codewords.

Interleaved codes can be used in linear SDMM schemes, since errors are always in the same positions in the codewords. In [35] it was shown that it is possible to correct up to $d-2$ errors with interleaved Reed–Solomon codes, where d is the minimum distance of the underlying Reed–Solomon code. This is nearly twice as many errors as the non-interleaved code. Hence, the cost of a Byzantine worker is the same as the cost of a straggling worker; both require one additional worker node to be added.

Even though it is possible to correct up to $d-2$ errors caused by Byzantine workers, this is possible only for some error patterns. The success probability of an interleaved Reed–Solomon decoder was analyzed in [35, 36], when the errors are assumed to be uniformly distributed burst errors. This assumption is not necessarily valid when discussing Byzantine workers in a linear SDMM scheme. It is an interesting future research topic to analyze the error correction capabilities of interleaved codes against Byzantine workers in the linear SDMM framework.

4 Cooperative SDMM

There has been substantial research on ways to lower the communication cost of secure distributed matrix multiplication schemes. For example, GASP codes choose the polynomials $f(x)$ and $g(x)$ in a clever way to reduce the number of terms in the degree table. A novel way to reduce the download cost from the user's perspective was introduced by the author in [3], where the collusion ability of the nodes is utilized to outsource some of the communication cost to the worker nodes. The total communication cost over the network is not reduced, but rather some of it is moved to the worker nodes. This might be advantageous if the communication cost between nodes is lower than the communication between the worker nodes and the user. The new communication cost between the worker nodes is measured by the *cooperation cost* of the scheme. For the noncooperative schemes, the cooperation cost is zero as there is no communication between the nodes. The total communication cost is the sum of the upload cost, download cost, and the cooperation cost.

Some care has to be taken to make sure that the security of the scheme is not compromised. An important assumption in noncooperative SDMM schemes is that there are at most X colluding workers in any given collusion set. The authors of [3] present two different cooperative schemes, which have different assumptions on the cooperation capabilities of the worker nodes. The information-theoretic scheme is able to provide the same information-theoretic security as the noncooperative SDMM schemes seen earlier. The encryption-based scheme is able to lower the download cost even further, with the drawback that the scheme is only computationally secure.

The collusion and cooperation abilities of the network are characterized by the collusion and cooperation graphs, G_{coll} and G_{coop}, respectively. These are simple graphs on N nodes, where N is the number of worker nodes. Any two nodes have an edge between them in the collusion graph if they are able to collude, and similarly in the cooperation graph. The X-collusion assumption can then be expressed in terms of the collusion graph G_{coll}. We say that a network has X-collusion if the largest connected component of the collusion graph G_{coll} is of size X.

4.1 Information-theoretic cooperation

We will first consider an information-theoretically secure cooperative SDMM scheme. The main idea of the scheme is that any colluding nodes also have the ability to cooperate, *i.e.*, $G_{\text{coll}} = G_{\text{coop}}$. The following example shows how the recovery phase of a secret sharing scheme can be done by downloading less compared to a standard secret sharing scheme.

Example 4.1.1. Consider a secret sharing scheme with n nodes. Let $G_{\text{coll}} = (V, E)$ be the collusion graph of the nodes, with γ connected components. The collusion sets are the connected components of G_{coll} and are denoted by $V_1, \dots, V_\gamma$ such that $\bigsqcup_{c=1}^{\gamma} V_c = V$. Let $\tilde{a}_1, \dots, \tilde{a}_n$ be the shares of the nodes. The secret can be recovered by some linear combination of the shares, say $a = \sum_i \lambda_i \tilde{a}_i$. These shares can be thought of as matrices over a field and the coefficients as field elements, but more general intepretations are possible as well.

Each collusion set chooses a designated node, which is sent the shares of the other nodes in the set. Then, each designated node computes their share of the linear combination, *i.e.*, $\sum_{i\in V_c}\lambda_i\tilde{a}_i$. The designated node sends this to the user, who computes the sum of the answers it received, which equals

$$\sum_{c=1}^{\gamma}\sum_{i\in V_c}\lambda_i\tilde{a}_i=\sum_{i}\lambda_i\tilde{a}_i=a.$$

Therefore, the user was able to recover the share by downloading only from γ nodes.

The above recovery of the secret in a secret sharing scheme can easily be utilized in SDMM, since the download phase in SDMM is the recovery phase of a secret sharing scheme. The following cooperative scheme is based on the secure MatDot scheme from [15] and the cooperative version of the scheme was presented in [3].

Construction 4.1.2 (Cooperative secure MatDot)**.** Let p be the partitioning parameter and let $A\in\mathbb{F}_q^{t\times s}$ and $B\in\mathbb{F}_q^{s\times r}$ be matrices. Partition A and B to block matrices such that

$$A=\begin{pmatrix}A_1 & \dots & A_p\end{pmatrix},\quad B=\begin{pmatrix}B_1\\ \vdots\\ B_p\end{pmatrix}.$$

The product of A and B is

$$AB=\sum_{j=1}^{p}A_jB_j.$$

Let $R_k\in\mathbb{F}_q^{t\times\frac{s}{p}}$ and $S_{k'}\in\mathbb{F}_q^{\frac{s}{p}\times r}$ be chosen uniformly at random and independently of each other. Now, let us form the polynomials

$$f(x)=\sum_{j=1}^{p}A_jx^{j-1}+\sum_{k=1}^{X}R_kx^{p+k-1},$$
$$g(x)=\sum_{j'=1}^{p}B_{j'}x^{p-j'}+\sum_{k'=1}^{X}S_{k'}x^{p+k'-1}.$$

Choose evaluation points $\alpha_1,\dots,\alpha_N$, where N is the number of worker nodes. The points should be distinct and nonzero. Evaluate the polynomials $f(x)$ and $g(x)$ at the evaluation points. Each worker node i is sent the encoded matrices

$$\widetilde{A}_i=f(\alpha_i),\quad \widetilde{B}_i=g(\alpha_i)$$

through a secure communication channel. Worker i computes the matrix product $\widetilde{C}_i=\widetilde{A}_i\widetilde{B}_i\in\mathbb{F}_q^{t\times r}$. The resulting products are now evaluations of the polynomial $h(x)=f(x)g(x)$. The product AB is the coefficient of x^{p-1} of the $2p+2X-2$ degree polynomial $h(x)$, as seen in Construction 3.3.1. Therefore, the result can be obtained by some linear combination of any of $K=2p+2X-1$ evaluations.

Now, we assume that the fastest K worker nodes are in collusion sets of size X, except for one, which has size K mod X. This means that there are $\lceil\frac{K}{X}\rceil$ colluding

sets. We assume that each node is able to cooperate within their colluding set. Each collusion set chooses a designated node, and each node in that collusion set sends their encoded product $\widetilde{C}_i$ to the designated node.

Once the designated worker has received their pieces from the collusion set, the designated worker computes the linear combination

$$\sum_{i \in V_c} \ell_i^{(p-1)} \widetilde{C}_i,$$

where $\ell_1(x), \ldots, \ell_K(x)$ are the Lagrange basis polynomials on the evaluation points of the K fastest nodes. The designated workers then send these results to the user.

Once the user has received all the responses, they can compute the sum

$$\sum_{c=1}^{\lceil \frac{K}{X} \rceil} \sum_{i \in V_c} \ell_i^{(p-1)} \widetilde{C}_i = \sum_{i=1}^{K} \ell_i^{(p-1)} \widetilde{C}_i = AB.$$

This scheme is still information-theoretically secure, since any node does not receive any additional information outside of their collusion set. Therefore, the underlying secure MatDot scheme provides for security under X-collusion.

From the user's point of view, the scheme works in a similar way to the noncooperative MatDot scheme, except the user needs to download only from $\lceil \frac{K}{X} \rceil$ nodes. As a result, the total download cost for the user is $tr\lceil \frac{K}{X} \rceil$ and the cooperation cost is $tr(K - \lceil \frac{K}{X} \rceil)$, where $K = 2p + 2X - 1$ is the recovery threshold.

It is easy to see that other noncooperative SDMM schemes can also be converted to information-theoretically secure cooperative schemes using the above construction. It depends on the partitioning scheme whether the cooperation is able to reduce the download cost. The outer product partitioning schemes are able to achieve low download cost since the encoded products $\widetilde{C}_i$ are only $\frac{1}{mn}$ of the size of the final product AB. The inner product partitioning schemes do not have this property since the encoded products are the same size as the final product AB. In the cooperative construction, the responses from the designated workers will be the same size as the final product AB, which means that the outer product partitioning schemes, including the GASP code, will not achieve lower download costs with the cooperative scheme.

The cooperative scheme above assumes that each of the collusion sets in the set of the fastest K nodes consists of sets of X nodes and the remainder K mod X. This assumption is not realistic in most situations where the fastest nodes are dynamically determined. Additionally, the download cost in the cooperative scheme improves by adding more collusion to the network, which seems to be contradictory from the point of view of the security. If we do not assume that the fastest K nodes form maximal sized collusion sets, then the download cost of the scheme increases.

When the designated workers compute the linear combinations using the Lagrange interpolation polynomials, they need to know exactly which nodes are part of the fastest K nodes. This requires some communication between the different collusion sets. Such communication can potentially go through the user, as the amount of information that needs to be transmitted for that is relatively small.

In this cooperative scheme, the nodes are able to organize the cooperation without the user's help, *i.e.*, the user does not need to know which nodes are potentially colluding with each other. However, as colluding is seen as something that goes against the rules of the scheme, it is not clear why the nodes would use that capability to help the user.

4.2 Encryption-based cooperation

The information-theoretically secure cooperative SDMM scheme assumes that worker nodes that cooperate can also collude. However, the download cost for the user can be further reduced if more nodes are able to cooperate. Therefore, some other measures need to be taken to provide security to any X colluding workers. In the previous scheme, we were limited by being able to send at most X shares to any particular designated node, since otherwise the scheme would not be secure. Using techniques from encryption, we are able to get rid of this restriction. The encryption must still allow the designated nodes to process the data such that the total download cost is low.

