

Secure Linear Algebra Using Linearly Recurrent Sequences

Eike Kiltz*

Enav Weinreb[†]

Abstract

In this work we present secure two-party protocols for various core problems in linear algebra. Our main building block is a protocol to obliviously decide singularity of an encrypted matrix: Bob holds an $n \times n$ matrix M , encrypted with Alice's secret key, and wants to learn whether the matrix is singular or not (and nothing beyond that). We give an interactive protocol between Alice and Bob that solves the above problem with optimal communication complexity while at the same time achieving low round complexity. More precisely, the number of communication rounds in our protocol is $\text{polylog}(n)$ and the overall communication is roughly $O(n^2)$ (note that the input size is n^2). At the core of our protocol we exploit some nice mathematical properties of linearly recurrent sequences and their relation to the characteristic polynomial of the matrix M , following [Wiedemann, 1986]. With our new techniques we are able to improve the round complexity of the communication efficient solution of [Nissim and Weinreb, 2006] from $n^{0.275}$ to $\text{polylog}(n)$.

Based on our singularity protocol we further extend our result to the problems of securely computing the rank of an encrypted matrix and solving systems of linear equations.

Key words. Secure Linear Algebra, Linearly Recurrent Sequences, Wiedemann's Algorithm.

1 Introduction

Linear algebra plays a central role in computer science in general and in cryptography in particular. Numerous cryptographic applications such as private information retrieval, secret sharing schemes, multi-party secure computation, and many more make use of linear algebra. In particular, the ability to efficiently solve a set of linear equations constitutes an important algorithmic and cryptographic tool. In this work we design efficient and secure protocols for various linear algebraic problems. Our protocols enjoy both low communication and round complexity.

We concentrate on the following problem. Alice holds the private key of a public-key homomorphic encryption system, and Bob holds a square matrix M , encrypted by Alice's public key. Alice and Bob wish to decide whether M is singular while leaking no other information on M . Many linear algebraic tasks are efficiently reducible to this problem. Our protocol is based on an algorithm by Wiedemann for "black-box linear algebra" [24] which is highly efficient when applied to sparse matrices. This algorithm uses *linearly recurrent sequences* and their relation to the *greatest common divisor* problem for polynomials (see Section 3). Somehow surprisingly, we design a secure protocol based on this algorithm which is applicable

*CWI Amsterdam, The Netherlands. kiltz@cwi.nl. Supported by the research program Sentinels (<http://www.sentinels.nl>). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

[†]Dept. of Computer Science, Ben-Gurion University, Beer-Sheva 84105, Israel. weinrebe@cs.bgu.ac.il. Partially supported by the Frankel Center for Computer Science.

to *general* matrices. Previous secure protocols for linear algebraic problems were based on basic linear algebra techniques like Gaussian Elimination. Our protocols exploit more advanced properties of linear systems, and thus achieve improved complexity bounds.

Cramer and Damgård initiated the study of secure protocols for solving various linear algebraic problems [7]. Their work was done in the information theoretical multi-party setup, with the main focus on reducing the protocols' round complexity to a constant. The communication complexity of their protocols is $\Omega(n^3)$ while the size of the inputs is merely $O(n^2)$. Another known approach for designing secure protocols for the mentioned linear algebraic problems is to apply the garbled circuit method of Yao [25]. The communication complexity of such protocols is related to the Boolean circuit complexity of the underlying problems. However, as these problems are strongly related to the problem of matrix multiplication [5], the communication complexity of the resulting protocol is essentially the circuit complexity of the latter. The best known upper bound for this problem is $O(n^\omega)$ [6] for $\omega \cong 2.38$, which is still larger than the input size. In a recent paper, Nissim and Weinreb [19] introduced a protocol with communication complexity¹ of roughly $O(n^2)$. However, their protocol, which relies on the Gaussian elimination procedure, has round complexity $\Omega(n^{0.275})$, which is considered relatively high.

We design a protocol for deciding singularity which gets the best of previous results, both in terms of communication and round complexity, up to a poly-logarithmic factor. We achieve communication complexity of roughly $O(n^2)$ and $\text{polylog}(n)$ round complexity, assuming (similar to [19]) the existence of homomorphic public-key encryption schemes. This leads to communication and round efficient protocols for many linear algebraic problems. For example, consider the linear subspace intersection problem, in which each of Alice and Bob holds a subspace of \mathbb{F}^n and they wish to securely decide whether there is a non-zero vector in the intersection of their input subspaces. Even for *insecure* computation, it is shown in [2] that the deterministic communication complexity of the problem is $\Omega(n^2)$. This result agrees with ours up to a poly-logarithmic factor.² Hence, our secure protocol for this problem is optimal up to poly-logarithmic factor, both in terms of communication and round complexity. Our protocols also give rise to communication and round efficient secure protocols for problems reducible to linear algebra, e.g., perfect matching, and functions with low span program complexity [16].

Our Techniques. A simple reduction turns the problem of deciding if an encrypted input matrix M is singular, into deciding whether a system $M\mathbf{x} = \mathbf{v}$ is solvable for a randomly chosen vector \mathbf{v} . The main technical tool we use for our protocols are linearly recurrent sequences. In a linear system $M\mathbf{x} = \mathbf{v}$, where M is an $n \times n$ matrix and \mathbf{v} is a vector, the vectors $\mathbf{v}, M\mathbf{v}, M^2\mathbf{v}, \dots, M^{2n}\mathbf{v}$ are clearly linearly dependent. Roughly speaking, the scalars of this linear dependency are related to the characteristic and minimal polynomials of the matrix M . It turns out that computing a polynomial called the *minimal polynomial* of the sequence $\mathbf{v}, M\mathbf{v}, M^2\mathbf{v}, \dots, M^{2n}\mathbf{v}$ is sufficient for deciding the solvability of the original linear system. This polynomial, in turn, can be computed from $\mathbf{v}, M\mathbf{v}, M^2\mathbf{v}, \dots, M^{2n}\mathbf{v}$ using the extended Euclidean algorithm for GCD of polynomials.

In our protocol Bob holds $\text{Enc}(M)$ and \mathbf{v} , where $\text{Enc}(\cdot)$ is a public-key homomorphic encryption scheme, and Alice holds the private decryption key. In the first step, Bob needs to compute $\text{Enc}(\mathbf{v}), \text{Enc}(M\mathbf{v}), \text{Enc}(M^2\mathbf{v}), \dots, \text{Enc}(M^{2n}\mathbf{v})$. The homomorphic encryption scheme does not allow for multiplication of encrypted values and thus Bob needs the “help” of Alice to perform the computations. However, he cannot disclose the values of M or \mathbf{v} to her. We give a method to securely compute $\text{Enc}(\mathbf{v}), \text{Enc}(M\mathbf{v}),$

¹We omit polylogarithmic factor from the complexity bounds discussed in the introduction.

²Although determining the *randomized* communication complexity of subspace intersection is an open problem, it serves as an evidence that our upper bound may be tight.

$\text{Enc}(M^2\mathbf{v}), \dots, \text{Enc}(M^{2n}\mathbf{v})$ within $2 \log n$ rounds of communication. To securely compute the minimal polynomial of the encrypted sequence using the extended Euclidean algorithm, we apply the well-known general result by Yao [25], allowing for secure computation of a function with communication complexity proportional to the Boolean circuit complexity of the computed function. We show a general method to apply the construction of Yao, from a starting point where Bob holds an encrypted input and Alice holds the decryption key. We stress that trying to apply Yao’s construction on the original linear algebraic problem would result in an $\Omega(n^\omega)$ communication protocol.

The above mentioned protocol enables *deciding* whether a linear system is solvable. In order to actually *find* a random solution to the system $M\mathbf{x} = \mathbf{y}$, given encryptions of M and \mathbf{y} , we apply an algorithm by Kalfoten and Saunders [15]. The technical difficulty in applying this algorithm is that it depends on the rank of the matrix M . Computing the rank of M in the clear would compromise the privacy of the protocol. We overcome this problem by designing a protocol for computing an encryption of the rank of an encrypted matrix. As the rank of a matrix is a basic concept in linear algebra, this protocol may be of independent interest. We use the fact that multiplying a rank r matrix from the right and from the left by non-singular matrices perturbs the matrix in a way that with high probability the top-left $r \times r$ sub-matrix of the perturbed matrix is non-singular [4]. The rank is computed then using oblivious binary search. We then show how to implement a secure version of the Kalfoten-Saunders algorithm using only an encrypted form of the rank of M .

Organization. In Section 2 we discuss the setting and some basic building blocks. In Section 3 we define linearly recurrent sequences and discuss their basic properties. Then, in Section 4, we introduce our main protocol for deciding singularity of an encrypted matrix. In Section 6, we design a protocol for solving an encrypted linear system, based on a protocol for computing the rank of a matrix, which is presented in Section 5. Then, in Section 7, we present an implementation of our basic sub-protocols using the garbled circuit method of Yao, and finally, in Section 8, we demonstrate some applications of our secure protocols.

