Secure Linear Algebra Using Linearly Recurrent Sequences

Eike Kiltz*, Payman Mohassel**, Enav Weinreb***, and Matthew Franklin†

Abstract. In this work we present secure two-party protocols for various core problems in linear algebra. Our main result is a protocol to obliviously decide singularity of an encrypted matrix: Bob holds an $n \times n$ matrix, encrypted with Alice's secret key, and wants to learn whether or not the matrix is singular (while leaking nothing further). We give an interactive protocol between Alice and Bob that solves the above problem in $O(\log n)$ communication rounds and with overall communication complexity of roughly $O(n^2)$ (note that the input size is n^2). Our techniques exploit certain nice mathematical properties of linearly recurrent sequences and their relation to the minimal and characteristic polynomial of the input matrix, 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 $O(n^{0.275})$ to $O(\log n)$.

At the core of our results we use a protocol that securely computes the minimal polynomial of an encrypted matrix. Based on this protocol we exploit certain algebraic reductions to further extend our results to the problems of securely computing rank and determinant, and to solving systems of linear equations (again with low round and communication complexity).

Keywords. 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, and multi-party secure computation make use of linear algebra. In particular, the ability to solve a set of linear equations is an important algorithmic and cryptographic tool. In this work we design efficient and secure protocols for various linear algebra problems. Our protocols enjoy both low communication and round complexity.

* CWI Amsterdam, The Netherlands. kiltz@cwi.nl.

** Department of Computer Science, University of California, Davis CA 95616. mohassel@cs.ucdavis.edu.

*** Dept. of Computer Science, Technion, Haifa, Israel. weinreb@cs.technion.ac.il.

† Department of Computer Science, University of California, Davis CA 95616. franklin@cs.ucdavis.edu.

The secure computation of many linear algebra tasks efficiently reduces to the following problem. Alice holds the private key of a public-key homomorphic encryption scheme, and Bob holds a square matrix A whose entries are encrypted under Alice's public key. Alice and Bob wish to decide whether A is singular while leaking no other information on A. 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). Somewhat surprisingly, we design a secure protocol based on this algorithm which is applicable to *general* matrices. Previous secure protocols for linear algebra problems used basic linear algebra techniques such as Gaussian Elimination. Our protocols exploit more advanced properties of linear systems to achieve improved complexity bounds.

Cramer and Damgård initiated the study of secure protocols for solving various linear algebra problems [6]. Their work was done in the information theoretic multi-party setting, with the main focus on achieving constant round complexity. The communication complexity of their protocols is $\Omega(n^3)$ while the size of the inputs is just $O(n^2)$. A generic approach for designing secure protocols is to apply the garbled circuit method of Yao [25], for which the communication complexity is related to the Boolean circuit complexity of the underlying function. However, these linear algebra functions are strongly related to the problem of matrix multiplication [4], with essentially the same circuit complexity. The best known upper bound for matrix multiplication is $O(n^\omega)$ [5] for $\omega \cong 2.38$, which is still larger than the input size. In a recent paper, Nissim and Weinreb [19] introduced an oblivious singularity protocol with communication complexity 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. The need for low round complexity is motivated by the fact that in most practical systems the time spent on sending and receiving messages is large compared to local computation time.

Our Results. We design a secure protocol for deciding singularity of a matrix, which gets the best of previous results, both in terms of communication and round complexity, up to a logarithmic factor. We achieve communication complexity of roughly $O(n^2)$ and $O(\log n)$ round complexity. Our constructions are secure, assuming the existence of a *homomorphic* public-key encryption scheme and a secure instantiation of Yao's garbled circuit protocol. The latter can be constructed using an appropriate symmetric key encryption and an oblivious transfer protocol which is secure against semi-honest adversaries. Using the protocol for deciding singularity, we design a secure protocol for solving a linear system $Ax = y$ based on an algorithm by Kaltofen and Saunders [14]. The technical difficulty in applying this algorithm is that it depends on the rank of the matrix A. Computing the rank of A 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 is of independent interest. The above techniques also yield communication and round efficient secure protocols for computing the minimal polynomial and the determinant of an encrypted matrix. Our results give rise to communication and round efficient secure protocols for problems that are reducible to linear algebra, e.g., perfect matching and problems with low span program complexity [15]. We summarize our main protocols in Table 1. Note that the outputs of our protocols are always encrypted, which in particular enables composition of our protocols. Thus, our protocols may be conveniently used as sub-protocols in other secure protocols.