The following encryption-based cooperative schemes were first presented by the author in [3]. The example below shows how a pseudorandom generator can be used to efficiently recover a shared secret.

Example 4.2.1. Consider a secret sharing scheme with N shares $\tilde{a}_1, \dots, \tilde{a}_N$. The secret can be recovered with some linear combination of the shares, *i.e.*,

$$a = \sum_i \lambda_i \tilde{a}_i.$$

Each worker node now chooses a random encryption key k_i and generates a pseudorandom value $z_i = G(k_i)$ using a pseudorandom generator G. The encryption key is sent directly to the user through a secure communication channel. Each worker then computes $\tilde{a}_i + z_i$ and sends that to a designated node, say node 1. Node 1 can then compute

$$\sum_i \lambda_i (\tilde{a}_i + z_i),$$

and send it to the user. The user can then use the pseudorandom generator G and the encryption keys to compute $z_1, \dots, z_N$, and then

$$\sum_i \lambda_i (\tilde{a}_i + z_i) - \sum_i \lambda_i z_i = \sum_i \lambda_i \tilde{a}_i = a.$$

Therefore, the user recovers the secret by downloading data from just one node.

Clearly, no additional information is revealed to any node $i = 2, \dots, N$ because they only send data and do not receive anything they would not receive in a noncooperative secret sharing scheme. Only node 1 receives additional data, in particular, they receive the values $\tilde{a}_i + z_i$. These values are encryptions of $\tilde{a}$ with the encryption scheme from Construction 2.4.6. As demonstrated in Proposition 2.4.7, this scheme is secure against eavesdropping attacks, *i.e.*, the node 1 does not gain any additional information that they are able to utilize, since we assume it has bounded computational resources.

In the above scheme, if the shares are finite field values from some reasonably-sized finite field, then the encryption keys are likely larger than the shares themselves. Hence, the scheme would not reduce the download cost for the user since transmitting the keys would be worse than just transmitting the shares. However, if the shares are some large matrices, then the size of the key can be negligible compared to the size of the share. In that case, the download cost would be significantly reduced. Currently, a common size for encryption keys would be 128–256 bits, which is much smaller than even a 100×100 matrix, where each finite field takes 8 bits to represent. Such a matrix would take 80000 bits of storage.

Definition 4.2.2. An SDMM scheme is said to be *computationally secure against X-collusion* if the data accessible to any colluding set $\mathcal{X} \subset [N]$, $|\mathcal{X}| \leq X$, is computationally indistinguishable from uniformly random noise.

Construction 4.2.3 (Encryption-based cooperative MatDot)**.** Let $A \in \mathbb{F}_q^{t\times s}$ and $B \in \mathbb{F}_q^{s\times r}$ be the input matrices, and p the partitioning parameter. The encoded pieces are computed as in Construction 3.3.1. The pieces $\widetilde{A}_i$ and $\widetilde{B}_i$ are then sent to node i through a secure communication channel. Worker node i computes $\widetilde{C}_i = \widetilde{A}_i\widetilde{B}_i$. The encryption is done in the same way as in Construction 2.4.6, *i.e.*, a pseudorandom value Z_i is chosen using a pseudorandom generator G and a secret key k_i. The key k_i is sent to the user using a secure communication channel and the encrypted value $\widehat{C}_i = \widetilde{C}_i + Z_i$ is sent to a designated worker node, say node 1.

Once the designated worker has received enough responses from the other nodes, it computes the linear decoding process using the encrypted responses. In this case the designated node needs $K = 2p + 2X - 1$ values, since that is the recovery threshold of the secure MatDot code as seen in Construction 3.3.1. Let $\mathcal{I} = \{i_1, \ldots, i_K\} \subset [N]$ be the fastest K worker nodes to send a response to the designated node. Then, the designated node computes the Lagrange interpolation polynomials $\ell_{i_1}, \ldots, \ell_{i_K}$ for the evaluation points $\alpha_{i_1}, \ldots, \alpha_{i_K}$ and computes the linear combination

$$\sum_{i\in\mathcal{I}} \ell_i^{(p-1)}\widehat{C}_i.$$

The designated worker can now send this to the user.

Once the user receives the response from the designated worker, they can compute

$$\sum_{i\in\mathcal{I}} \ell_i^{(p-1)}\widehat{C}_i - \sum_{i\in\mathcal{I}} \ell_i^{(p-1)}Z_i = \sum_{i\in\mathcal{I}} \ell_i^{(p-1)}\widetilde{C}_i = AB.$$

This is possible, since the user knows the secret keys k_i and can therefore compute $Z_i = G(k_i)$. It is now clear that the download cost for the user is only tr, since that is the size of the final matrix AB. The cooperation cost is now $(K-1)tr$, since $K-1$ fastest worker nodes need to send their encrypted pieces to the designated node. The designated node does not need to send their own piece to themselves, since they already have it. It is worth noting that the total communication cost is the same as in the noncooperative version, but most of it is outsourced to the worker nodes.

Again, it is worth noting that sending the encryption keys from the worker nodes to the user is insignificant compared to the sizes of the matrices. Therefore, we can ignore them in our analysis. Now, we wish to show that the above construction is indeed computationally secure against any X colluding worker nodes.

Theorem 4.2.4. *The cooperative MatDot scheme based on encryption is computationally secure against any X colluding worker nodes.*

Proof. Let $\mathcal{X} \subset [N]$, $|\mathcal{X}| \leq X$, be a set of colluding workers. If $\mathcal{X}$ does not contain the designated worker, which receives the encrypted pieces, then this set of nodes only has access to at most X encoded pieces. As the underlying secure MatDot scheme is information-theoretically secure against X-collusion, it is also computationally secure against these colluding nodes.

The nontrivial part of the proof is to show that the access to the K encrypted pieces does not reveal too much information to a computationally bounded adversary. If the colluding set contains the designated node, then they have access to the encoded pieces $\widetilde{A}_{\mathcal{X}}$ and $\widetilde{B}_{\mathcal{X}}$, and the encrypted pieces $\widehat{C}_{\mathcal{I}}$ for the recovery set $\mathcal{I} \subset [N]$. Assume, for the sake of contradiction, that this data is not computationally indistinguishable from uniform noise. The encoded pieces $\widetilde{A}_{\mathcal{X}}$ and $\widetilde{B}_{\mathcal{X}}$ are clearly uniformly distributed, since the underlying MatDot scheme is information-theoretically secure against X-collusion. Therefore, the encrypted pieces $\widehat{C}_{\mathcal{I}}$ are not computationally indistinguishable from uniform noise, *i.e.*, there is a distinguisher that recognizes the data from uniform noise with nonnegligible probability. Thus, we can build an efficient adversary against one of the encryptions by simulating the other encrypted pieces in $\widehat{C}_{\mathcal{I}}$ and passing that to the above distinguisher. This contradicts the IND-EAV security of the encryption scheme presented in Construction 2.4.6. Thus, the cooperative MatDot scheme based on encryption is computationally secure against X-collusion. □

It is clear that any noncooperative linear SDMM scheme can be transformed to a cooperative SDMM scheme using the formula presented in Construction 4.2.3. Such a scheme always achieves the download cost of tr, *i.e.*, the size of the matrix AB. It is clear that this is the lowest possible download cost. However, the communication cost between the worker nodes naturally increases as the total communication cost cannot be lower compared to the noncooperative SDMM scheme. In the secure MatDot code, the total communication cost is not increased by transforming to the cooperative MatDot code. Such improvement is not possible for codes with other types of partitioning. For example, in the cooperative GASP scheme, the workers send encoded matrices of size $\frac{t}{m} \times \frac{r}{n}$ to the designated worker. Therefore, the cooperation cost is $(K-1)\frac{tr}{mn}$ and the total communication cost is $(K-1)\frac{tr}{mn} + tr > K\frac{tr}{mn}$, when $m, n > 1$. This increase in total communication cost is only minor.

In the case that not all of the worker nodes are able to cooperate with each other, it is possible to adapt the construction of the cooperative SDMM scheme presented above. In such a scheme, the workers that are able to cooperate will send their encrypted pieces to some designated node. There can therefore be multiple designated nodes for different cooperation sets. The download cost for the user would then be $N_C tr$, where N_C is the number of cooperation sets. There needs

to still be some communication between the different cooperation sets, since the designated nodes need to know what decoding matrix to use, which depends on which worker nodes are part of the recovery set. This communication can again go through the user, since it is small in size. As a special case of this scheme we get the information-theoretically secure cooperative SDMM scheme when all the cooperation sets contain at most X nodes. In case the cooperation sets and the collusion sets coincide, it is not necessary to use the encryption, since any node will access only X encoded pieces, which does not break the security assumption.

The pseudorandom generator used to build the encryption-based cooperative SDMM scheme must generate a pseudorandom matrix of appropriate size in the finite field $\mathbb{F}_q$. The majority of practical applications of pseudorandom generators, and encryption algorithms in general, are done over binary, which means they are not immediately applicable. A commonly used cipher, such as AES, can be used in a practical application by reducing the output of the cipher modulo the size of the field. This will give a pseudorandom function that outputs elements in an arbitrary set, such as a field.

The reason the IND-EAV security definition was chosen instead of something stronger is that the keys are never reused. It is possible to expand the construction to work with a stronger security definition such as IND-CPA. This requires using randomness in the encryption process in addition to the pseudorandomness. This will increase the download cost slightly, but is not a major consideration.