2 General Framework

Notation. Let \mathbb{F} be a finite field with p elements, and denote $k = \log p$. For an encryption scheme, we denote by λ its security parameter. W.l.o.g., we assume that the result of encrypting a field element is of length $O(\lambda + k) = O(\lambda)$. We denote by $\mathbf{neg}(n)$ a function that is negligible in n , i.e., $\mathbf{neg}(n) = n^{-\omega(1)}$. We view a vector $\mathbf{v} \in \mathbb{F}^n$ as a column vector. To denote a row vector we use \mathbf{v}^\top . Finally, we use the $\tilde{O}(\cdot)$ notation to hide any poly-logarithmic factors, that is, $\tilde{O}(f(n)) = O(f(n)\text{polylog}(n))$.

Homomorphic encryption schemes. Our constructions use semantically-secure public-key encryption schemes that allow for simple computations on encrypted data. In particular, we use encryption schemes where the following operations can be performed without knowledge of the private key: (i) Given two encryptions $\text{Enc}(m_1)$ and $\text{Enc}(m_2)$, we can efficiently compute a random encryption $\text{Enc}(m_1 + m_2)$; and (ii) Given an encryption $\text{Enc}(m)$ and $c \in \mathbb{F}$, we can efficiently compute a random encryption $\text{Enc}(cm)$. Several constructions of homomorphic encryption schemes are known, each with its particular properties (see e.g. [22, 14, 11, 21, 23, 20, 3, 18, 8]). These have been in use in a variety of cryptographic protocols. Over $\mathbb{F} = \text{GF}(2)$, for example, the encryption scheme of Goldwasser and Micali [14], based on quadratic residuosity, is sufficient for our constructions.

For a vector $\mathbf{v} \in \mathbb{F}^n$, we denote by $\text{Enc}(\mathbf{v})$ the coordinate-wise encryption of \mathbf{v} . That is, if $\mathbf{v} = \langle a_1, \dots, a_n \rangle$ where $a_1, \dots, a_n \in \mathbb{F}$, then $\text{Enc}(\mathbf{v}) = \langle \text{Enc}(a_1), \dots, \text{Enc}(a_n) \rangle$. Similarly, for a matrix $M \in$

$\mathbb{F}^{m \times n}$, we denote by $\text{Enc}(M)$ the $m \times n$ matrix such that $\text{Enc}(M)[i, j] = \text{Enc}(M[i, j])$. An immediate consequence of the above properties of homomorphic encryption schemes is the ability to perform the following operations without knowledge of the secret key: (i) Given encryptions of two vectors $\text{Enc}(\mathbf{v}_1)$ and $\text{Enc}(\mathbf{v}_2)$, we can efficiently compute $\text{Enc}(\mathbf{v}_1 + \mathbf{v}_2)$, and similarly with matrices. (ii) Given an encryption of a vector $\text{Enc}(\mathbf{v})$ and a constant $c \in \mathbb{F}$, we can efficiently compute $\text{Enc}(c\mathbf{v})$. (iii) Given an encryption of a matrix $\text{Enc}(M)$ and a matrix M' of the appropriate dimensions, we can efficiently compute $\text{Enc}(MM')$ and $\text{Enc}(M'M)$, as any entry in the result matrix is a linear combination of some encrypted matrix entries.

In some cases we want to use the encryption scheme Enc to encrypt an integer ℓ where $0 \leq \ell \leq n$. We do this by encrypting the binary representation of ℓ , bit by bit. If $\ell = \sum_{i=0}^{\log n} \ell_i 2^i$, where $\ell_i \in \{0, 1\}$ for every $0 \leq i \leq \log n$, then we use the notation $\text{Enc}_{\text{bin}}(\ell) = (\text{Enc}(\ell_{\log n}), \dots, \text{Enc}(\ell_1), \text{Enc}(\ell_0))$.

Adversary model. Our protocols are constructed for the two-party semi-honest adversary model. Roughly speaking, both parties are assumed to act in accordance with their prescribed actions in the protocol. Each party may, however, collect any information he/she encounters during the protocol run, and try to gain some information about the other party's input. We will compose our protocols in a modular manner and will argue about their privacy using well-known sequential composition theorems [13] in the semi-honest adversary model.

Complexity Measures. Any interaction between Alice and Bob in the protocol is called a *round* of communication. The total number of such interactions consists the *round complexity* of the protocol. In each round some data is sent from Bob to Alice or from Alice to Bob. The size of all the data (i.e. the total number of bits) that is communicated between Alice and Bob during the whole execution of the protocol is called the *communication complexity* of the protocol. We make the convention to count the communication complexity of our protocols in terms of multiples of λ , i.e. we count the number of encrypted values $\text{Enc}(\cdot)$ exchanged between Alice and Bob.

2.1 Useful Sub-Protocols

In our protocols Bob holds data encrypted by a public key homomorphic encryption scheme, while Alice holds the private decryption key. We view our protocols as algorithms Bob executes on his encrypted input. As mentioned above, the homomorphic encryption allows Bob to locally perform several simple manipulations on his input. However, some operation require the help of Alice. In Table 1, we list a set of operations Bob can perform using the help of Alice. These operations determine the communication and round complexity of our protocols, and thus we added the complexities of each sub-protocol to the table. Later, in Section 7, we show how to implement these sub-protocols within the mentioned complexity bounds, enabling Bob to use Alice's help to perform the described operations without disclosing any of his data to her.

As an example of a protocol where Bob uses Alice's help, we now present Protocol MATRIX MULT for encrypted matrix multiplication. Bob holds the encryptions $\text{Enc}(A)$ and $\text{Enc}(B)$ of two matrices $A \in \mathbb{F}^{n \times \ell}$ and $B \in \mathbb{F}^{\ell \times m}$. Alice holds the private decryption key. In the end of the protocol Bobs should hold the encryption $\text{Enc}(AB)$ of the product matrix $AB \in \mathbb{F}^{n \times m}$. Bob chooses two random matrices $R_A \in \mathbb{F}^{n \times \ell}$ and $R_B \in \mathbb{F}^{\ell \times m}$ and sends Alice the two matrices $\text{Enc}(A + R_A)$ and $\text{Enc}(B + R_B)$, which can be locally computed using the homomorphic properties of $\text{Enc}(\cdot)$. Alice decrypts these matrices and returns $\text{Enc}((A + R_A) \cdot (B + R_B))$ to Bob. Finally Bob locally computes $\text{Enc}(AB) = \text{Enc}((A + R_A)(B + R_B)) - \text{Enc}(AR_B) - \text{Enc}(R_AB) - \text{Enc}(R_AR_B)$. The protocol runs in two rounds and the communication complexity of this protocol is $n\ell + \ell m + nm$. The security proof for this protocol is straightforward.

Protocol name	INPUT/OUTPUT	Communication complexity	Rounds
MATRIX MULT	Input: $\text{Enc}(A), \text{Enc}(B)$ ($A \in \mathbb{F}^{n \times \ell}, B \in \mathbb{F}^{\ell \times m}$) Output: $\text{Enc}(A \cdot B)$	$\ell n + \ell m + nm$	2
NONZERO	Input: $\text{Enc}(x)$ ($x \in \mathbb{F}$) Output: $\text{Enc}(1)$ if $x \neq 0$, $\text{Enc}(0)$ if $x = 0$	$O(k)$	2
EQUAL	Input: $\text{Enc}_{\text{bin}}(x), \text{Enc}_{\text{bin}}(y)$ ($0 \leq x, y \leq n$) Output: $\text{Enc}(1)$ if $x = y$, $\text{Enc}(0)$ if $x \neq y$	$O(k)$	$O(1)$
UNARY	Input: $\text{Enc}_{\text{bin}}(r)$ ($0 \leq r \leq n$) Output: $\text{Enc}(\delta)$, $\delta \in \mathbb{F}^n$, $\delta_i = 1$ if $r \geq i$, $\delta_i = 0$ otherwise.	$O(kn)$	$O(1)$

Table 1: Basic sub-protocols and their complexities.

3 Linearly Recurrent Sequences

We reduce the problem of deciding if a linear system $Mx = v$ is solvable, to computing the minimal polynomial of a certain *linearly recurrent sequence*. In this section we formally define linearly recurrent sequences and discuss some of their basic properties. We follow the exposition given in [12].

Let \mathbb{F} be field and V be a vector space over \mathbb{F} . An infinite sequence $\mathbf{a} = (a_i)_{i \in \mathbb{N}} \in V^{\mathbb{N}}$ is linearly recurrent (over \mathbb{F}) if there exists $n \in \mathbb{N}$ and $f_0, \dots, f_n \in \mathbb{F}$ with $f_n \neq 0$ such that $\sum_{j=0}^n f_j a_{i+j} = 0$, for all $i \in \mathbb{N}$. The polynomial $f = \sum_{j=0}^n f_j x^j$ of degree n is called a *characteristic polynomial* of \mathbf{a} .