Our protocols are designed under the assumption that Bob holds an encrypted version of the input and Alice holds the decryption key. In practice, secure linear algebra is often needed when the inputs of Alice and Bob are in the clear. However, applying simple reductions, we are able to give improved secure protocols for many natural problems of this kind. 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 logarithmic factor.[1] Another natural problem that we can compute securely and efficiently is solving a shared system of linear equations. Here Alice and Bob both hold independent systems of linear equations in the same variables. They jointly want to compute a solution vector that satisfies both sets of equations, without revealing anything about their secret inputs.

Protocol name	INPUT		OUTPUT	
	Bob	Alice	Bob	Alice
MINPOLY	$\mathsf{Enc}(A)$	SK	$\mathsf{Enc}(m_A)$	—
SINGULAR	$\mathsf{Enc}(A)$	SK	$\mathsf{Enc}(\det(A) = 0?)$	—
RANK	$\mathsf{Enc}(A)$	SK	$\mathsf{Enc}(\mathrm{rank}(A))$	—
DET	$\mathsf{Enc}(A)$	SK	$\mathsf{Enc}(\det(A))$	—
LINEAR SOLVE	$\mathsf{Enc}(A), \mathsf{Enc}(x)$	SK	$\mathsf{Enc}(y)$ (y random s.t. $Ax = y$)	—

Table 1. Basic linear algebra protocols with $O(n^2 \log n \cdot \log |\mathbb{F}|)$ communication complexity and $O(\log n)$ rounds. Here $A \in \mathbb{F}^{n \times n}$ is a matrix and $x \in \mathbb{F}^n$ is a vector.

Techniques. Our protocols rely on random reductions from computing linear algebra properties of a matrix $A \in \mathbb{F}^{n \times n}$ to computing the *minimal polynomial* m_A of a certain matrix related to A [24, 14, 11]. In particular, the singularity of A is related to the constant coefficient of this minimal polynomial, and the rank of A to its degree.

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

Since no efficient secure protocol for computing the minimal polynomial of a shared matrix is known, we exploit another probabilistic reduction from this problem to computing the minimal polynomial of a particular *linearly recurrent sequence*. A sequence of field elements $\mathbf{a} = (a_i)_{i\in\mathbb{N}} \in \mathbb{F}^{\mathbb{N}}$ is linearly recurrent of order $n \in \mathbb{N}$ if there exist $f_0, \ldots, f_n \in \mathbb{F}$ with $f_n \neq 0$ such that $\sum_{j=0}^{n} f_j a_{i+j} = 0$, for all $i \in \mathbb{N}$. The field elements $f_0, \ldots, f_{2n-1}$ completely characterize the sequence $\mathbf{a}$ and are, roughly speaking, related to its minimal polynomial (see Section 3). Picking u, v uniformly in $\mathbb{F}^n$, the minimal polynomial of the linearly recurrent sequence $\mathbf{a} = (\mathbf{u}^\top A^i \mathbf{v})_{i\in\mathbb{N}}$ of order n coincides, with high probability, with the minimal polynomial of matrix A [11].

To securely compute the minimal polynomial of the above sequence, we first show how to compute the first $2n$ elements of $\mathbf{a}$ in low round and communication complexity. Then we use the Berlekamp/Massey algorithm [17] to reduce the problem to computing the *extended greatest common divisor* of two polynomials derived from the first $2n$ elements of sequence $\mathbf{a}$. Finally, we exploit the fact the the Boolean circuit complexity of the extended GCD algorithm is significantly smaller than that of the original linear algebraic function to apply Yao's garbled circuit method here. Moreover, we show a general technique to apply Yao's garbled circuit method from a starting point where Bob holds an encrypted input and Alice holds the decryption key. As we discussed earlier in the introduction, trying to apply Yao's construction directly to the original linear algebra problems would result in $\Omega(n^\omega)$ communication complexity.

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 show how to compute the minimal polynomial of an encrypted matrix. We design protocols for deciding singularity, computing rank and determinant of an encrypted matrix, and solving an encrypted linear system in Section 5. The appendices contain some additional details and applications of our secure protocols.