5 SDMM over the analog domain

This far, we have discussed everything over finite fields, which is a natural domain for coding theory. Furthermore, as shown in Sections 2.2.1 and 2.2.2, information theory is simpler for finite domains than for continuous domains. Computations over finite fields, on the other hand, are impractical in real-world applications because finite field arithmetic is computationally expensive and most real-world data is either integers or real numbers. Some computations, especially those over the integers, can be approximated using large prime fields. However, such computations are slow because modular arithmetic is computationally more expensive than integer operations. Therefore, secure distributed computation schemes over real numbers would be useful.

The first idea would be to convert the earlier schemes directly to work over real numbers. There are a few problems with this approach. The first is that there is no uniform distribution over the real numbers, so the distribution of the random matrices is not immediately clear. The second problem concerns numerical stability, since real numbers cannot be represented with arbitrary precision on a computer. Therefore, each computation presents some numerical errors to the result. To get a useful result, these errors should be kept to a minimum.

According to Proposition 2.2.13 the uniform distribution over a finite field has the maximal entropy. Over the real and complex numbers, this is achieved by the normal distribution according to Proposition 2.2.21 and Theorem 2.2.22. Using the maximal entropy distribution to hide information is a natural choice, since we want to introduce the maximal amount of noise to the computation. The variance of the normal distribution will need to be computed such that the required security properties are fulfilled.

Matrix multiplication using floating point numbers introduces some numerical errors to the computation since floating point numbers have finite precision. This error is proportional to the sizes of the elements in the matrices [37]. The decoding process of an SDMM scheme involves solving a system of linear equations, which amplifies the relative errors from the matrix multiplication by the condition number of the decoding matrix. Therefore, we wish to make the condition number of the decoding matrix as small as possible.

Secure distributed computation has been recently researched in [38, 39] in the context of Lagrange coded computation over the complex numbers. Secret sharing and computation over real numbers has been researched in [40]. The results presented in this section were first published by the author in [4].

5.1 Secret sharing over the analog domain

As seen in Section 2.3 secret sharing is a way of distributing a secret to N parties such that some subset of those parties can use their shares to reconstruct the secret. A (K, N)-threshold secret sharing scheme allows for any K out of the N parties to recover the secret, which is usually considered to be a finite field value. Secret sharing over the real numbers was recently considered in [40]. The following example

shows how such a scheme works and how the security is determined.

Example 5.1.1. Consider a $(2,3)$-threshold secret sharing scheme, where any 2 of the 3 parties can recover the secret. Let $a \in \mathbb{R}$ be the secret. The shares are constructed with

$$\tilde{a}_i = a + r\alpha_i,$$

for $i = 1, 2, 3$, where $r \in \mathbb{R}$ is drawn at random from some distribution independently of a. Without loss of generality, assume that parties 1 and 2 want to recover the secret from their shares $\tilde{a}_1$ and $\tilde{a}_2$. Consider the linear combination

$$\alpha_2\tilde{a}_1 - \alpha_1\tilde{a}_2 = (\alpha_2 - \alpha_1)a + (\alpha_2\alpha_1 - \alpha_1\alpha_2)r = (\alpha_2 - \alpha_1)a.$$

The secret a can then be recovered, provided that the points $\alpha_1, \alpha_2, \alpha_3$ are distinct.

Let us now analyze the amount of information leaked from one share. To do this we consider the values a and r to come from some probability distributions as the values of the random variables $\mathbf{a}$ and $\mathbf{r}$, respectively. Similarly, the share $\tilde{a}_i$ is the value of the random variable $\tilde{\mathbf{a}}_i$. Using the definition of mutual information, we get that

$$\begin{aligned} I(\tilde{\mathbf{a}}_i; \mathbf{a}) &= h(\tilde{\mathbf{a}}_i) - h(\tilde{\mathbf{a}}_i \mid \mathbf{a}) \\ &= h(\mathbf{a} + \mathbf{r}\alpha_i) - h(\mathbf{a} + \mathbf{r}\alpha_i \mid \mathbf{a}) \\ &= h(\mathbf{a} + \mathbf{r}\alpha_i) - h(\mathbf{r}\alpha_i). \end{aligned}$$

The last step follows from the independence of $\mathbf{a}$ and $\mathbf{r}$. Assuming that the distribution of $\mathbf{a}$ is continuous, we can use Proposition 2.2.21. The variance of the sum is given by $\mathrm{Var}(\mathbf{a} + \mathbf{r}\alpha_i) = \sigma_a^2 + \alpha_i^2\sigma_r^2$. Therefore,

$$h(\mathbf{a} + \mathbf{r}\alpha_i) \leq \frac{1}{2}\log(2\pi e(\sigma_a^2 + \alpha_i^2\sigma_r^2)).$$

On the other hand, we may choose that $\mathbf{r}$ is normally distributed with mean 0 and variance σ_r^2. Therefore,

$$h(\mathbf{r}\alpha_i) = \frac{1}{2}\log(2\pi e\alpha_i^2\sigma_r^2).$$

As a result, the amount of leaked information from share $\tilde{\mathbf{a}}_i$ is

$$I(\tilde{\mathbf{a}}_i; \mathbf{a}) \leq \frac{1}{2}\log(1 + \frac{\sigma_a^2}{\alpha_i^2\sigma_r^2}).$$

By increasing σ_r^2 we can make the amount of leaked information as small as we wish. It is also clear that we cannot have the situation where $\alpha_i = 0$, since the share $\tilde{a}_i$ would leak the secret directly.

The computation in the example is just an upper bound, but an explicit number can be computed if the distribution of $\mathbf{a}$ is known. If $\mathbf{a}$ is normally distributed, then the upper bound is reached with equality by Proposition 2.2.21. Perfect security is not possible if $\sigma_a^2 \neq 0$ since the real numbers do not have a uniform distribution.

It is worth noticing that the amount of leaked information depends on the value of α_i^2, which means that some shares leak more information than others. In the above example, we use the basis $\{1, x, x^2, \dots\}$, which introduces the problem that the random values are scaled by different constants, which makes them contribute different amounts to the entropy. For example, if $\alpha_i < 1$, then $\alpha_i^{N-1} \ll 1$, which means that the coefficient of x^{N-1} does not contribute to the entropy as much. In [40] the basis is chosen to be Lagrange basis polynomials, which evens out the contribution from the different random parts. The structure of the existing SDMM schemes relies heavily on the basis that is used, so for SDMM it is not practical to change the basis polynomials to Lagrange basis polynomials.

5.2 Analog secure MatDot code

The decoding matrix for the MatDot scheme and the GASP_{big} scheme is a Vandermonde matrix. In [41] it was shown that the condition number of a $K \times K$ Vandermonde matrix with real evaluation points with respect to the 2-norm is exponential in K. This means that it is numerically unstable to invert the Vandermonde matrix with real evaluation points, when K is large. However, choosing the evaluation points as the Kth roots of unity in the complex numbers, it is possible to get the condition number to the minimum value of 1. This follows from the fact that such a Vandermonde matrix is unitary. Therefore, in the case that there are no straggling workers, we can choose the evaluation points as Kth roots of unity and get the minimal condition number for the Vandermonde matrix. In the case that there are straggling workers, the evaluation points can be chosen as the Nth roots of unity. Hence, the decoding matrix will be a $K \times K$ Vandermonde matrix on some K Nth roots of unity. In [42] it was shown that the condition number of such a matrix grows polynomially in N if the number of stragglers is constant. This means that it is feasible to invert such a matrix in a numerically stable way.

The secure MatDot scheme is straightforward to extend to work with the complex numbers. This construction was first presented by the author in [4].

Construction 5.2.1 (Analog secure MatDot)**.** The input matrices are $A \in \mathbb{C}^{t \times s}$ and $B \in \mathbb{C}^{s \times r}$. We assume that these matrices come from some continuous probability distribution over the complex numbers. The matrices are then split to p pieces, assuming that $p \mid s$. Hence, the partitioned matrices can be written as

$$A = \begin{pmatrix} A_1 & \dots & A_p \end{pmatrix}, \quad B = \begin{pmatrix} B_1 \\ \vdots \\ B_p \end{pmatrix}.$$

The partitions are of the size $A_j \in \mathbb{C}^{t \times \frac{s}{p}}$ and $B_{j'} \in \mathbb{C}^{\frac{s}{p} \times r}$. The product AB is then the dot product of the partitions, *i.e.*,

$$AB = \sum_{j=1}^{p} A_j B_j.$$

Matrices $R_1, \ldots, R_X \in \mathbb{C}^{t \times \frac{s}{p}}$ and $S_1, \ldots, S_X \in \mathbb{C}^{\frac{s}{p} \times r}$ are drawn at random such that each entry is distributed according to a circular symmetric complex normal distribution with zero mean and a set variance σ_r^2 and σ_s^2, respectively. As was done in the linear SDMM, we will encode the matrices with $[N, p+X]$ codes $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively. Choose the points $\alpha_i = \zeta^i$, for $i \in [N]$, to be the evaluation points, where ζ is a primitive Nth root of unity. Let

$$F = \begin{pmatrix} 1 & \ldots & 1 \\ \alpha_1 & \ldots & \alpha_N \\ \vdots & \ddots & \vdots \\ \alpha_1^{p-1} & \ldots & \alpha_N^{p-1} \\ \alpha_1^{p} & \ldots & \alpha_N^{p} \\ \vdots & \ddots & \vdots \\ \alpha_1^{p+X-1} & \ldots & \alpha_N^{p+X-1} \end{pmatrix}, \quad G = \begin{pmatrix} \alpha_1^{p-1} & \ldots & \alpha_N^{p-1} \\ \vdots & \ddots & \vdots \\ \alpha_1 & \ldots & \alpha_N \\ 1 & \ldots & 1 \\ \alpha_1^{p} & \ldots & \alpha_N^{p} \\ \vdots & \ddots & \vdots \\ \alpha_1^{p+X-1} & \ldots & \alpha_N^{p+X-1} \end{pmatrix}$$

be the generator matrices for $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively. The encoded pieces are computed as

$$\widetilde{A} = (\widetilde{A}_1, \ldots, \widetilde{A}_N) = (A_1, \ldots, A_p, R_1, \ldots, R_X)F,$$
$$\widetilde{B} = (\widetilde{B}_1, \ldots, \widetilde{B}_N) = (B_1, \ldots, B_p, S_1, \ldots, S_X)G.$$