We now define a multiplication of a sequence by a polynomial. For $f = \sum_{j=0}^n f_j x^j \in \mathbb{F}[x]$ and $\mathbf{a} = (a_i)_{i \in \mathbb{N}} \in V^{\mathbb{N}}$, we set

$$f \bullet \mathbf{a} = \left(\sum_{j=0}^n f_j a_{i+j} \right)_{i \in \mathbb{N}} \in V^{\mathbb{N}}.$$

This makes $\mathbb{F}^{\mathbb{N}}$, together with \bullet , into an $\mathbb{F}[x]$ -module.³

The property of being a characteristic polynomial can be expressed in terms of the operation \bullet . A polynomial $f \in \mathbb{F}[x] \setminus \{0\}$ is a characteristic polynomial of $\mathbf{a} \in \mathbb{F}^{\mathbb{N}}$ if and only if $f \bullet \mathbf{a} = \mathbf{0}$ where $\mathbf{0}$ is the all-0 sequence. The set of all characteristic polynomials of a sequence $\mathbf{a} \in \mathbb{F}^{\mathbb{N}}$, together with the zero polynomial form an ideal in $\mathbb{F}[x]$. This ideal is called the *annihilator* of \mathbf{a} and denoted by $\text{Ann}(\mathbf{a})$. Since any ideal in $\mathbb{F}[x]$ is generated by a single polynomial, either $\text{Ann}(\mathbf{a}) = \{0\}$ or there is a unique monic polynomial $m \in \text{Ann}(\mathbf{a})$ of least degree such that $\langle m \rangle = \{rm : r \in \mathbb{F}[x]\} = \text{Ann}(\mathbf{a})$. This polynomial is called the *minimal polynomial* of \mathbf{a} and divides any other characteristic polynomial of \mathbf{a} . We denote the minimal polynomial of \mathbf{a} by $m_{\mathbf{a}}$. The degree of $m_{\mathbf{a}}$ is called the *recursion order* of \mathbf{a} .

Let $M \in \mathbb{F}^{n \times n}$ be a matrix, and $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$ be vectors. We will be interested in the following three sequences: (i) $\mathbf{A} = (M^i)_{i \in \mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}^{n \times n}$. (ii) $\mathbf{a} = (M^i \mathbf{v})_{i \in \mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}^n$. (iii) $\mathbf{a}' = (\mathbf{u}^T M^i \mathbf{v})_{i \in \mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}$. We denote by $f_M = \det(M - xI_n) = \sum_{j=1}^n f_j x^j$, the characteristic polynomial of M .

Claim 3.1 Consider $m_{\mathbf{a}'}, m_{\mathbf{a}}, m_{\mathbf{A}}$, the minimal polynomials of the sequences $\mathbf{a}', \mathbf{a}, \mathbf{A}$ respectively. Then $m_{\mathbf{a}'} | m_{\mathbf{a}} | m_{\mathbf{A}} | f_M$.

³Roughly speaking, a module is something similar to a vector space, with the only difference that the ‘‘scalars’’ may be elements of an arbitrary ring instead of a field. See any linear algebra textbook for a formal definition.

Proof: We first show $m_{\mathbf{A}}|f_M$. By the Cayley-Hamilton Theorem $f_M(M) = 0$. Consequently,

$$f_M \bullet \mathbf{A} = \left(\sum_{j=0}^n f_j M^{i+j} \right)_{i \in \mathbb{N}} = (M^i f_M(M))_{i \in \mathbb{N}} = \mathbf{0},$$

and $f_M(M)$ is a characteristic polynomial of \mathbf{A} . Therefore $m_{\mathbf{A}}$, the minimal polynomial of \mathbf{A} , divides f_M .

Next, to prove $m_{\mathbf{a}}|m_{\mathbf{A}}$, write $m_{\mathbf{A}} = \sum_{i=0}^n a_i x^i$. As $m_{\mathbf{A}} \bullet \mathbf{A} = \mathbf{0}$, we get that $(\sum_{j=0}^n a_j M^{i+j})_{i \in \mathbb{N}} = \mathbf{0}$. Hence,

$$m_{\mathbf{A}} \bullet \mathbf{a} = \left(\sum_{j=0}^n a_j (M^{i+j} \cdot \mathbf{v}) \right)_{i \in \mathbb{N}} = \left(\left(\sum_{j=0}^n a_j M^{i+j} \right) \mathbf{v} \right)_{i \in \mathbb{N}} = (0 \cdot \mathbf{v})_{i \in \mathbb{N}} = \mathbf{0}.$$

Therefore $m_{\mathbf{A}}$ is a characteristic polynomial of \mathbf{a} as well, and this $m_{\mathbf{a}}|m_{\mathbf{A}}$. The proof of $m_{\mathbf{a}'}|m_{\mathbf{a}}$ is similar. \square

Corollary 3.2 *The sequences $\mathbf{a}, \mathbf{a}', \mathbf{A}$ are linearly recurrent of order at most n .*

4 Deciding Matrix Singularity

In this section we consider the following problem: Bob holds an $n \times n$ dimensional matrix $\text{Enc}(M)$ over a finite field \mathbb{F} , encrypted under a public-key homomorphic encryption scheme. Alice holds the private decryption key, and they wish to decide whether M is singular, or equivalently, whether $\det(M) = 0$. As mentioned in Section 2, we view the protocol as an algorithm executed by Bob, in which some operations require the help of Alice. The implementation of the secure sub-protocols for these operations is presented in Section 7. Our protocol is based on an algorithm by Wiedemann for “black-box linear algebra” [24]. As a simple first step, we reduce the problem into checking if a certain random linear system is solvable. A second reduction leads us to the problem of computing the minimal polynomial of a linearly recurrent sequence. We then show how to securely compute this particular sequence, and finally, how to compute its minimal polynomial to solve the original matrix singularity problem.

Our first step is to reduce the problem of deciding whether $\det(M) = 0$ to deciding whether the linear system $A\mathbf{x} = \mathbf{v}$ is solvable for some random vector $\mathbf{v} \in \mathbb{F}^n$. If M is non-singular then, obviously, the linear system must be solvable. On the other hand, if $\det(M) = 0$, then with probability at least $1/|\mathbb{F}| \geq 1/2$, the linear system has no solution. Thus, repeating this experiment $\omega(\log n)$ solves the original problem with overwhelming success probability.

Next, we reduce the problem of deciding whether the linear system $A\mathbf{x} = \mathbf{v}$ is solvable to computing $m_{\mathbf{a}}$, the minimum polynomial of the recurrent sequence of vectors $\mathbf{a} = (M^i \mathbf{v})_{i \in \mathbb{N}}$.

Claim 4.1 ([12]) (i) *If $m_{\mathbf{a}}(0) \neq 0$ then the system $A\mathbf{x} = \mathbf{v}$ is solvable.* (ii) *If $m_{\mathbf{a}}(0) = 0$ then $\det(M) = 0$.*

Proof: (i) Since by Claim 3.2 the order of \mathbf{a} is at most n , we can write $m_{\mathbf{a}} = \sum_{i=0}^n m_i x^i$. As $m_{\mathbf{a}}$ is the minimal polynomial of \mathbf{a} , we get that

$$m_n M^n \mathbf{v} + m_{n-1} M^{n-1} \mathbf{v} + \dots + m_1 M \mathbf{v} + m_0 I \mathbf{v} = \mathbf{0}.$$

Since $m_0 = m_{\mathbf{a}}(0)$ is different to 0, we get

$$-m_0^{-1} (m_n M^n \mathbf{v} + m_{n-1} M^{n-1} \mathbf{v} + \dots + m_1 M \mathbf{v}) = \mathbf{v}$$

and hence

$$M(-m_0^{-1}(m_n M^{n-1} \mathbf{v} + m_{n-1} M^{n-2} \mathbf{v} + \dots + m_1 I \mathbf{v})) = \mathbf{v}.$$

Therefore, the system $M\mathbf{x} = \mathbf{v}$ is solvable.

(ii) Let f_M be the characteristic polynomial of M . By Claim 3.1, it holds that $m_{\mathbf{a}} | f_M$. Since $m_{\mathbf{a}}(0) = 0$ we get that $x | m_{\mathbf{a}}$ and thus $x | f_M$. Therefore, the constant coefficient of f_M is 0. As the constant coefficient of the characteristic polynomial f_M is $-\det(M)$, we get that $\det(M) = 0$. \square

We are interested in finding the minimal polynomial of a linearly recurrent sequence of vectors $\mathbf{a} = (M^i \mathbf{v})_{i \in \mathbb{N}}$. This is done by picking a random vector $\mathbf{u} \in \mathbb{F}^n$ and further reducing the problem to computing the minimal polynomial of the linearly recurrent sequence of field elements $\mathbf{a}' = (\mathbf{u}^\top M^i \mathbf{v})_{i \in \mathbb{N}}$. The correctness of the reduction is proved in Lemma 12.17. in [12]. In particular, it is proved that for fields of size at least $2n$, the minimal polynomials of \mathbf{a} and \mathbf{a}' are equal, with at least a constant probability.