2 General Framework

Homomorphic encryption schemes. As a first step in our protocols, we reduce the original linear algebra problems to a state where Bob holds data encrypted by a public key homomorphic encryption scheme, and Alice holds the private decryption key. Our constructions use semantically-secure public-key encryption schemes that allow for simple computations on encrypted data. In particular, we use encryption schemes where given two encryptions $\mathsf{Enc}(m_1)$ and $\mathsf{Enc}(m_2)$, we can efficiently compute a random encryption $\mathsf{Enc}(m_1 + m_2)$. Note that this implies that given an encryption $\mathsf{Enc}(m)$ and $c \in \mathbb{F}$, we can efficiently compute a random encryption $\mathsf{Enc}(cm)$. We will be working with encryption of elements in a finite field. Paillier's [20] cryptosystem is an appropriate choice for this purpose. One minor issue is that the domain of Paillier's cryptosystem is the ring Z_n, where n is the product of two large and secret primes. Note that Z_n has all of the properties of a finite field except that some of the non-zero elements in Z_n are not invertible. We assume that all the non-zero values used by our protocols are invertible elements of Z_n. This assumption is reasonable since otherwise one could use our protocols to factor n. Particularly, an extended GCD algorithm on any element x used by our protocols and n, would either find the inverse of $x \bmod n$, or find a non-trivial factor of n. So in the context of our paper, we can describe computations in Z_n as if it was a finite field. Several other constructions of homomorphic encryption schemes are known, each with their particular properties (see e.g. [22, 13, 10, 21, 23, 3, 18, 7]).

We view our protocols as algorithms that Bob executes on his encrypted input. As mentioned above, the homomorphic encryption allows Bob to locally perform several simple computations on his input. However, other computations require the help of Alice. As a simple example of a protocol where Bob uses Alice's help, consider the following (folklore) protocol MATRIX MULT for encrypted matrix multiplication.

Bob holds the encryptions $\mathsf{Enc}(A)$ and $\mathsf{Enc}(B)$ of two matrices $A \in \mathbb{F}^{n\times\ell}$ and $B \in \mathbb{F}^{\ell\times m}$. Alice holds the private decryption key. At the end of the protocol Bob should hold the encryption $\mathsf{Enc}(AB)$ 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 $\mathsf{Enc}(A + R_A)$ and $\mathsf{Enc}(B + R_B)$, which he can locally compute using the homomorphic properties of $\mathsf{Enc}(\cdot)$. Alice decrypts these matrices and returns $\mathsf{Enc}((A+R_A)\cdot(B+R_B))$ to Bob. Finally Bob locally computes $\mathsf{Enc}(AB) = \mathsf{Enc}((A+R_A)(B+R_B))-\mathsf{Enc}(AR_B)-\mathsf{Enc}(R_AB)-\mathsf{Enc}(R_AR_B)$. 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.

Notation. We denote by $\mathbf{neg}(x)$ a function that is negligible in x, i.e., $\mathbf{neg}(x) = x^{-\omega(1)}$. Let $\mathbb{F}$ be a finite field with p elements, and denote $k = \log p$. To make our complexity statements simpler, we make the assumption that the size of the field $\mathbb{F}$ is not too big[2] with respect to the dimensions of the matrix, i.e. $\log|\mathbb{F}| = k = O(n)$. Our protocols usually work with error probability of about $n/|\mathbb{F}|$. That is, we also assume the field size to be super-polynomial in n. If the field size is too small, we can always work over an extension field of appropriate size. This may add a small multiplicative factor polylog(n) to the communication complexity of the protocol. For example, for the case of $\mathbb{F}_2$ we could view the elements as if they were from $(\mathbb{F}_2)^\alpha$ for $\alpha = (\log n)^{1+\epsilon}$. This would add a factor of $(\log n)^{1+\epsilon}$ to the communication complexity, and reduce the error probability to $\mathbf{neg}(n)$.

For an encryption scheme, we denote by λ its security parameter. We assume that the result of encrypting a field element is of length $O(\lambda+k)$. As a convention, the complexities of our protocols count the number of encrypted field elements that are communicated during the protocol.