This is equivalent to evaluating certain polynomials at the points α_i, as was done in Construction 3.3.1. The encoded pieces can also be written as

$$\widetilde{A} = \underbrace{(A_1, \ldots, A_p)F^{\leq p}}_{=A'} + \underbrace{(R_1, \ldots, R_X)F^{>p}}_{=R'},$$
$$\widetilde{B} = \underbrace{(B_1, \ldots, B_p)G^{\leq p}}_{=B'} + \underbrace{(S_1, \ldots, S_X)G^{>p}}_{=S'}.$$

The security of the scheme can be proven using the properties of $F^{>p}$ and $G^{>p}$. The encoded pieces $\widetilde{A}_i$ and $\widetilde{B}_i$ are sent to worker node $i \in [N]$, which computes $\widetilde{C}_i = \widetilde{A}_i \widetilde{B}_i$. These products can be collected as the star product

$$\widetilde{C} = \widetilde{A} \star \widetilde{B}.$$

As seen earlier in Construction 3.3.1, the product AB can be retrieved as a linear combination of any $K = 2p + 2X - 1$ pieces of $\widetilde{C}$, by computing

$$\widetilde{C}_{\mathcal{I}} H_{\mathcal{I}}^{-1},$$

where H is the transpose of the $N \times K$ Vandermonde matrix on the points $\alpha_1, \ldots, \alpha_N$, and $\mathcal{I} \subset [N]$ is any set of K nodes. Then AB is the pth element in the resulting vector.

This construction is similar to the one presented in 3.5.2, except that the distribution of the random matrices is different and the choice of the evaluation points is fixed. These help fix the problems regarding security and numerical stability, respectively.

5.3 Analog GASP code

In this section we present the GASP code in the analog setting. This construction follows Construction 3.4.4 and was first presented by the author in [4].

Construction 5.3.1 (Analog $\mathrm{GASP}_{\mathrm{big}}$)**.** The input matrices are $A \in \mathbb{C}^{t\times s}$ and $B \in \mathbb{C}^{s\times r}$. We assume that these matrices come from some continuous probability distribution over the complex numbers. The matrices are then split to m and n pieces, respectively, assuming that $m \mid t$ and $n \mid r$. Hence, the partitioned matrices can be written as

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_m \end{pmatrix}, \quad B = \begin{pmatrix} B_1 & \dots & B_n \end{pmatrix}.$$

The partitions are of the size $A_j \in \mathbb{C}^{\frac{t}{m}\times s}$ and $B_{j'} \in \mathbb{C}^{s\times \frac{r}{n}}$. The product AB is then the outer product of the partitions, *i.e.*,

$$AB = \begin{pmatrix} A_1B_1 & \dots & A_1B_n \\ \vdots & & \vdots \\ A_mB_1 & \dots & A_mB_n \end{pmatrix}.$$

Matrices $R_1, \dots, R_X \in \mathbb{C}^{\frac{t}{m}\times s}$ and $S_1, \dots, S_X \in \mathbb{C}^{s\times \frac{r}{n}}$ are drawn at random such that each entry is distributed according to a circular symmetric complex normal distribution with zero mean and a set variance σ_r^2 and σ_s^2, respectively. As was done in the linear SDMM, we will encode the matrices with $[N, m+X]$ and $[N, n+X]$ codes $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively. Choose the points $\alpha_i = \zeta^i$ for $i \in [N]$ to be the evaluation points, where ζ is the primitive Nth root of unity. Let

$$F = \begin{pmatrix} 1 & \dots & 1 \\ \alpha_1 & \dots & \alpha_N \\ \vdots & \ddots & \vdots \\ \alpha_1^{m-1} & \dots & \alpha_N^{m-1} \\ \alpha_1^{mn} & \dots & \alpha_N^{mn} \\ \vdots & \ddots & \vdots \\ \alpha_1^{mn+X-1} & \dots & \alpha_N^{mn+X-1} \end{pmatrix}, \quad G = \begin{pmatrix} 1 & \dots & 1 \\ \alpha_1^{m} & \dots & \alpha_N^{m} \\ \vdots & \ddots & \vdots \\ \alpha_1^{m(n-1)} & \dots & \alpha_N^{m(n-1)} \\ \alpha_1^{mn} & \dots & \alpha_N^{mn} \\ \vdots & \ddots & \vdots \\ \alpha_1^{mn+X-1} & \dots & \alpha_N^{mn+X-1} \end{pmatrix}$$

be the generator matrices for $\mathcal{C}_A$ and $\mathcal{C}_B$, respectively. The encoded pieces are computed as

$$\widetilde{A} = (\widetilde{A}_1, \dots, \widetilde{A}_N) = (A_1, \dots, A_p, R_1, \dots, R_X)F,$$
$$\widetilde{B} = (\widetilde{B}_1, \dots, \widetilde{B}_N) = (B_1, \dots, B_p, S_1, \dots, S_X)G.$$

Again we can write this as

$$\widetilde{A} = \underbrace{(A_1, \dots, A_m)F^{\leq m}}_{=A'} + \underbrace{(R_1, \dots, R_X)F^{>m}}_{=R'},$$
$$\widetilde{B} = \underbrace{(B_1, \dots, B_n)G^{\leq n}}_{=B'} + \underbrace{(S_1, \dots, S_X)G^{>n}}_{=S'}.$$

In the scheme, the encoded pieces $\widetilde{A}_i$ and $\widetilde{B}_i$ are sent to worker node $i \in [N]$, which computes $\widetilde{C}_i = \widetilde{A}_i \widetilde{B}_i$. These products can be collected as the star product

$$\widetilde{C} = \widetilde{A} \star \widetilde{B}.$$

As seen earlier in Construction 3.4.4, the product AB can be retrieved as a linear combination of any $K = 2mn + 2X - 1$ pieces of $\widetilde{C}$, by computing

$$\widetilde{C}_{\mathcal{I}} H_{\mathcal{I}}^{-1},$$

where H is the transpose of the $N \times K$ Vandermonde matrix on the points $\alpha_1, \ldots, \alpha_N$, and $\mathcal{I} \subset [N]$ is any set of K nodes. Then the submatrices $A_j B_{j'}$ are the first mn components of the resulting vector.

The general GASP construction presented in 3.4.3 is not as simple to convert to the analog setting. This is because the choice of the evaluation points is not as simple in that scheme. In [16] it was shown that the evaluation points can always be chosen in a large enough field, when the GASP code is considered over finite fields. A similar result for the analog setting is not clear in generality.

5.4 Analyzing security

Perfect security is not possible over a field with no uniform distribution, as discussed in Section 5.1. Therefore, we need to change the security definition of analog SDMM schemes to only leak some set amount of information about the matrices A and B. The security of the analog SDMM model can be characterized by the following definition.

Definition 5.4.1. An analog SDMM scheme is (X, δ)-secure for $\delta > 0$ if

$$I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}}) \leq \delta$$

for all $\mathcal{X} \subset [N]$, $|\mathcal{X}| \leq X$.

We can prove the security of analog SDMM schemes that follow the same construction as the linear SDMM schemes with the random matrices chosen in the same way as the two constructions above. The condition for security is the same as the condition in Theorem 3.5.5.

Theorem 5.4.2. *Consider an analog SDMM scheme and let $\mathcal{X} \subset [N]$, $|\mathcal{X}| = X$, be a set of colluding nodes. If $F^{>m}$ and $G^{>n}$ have the MDS property, then for all $\delta > 0$*

$$I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}}) \leq \delta$$

for large enough σ_r^2 and σ_s^2.

Proof. To make the problem simpler, let us concentrate on each of the matrices individually. This is possible since

$$\begin{aligned} I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}}) &= I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}) + I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{B}}_{\mathcal{X}} \mid \widetilde{\mathbf{A}}_{\mathcal{X}}) \\ &= h(\widetilde{\mathbf{A}}_{\mathcal{X}}) - h(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}, \mathbf{B}) + h(\widetilde{\mathbf{B}}_{\mathcal{X}} \mid \widetilde{\mathbf{A}}_{\mathcal{X}}) - h(\widetilde{\mathbf{B}}_{\mathcal{X}} \mid \widetilde{\mathbf{A}}_{\mathcal{X}}, \mathbf{A}, \mathbf{B}) \\ &\leq h(\widetilde{\mathbf{A}}_{\mathcal{X}}) - h(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}) + h(\widetilde{\mathbf{B}}_{\mathcal{X}}) - h(\widetilde{\mathbf{B}}_{\mathcal{X}} \mid \mathbf{B}) \\ &= I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}}) + I(\mathbf{B}; \widetilde{\mathbf{B}}_{\mathcal{X}}). \end{aligned}$$

The inequality follows from (1) $\widetilde{\mathbf{A}}_{\mathcal{X}}$ is conditionally independent of $\mathbf{B}$ given $\mathbf{A}$, (2) conditioning lowers the entropy, and (3) $\widetilde{\mathbf{B}}_{\mathcal{X}}$ is conditionally independent of $\widetilde{\mathbf{A}}_{\mathcal{X}}$ and $\mathbf{A}$ given $\mathbf{B}$. Therefore, it is enough to consider bounding $I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}})$ and $I(\mathbf{B}; \widetilde{\mathbf{B}}_{\mathcal{X}})$ from above. Let us analyze only the first bound, since the other one follows similarly.