Lemma 4.2 ([12]) *Let $M \in \mathbb{F}^{n \times n}$, $\mathbf{v} \in \mathbb{F}^n$, $m_{\mathbf{a}}$ be the minimal polynomial of the sequence $\mathbf{a} = (M^i \mathbf{v})_{i \in \mathbb{N}}$. Then the probability p that $m_{\mathbf{a}}$ is the minimal polynomial of the sequence $\mathbf{a}' = (\mathbf{u}^\top M^i \mathbf{v})_{i \in \mathbb{N}}$ for a $\mathbf{u} \in \mathbb{F}^n$ chosen uniformly at random satisfies $p \geq 1 - \deg(m_{\mathbf{a}})/|\mathbb{F}|$.*

To compute $m_{\mathbf{a}'}$, the minimal polynomial of the sequence \mathbf{a}' , we first need to compute a prefix of the sequence itself. As we will later see, the $2n$ first entries of the sequence will suffice. As the communication complexity of the sub-protocol for matrix multiplication is linear in the matrix size, we are interested in computing $(\text{Enc}(\mathbf{u}^\top M^i \mathbf{v}))_{0 \leq i \leq 2n-1}$ using the least number of matrix multiplication operations. We next show how to compute the sequence using $2 \log n + 1$ matrix multiplication operations.

First compute $\text{Enc}(M^{2^j})$ for $0 \leq j \leq \log n$. This can be easily done in $\log n$ sequential matrix multiplications. For two matrices X and Y of matching size let $X|Y$ be the matrix obtained by concatenating X with Y . Then compute the following using sequential $\log n$ matrix multiplications: (Note that all the matrices are of dimensions at most $n \times n$.)

$$\begin{aligned} \text{Enc}(M\mathbf{v}) &= \text{Enc}(M)\mathbf{v} \\ \text{Enc}(M^3\mathbf{v}|M^2\mathbf{v}) &= \text{Enc}(M^2) \cdot \text{Enc}(M\mathbf{v}|\mathbf{v}) \\ \text{Enc}(M^7\mathbf{v}|M^6\mathbf{v}|M^5\mathbf{v}|M^4\mathbf{v}) &= \text{Enc}(M^4) \cdot \text{Enc}(M^3\mathbf{v}|M^2\mathbf{v}|M\mathbf{v}|\mathbf{v}) \\ \vdots &= \vdots \\ \text{Enc}(M^{2n-1}\mathbf{v}|M^{2n-2}\mathbf{v}|\dots|M^n\mathbf{v}) &= \text{Enc}(M^n) \cdot \text{Enc}(M^{n-1}\mathbf{v}|M^{n-2}\mathbf{v}|\dots|\mathbf{v}) \end{aligned}$$

Finally, multiply each vector $\text{Enc}(M^i \mathbf{v})$ from the left by \mathbf{u}^\top to get $\text{Enc}(\mathbf{u}^\top M^i \mathbf{v})$ for $0 \leq i \leq 2n - 1$.

Our next step is to compute the minimal polynomial. By Corollary 3.2, the order of the sequence \mathbf{a}' is at most n . To compute the minimal polynomial of the sequence \mathbf{a}' given the encryption of its first $2n$ elements, we use the sub-protocol MINPOLY. The protocol is based on the well-known general construction by Yao [25], which supplies a constant round secure protocol for a function f , with communication complexity that is linear in the size of the boolean circuit implementing f . Since the circuit size of the well-known Berlekamp-Massey algorithm [17] for computing the minimal polynomial is $O(n^2 k \log k)$ we get a sub-protocol with $\tilde{O}(n^2 k)$ communication complexity and constant rounds. A constant round implementation of the sub-protocol MINPOLY with $\tilde{O}(nk)$ communication complexity appears in Section 7.

To summarize, Protocol SINGULAR decides if a given encrypted square matrix is singular. To get a constant success probability in each iteration of the protocol, we need $|\mathbb{F}|$ to be at least $2n$. In case $|\mathbb{F}| < 2n$, we work over an extension field, which costs a logarithmic factor in the communication complexity⁴.

⁴It is not hard to derive a homomorphic encryption scheme over an extension field of \mathbb{F} given a homomorphic encryption scheme over \mathbb{F} .

Protocol SINGULAR

Input: $\text{Enc}(M)$ where $M \in \mathbb{F}^{n \times n}$

Output: $\text{Enc}(0)$ if $\det(M) = 0$ and $\text{Enc}(1)$ otherwise.

Repeat the following $\omega(\log n)$ times:

1. Pick random vectors $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^n$.
For $i = 0 \dots 2n - 1$ compute the values $a'_i = \text{Enc}(\mathbf{u}^\top M^i \mathbf{v})$ using $2 \log n$ executions of the matrix multiplication protocol.
2. Execute Protocol MINPOLY to compute $\text{Enc}(m_{\mathbf{a}'})$, an encryption of the minimal polynomial of the sequence $\mathbf{a}' = (a'_i)_{0 \leq i < 2n-1}$.
3. Execute Protocol NONZERO on $\text{Enc}(m_{\mathbf{a}'}(0))$.

Compute the logical AND of the $\omega(\log n)$ results and answer accordingly.

The following theorem summarizes the properties of Protocol SINGULAR. Due to lack of space, the proof is moved to Appendix A.3.

Theorem 4.3 *Let $\text{Enc}(M)$ be an encrypted $n \times n$ matrix over a finite field \mathbb{F} , such that $|\mathbb{F}| \geq 2n$. Then Protocol SINGULAR securely checks if M is singular with probability $1 - \mathbf{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$, where $k = \log |\mathbb{F}|$.*

5 Computing the Rank

In this section we show how to compute $\text{Enc}_{\text{bin}}(\text{rank}(M))$ given an encryption $\text{Enc}(M)$ of a matrix $M \in \mathbb{F}^{m \times n}$. That is, if $r = \text{rank}(M)$, we are interested in outputting $\text{Enc}_{\text{bin}}(r)$. We compute the encryption of the binary representation of r bit by bit. We first show Protocol RANK_{\geq} that decides whether $\text{rank}(M) \geq \ell$ given a bit-wise encryption of a positive integer ℓ .

Protocol RANK_{\geq}

Input: $\text{Enc}(M)$ where $M \in \mathbb{F}^{m \times n}$, for $n \leq m$ and $\text{Enc}_{\text{bin}}(\ell)$ where $1 \leq \ell \leq n$.^a

Output: $\text{Enc}(1)$ if $\text{rank}(M) \geq \ell$ and $\text{Enc}(0)$ otherwise.

Perform the following $\omega(\log n)$ times:

1. Locally compute $\text{Enc}(M') = P \cdot \text{Enc}(M) \cdot Q$ where P and Q are random non-singular $m \times m$ and $n \times n$ matrices respectively.
2. Compute $\text{Enc}(\delta)$ (with $\delta_i = 1$ if $\ell \geq i$ and $\delta_i = 0$ otherwise) from $\text{Enc}_{\text{bin}}(\ell)$ using the UNARY protocol. Create $\text{Enc}(\Delta)$, where Δ is the $n \times n$ diagonal matrix $\Delta = \text{diag}(\delta_1, \delta_2, \dots, \delta_n)$. Compute

$$\text{Enc}(M'_\ell) = \text{Enc}(M') \cdot \text{Enc}(\Delta) + I_n - \text{Enc}(\Delta),$$

where I_n is the $n \times n$ identity matrix. Note that $M'_\ell \in \mathbb{F}^{n \times n}$ is now the matrix that coincides with M' in the top-left $\ell \times \ell$ sub-matrix and with the unit matrix in the $n - \ell \times n - \ell$ bottom-right sub-matrix.

3. Execute Protocol SINGULAR on M'_ℓ and output accordingly.

Compute and output the logical OR of the results of the $\omega(\log n)$ iterations.

^ain $m < n$ execute the protocol on $\text{Enc}(M^\top)$.

The protocol relies on the following simple linear algebraic claim.

Claim 5.1 ([4]) *Let $n \leq m$ be positive integers, \mathbb{F} be a finite field, and M be a $m \times n$ matrix over \mathbb{F} . Suppose $r \leq \text{rank}(M)$ and let P and Q be $m \times m$ and $n \times n$ randomly chosen full rank matrices over \mathbb{F} . Let $M' = PMQ$, and denote the top-left $r \times r$ sub-matrix of M' by N' . Then with constant probability $\text{rank}(N') = r$.*

The proof of the next claim is moved to appendix A.1 due to lack of space.

Claim 5.2 *Protocol RANK_{\geq} securely computes whether $\text{rank}(M) \geq \ell$ with probability $1 - \text{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$.*

Next, in Protocol RANK , we perform $\log n$ executions of Protocol RANK_{\geq} , to compute the encryption of the rank of M using a binary search. The protocol starts with $\ell = 2^{\lfloor \log n \rfloor}$ and checks if the rank of M is greater or equal to ℓ . In the first case the most significant bit of M 's rank, r_{ℓ} , is set to 1 and the check is repeated with the new value $\ell = 2^{\lfloor \log n \rfloor} + 2^{\lfloor \log n \rfloor - 1}$ to determine the second most significant bit of the rank. In the latter case the the most significant bit of M 's rank, r_{ℓ} , is set to 0 and the check is repeated with the new value $\ell = 2^{\lfloor \log n \rfloor - 1}$. This is repeated until all bits of the rank are determined. Note that the search is performed obliviously, as the threshold rank given to Protocol RANK_{\geq} is encrypted. The full description of protocol RANK and is given in Appendix A.1, together with the proof of Theorem 5.3.