We view a vector $\mathbf{v} \in \mathbb{F}^n$ as a column vector. To denote a row vector we use $\mathbf{v}^\top$. For a vector $\mathbf{v} \in \mathbb{F}^n$, we denote by $\mathsf{Enc}(\mathbf{v})$ the coordinate-wise encryp-

[2] For bigger fields the complexity of our protocols grows at most by an additional factor of $\log k$.

tion of $\mathbf{v}$. That is, if $\mathbf{v} = \langle a_1, \ldots, a_n \rangle$ where $a_1, \ldots, a_n \in \mathbb{F}$, then $\mathsf{Enc}(\mathbf{v}) = \langle \mathsf{Enc}(a_1), \ldots, \mathsf{Enc}(a_n) \rangle$. Similarly, for a matrix $A \in \mathbb{F}^{m \times n}$, we denote by $\mathsf{Enc}(A)$ the $m \times n$ matrix such that $\mathsf{Enc}(A)[i,j] = \mathsf{Enc}(A[i,j])$. 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 $\mathsf{Enc}(\mathbf{v_1})$ and $\mathsf{Enc}(\mathbf{v_2})$, we can efficiently compute $\mathsf{Enc}(\mathbf{v_1} + \mathbf{v_2})$, and similarly with matrices. (ii) Given an encryption of a vector $\mathsf{Enc}(\mathbf{v})$ and a constant $c \in \mathbb{F}$, we can efficiently compute $\mathsf{Enc}(c\mathbf{v})$. (iii) Given an encryption of a matrix $\mathsf{Enc}(A)$ and a matrix A' of the appropriate dimensions, we can efficiently compute $\mathsf{Enc}(AA')$ and $\mathsf{Enc}(A'A)$, as any entry in the resulting matrix is a linear combination of some encrypted matrix entries.

Adversary model. Our protocols are constructed for the two-party semi-honest adversary model.[3] 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 [12] in the semi-honest adversary model. Designing communication and round efficient secure protocols for linear algebraic problems in the malicious model remains an open problem.

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 the number of encrypted values $\mathsf{Enc}(\cdot)$ exchanged between Alice and Bob.

2.1 Applying Yao's Garbled Circuit Method

In Yao's garbled circuit method [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 $O(G + m' + \ell')$.[4]

[3] Getting the same results in the multi-party information theoretic setting remains an open problem. In particular, our protocols reduce the linear algebra problems into a variant of the extended GCD problem for polynomials. Unfortunately, a communication and round efficient protocol for this problem is not known in the multi-party information theoretic setting.

[4] Here 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

In our (homomorphic encryption) setting, we typically get to a state where Bob holds $\mathsf{Enc}(y)$ and Alice holds a private decryption key, and they wish for Bob to learn $\mathsf{Enc}(f(y))$ while Alice learns nothing, for some function f computed by a given circuit. In our protocols, we sometimes need to switch from this "homomorphic encryption setting" to the setting of Yao's garbled circuit to perform some tasks more efficiently. Then, we change from this setting to the "homomorphic encryption setting" and continue. For completeness, next we explain a simple way of doing so securely and efficiently.

From Homomorphic Encryption to Yao's Assume that Bob is holding $\mathsf{Enc}(a)$ where $a \in \mathbb{F}$. Parties want to switch to a circuit C_f that computes the function $f(a, ...)$ on a and other inputs, without revealing the value of a.

Bob generates a random $r \in \mathbb{F}$ and sends $\mathsf{Enc}(a+r)$ to Alice. Alice decrypts to get $a+r$. Now, parties create a circuit C'_f such that Bob feeds r to C'_f as his part of the input, and Alice feeds $a+r$ to C'_f as her part of the input. They also add the additional circuitry that subtracts $(a+r)-r = a$, and use the output of this circuitry in the same way that a would be used in C_f. Everything else will stay the same as it was in C_f. The circuit for subtraction requires $O(k)$ gates. This does not affect the overall complexity of the circuit.

From Yao's to Homomorphic Encryption Assume that Bob and Alice want to apply Yao's garbled circuit method to compute the function f, and C_f is an appropriate circuit for this task. Lets denote the output of f by $o \in \mathbb{F}$. Then, parties want to have Bob hold $\mathsf{Enc}(o)$ without revealing o itself. In what follows, we assume that Bob creates the circuit and Alice evaluates it (for more information on Yao's protocol see [16]).