Using the definition of mutual information, we get that

$$I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}}) = h(\widetilde{\mathbf{A}}_{\mathcal{X}}) - h(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}).$$

Let us analyze the first term. We assume that each entry in $\mathbf{A}$ is independent and identically distributed to make the proof simpler. Such an assumption is not necessary for the proof to work. We can deduce from the union bound on entropy and the independence of the entries of $\widetilde{\mathbf{A}}$ that

$$\begin{aligned} h(\widetilde{\mathbf{A}}_{\mathcal{X}}) &= \frac{ts}{m} h(\tilde{\mathbf{a}}_{\mathcal{X}}) \\ &\leq \frac{ts}{m} \log((\pi e)^X \det(\Sigma_{\tilde{a}})) \\ &= \frac{ts}{m} \log((\pi e)^X \det(\Sigma_{\tilde{a}})), \end{aligned}$$

where $\tilde{\mathbf{a}}$ denotes a single entry in the matrix and the inequality follows from Theorem 2.2.22. We may use the Theorem, since $\tilde{\mathbf{a}}_{\mathcal{X}} = \mathbf{a}'_{\mathcal{X}} + \mathbf{r}'_{\mathcal{X}}$ is a continuous random vector of length $|\mathcal{X}|$ over the complex numbers. The notation is from Constructions 5.2.1 and 5.3.1. Additionally, we need that the covariance matrix of $\tilde{a}$, denoted by $\Sigma_{\tilde{a}}$ is nonsingular.

For the other term we get

$$h(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}) = h(\mathbf{A}'_{\mathcal{X}} + \mathbf{R}'_{\mathcal{X}} \mid \mathbf{A}).$$

Now, $\mathbf{A}'_n$ is a deterministic function of $\mathbf{A}$, so the conditional entropy is determined directly by the entropy of the random matrices. We can again separate the components of the random matrices, since all the components are independent. Hence, we get

$$h(\widetilde{\mathbf{A}}_{\mathcal{X}} \mid \mathbf{A}) = h(\mathbf{R}'_{\mathcal{X}}) = \frac{ts}{m} h(\mathbf{r}'_{\mathcal{X}}) = \frac{ts}{m} \log((\pi e)^X \det(\Sigma_{r'}))$$

by Theorem 2.2.22, since $\mathbf{r}'_{\mathcal{X}}$ is circular symmetric complex normally distributed. Therefore,

$$I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}}) \leq \frac{ts}{m} \log\left(\frac{\det(\Sigma_{\tilde{a}})}{\det(\Sigma_{r'})}\right).$$

We now need to set an upper bound for the quotient of the determinants. Let us analyze the covariance matrices. The elements of $\Sigma_{\tilde{a}}$ are

$$\mathrm{Cov}(\mathbf{a}'_i + \mathbf{r}'_i, \mathbf{a}'_j + \mathbf{r}'_j) = \mathrm{Cov}(\mathbf{a}'_i, \mathbf{a}'_j) + \mathrm{Cov}(\mathbf{r}'_i, \mathbf{r}'_j),$$

which follows from the independence of $\mathbf{a}'_{\mathcal{X}}$ and $\mathbf{r}'_{\mathcal{X}}$. Therefore, we can write $\Sigma_{\tilde{a}} = \Sigma_{a'} + \Sigma_{r'}$. Now, the covariance matrix $\Sigma_{a'}$ can be written as

$$\Sigma_{a'} = \mathrm{Var}(\mathbf{a}F_{\widetilde{\mathcal{X}}}^{\leq m}) = (F_{\widetilde{\mathcal{X}}}^{\leq m})^* \Sigma_a F_{\widetilde{\mathcal{X}}}^{\leq m} = \sigma_a^2 (F_{\widetilde{\mathcal{X}}}^{\leq m})^* F_{\widetilde{\mathcal{X}}}^{\leq m},$$

since $\Sigma_a = \sigma_a^2 I_m$ is the covariance matrix of the partitions of $\mathbf{A}$. The covariance matrix $\Sigma_{r'}$ can be written as

$$\Sigma_{r'} = \mathrm{Var}(\mathbf{r}F_{\mathcal{X}}^{>m}) = (F_{\mathcal{X}}^{>m})^* \Sigma_r F_{\mathcal{X}}^{>m} = \sigma_r^2 (F_{\mathcal{X}}^{>m})^* F_{\mathcal{X}}^{>m},$$

since $\Sigma_r = \sigma_r^2 I_X$ as the entries in the random matrices are i.i.d.. Now, if $F^{>m}$ has the MDS property, then $\Sigma_{r'}$ is invertible, since $F_{\mathcal{X}}^{>m}$ is invertible. Now, write $U = F_{\widetilde{\mathcal{X}}}^{\leq m}$ and $L = F_{\mathcal{X}}^{>m}$. We can use the matrix determinant lemma (Lemma A.4) and Hadamard's inequality (Lemma A.3) to get

$$\begin{aligned}
\det(\Sigma_{\tilde{a}}) &= \det(\Sigma_{r'} + \Sigma_{a'}) \\
&= \det(\Sigma_{r'} + \sigma_a^2 U^* U) \\
&= \det(\Sigma_{r'}) \det(I_m + \sigma_a^2 U \Sigma_{r'}^{-1} U^*) \\
&\leq \det(\Sigma_{r'}) \prod_{i=1}^{m} (1 + \sigma_a^2 (U \Sigma_{r'}^{-1} U^*)_{ii})
\end{aligned}$$

Therefore,

$$\begin{aligned}
I(\mathbf{A}; \widetilde{\mathbf{A}}_{\mathcal{X}}) &\leq \frac{ts}{m} \log \left(\prod_{i=1}^{m} (1 + \sigma_a^2 (U \Sigma_{r'}^{-1} U^*)_{ii}) \right) \\
&= \frac{ts}{m} \sum_{i=1}^{m} \log(1 + \sigma_a^2 (U \Sigma_{r'}^{-1} U^*)_{ii}) \\
&\leq \frac{ts\sigma_a^2}{m \ln 2} \sum_{i=1}^{X} (U \Sigma_{r'}^{-1} U^*)_{ii} \\
&= \frac{ts\sigma_a^2}{m \ln 2} \mathrm{Tr}(U \Sigma_{r'}^{-1} U^*) \\
&= \frac{ts}{m \ln 2} \frac{\sigma_a^2}{\sigma_r^2} \mathrm{Tr}(U (L^* L)^{-1} U^*).
\end{aligned}$$

We get a similar result for $I(\mathbf{B}; \widetilde{\mathbf{B}}_{\mathcal{X}})$. Thus, we can make the total information leakage arbitrarily small by increasing the variances σ_a^2 and σ_b^2. □

For the security of the analog SDMM schemes, we need to assume that the input matrices come from some continuous distribution. This is not an unreasonable assumption, since real world data often comes from a continuous distribution. The problem might come from the fact that, for a real implementation, we need to know

the covariance matrix of the entries of the input matrices A and B to compute the $\Sigma_{a'}$ and $\Sigma_{b'}$ covariance matrices. The proof is made simple if each matrix element is independent and identically distributed, since then the covariance matrices are multiples of the identity matrix. It is an interesting future research question to compute an upper bound for the leaked information in the case that the entries in the matrices are not independent and the full covariance matrix is not known.

For an actual implementation it is important to be able to compute the required variances σ_r^2 and σ_s^2 for a given security parameter δ. For simplicity, we take $\sigma_r^2 = \sigma_s^2 = \sigma^2$. We can then compute the variance with the formulas in the following corollaries to the above theorem.

Corollary 5.4.3. *The analog secure MatDot scheme is (X, δ)-secure if*

$$\sigma^2 \geq \max_{\substack{\mathcal{X}\subset[N]\\|\mathcal{X}|=X}} \frac{s}{\delta p \ln 2} \left(t \operatorname{Tr}(U(L^*L)^{-1}U^*) + r \operatorname{Tr}(V(M^*M)^{-1}V^*). \right),$$

where U is the $p \times X$ Vandermonde matrix on the nodes $\alpha_{\mathcal{X}}$, V is the $p \times X$ inverted Vandermonde matrix on the nodes $\alpha_{\mathcal{X}}$, and L and M are the $X \times X$ generalized Vandermonde matrices starting at the power p on the nodes $\alpha_{\mathcal{X}}$.

The proof of this corollary follows directly from the above theorem by using the definition of the upper and lower generator matrices from Construction 5.2.1.

Corollary 5.4.4. *The analog* GASP_{big} *scheme is (X, δ)-secure if*

$$\sigma^2 \geq \max_{\substack{\mathcal{X}\subset[N]\\|\mathcal{X}|=X}} \frac{s}{\delta \ln 2} \left(\frac{t}{m} \operatorname{Tr}(U(L^*L)^{-1}U^*) + \frac{r}{n} \operatorname{Tr}(V(M^*M)^{-1}V^*). \right),$$

where U is the $p \times X$ Vandermonde matrix on the nodes $\alpha_{\mathcal{X}}$, V is the $p \times X$ Vandermonde matrix on the nodes α_i^m, for $i \in \mathcal{X}$, and L and M are the $X \times X$ generalized Vandermonde matrices starting at the power mn on the nodes $\alpha_{\mathcal{X}}$.

The formula given above still depends on the subset $\mathcal{X} \subset [N]$. Some numerical computations show that the leaked information for the analog secure MatDot scheme is maximized when the colluding set is chosen to be

$$\mathcal{X} = \{e^{\imath 2\pi i/N}, e^{\imath 2\pi (i+1)/N}, \ldots, e^{\imath 2\pi (i+X-1)/N}\},$$

where $\imath$ is the imaginary unit and i is some integer, *i.e.*, when the evaluation points are consecutive. This choice for a maximizing set is natural, since that choice clearly minimizes the determinant of the $(F_{\mathcal{X}}^{>p})^* F_{\mathcal{X}}^{>p}$ matrix, which controls the entropy of the noise $\mathbf{R}'_{\mathcal{X}}$. That determinant is the square of the product of distances between the different evaluation points, as the matrix is a product of a Vandermonde matrix with its conjugate transpose. That product is clearly minimized when the evaluation points are chosen to be as close to each other as possible, which means that the points need to be consecutive.