Theorem 5.3 *Let $\text{Enc}(M)$ be an encrypted $m \times n$ matrix over a finite field \mathbb{F} . Protocol RANK securely computes $\text{Enc}_{\text{bin}}(\text{rank}(M))$ with probability $1 - \text{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$, where $k = \log |\mathbb{F}|$.*

6 Solving Linear Equations

In this section we discuss the problem of solving a system of linear equations. Given encryptions $\text{Enc}(M)$ and $\text{Enc}(\mathbf{y})$, representing the linear system $M\mathbf{x} = \mathbf{y}$, where $M \in \mathbb{F}^{m \times n}$ and $\mathbf{y} \in \mathbb{F}^m$, we are interested in outputting an encryption $\text{Enc}(\mathbf{x})$ of a random solution to the system, if the system is solvable. We present a protocol that relies on Protocol SINGULAR .

The easy case to deal with is where M is a non-singular square matrix. In this case it is enough to compute $\text{Enc}(M^{-1})$ and then execute Protocol MATRIX MULT once to compute $\text{Enc}(M^{-1})\text{Enc}(\mathbf{y}) = \text{Enc}(M^{-1}\mathbf{y})$, which is the unique solution to the system (and hence is also a random solution). To compute $\text{Enc}(M^{-1})$ from $\text{Enc}(M)$ we use Protocol MATRIX INVERT . The protocol gets an encrypted $n \times n$ matrix $\text{Enc}(M)$ as input, and outputs an encryption of a matrix and an encryption of a field element. If M is invertible, the protocol outputs $\text{Enc}(M^{-1}), \text{Enc}(1)$. Otherwise, it outputs $\text{Enc}(R^{-1}), \text{Enc}(0)$, where R is a random non-singular $n \times n$ matrix. The protocol uses Protocol SINGULAR as a sub-protocol. Its implementation is simple and is described in Appendix A.2.

To reduce the general case to the non-singular case we will show how to apply an algorithm by Kaltofen and Saunders [15] in the secure setting. The algorithm solves $M\mathbf{x} = \mathbf{y}$ in the following way: (i) Perturb the linear system $M\mathbf{x} = \mathbf{y}$ to get a system $M'\mathbf{x} = \mathbf{y}'$ with the same solution space. The perturbation has the property that, with high probability, if M is of rank r , then $M'_{r \times r}$, the top-left $r \times r$ sub-matrix of M' , is non-singular. (ii) Pick a random vector $\mathbf{u} \in \mathbb{F}^n$ and set \mathbf{y}'_r to be the upper r coordinates of the vector $\mathbf{y}' + M'\mathbf{u}$. (iii) Solve the linear system $M'_{r \times r}\mathbf{x}_r = \mathbf{y}'_r$, and denote the solution by \mathbf{u}_r . (iv) Let $\mathbf{u}^* \in \mathbb{F}^n$ be a vector whose upper part is \mathbf{u}_r and its lower part is zero. It can be shown that $\mathbf{x} = \mathbf{u}^* - \mathbf{u}$ is a uniform random solution to the system $M'\mathbf{x} = \mathbf{y}'$ and thus is a uniform random solution to the original system. The

correctness proof for this algorithm may be found in [15, Theorem 4]. Note that this algorithm is correct assuming that the system $M\mathbf{x} = \mathbf{y}$ is solvable.

Implementing the above algorithm in a secure protocol is not straightforward. On one hand, we need to compute r , the rank of M , in order to invert the top-left sub-matrix of M . On the other hand, computing r violates the privacy of the protocol, as r cannot be extracted from a random solution to the linear system. We overcome this problem using Protocol RANK from Section 5, to compute an encryption of r .

Next, we show how to implement the Kaltofen-Saunders algorithm having only an encryption of $r = \text{rank}(M)$. The key idea is that we can work with the $r \times r$ top-left sub-matrix of the perturbed matrix M' , without knowing the value of r in the clear.

Protocol LINEAR SOLVE

Input: $\text{Enc}(M)$ where $M \in \mathbb{F}^{m \times n}$ and $n \leq m$, and $\text{Enc}(\mathbf{y})$ where $\mathbf{y} \in \mathbb{F}^m$. This protocol assumes the system $M\mathbf{x} = \mathbf{y}$ is solvable.

Output: $\text{Enc}(\mathbf{x})$ where $\mathbf{x} \in \mathbb{F}^n$ is a random solution to the system $M\mathbf{x} = \mathbf{y}$.

1. Execute Protocol RANK on $\text{Enc}(M)$ to compute $\text{Enc}(r)$ where $r = \text{rank}(M)$.
2. Repeat the following $\omega(\log n)$ times:
 - (a) Locally compute $\text{Enc}(M') = P \cdot \text{Enc}(M) \cdot Q$ and $\text{Enc}(\mathbf{y}') = P \cdot \text{Enc}(\mathbf{y})$ where P and Q are random non-singular $m \times m$ and $n \times n$ matrices respectively.
 - (b) Compute the encrypted matrix $\text{Enc}(M'_r)$ as in Step 2 of Protocol RANK \geq .
 - (c) Compute $(\text{Enc}(H), \text{Enc}(b))$ using protocol MATRIX INVERT, where $H = (M'_r)^{-1} \in \mathbb{F}^{n \times n}$ and $b = 1$ if M'_r is non-singular. If M'_r is singular then $b = 0$, indicating that H is a random matrix, and that this iteration of the protocol is faulty.
 - (d) Pick a random vector $\mathbf{u} \in \mathbb{F}^n$ and set $\text{Enc}(\mathbf{y}'_r)$ for $\mathbf{y}'_r \in \mathbb{F}^n$ to be a vector whose upper r coordinates are the upper r coordinates of $\text{Enc}(\mathbf{y}') + \text{Enc}(M')\mathbf{u}$ and lower $n-r$ coordinates are $\text{Enc}(0)$. This can be easily using the protocol UNARY.
 - (e) Execute the matrix multiplication protocol MATRIX MULT to compute $\text{Enc}(\mathbf{u}_r) = \text{Enc}((M'_r)^{-1})\text{Enc}(\mathbf{y}'_r)$.
 - (f) Compute $\text{Enc}(Q^{-1}(\mathbf{u} - \mathbf{u}_r))$.
3. Output $\text{Enc}(Q^{-1}(\mathbf{u} - \mathbf{u}_r))$ for a round in which $b = 1$.

Some remarks are in place. First, note that the protocol is valid only for solvable linear system. To check if a system is solvable, it is sufficient to compare the rank of the matrices M and $M|\mathbf{y}$ where $|$ stands for concatenation. The encryption of the rank of these matrices can be computed using Protocol RANK, while the comparison can be done using protocol EQUAL.

Next we discuss the repetitions in Protocol LINEAR SOLVE. In Step 2a, we multiply the matrix M from the right and from the left by random non-singular matrices to get the matrix M' . By Claim 5.1, the top left sub-matrix of M' is of rank r with constant probability. If this is the case, then the rest of the protocol follows the Kaltofen-Saunders algorithm, and thus its correctness is implied by [15, Theorem 4]. If the top-left sub-matrix of M' is not full rank, we get that $b = 0$, and the output is discarded.⁵ Repeating the process $\omega(\log n)$ iteration ensures success probability of $1 - \text{neg}(n)$.

⁵Outputting a result vector for which $b \neq 0$ is simple. Keep an encrypted value c that is 1 if and only if all the values of b in previous iterations were 0. Set the output to be $\text{Enc}(c) \cdot \text{Enc}(\mathbf{u} - \mathbf{u}_r)$ after every iteration. The output of the protocol will be the output of the first iteration in which $b = 1$.

As a final note, we stress that the requirement that $n \leq m$ is made only for simplicity of presentation. Otherwise, M'_r would have been of dimension $\min(m, n) \times \min(m, n)$ instead of $n \times n$, and the changes needed in the rest of the protocol are minor. The following Theorem concludes the properties of Protocol LINEAR SOLVE.

Theorem 6.1 *Let $\text{Enc}(M)$ be an encrypted $m \times n$ matrix over a finite field \mathbb{F} , and let $\text{Enc}(\mathbf{y})$ be an encrypted vector $\mathbf{y} \in \mathbb{F}^m$. Protocol LINEAR SOLVE securely computes $\text{Enc}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{F}^n$ is a random solution of $M\mathbf{x} = \mathbf{y}$, with probability $1 - \mathbf{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$, where $k = \log |\mathbb{F}|$.*

7 Implementation of the Sub-Protocols

In this section we will show how to implement the various sub-protocols presented in Section 2.1.

7.1 Applying Yao's Garbled Circuit Method

In this section we show how to apply the well-known *garbled circuit* method of Yao [25] to implement some of the sub-protocols described in Table 1. In our protocols Bob typically holds an input $\text{Enc}(\mathbf{x})$, where \mathbf{x} is a vector of field elements encrypted by a public-key homomorphic encryption scheme. Alice holds the private decryption key. We show a general way to design a constant round secure protocols for a function f , with communication complexity proportional to the Boolean circuit complexity of f . We design the protocols such that Alice learns no information, while Bob learns $\text{Enc}(f(\mathbf{x}))$.