Bob generates a random value $r \in F$. Parties create a circuit C'_f such that Bob feeds r to C'_f as part of his input. C'_f is the same as C_f except that parties add the additional circuity to the end of the circuit to add r to o and output $o+r$ instead of o. Note that only Alice receives the output. She encrypts and sends $\mathsf{Enc}(o+r)$ to Bob. Bob computes $\mathsf{Enc}(o) = \mathsf{Enc}(o+r) - \mathsf{Enc}(r)$ on his own.

The circuit for addition requires $O(k)$ gates and does not affect the overall complexity of the circuit. Parties can use the above two transformation on the same circuit if the goal is to change back and forth between the two different settings.

3 Linearly Recurrent Sequences

We reduce our various problems from linear algebra 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 [11].

---

of the circuit which is usually done by sending the output of a pseudorandom bit generator) have a communication complexity $O(\lambda)$ (where $\lambda = |\mathsf{Enc}(\cdot)|$) for each execution.

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 $f_0, \ldots, f_n \in \mathbb{F}$ 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

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

This makes $\mathbb{F}^{\mathbb{N}}$, together with $\bullet$, into an $\mathbb{F}[x]$-module.[5]

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 $\mathrm{Ann}(\mathbf{a})$. Since any ideal in $\mathbb{F}[x]$ is generated by a single polynomial, either $\mathrm{Ann}(\mathbf{a}) = \{0\}$ or there is a unique monic polynomial $m \in \mathrm{Ann}(\mathbf{a})$ of least degree such that $\langle m\rangle = \{rm : r \in \mathbb{F}[x]\} = \mathrm{Ann}(\mathbf{a})$. 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 $A \in \mathbb{F}^{n\times n}$ be a matrix, and $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$ be vectors. We will be interested in the following three sequences:

- $\mathbf{A} = \mathbf{A}_A = (A^i)_{i\in\mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}^{n\times n}$.
- $\mathbf{a} = \mathbf{a}_{A,\mathbf{v}} = (A^i\mathbf{v})_{i\in\mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}^n$.
- $\mathbf{a}' = \mathbf{a}'_{A,\mathbf{u},\mathbf{v}} = (\mathbf{u}^\top A^i\mathbf{v})_{i\in\mathbb{N}}$ where the sequence elements are from $V = \mathbb{F}$.

Definition 1. *The minimal polynomial of a matrix $A \in \mathbb{F}^{n\times n}$ is defined as $m_A = m_{\mathbf{A}}$, i.e. as the minimal polynomial of the sequence $\mathbf{A} = (A^i)_{i\in\mathbb{N}}$.*

By our definition of the minimal polynomial of a sequence the minimal polynomial of A can alternatively be characterized as the unique monic polynomial $p(x)$ over $\mathbb{F}$ of least degree such that $p(A) = 0$.

We denote by $f_A = \det(xI_n - A) = \sum_{i=0}^{n} f_j x^j$ the characteristic polynomial of matrix $A \in \mathbb{F}^{n\times n}$. Note that f_A is monic.

Lemma 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 $m_{\mathbf{a}'}|m_{\mathbf{a}}|m_{\mathbf{A}}|f_A$.*

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

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

[5] Roughly 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. A formal definition can be found in many linear algebra textbooks (e.g., [11]).

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

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 $(\sum_{j=0}^{n} a_j A^{i+j})_{i\in\mathbb{N}} = \mathbf{0}$. Hence,

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

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

Corollary 1. *The sequences* $\mathbf{a}, \mathbf{a}', \mathbf{A}$ *are linearly recurrent of order at most* n.

We will use the following useful result (e.g. see [8, page 92]).

Lemma 2. *The minimal polynomial* $m_A(x)$ *of* A *divides the characteristic polynomial* $f_A(x)$ *of* A*, and both polynomials have the same irreducible factors.*

Since $f_A(0) = \det(-A) = (-1)^n \det(A)$, we obtain:

Corollary 2. $m_A(0) = 0$ *if and only if* $\det(A) = 0$.

Corollary 3. *If* f_A *is square-free, then* $m_A = f_A$*, which implies that* $m_A(0) = f_A(0) = (-1)^n \cdot det(A)$.