Even though we know which evaluation points maximize the entropy of the noise, we cannot say anything about what maximizes the upper bound in Corollary 5.4.3.

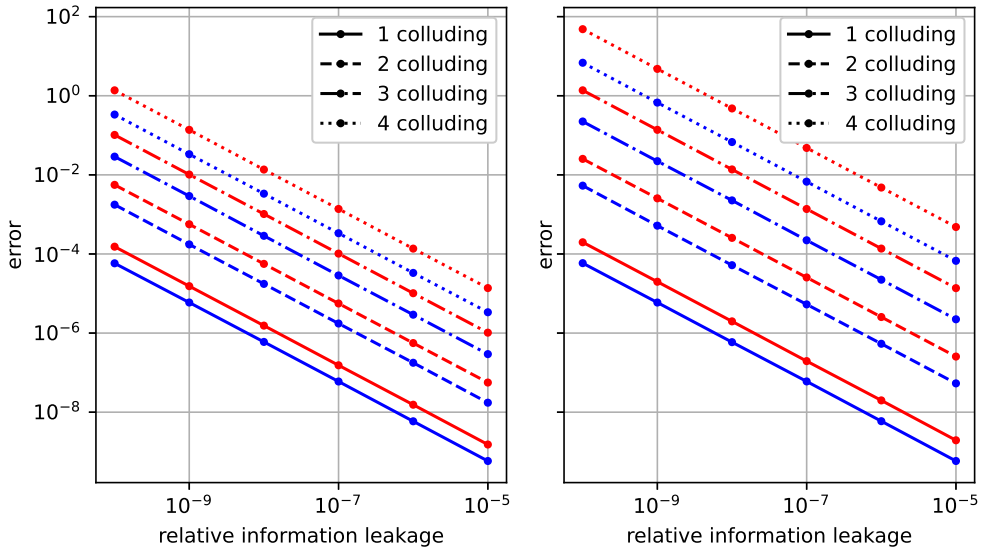

Figure 1: Comparison of the effect of colluding workers on the relationship between the leaked information and the numerical error. The blue lines correspond to analog secure MatDot with partitioning parameter $p = 4$ (left) and $p = 9$ (right). The red lines correspond to the analog GASP_{big} with partitioning parameters $m = n = 2$ (left) and $m = n = 3$ (right).

However, it is easy to see that the upper bound is invariant by shifting the evaluation points by a constant amount around the unit circle. This follows from the fact that such a shift can be achieved by multiplying U, V, L, M by certain diagonal matrices, which cancel each other out in the trace. This fact supports the fact that the upper bound is maximized when the evaluation points are consecutive, which would help compute the upper bound without having to go through every possible colluding set.

5.5 Analyzing numerical accuracy

To make the computation secure, we need to make the variance σ^2 sufficiently large. By doing this, we add large numbers to our original input data, which means that the elements in the encoded matrices are on average large compared to the elements in the unencoded matrices. As the numerical error of matrix multiplication is proportional to the sizes of the elements, the size of the error compared to the original data grows as the variance grows [37]. This introduces a trade-off between security and numerical accuracy.

Analyzing the additional error introduced by the analog SDMM process can be done by numerical simulation. It is an interesting future research question to compute bounds on the error without performing the numerical simulation so that the scheme can be used in a real application where performing numerical simulations defeats the point of performing SDMM.

We perform the numerical simulation by running the algorithm 1000 times and recording the mean error as a function of the specified security level. For each round,

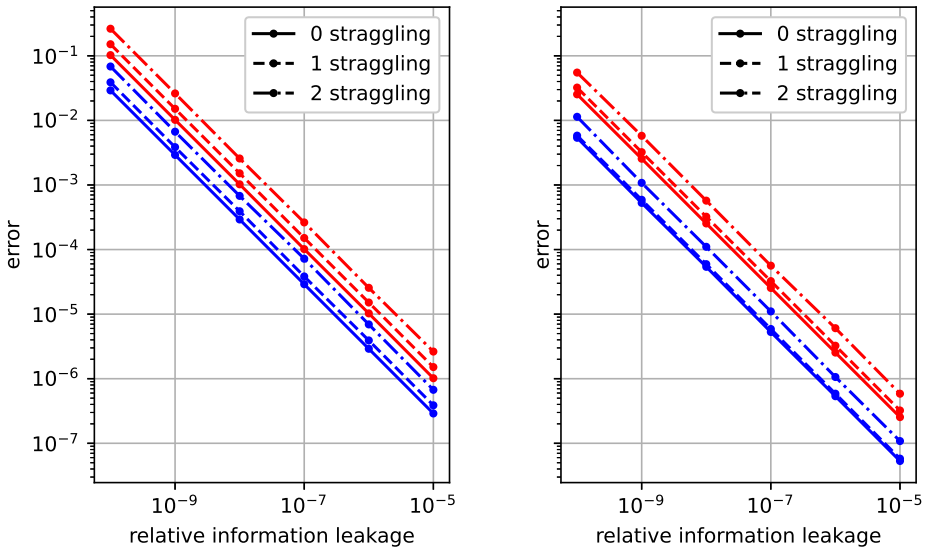

Figure 2: Comparison of the effect of straggling workers on the relationship between the leaked information and the numerical error. The blue lines correspond to the analog secure MatDot with partitioning parameter $p = 4$ (left) and $p = 9$ (right). The red lines correspond to the analog GASP_{big} with partitioning parameters $m = n = 2$ (left) and $m = n = 3$ (right). The number of colluding workers is $X = 3$ (left) and $X = 2$ (right).

new 36×36 input matrices are chosen, such that each element is independently chosen from the standard normal distribution. The variance is chosen to be as low as possible using Corollaries 5.4.3 and 5.4.4. Figures 1 and 2 show the resulting graphs. The error is measured as

$$\|\widehat{AB} - AB\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm, AB is the product of A and B computed without SDMM, and $\widehat{AB}$ is the product of A and B computed with SDMM. Frobenius norm is used since it is a natural choice, when the condition number of the generator matrices is measured using the operator norm induced by the 2-norm.

It is worth noting that we are not comparing the result of the SDMM computation to the exact product, but rather to the product obtained using a conventional matrix multiplication algorithm, which contains some numerical errors. The experiments were performed using IEEE double precision floating point numbers. Leaked information is measured as the upper bound of

$$\frac{I(\mathbf{A}, \mathbf{B}; \widetilde{\mathbf{A}}_{\mathcal{X}}, \widetilde{\mathbf{B}}_{\mathcal{X}})}{h(\mathbf{A}, \mathbf{B})},$$

which is computed using Theorem 5.4.2. [2]

[2]The source code for the generation of the plots can be found at https://github.com/okkomakkonen/sdmm-over-complex

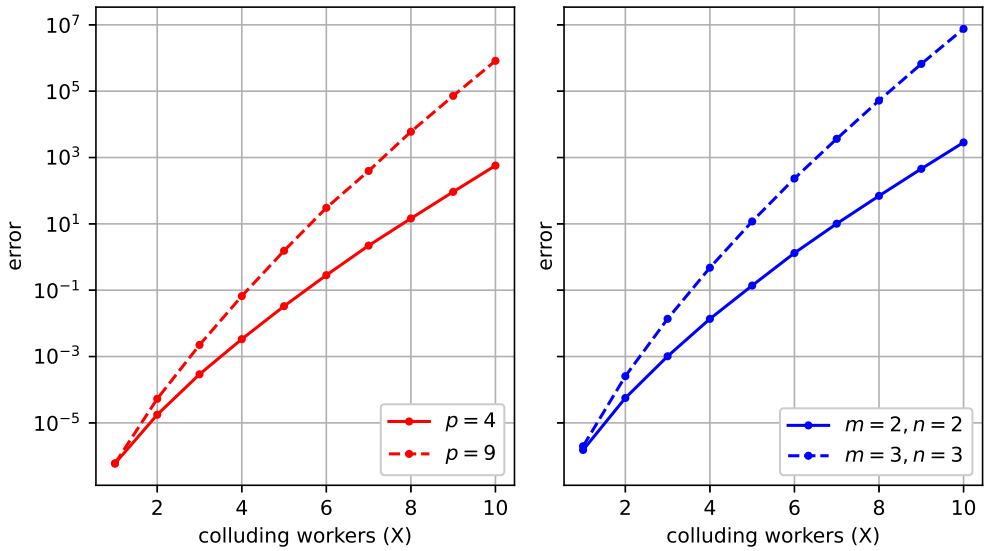

Figure 3: Comparison of the effect of the number of colluding workers on the numerical error. The left figure contains the analog secure MatDot scheme with partitioning parameter p and X colluding workers. The right figure contains the analog GASP_{big} scheme with partitioning parameters m and n and X colluding workers. The relative information leakage is fixed at 10^{-8}.

From each of the figures, it is clear that there is an inverse relationship between security and numerical accuracy. This is explained by the fact that for large enough values of variance, the input data, which is small in comparison, is irrelevant, which means that the standard deviation σ scales the magnitude of the encoded matrices. As the numerical error of matrix multiplication is directly proportional to the product of the magnitudes of the inputs, the error is directly proportional to the variance σ^2. This does not apply to low values of the variance, where the actual input data also has an affect, which can also be numerically confirmed.

From Figures 1 and 3 it is clear that increasing the number of colluding nodes also increases the numerical error. The increase seems to be linear in X, which is expected since for large values of the variance, the number of random matrices determines the size of the elements in the encoded matrices. As the number of colluding workers increases the number of random matrices, the magnitude of the elements of the encoded matrices is proportional to the number of colluding workers.