In Yao's two-party protocol [25] Alice and Bob hold private binary inputs x and y , respectively, and wish to jointly compute a functionality $f(x, y)$, such that Alice learns $f(x, y)$ and Bob learns nothing. Let f be a functionality with m' inputs and ℓ' outputs, which can be computed by a Boolean circuit of size G . Then the construction of Yao results in a protocol that runs in a constant number of rounds and communication complexity $O(G + m' + \ell')$.⁶

Yao's method can be used in our setting as follows: Suppose Bob holds an encryption $\text{Enc}(\mathbf{x})$, where $\mathbf{x} = (x_1, \dots, x_\ell)$ is a vector of field elements. Let $g : \mathbb{F}^\ell \rightarrow \mathbb{F}^m$ be a functionality. Bob wants to securely compute the value $\text{Enc}(g(\mathbf{x}))$. The idea is to first mask the input \mathbf{x} with a random vector \mathbf{r} . Then execute Yao's protocol on a modified functionality f that first un.masks the input, then runs the functionality g , and finally re-masks the output with another random vector \mathbf{s} ; Details follow.

First Bob chooses random vectors $\mathbf{r} = (r_1, \dots, r_\ell) \in \mathbb{F}^\ell$ and $\mathbf{s} = (s_1, \dots, s_m) \in \mathbb{F}^m$ and sends $\text{Enc}(\mathbf{x} + \mathbf{r})$ to Alice. Alice decrypts to get the vector $\mathbf{x} + \mathbf{r} \in \mathbb{F}^\ell$ and converts each field element to a binary string. Now Alice and Bob execute Yao's protocol on the following functionality f : The input of Alice is $\mathbf{x} + \mathbf{r}$ and the input of Bob is \mathbf{r}, \mathbf{s} . f first un.masks the vector x by computing the bits of $(\mathbf{x} + \mathbf{r}) - \mathbf{r}$. Then it runs the functionality g on the input \mathbf{x} and masks the output vector $g(\mathbf{x})$ by adding the vector \mathbf{s} . This vector, $g(\mathbf{x}) + \mathbf{s}$, is given back to Alice who encrypts it and sends $\text{Enc}(g(\mathbf{x}) + \mathbf{s})$ to Bob. Now Bob who knows the random mask \mathbf{s} in the cleartext can reconstruct $\text{Enc}(g(\mathbf{x}))$.

The privacy of the protocol is implied by the privacy of Yao's protocol. We now compute its communication complexity in terms of $C(f)$, the circuit complexity of the functionality f as described above. The number of input and output bits of f is $\ell' = k\ell$ and $m' = km$, where $k = \log |\mathbb{F}|$. Addition and subtraction

⁶Here we make the (simplifying but reasonable) assumption that the primitives used in [25] (i.e., the 1-out-of-2 oblivious transfer protocol and sending one garbled gate of the circuit which is usually done by sending the output of a pseudorandom bit generator) have a communication complexity $O(\lambda)$ (where $\lambda = |\text{Enc}(\cdot)|$) for each execution.

Protocol name	INPUT	OUTPUT
INTERSECTION DECIDE	Alice : Subspace $V_a \subseteq \mathbb{F}^n$ Bob: Subspace $V_b \subseteq \mathbb{F}^n$	Is $V_a \cap V_b = \{\mathbf{0}\}$?
COMMON LINEAR EQUATIONS	Alice: $M_A \in \mathbb{F}^{n_a \times n}$, $\mathbf{v}_a \in \mathbb{F}^{n_a}$ Bob: $M_B \in \mathbb{F}^{n_b \times n}$, $\mathbf{v}_b \in \mathbb{F}^{n_b}$	random \mathbf{x} with $M_A \mathbf{x} = \mathbf{v}_a$, $M_B \mathbf{x} = \mathbf{v}_b$
DETERMINANT	Alice: $M_A \in \mathbb{F}^{n \times n}$ Bob: $M_B \in \mathbb{F}^{n \times n}$	$\det(M_A + M_B)$

Table 2: Linear algebra protocols with $\tilde{O}(n^2)$ communication complexity and $\text{polylog}(n)$ rounds.

can be done with a circuit of size $O(k)$, thus for masking and unmasking we need $O(k(\ell + m))$ gates. Therefore the overall circuit size $G(f)$ of the functionality f is $O(C(g) + k(\ell + m))$. This leads to the following results.

Lemma 7.1 *Let $g : \mathbb{F}^\ell \rightarrow \mathbb{F}^m$ be a public functionality that can be represented by a binary circuit of size C . Suppose Bob holds an encrypted input vector $\text{Enc}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{F}^\ell$. Then there exists a secure two-party protocol that runs in constant rounds and $O(C + k(m + \ell))$ communication complexity such that Bob learns the encrypted vector $\text{Enc}(\mathbf{y})$, where $\mathbf{y} = g(\mathbf{x}) \in \mathbb{F}^m$, and Alice learns no information from the protocol.*

In view of Lemma 7.1, the implementation of Protocols NONZERO, EQUAL and UNARY reduces to implementing simple boolean circuits. For lack of space, we discuss these protocols in Appendix A.4.

7.2 Minimal Polynomial

Using the well-known Berlekamp/Massey algorithm [17] there exists an algebraic circuit of size $O(n^2)$ that computes the minimal polynomial from a sequence $\mathbf{a} = (a_i)_{i \in \mathbb{N}}$ of maximal recursion order n . Further efficiency improvement can be obtained by noting that computing the minimal polynomial can actually be reduced to computing the greatest common division (GCD) of two polynomial of degree $2n$. For completeness we give further details in Appendix C. Using the fast Extended Euclidean algorithm [12, Chapter 11] the latter one can be carried out using an algebraic circuit of size $O(n \log n) = \tilde{O}(n)$.

By implementing each algebraic operation over \mathbb{F} with a binary circuit of size $O(k \log k \log \log k) = \tilde{O}(k)$ we get a binary circuit of size $\tilde{O}(nk)$ for computing the minimal polynomial.

Lemma 7.2 *Suppose Bob holds encrypted vectors $\text{Enc}(a_0), \dots, \text{Enc}(a_{2n-1})$, where $\mathbf{a} = (a_i)_{i \in \mathbb{N}}$ is a linearly recurrent sequence of order at most n . There exists a secure two-party protocol MINPOLY that runs in constant rounds and $\tilde{O}(nk)$ communication complexity that returns the encrypted minimal polynomial $\text{Enc}(m_a)$ of \mathbf{a} to Bob.*

8 Applications

In previous sections we described protocols whose input was an encrypted. In this section we give communication and round efficient protocols for a set of problems in linear algebra, improving upon previous results in the two-party setting. We summarize our results in Table 2 and refer to Appendix B for the exact problem definitions and the protocols.

References

- [1] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *PODC '89: Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 201–209, New York, NY, USA, 1989. ACM Press.
- [2] A. Beimel and E. Weinreb. Separating the power of monotone span programs over different fields. In *Proc. of the 44th IEEE Symp. on Foundations of Computer Science*, pages 428–437, 2003.
- [3] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *the Second Theory of Cryptography Conference – TCC 2005*, pages 325–341, 2005.
- [4] A. B. Borodin, J. von zur Gathen, and J. E. Hopcroft. Fast parallel matrix and GCD computations. Technical report, Cornell University, Ithaca, NY, USA, 1982.
- [5] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*. Springer-Verlag, Berlin, 1997.
- [6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 1–6. ACM Press, 1987.
- [7] R. Cramer and I. Damgaard. Secure distributed linear algebra in a constant number of rounds. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 119–136. Springer-Verlag, 2001.
- [8] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology – EUROCRYPT '97*, Lecture Notes in Computer Science, pages 103–118. Springer-Verlag, 1997.
- [9] I. Damgaard, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *the Third Theory of Cryptography Conference – TCC 2006*, 2006. To Appear.
- [10] J. L. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inf. Theory*, it-33(3):428–431, 1987.
- [11] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.
- [12] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 1999.
- [13] O. Goldreich. *Foundations of Cryptography, Volume II Basic Applications*. Cambridge University Press, 2004.
- [14] S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377, New York, NY, USA, 1982. ACM Press.

- [15] E. Kaltofen and D. Saunders. On Wiedemann’s method of solving sparse linear systems. In *AAECC-9: Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 29–38, London, UK, 1991. Springer-Verlag.
- [16] M. Karchmer and A. Wigderson. On span programs. In *Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111, 1993.
- [17] J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory*, it-15:122–127, 1969.
- [18] D. Naccache and J. Stern. A new public-key cryptosystem based on higher residues. In *ACM CCS 98*, pages 59–66, 1998.
- [19] K. Nissim and E. Weinreb. Communication efficient secure linear algebra. In *the Third Theory of Cryptography Conference – TCC 2006*, 2006. To Appear.
- [20] P. Pallier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT ’99*, pages 223–238, 1999.
- [21] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology – EUROCRYPT ’91*, pages 522–526, 1991.
- [22] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
- [23] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC1. In *FOCS ’99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 554, Washington, DC, USA, 1999. IEEE Computer Society.
- [24] D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theor.*, 32(1):54–62, 1986.
- [25] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th IEEE Symp. on Foundations of Computer Science*, pages 162–167, 1986.

A Full Protocols

A.1 Matrix Rank

Protocol RANK

Input: $\text{Enc}(M)$ where $M \in \mathbb{F}^{m \times n}$, for $n \leq m$.