4 Computing the Minimal Polynomial of a Matrix

In this section we consider the following problem: Bob holds an $n \times n$ dimensional matrix $\mathsf{Enc}(A)$ over a finite field $\mathbb{F}$, encrypted under a public-key homomorphic encryption scheme. Alice holds the private decryption key. We design a secure two-party protocol such that in the end Bob holds an encryption of m_A, the minimal polynomial of A. Computing the minimal polynomial of matrix A can be reduced to computing the minimal polynomial of the linearly recurrent sequence of field elements $\mathbf{a}' = (\mathbf{u}^\top A^i \mathbf{v})_{i\in\mathbb{N}}$. The correctness of the reduction is proved in Exercise 12.15 in [11].

Lemma 3. *Let* $A \in \mathbb{F}^{n\times n}$ *and let* m_A *be the minimal polynomial of matrix* A*. For* $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$ *chosen uniformly at random, we have* $m_A = m_{\mathbf{a}'}$ *with probability at least* $1 - 2\deg(m_A)/|\mathbb{F}|$.

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 $(\mathsf{Enc}(\mathbf{u}^\top A^i \mathbf{v}))_{0\le i\le 2n-1}$ using the least number of matrix multiplication operations.

We now show how to compute the sequence using $2\log n$ matrix multiplication operations. First compute $\mathsf{Enc}(A^{2^j})$ for $0 \le j \le \log n$. 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 a sequence of $\log n$ matrix multiplications: (Note that all the matrices are of dimensions at most $n \times n$.)

$$\begin{array}{lcr}
\mathsf{Enc}(A\mathbf{v}) & = & \mathsf{Enc}(A) \cdot \mathbf{v} \\
\mathsf{Enc}(A^3\mathbf{v}|A^2\mathbf{v}) & = & \mathsf{Enc}(A^2) \cdot \mathsf{Enc}(A\mathbf{v}|\mathbf{v}) \\
\mathsf{Enc}(A^7\mathbf{v}|A^6\mathbf{v}|A^5\mathbf{v}|A^4\mathbf{v}) & = & \mathsf{Enc}(A^4) \cdot \mathsf{Enc}(A^3\mathbf{v}|A^2\mathbf{v}|A\mathbf{v}|\mathbf{v}) \\
\vdots & = & \vdots \\
\mathsf{Enc}(A^{2n-1}\mathbf{v}|A^{2n-2}\mathbf{v}|\ldots|A^n\mathbf{v}) & = & \mathsf{Enc}(A^n) \cdot \mathsf{Enc}(A^{n-1}\mathbf{v}|A^{n-2}\mathbf{v}|\ldots|A\mathbf{v}|\mathbf{v})
\end{array}$$

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

Our next step is to compute the minimal polynomial. By Corollary 1, 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 following sub-protocol. 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 A.2. Using the fast Extended Euclidean algorithm [11, Chapter 11] the latter one can be carried out using an algebraic circuit of size $O(n \log n)$. By implementing each algebraic operation over $\mathbb{F}$ with a binary circuit of size $O(k \log k \log\log k)$ we get a binary circuit of size $O(nk \log n \log k \log\log k)$ for computing the minimal polynomial. We will use the fact that the size of this circuit is $O(n^2 k \log n)$, and so it will not be the dominate part in the overall complexity of our protocol (since we assume $|\mathbb{F}| = 2^{O(n)}$ and thus $k = \log|\mathbb{F}| = O(n)$). Using the techniques from Section 2.1 we now apply Yao's protocol to this circuit and obtain the following result.

Lemma 4. *Suppose Bob holds a sequence* $\mathsf{Enc}(\mathbf{a}') = (\mathsf{Enc}(a'_0), \ldots, \mathsf{Enc}(a'_{2n-1}))$, *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 that runs in constant rounds and* $O(n^2 k \log n)$ *communication complexity that returns the encrypted minimal polynomial* $\mathsf{Enc}(m_{\mathbf{a}'})$ *of* $\mathbf{a}'$ *to Bob.*

The following protocol computes the minimal polynomial of matrix A.

Protocol MINPOLY

Input: $\mathsf{Enc}(A)$ where $A \in \mathbb{F}^{n \times n}$
Output: $\mathsf{Enc}(m_A)$.