In the case that there are no stragglers, the evaluation points are chosen as the roots of unity, which means that the corresponding Vandermonde matrix used in the decoding process is unitary. Hence, the condition number is 1. In the case of straggling workers, the Vandermonde matrix will not be unitary, since it does not contain all of the Nth roots of unity. In [42] it was shown that the condition number grows polynomially in the number of nodes, when the number of stragglers is constant. This behaviour can be seen in Figure 2, where the numerical error grows as the number of straggling workers grows.

From Figures 1, 2, and 3 it is clear that the analog MatDot code achieves better numerical accuracy compared to the analog GASP_{big} code when $p = mn$ and the other parameters are the same. However, the analog GASP_{big} code achieves a better download cost by using the outer product partitioning.

6 Applying SDMM in eHealth

The term eHealth refers to the digitalization of healthcare, allowing for more efficient use of resources and providing healthcare remotely. Such applications have an inherent need for confidentiality and security of the data that is being handled. The techniques presented in this thesis have a variety of applications within eHealth.

Private information retrieval (PIR) is a way of downloading a file from a distributed database without revealing the index of the file to the nodes. Many PIR schemes have a similar structure as the linear SDMM schemes in this thesis. As such, PIR can be seen as a subset of SDMM, where one of the matrices describes the index of the file and the other matrix describes the database containing the files. Private information retrieval can be used to store encrypted data on a distributed storage system without the database manager knowing which files are being accessed.

Secure distributed matrix multiplication can be used in place of matrix multiplication if the size of the data is too large to compute the product locally. Another reason for using SDMM is that the data is already distributed, in which case the product can be computed using the distributed system.

Matrix multiplication can be used in various machine learning applications, such as computing and training an artificial neural network. The backpropagation algorithm used in the neural network multiplication involves computing multiple matrix-vector multiplications, where the matrix contains the weights from the artificial neural network. The vectors contain the training data, which means that the matrix-vector multiplications can be computed as a batch, where the vectors are stacked into a matrix. Therefore, matrix-matrix SDMM can be used. Machine learning can be used in many applications in eHealth.

The processing of medical images from MRI and PET machines involves linear algebra operations over a large amount of data. Secure distributed matrix multiplication can be used to speed up the computation without revealing confidential data to the worker nodes.

A common problem in applied mathematics, data science, and statistics is linear regression, where we want to find the hyperplane of best fit when given a number of observations of a linear relationship. We are given the observations $\{x_i, y_i\}_{i\in[s]}$, where $x_i \in \mathbb{R}^t$ and $y_i \in \mathbb{R}$ and we want to find the coefficients $\beta_1, \ldots, \beta_t$ that minimize

$$\sum_{i=1}^{s} (\beta_1 x_i^{(1)} + \cdots + \beta_t x_i^{(t)} - y_i)^2.$$

It is well-known that the solution to the least-squares linear regression problem can be computed with

$$\beta = (X^T X)^{-1} X^T y,$$

where $X \in \mathbb{R}^{s\times t}$ is the matrix formed by stacking the given vectors $\{x_i\}_{i\in[s]}$, and $y \in \mathbb{R}^s$ is the vector of y_i's. The computation of β involves the computation of the Gram matrix $X^T X$, computing the inverse of $X^T X \in \mathbb{R}^{t\times t}$, and finally computing the product $(X^T X)^{-1} X^T y$. If $s \gg t$, then the computation of $X^T X$ is the the most expensive operation in the computation. This computation can be performed using SDMM.

7 Conclusions and future work

In this work, we have presented a novel framework for SDMM schemes and presented some existing SDMM schemes as special cases of the general linear SDMM scheme. Additionally, we used the linear SDMM framework to discuss cooperative SDMM, which is able to provide more flexible communication to the user. Finally, we discussed analog SDMM, which is the most practical mode of SDMM for real-world uses.

An interesting future research problem is that of error correction in SDMM, especially in the context of finite field SDMM, using interleaved codes. Section 3.6 provides the basic framework of how error correction could work, but there are still open questions regarding the probability of successful decoding. The research on interleaved codes is quite mature at this point, but there has not been any research on using interleaved codes in SDMM.

The section 5.4 presented a basic proof for the security of analog SDMM schemes based on the assumption that the input matrices contain i.i.d. elements. It is an interesting future research question to formulate a similar proof for a more general case where the distribution of the input matrices is more complicated or not completely known.

References

[1] V. Strassen, "Gaussian elimination is not optimal," *Numerische mathematik*, vol. 13, no. 4, pp. 354–356, 1969.

[2] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[3] J. Li, O. Makkonen, C. Hollanti, and O. W. Gnilke, "Efficient recovery of a shared secret via cooperation: Applications to SDMM and PIR," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 871–884, 2022.

[4] O. Makkonen and C. Hollanti, "Analog secure distributed matrix multiplication over complex numbers," *arXiv preprint arXiv:2202.03352*, 2022.

[5] S. Ling and C. Xing, *Coding theory: a first course*. Cambridge University Press, 2004.

[6] W. C. Huffman and V. Pless, *Fundamentals of error-correcting codes*. Cambridge University Press, 2003.

[7] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[8] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley-Interscience, 2006.

[9] R. G. Gallager, *Information theory and reliable communication*. Springer, 1968.

[10] F. D. Neeser and J. L. Massey, "Proper complex random processes with applications to information theory," *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1293–1302, 1993.

[11] C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949.

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

[13] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2007.

[14] O. Goldreich, *Foundations of cryptography: a primer*. Now Publishers Inc, 2005.

[15] M. Aliasgari, O. Simeone, and J. Kliewer, "Private and secure distributed matrix multiplication with flexible communication load," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2722–2734, 2020.

[16] R. G. D'Oliveira, S. El Rouayheb, and D. Karpuk, "GASP codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4038–4050, 2020.

[17] R. G. D'Oliveira, S. El Rouayheb, D. Heinlein, and D. Karpuk, "Degree tables for secure distributed matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 907–918, 2021.

[18] R. G. D'Oliveira, S. El Rouayheb, D. Heinlein, and D. Karpuk, "Notes on communication and computation in secure distributed matrix multiplication," in *2020 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–6, IEEE, 2020.

[19] Z. Jia and S. A. Jafar, "On the capacity of secure distributed batch matrix multiplication," *IEEE Transactions on Information Theory*, vol. 67, no. 11, pp. 7420–7437, 2021.

[20] Z. Jia and S. A. Jafar, "Cross subspace alignment codes for coded distributed batch computation," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2821–2846, 2021.

[21] J. Kakar, S. Ebadifar, and A. Sezgin, "Rate-efficiency and straggler-robustness through partition in distributed two-sided secure matrix computation," *arXiv preprint arXiv:1810.13006*, 2018.

[22] J. Kakar, S. Ebadifar, and A. Sezgin, "On the capacity and straggler-robustness of distributed secure matrix multiplication," *IEEE Access*, vol. 7, pp. 45783–45799, 2019.

[23] S. Ebadifar, J. Kakar, and A. Sezgin, "The need for alignment in rate-efficient distributed two-sided secure matrix computation," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[24] H. Yang and J. Lee, "Secure distributed computing with straggling servers using polynomial codes," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 1, pp. 141–150, 2018.

[25] N. Mital, C. Ling, and D. Gunduz, "Secure distributed matrix computation with discrete Fourier transform," *arXiv preprint arXiv:2007.03972*, 2020.

[26] Q. Yu and A. S. Avestimehr, "Entangled polynomial codes for secure, private, and batch distributed matrix multiplication: Breaking the "cubic" barrier," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 245–250, IEEE, 2020.

[27] J. Li and C. Hollanti, "Private and secure distributed matrix multiplication schemes for replicated or MDS-coded servers," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 659–669, 2022.

[28] J. Zhu and X. Tang, "Secure batch matrix multiplication from grouping Lagrange encoding," *IEEE Communications Letters*, vol. 25, no. 4, pp. 1119–1123, 2021.

[29] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and S. A. Avestimehr, "Lagrange coded computing: Optimal design for resiliency, security, and privacy," in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1215–1225, PMLR, 2019.

[30] M. Kim and J. Lee, "Private secure coded computation," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 1097–1101, IEEE, 2019.

[31] W.-T. Chang and R. Tandon, "On the upload versus download cost for secure and private matrix multiplication," in *2019 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2019.

[32] R. Bitar, P. Parag, and S. El Rouayheb, "Minimizing latency for secure coded computing using secret sharing via staircase codes," *IEEE Transactions on Communications*, vol. 68, no. 8, pp. 4609–4619, 2020.

[33] R. Bitar, Y. Xing, Y. Keshtkarjahromi, V. Dasari, S. El Rouayheb, and H. Seferoglu, "Private and rateless adaptive coded matrix-vector multiplication," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, pp. 1–25, 2021.

[34] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[35] G. Schmidt, V. R. Sidorenko, and M. Bossert, "Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 2991–3012, 2009.

[36] L. Holzbaur, H. Liu, A. Neri, S. Puchinger, J. Rosenkilde, V. Sidorenko, and A. Wachter-Zeh, "Success probability of decoding interleaved alternant codes," in *2020 IEEE Information Theory Workshop (ITW)*, pp. 1–5, IEEE, 2021.

[37] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[38] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Privacy-preserving distributed learning in the analog domain," *arXiv preprint arXiv:2007.08803*, 2020.

[39] M. Soleymani, H. Mahdavifar, and A. S. Avestimehr, "Analog Lagrange coded computing," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 1, pp. 283–295, 2021.

[40] K. Tjell and R. Wisniewski, "Privacy in distributed computations based on real number secret sharing," *arXiv preprint arXiv:2107.00911*, 2021.

[41] W. Gautschi, "How (un) stable are Vandermonde systems?," in *Asymptotic and computational analysis*, pp. 193–210, CRC Press, 1990.