Output: $\text{Enc}_{\text{bin}}(r)$ where $r = \text{rank}(M)$.

1. For every $i = \lfloor \log n \rfloor$ down to $i = 0$ do:
Set $\text{Enc}(r_i)$ to be the output of $\text{Protocol RANK}_{\geq}$ on $\text{Enc}(M)$ and $\text{Enc}_{\text{bin}}(\ell)$, where $\ell = (\sum_{k=i+1}^{\lfloor \log n \rfloor} r_k 2^k) + 2^i$. Note that the binary presentation of $\text{Enc}_{\text{bin}}(\ell)$ can be composed from the already computed bits of $\text{Enc}_{\text{bin}}(r)$.
2. Output $\text{Enc}_{\text{bin}}(r) = (\text{Enc}(r_{\lfloor \log n \rfloor}), \dots, \text{Enc}(r_0))$.

Theorem 5.3 Let $\text{Enc}(M)$ be an encrypted $m \times n$ matrix over a finite field \mathbb{F} . Protocol RANK securely computes $\text{Enc}_{\text{bin}}(\text{rank}(M))$ with probability $1 - \mathbf{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$, where $k = \log |\mathbb{F}|$.

Proof: Protocol RANK uses $\log n$ executions of protocol RANK_{\geq} to perform a binary search for the rank of M . The protocol is correct if and only if all the executions of RANK_{\geq} are correct, which happens with probability $1 - \mathbf{neg}(n)$. The bounds on round and communication complexity follow by the respective bounds on the RANK_{\geq} protocol from Claim 5.2. \square

A.2 Matrix Inversion

Bob holds an encrypted matrix $\text{Enc}(M)$. Alice holds the private decryption key. Based on the shared field inversion protocol from Bar-Ilan and Beaver [1] we design a protocol with the following properties. The protocol outputs an encryption of a matrix and an encryption of a field element. If M is invertible then in the end of the execution Bob holds $(\text{Enc}(M^{-1}), \text{Enc}(1))$ while if M is singular Bob gets $(\text{Enc}(R), \text{Enc}(0))$ for a random non-singular matrix R . Alice learns nothing in the protocol.

Protocol MATRIX INVERT

Input: $\text{Enc}(M)$ where $M \in \mathbb{F}^{n \times n}$.

Output: $(\text{Enc}(M^{-1}), \text{Enc}(1))$ if M is invertible and $(\text{Enc}(R^{-1}), \text{Enc}(0))$, where R is a random non-singular $n \times n$ matrix, if M is singular.

1. Alice and Bob execute Protocol SINGULAR on $\text{Enc}(M)$. Denote the output of this step by $\text{Enc}(b)$.
2. Bob picks a random $n \times n$ non-singular encrypted matrix R . Bob uses the help of Alice to compute the matrix $\text{Enc}(\tilde{M}) = \text{Enc}(M) \cdot \text{Enc}(b) + \text{Enc}(R) \cdot \text{Enc}(1 - b)$.
3. Bob picks another $n \times n$ random non-singular matrix Q .
4. Bob computes the encrypted matrix $\text{Enc}(Q\tilde{M})$ by multiplying $\text{Enc}(\tilde{M})$ from the left by the matrix Q , and sends $\text{Enc}(Q\tilde{M})$ to Alice.
5. Alice decrypts $\text{Enc}(Q\tilde{M})$ and compute $(Q\tilde{M})^{-1} = \tilde{M}^{-1}Q^{-1}$. Alice encrypts $\tilde{M}^{-1}Q^{-1}$ and sends Bob $\text{Enc}(\tilde{M}^{-1}Q^{-1})$.
6. Bob computes $\text{Enc}(\tilde{M}^{-1}) = \text{Enc}(\tilde{M}^{-1}Q^{-1})Q$.
7. Bob locally outputs $\text{Enc}(\tilde{M}^{-1}), \text{Enc}(b)$.

It is easy to see that the matrix \tilde{M} is always invertible. In case M is invertible $\tilde{M} = M$, otherwise \tilde{M} is a random non-singular matrix. In both cases, Alice gets a random non-singular matrix $Q\tilde{M}$, and thus learns no information in the protocol. In case M is invertible, Bob learns $\text{Enc}(M^{-1})$. Since Bob only learns encrypted values from the protocol, he gets no information on the value of M .

A.3 Proofs

Theorem 4.3 Let $\text{Enc}(M)$ be an encrypted $n \times n$ matrix over a finite field \mathbb{F} , such that $|\mathbb{F}| \geq 2n$. Then Protocol SINGULAR securely checks if M is singular with probability $1 - \mathbf{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$, where $k = \log |\mathbb{F}|$. **Proof:** We first prove that if $\det(M) \neq 0$ then the output of the protocol is $\text{Enc}(1)$. If in any iteration $m_{\mathbf{a}'}(0) = 0$, this means that

the constant coefficient of $m_{\mathbf{a}'}$ is 0, thus $x|m_{\mathbf{a}'}$. By Claim 3.1, $m_{\mathbf{a}'}|f_M$, where f_M is the characteristic polynomial of the matrix M . Hence, the constant coefficient of f_M is 0, which implies $\det(M) = 0$. Hence if M is non-singular, the output of the entire protocol must be $\text{Enc}(1)$.

On the other hand, if $\det(M) = 0$ then, by part (i) of Claim 4.1, if the following two events happen, the output of an iteration is $\text{Enc}(0)$: (i) The system $Mx = v$ is not solvable. (ii) $m_{\mathbf{a}'} = m_{\mathbf{a}}$. The probability of event (i) is at least $(1 - 1/|\mathbb{F}|) \geq 1 - 1/2n > 1/2$. The probability of event (ii), by Lemma 4.2, is at least $1 - \deg(m_{\mathbf{a}})/|\mathbb{F}| > 1 - n/2n = 1/2$. Therefore, with probability at least $1/4$ the output of the iteration is $\text{Enc}(0)$. Hence the probability that $\det(M) = 0$ and still in all the $\omega(\log n)$ iterations the output is $\text{Enc}(1)$ is $\mathbf{neg}(n)$.

We account the communication and round complexity in each iteration of the protocol as follows: in the first step we have $2 \log n$ sequential executions of the matrix multiplication protocol, where each single execution requires $O(n^2)$ communication and constant round. The complexity of the second step is, as discussed above, $\tilde{O}(nk)$ of communication and constant rounds. According to Table 1, the equality protocol in the third step needs $O(k)$ communication and constant rounds. Computing the logical AND of the $\omega(\log n)$ iteration results can be done in 2 rounds and communication which does not effect the asymptotic complexity of the protocol. Hence, after $\omega(\log n)$ repetitions we get a communication complexity of $\tilde{O}(\log n(n^2 + nk)) = \tilde{O}(n^2k)$ and a round complexity of $\omega(\log^2(n)) = \text{polylog}(n)$, as required.

Security of the protocol follows by security of the sub-protocols used. We stress that even though Bob knows the vectors \mathbf{u} and \mathbf{v} in the clear this does not violate privacy of the protocol since the vectors are random vectors and therefore could easily be simulated in a formal proof. However, an implementation of the protocol within the same complexity and using encrypted vectors is also possible. \square

Claim 5.2 Protocol RANK_{\geq} securely computes whether $\text{rank}(M) \geq \ell$ with probability $1 - \mathbf{neg}(n)$, communication complexity $\tilde{O}(n^2k)$ and round complexity $\text{polylog}(n)$.

Proof: Suppose $\text{rank}(M) \geq \ell$. Then, by Claim 5.1, in each iteration the matrix M'_ℓ is non-singular with constant probability. As Protocol SINGULAR is correct with probability $1 - \mathbf{neg}(n)$, we get that in each iteration, with constant probability, the output of SINGULAR is $\text{Enc}(1)$. Hence the probability that in any of the iterations the output of SINGULAR is $1 - \mathbf{neg}(n)$.

On the other hand, if $\text{rank}(M) < \ell$, the the matrix M'_ℓ is singular in all rounds. Therefore, in each iteration the output of SINGULAR is $\text{Enc}(0)$ with probability $1 - \mathbf{neg}(n)$. Therefore, the probability that in any round the output of SINGULAR is $\text{Enc}(0)$ is $\mathbf{neg}(n)$. Hence the output of the protocol is correct with the desired probability. The bounds on the round and communication complexity follow from the complexity bounds of the Protocol SINGULAR (See Theorem 4.3). Note that computing the logical OR of the $\omega(\log n)$ iteration results can be done in 2 rounds and communication which does not effect the asymptotic complexity of the protocol. \square

A.4 Protocols NONZERO, EQUAL and UNARY

Let $x \in \mathbb{F}$. There clearly exist a binary circuit of size $O(k)$ that checks for $x = 0$. Applying Lemma 7.1 leads to the following implementation of Protocol NONZERO.

Lemma A.1 *Suppose Bobs holds an encrypted field element $\text{Enc}(x)$. There exists a secure two-party protocol that runs in constant rounds and $O(k)$ communication complexity that returns to Bob $\text{Enc}(1)$ if $x \neq 0$ and $\text{Enc}(0)$ if $x = 0$.*

Let $0 \leq r, i \leq n$ be two integers. There exist a binary circuit of size $O(\log n)$ that checks for $r \geq i$. Applying this circuit n times in parallel we get an implementation of Protocol UNARY.