1. Pick random vectors $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^n$.
   Compute $\mathsf{Enc}(\mathbf{a}')$, i.e. for $i = 0, \ldots, 2n-1$ compute the values $\mathsf{Enc}(a'_i) = \mathsf{Enc}(\mathbf{u}^\top A^i \mathbf{v})$ using $2 \log n$ executions of the matrix multiplication protocol.
2. Compute $\mathsf{Enc}(m_{\mathbf{a}'})$, an encryption of the minimal polynomial of the sequence $\mathbf{a}' = (a'_i)_{0 \leq i \leq 2n-1}$ using Yao's Protocol.
3. Return $\mathsf{Enc}(m_{\mathbf{a}'})$ as encryption of the minimal polynomial of matrix A.

The following theorem summarizes the properties of Protocol MINPOLY.

Theorem 1. *Let $\mathsf{Enc}(A)$ be an encrypted $n \times n$ matrix over a finite field $\mathbb{F}$. Then protocol* MINPOLY *securely computes $\mathsf{Enc}(m_A)$ with probability $1-2n/|\mathbb{F}|$, communication complexity $O(n^2 k \log n)$ and round complexity $O(\log n)$, where $k = \log |\mathbb{F}|$.*

5 Singularity, Rank, Determinant, and Linear Equations

In this section, we present our main basic linear algebra protocols from Table 1 for testing if a matrix is singular, computing the rank of a matrix, computing the determinant of a matrix, and solving a system of linear equations.

5.1 Testing Matrix Singularity

One possible implementation of a protocol to securely test matrix singularity is based on Corollary 2 stating that $m_A(0) = 0$ if and only if $\det(A) = 0$. Hence, testing singularity can be reduced to computing the minimal polynomial of the matrix A and checking if its constant term equals zero. By Theorem 1 its success probability is bounded by $1 - 2n/|\mathbb{F}|$ and a secure implementation is given by protocol MINPOLY from Section 4. We now present an alternative protocol achieving a slightly improved error bound by exploiting certain algebraic properties of the minimal polynomial of sequence $\mathbf{a}$.

Again we reduce matrix singularity to computing the minimal polynomial of $\mathbf{a}'$. Our reduction works in three steps. Our first step is to reduce the problem of deciding whether $\det(A) = 0$ to deciding whether the linear system $A\mathbf{x} = \mathbf{v}$ is solvable for some random vector $\mathbf{v} \in \mathbb{F}^n$. If A is non-singular then, obviously, the linear system must be solvable. On the other hand, if $\det(A) = 0$, then with probability at least $1 - 1/|\mathbb{F}|$, the linear system has no solution.

In the second step 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 $\mathbf{a} = (A^i \mathbf{v})_{i \in \mathbb{N}}$.

Lemma 5. *If $m_{\mathbf{a}}(0) \neq 0$ then the system $A\mathbf{x} = \mathbf{v}$ is solvable.*

Proof. Since by Corollary 1 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

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

Since $m_0 = m_{\mathbf{a}}(0)$ is non-zero, we get

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

and hence

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

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

In the third step we reduce computing the minimal polynomial of sequence $\mathbf{a} = (A^i \mathbf{v})_{i \in \mathbb{N}}$ to computing the minimal polynomial of $\mathbf{a}' = (\mathbf{u}^\top A^i \mathbf{v})_{i \in \mathbb{N}}$, where $\mathbf{u} \in \mathbb{F}^n$ is a random vector. The correctness of the reduction is proved in Lemma 12.17 in [11].

Lemma 6. *Let $A \in \mathbb{F}^{n \times n}$, $\mathbf{v} \in \mathbb{F}^n$, $m_{\mathbf{a}}$ the minimal polynomial of the sequence $\mathbf{a} = (A^i \mathbf{v})_{i \in \mathbb{N}}$. For a $\mathbf{u} \in \mathbb{F}^n$ chosen uniformly at random we have that $m_{\mathbf{a}}$ is the minimal polynomial of the sequence $\mathbf{a}' = (\mathbf{u}^\top A^i \mathbf{v})_{i \in \mathbb{N}}$ with probability at least $1 - \deg(m_{\mathbf{a}})/|\mathbb{F}|$.*

Protocol SINGULAR

Input: $\mathsf{Enc}(A)$ where $A \in \mathbb{F}^{n \times n}$
Output: $\mathsf{Enc}(0)$ if $\det(A) = 0$ and $\mathsf{Enc}(1)$ otherwise.