[42] A. Ramamoorthy and L. Tang, "Numerically stable coded matrix computations via circulant and rotation matrix embeddings," *IEEE Transactions on Information Theory*, 2021.

A Determinant theory

A Hermitian square matrix $A \in \mathbb{C}^{n\times n}$ is said to be *positive definite* if $x^*Ax > 0$ for all $x \in \mathbb{C}^n \setminus \{0\}$, and *positive semi-definite* if $x^*Ax \geq 0$ for all $x \in \mathbb{C}^n$. It is then clear that the sum of two positive (semi-)definite matrices is positive (semi-)definite. The eigenvalues of a positive (semi-)definite matrix are clearly nonnegative.

The determinant of a matrix can be expressed as the product of its eigenvalues, and the trace can be expressed as the sum of eigenvalues. When working with information theory it will often be convenient to compute determinants of block matrices. The following lemmas is useful for proving many results that are needed in Section 2.2.2.

Lemma A.1. *Let $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times m}$, $C \in \mathbb{C}^{m\times n}$, $D \in \mathbb{C}^{m\times m}$ be matrices. Then*

$$\det \begin{pmatrix} A & B \\ 0 & D \end{pmatrix} = \det(A)\det(D), \quad \det \begin{pmatrix} A & 0 \\ C & D \end{pmatrix} = \det(A)\det(D),$$

where 0 is the zero matrix of appropriate size.

Proof. Let us first consider the matrix

$$M = \begin{pmatrix} A & 0 \\ 0 & 1 \end{pmatrix}$$

of dimension $(n+1)\times(n+1)$. Using the definition of the determinant we get that

$$\begin{aligned} \det(M) &= \sum_{\sigma\in S_{n+1}} \operatorname{sgn}(\sigma) \prod_{i=1}^{n+1} M_{i,\sigma(i)} \\ &= \sum_{\tau\in S_n} \operatorname{sgn}(\tau) \prod_{i=1}^{n} A_{i,\tau(i)} \\ &= \det(A), \end{aligned}$$

where S_n is the symmetric group of order n. The equality follows, since $M_{n+1,\sigma(n+1)} \neq 0$ if and only if $\sigma(n+1) = n+1$. Thus, we can think of σ as an element of S_n, since it keeps $n+1$ constant. Therefore, the term $M_{n+1,\sigma(n+1)} = 1$ for such σ and $M_{i,\sigma(i)} = A_{i,\sigma(i)}$ for $i \in [n]$. Using recursion on the previous result we get that

$$\begin{aligned} \det \begin{pmatrix} A & 0 \\ 0 & I_m \end{pmatrix} &= \det \begin{pmatrix} \begin{pmatrix} A & 0 \\ 0 & I_{m-1} \end{pmatrix} & 0 \\ 0 & 1 \end{pmatrix} \\ &= \det \begin{pmatrix} A & 0 \\ 0 & I_{m-1} \end{pmatrix} \\ &\vdots \\ &= \det(A). \end{aligned}$$

Consider then the products of the following $(n+m)\times(n+m)$ block matrices

$$\begin{pmatrix} I_n & 0 \\ 0 & D \end{pmatrix}\begin{pmatrix} I_n & B \\ 0 & I_m \end{pmatrix}\begin{pmatrix} A & 0 \\ 0 & I_m \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & D \end{pmatrix}.$$

The determinant can be computed using the multiplicative property of the determinant and the previous result. Additionally, the determinant of a upper triangular matrix is the product of the diagonal elements. Hence, we get that

$$\det\begin{pmatrix} A & B \\ 0 & D \end{pmatrix} = \det(D)\cdot 1\cdot\det(A).$$

Similarly, we get

$$\begin{pmatrix} A & 0 \\ 0 & I_m \end{pmatrix}\begin{pmatrix} I_n & 0 \\ C & I_m \end{pmatrix}\begin{pmatrix} I_n & 0 \\ 0 & D \end{pmatrix} = \begin{pmatrix} A & 0 \\ C & D \end{pmatrix}.$$

Therefore,

$$\det\begin{pmatrix} A & 0 \\ C & D \end{pmatrix} = \det(A)\cdot 1\cdot\det(D).$$

□

Lemma A.2. *Let $A, B \in \mathbb{C}^{n\times n}$ be square matrices. Then*

$$\det\begin{pmatrix} A & B \\ B & A \end{pmatrix} = \det(A+B)\det(A-B).$$

Proof. A simple eigenvalue decomposition shows that

$$\begin{pmatrix} A & B \\ B & A \end{pmatrix} = \frac{1}{2}\begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix}\begin{pmatrix} A+B & 0 \\ 0 & A-B \end{pmatrix}\begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix}.$$

Hence, the determinant can be computed as the product of the determinants, where

$$\det\begin{pmatrix} I_n & I_n \\ I_n & -I_n \end{pmatrix} = (-2)^n,$$

and the determinant of the middle matrix follows from Lemma A.1. Therefore,

$$\begin{aligned}\det\begin{pmatrix} A & B \\ B & A \end{pmatrix} &= \frac{1}{2^{2n}}(-2)^n\det(A+B)\det(A-B)(-2)^n \\ &= \det(A+B)\det(A-B).\end{aligned}$$

□

Hadamard's inequality is an important inequality in information theory. It can be proven using Proposition 2.2.21 as was done in [8]. Here we present an alternative proof, which does not rely on information theory.

Lemma A.3 (Hadamard's inequality)**.** *Let $A \in \mathbb{C}^{n\times n}$ be a positive semi-definite matrix. Then*

$$\det(A) \leq \prod_{i=1}^{n} A_{ii}.$$

Proof. The diagonal elements of a positive semi-definite matrix are nonnegative, since $0 \leq e_i^* A e_i = A_{ii}$, where e_i is the ith standard basis vector. If A is has zero diagonal values, then the matrix is singular, which means that the inequality is clear. Thus, we may assume that the diagonal values are nonzero. The determinant can be expressed as the product of eigenvalues. Therefore, we can use the inequality of arithmetic and geometric means to get

$$\begin{aligned}\det(A) &= \prod_{i=1}^{n} \lambda_i \\ &= \prod_{i=1}^{n} A_{ii} \prod_{i=1}^{n} \frac{\lambda_i}{A_{ii}} \\ &\leq \prod_{i=1}^{n} A_{ii} \left(\frac{1}{n} \sum_{i=1}^{n} \frac{\lambda_i}{A_{ii}}\right)^n.\end{aligned}$$

The terms in the sum are the eigenvalues of the matrix A, where every column has been divided by the value on the diagonal. Therefore, the sum equals the trace of that matrix. As the diagonal elements are all ones, the trace is n. Thus,

$$\det(A) \leq \prod_{i=1}^{n} A_{ii}.$$

□

Lemma A.4 (Matrix determinant lemma)**.** *Let $A \in \mathbb{C}^{n\times n}$ be an invertible matrix and $U \in \mathbb{C}^{n\times m}$, $V \in \mathbb{C}^{m\times n}$. Then*

$$\det(A + UV) = \det(A)\det(I_m + VA^{-1}U).$$

Proof. The simple case of $A = I_n$ can be computed as follows. Consider the product of the following $(n+m)\times(n+m)$ block matrices

$$\begin{pmatrix} I_n & 0 \\ V & I_m \end{pmatrix} \begin{pmatrix} I_n + UV & U \\ 0 & I_m \end{pmatrix} \begin{pmatrix} I_n & 0 \\ -V & I_m \end{pmatrix} = \begin{pmatrix} I_n & U \\ 0 & I_m + VU \end{pmatrix},$$

where 0 is the zero matrix of appropriate size. Using the multiplicative property of the determinant and Lemma A.1 we get that

$$1 \cdot \det(I_n + UV) \cdot 1 = \det(I_m + VU).$$

Then

$$\det(A + UV) = \det(A)\det(I_n + (A^{-1}U)V) = \det(A)\det(I_m + VA^{-1}U)$$

by using the special case for the matrices $A^{-1}U$ and V. □

The following lemma can be seen as the inequality of arithmetic and geometric means for matrix determinants.

Lemma A.5. *Let $A, B \in \mathbb{C}^{n\times n}$ be positive definite matrices. Then*

$$\det(AB) \leq \det\left(\frac{A+B}{2}\right)^2,$$

with equality if and only if $A = B$.

Proof. Let us first prove the simpler case when $A = I_n$. Let λ_i, for $i \in [n]$, be the eigenvalues of B. Therefore, $1 + \lambda_i$ are the eigenvalues of $I_n + B$. Hence,

$$\det(B) = \prod_{i=1}^{n} \lambda_i \leq \left(\prod_{i=1}^{n} \frac{1+\lambda_i}{2}\right)^2 = \det\left(\frac{I_n+B}{2}\right)^2,$$

where the inequality follows from the elementary inequality $(\lambda_i + 1)^2 \geq 4\lambda_i$. Equality is reached if and only if $\lambda_i = 1$ for all $i \in [n]$, *i.e.*, $B = I_n$.

As B is positive definite, we can write $B = UU^*$ for some $U \in \mathbb{C}^{n\times n}$. Then, the general case then follows simply by

$$\begin{aligned}
\det(AB) &= \det(A)^2 \det(U^*) \det(A^{-1}) \det(U) \\
&= \det(A)^2 \det(U^* A^{-1} U) \\
&\leq \det(A)^2 \det\left(\frac{I_n + U^* A^{-1} U}{2}\right)^2 \\
&= \det(A)^2 \det\left(\frac{I_n + A^{-1} U U^*}{2}\right)^2 \\
&= \det\left(\frac{A+B}{2}\right)^2.
\end{aligned}$$

Using the previous result is justified, since $U^*A^{-1}U$ is positive definite. The fourth line follows from Lemma A.4. Equality is reached if and only if $U^*A^{-1}U = I_n$, *i.e.*, $A = UU^* = B$. □