Lemma A.2 *Suppose Bob holds encrypted $\text{Enc}_{\text{bin}}(r)$ with $0 \leq r \neq n$. There exists a secure two-party protocol that runs in constant rounds and $O(n \log n + kn) = O(kn)$ communication complexity that returns to Bob a vector $\delta \in \mathbb{F}^n$ such that $\delta_i = 1$ if $r \geq i$ and $\delta_i = 0$ otherwise.*

We remark that by the techniques from [9] it is further possible to implement the two protocols EQUAL and UNARY (without having to rely on Yao's general method) in constant rounds and communication complexity $\tilde{O}(k)$ and $\tilde{O}(kn)$, respectively.

The protocol for EQUAL is also easy to implement. Let $0 \leq x, y \leq n$. Designing a circuit of size $O(\log n)$ for this problem is straightforward.

Lemma A.3 *Suppose Bob holds encrypted $\text{Enc}_{\text{bin}}(x), \text{Enc}_{\text{bin}}(y)$ with $0 \leq x, y \neq n$. There exists a secure two-party protocol that runs in constant rounds and $O(\log n + k \log n) = O(k + \log n)$ communication complexity that returns to Bob $\text{Enc}(1)$ if $x = y$ and $\text{Enc}(0)$ otherwise.*

B Applications

B.1 Linear Subspace Intersection

Let \mathbb{F} be a finite field and n be a positive integer. Alice holds a subspace $V_A \subseteq \mathbb{F}^n$ of dimension $n_a \leq n$. The subspace V_A is represented by an $n_a \times n$ matrix A , where the rows of A span V_A . Similarly, Bob's input is a subspace $V_B \subseteq \mathbb{F}^n$ of dimension n_b , represented by an $n_b \times n$ matrix B . Letting $V_I = V_A \cap V_B$, Alice and Bob wish to securely study different properties of V_I .

In [19], constant round $O(n^2)$ protocols were designed for securely *computing* the subspace V_I , and for securely computing the rank of the subspace V_I . However, it turned out that the problem of securely *deciding* whether the subspace V_I is the trivial zero subspace seems harder to solve. Ignoring security issues, computing the intersection of the input subspaces is at least as hard as deciding whether they have a non trivial intersection. However, constructing a *secure* protocol for the latter turns to be somewhat harder as the players gain less information from its output.

The following claim from [19] reduces the problem of deciding subspace intersection, to computing whether a matrix is of full rank:

Claim B.1 ([19]) *Define $M = AB^\top$. Then $V_I \neq \{0\}$ if and only if the matrix M is not full rank.*

This gives rise to the following protocol:

Protocol INTERSECTION DECIDE

Input: Alice (resp. Bob) holds a $n_a \times n$ (resp. $n_b \times n$) matrix A (resp. B) over a finite field \mathbb{F} representing a subspace $V_A \subseteq \mathbb{F}^n$ (resp. $V_B \subseteq \mathbb{F}^n$). Let B^\top be a $n \times n'_b$ matrix that represents the subspace V_B^\top .

Output: If V_I is the trivial zero subspace, Alice outputs 1. Else, Alice outputs 0.

1. Alice generates keys for a homomorphic public key encryption system, and sends Bob $\text{Enc}(A)$ and the public key.
2. Bob locally computes $\text{Enc}(M)$, where $M \stackrel{\text{def}}{=} AB^\top$. Note that M is a $n_a \times n'_b$ matrix.
3. Alice and Bob execute Protocol RANK on $\text{Enc}(M)$. Denote by $\text{Enc}(r)$ the output of the protocol held by Bob.
4. Alice and Bob execute protocol EQUAL on $\min n_a, n'_b$ and $\text{Enc}(r)$. Bob sends the encrypted output to Alice who decrypts and outputs it.

This protocol has the same communication complexity as of the protocol designed in [19], that is $\tilde{O}(n^2)$. However, the round complexity of this protocol, which is $\text{polylog}(n)$ is substantially better⁷ than the round complexity of [19], which is $\Omega(n^{0.275})$. We note that the techniques in our paper are very different from those of [19].

B.2 Solving a Common Linear Equation System

Let \mathbb{F} be a finite field and n be a positive integer. Alice holds an $n_a \times n$ matrix M_A and a vector $\mathbf{v}_a \in F^{n_a}$. Similarly, Bob's input is an $n_b \times n$ matrix M_B and a vector $\mathbf{v}_b \in F^{n_b}$. Alice and Bob wish to securely compute a random vector $\mathbf{x} \in \mathbb{F}^n$ such that both $M_A \mathbf{x} = \mathbf{v}_a$ and $M_B \mathbf{x} = \mathbf{v}_b$.

This problem can be viewed as computing a random vector from the intersection of the affine subspaces representing the solutions to the systems $M_A \mathbf{x} = \mathbf{v}_a$ and $M_B \mathbf{x} = \mathbf{v}_b$. This problem was considered in [19], who designed a protocol of communication complexity $\tilde{O}(n^2)$ and round complexity $\Omega(n^{0.275})$. We show a protocol which improves the round complexity to $\text{polylog}(n)$ while keeping the communication complexity $\tilde{O}(n^2)$.

The protocol is simple: Alice generates keys for a homomorphic public key encryption system, and sends Bob $\text{Enc}(M_A)$, $\text{Enc}(v_a)$ and the public key. Bob encrypts his input to get the encrypted linear system.

$$\begin{pmatrix} \text{Enc}(M_A) \\ \text{Enc}(M_B) \end{pmatrix} \mathbf{x} = \begin{pmatrix} \text{Enc}(v_a) \\ \text{Enc}(v_b) \end{pmatrix}$$

Alice and Bob then execute Protocol LINEAR SOLVE after which Bob holds $\text{Enc}(\mathbf{x})$ where \mathbf{x} is a random solution to the common system. Finally, Bob sends $\text{Enc}(\mathbf{x})$ to Alice, which decrypts and outputs \mathbf{x} .

B.3 Computing the Determinant of a Shared Matrix

Alice, holding $M_A \in \mathbb{F}^{n \times n}$, and Bob, holding $M_B \in \mathbb{F}^{n \times n}$, share a matrix $M = M_A + M_B$. They wish to compute the *determinant* of M without leaking any other information of M . Again, we give a $\text{polylog}(n)$ round and $\tilde{O}(n^2)$ communication protocol for this problem, improving on previous results. The protocol is again simple and is similar to protocol MATRIX INVERT: Alice generates keys for a homomorphic public

⁷we note that the $\text{polylog}(n)$ factor in the round complexity of our protocol appears in the protocol of [19] as well. Hence our protocol is more round efficient for small values of n as well.

key encryption system, and sends Bob $\text{Enc}(M_A)$ and the public key. Bob encrypts his input and computes $\text{Enc}(M) = \text{Enc}(M_A) + \text{Enc}(M_B)$. Alice and Bob first execute protocol SINGULAR on $\text{Enc}(M)$ and let $\text{Enc}(b)$ be the result of the protocol. Bob sends $\text{Enc}(b)$ to Alice, who decrypts to get b . If $b = 0$, Alice outputs 0. Otherwise, Bob picks a random non-singular $n \times n$ matrix R , locally computes and sends Alice $\text{Enc}(MR)$. Alice decrypts and compute $\ell \det(MR)$ and sends Bob $\text{Enc}(\ell)$. Bob multiplies $\text{Enc}(\ell)$ by $\det(R)^{-1}$ to get $\text{Enc}(\det(M))$, and sends it to Alice, who decrypts and outputs the result.

C Computing the minimal polynomial using the GCD algorithm

In this section we demonstrate an algorithm from [10] how to efficiently compute the minimal polynomial of a sequence $\mathbf{a} = (a_i)_{i \in \mathbb{N}}$ of recursion order n using the Extended Euclidean Algorithm on polynomials. By the definition from Section 3 the minimal polynomial $m_{\mathbf{a}}$ of the sequence \mathbf{a} is the unique monic polynomial $m_{\mathbf{a}}(x) = m(x)$ of least degree $\leq n$ for which $m(x) \bullet \mathbf{a} = \mathbf{0}$. By division with remainder we can rewrite this as

$$m_{\mathbf{a}} \cdot (a_1 + a_2x + \dots + a_{2n}x^{2n-1}) - q(x) \cdot x^{2n} = r(x), \quad (1)$$

where $r(x)$ is a remainder polynomial of degree $< n$, and $q(x)$ is a quotient polynomial. Denote by $a(x)$ the sum $\sum_{i=1}^{2n} a_i x^{i-1}$. If we apply the extended GCD algorithm to the two polynomials $a(x)$ and x^{2n} , keeping track of remainders, we get two sequences $p_i(x), q_i(x)$ such that the $r_i := p_i(x) \cdot a(x) - q_i(x) \cdot x^{2n}$ form a series of polynomials whose degree is strictly decreasing. As soon as the degree of r_i is less than n , we have the required polynomials from (1) with $m_{\mathbf{a}}(x) = p_i(x)$, $q(x) = q_i$, and $r(x) = r_i(x)$.