1. Pick random vectors $\mathbf{u}, \mathbf{v} \in_R \mathbb{F}^n$.
   For $i = 0 \ldots 2n-1$ compute the values $a'_i = \mathsf{Enc}(\mathbf{u}^\top A^i \mathbf{v})$ using $2 \log n$ executions of the matrix multiplication protocol.
2. Compute $\mathsf{Enc}(m_{\mathbf{a}'})$, an encryption of the minimal polynomial of the sequence $\mathbf{a}' = (a'_i)_{0 \le i \le 2n-1}$ except that in the last step a circuit is used that returns 0 if $m_{\mathbf{a}'}(0) = 0$ and 1 otherwise using Yao's Protocol.

The following theorem summarizes the properties of Protocol SINGULAR.

Theorem 2. *Let $\mathsf{Enc}(A)$ be an encrypted $n \times n$ matrix over a finite field $\mathbb{F}$. Then Protocol* SINGULAR *securely checks if A is singular with probability $1 - (n+1)/|\mathbb{F}|$, communication complexity $O(n^2 k \log n)$ and round complexity $O(\log n)$, where $k = \log |\mathbb{F}|$.*

Proof. We first prove that if $\det(A) \neq 0$ then the output of the protocol is $\mathsf{Enc}(1)$. If $m_{\mathbf{a}'}(0) = 0$, this means that the constant coefficient of $m_{\mathbf{a}'}$ is 0, thus $x | m_{\mathbf{a}'}$. By Lemma 1, $m_{\mathbf{a}'} | f_A$, where f_A is the characteristic polynomial of the matrix A. Hence, the constant coefficient of f_A is 0, which implies $\det(A) = 0$. Hence if A is non-singular, the output of the entire protocol must be $\mathsf{Enc}(1)$.

On the other hand, if $\det(A) = 0$ then, by Lemma 5, if the following two events happen, the output of the protocol is $\mathsf{Enc}(0)$: (i) The system $Ax = v$ is not solvable. (ii) $m_{\mathbf{a}'} = m_{\mathbf{a}}$. The probability of event (i) is at least $(1 - 1/|\mathbb{F}|)$. The probability of event (ii), by Lemma 6, is at least $1 - \deg(m_{\mathbf{a}})/|\mathbb{F}| \geq 1 - n/|\mathbb{F}|$. Therefore, with probability at least $1 - (n+1)/|\mathbb{F}|$ the output is $\mathsf{Enc}(0)$. Security of the protocol follows by security of the sub-protocols used. Round and communication complexity of the protocol is easy to verify.

5.2 Computing the Rank

In this section we show how to compute $\mathsf{Enc}(\text{rank}(A))$ given an encryption $\mathsf{Enc}(A)$ of a matrix $A \in \mathbb{F}^{n\times n}$. To compute the rank of a matrix A, we use the following two results which are proved in [14].

Lemma 7. *Let A be a matrix in $\mathbb{F}^{n\times n}$ of (unknown) rank r. Let U and L be randomly chosen unit upper triangular and lower triangular Toeplitz matrices in $\mathbb{F}^{n\times n}$, and let $B = UAL$. Lets denote the $i \times i$ leading principal of B by B_i. The probability that $\det(B_i) \neq 0$ for all $1 \leq i \leq r$ is greater than $1 - n^2/|\mathbb{F}|$.*

Lemma 8. *Let matrix $B \in \mathbb{F}^{n\times n}$ have leading invertible principals up to B_r where r is the (unknown) rank of B. Let X be a randomly chosen diagonal matrix in $\mathbb{F}^{n\times n}$. Then, $r = \deg(m_{XB}) - 1$ with probability greater than $1 - n^2/|\mathbb{F}|$.*

The above two results lead to the following protocol for computing the rank of a matrix.

Protocol RANK

Input: $\mathsf{Enc}(A)$ where $A \in \mathbb{F}^{n\times n}$.
Output: $\mathsf{Enc}(r)$ where r is rank of A.

1. Generate random unit upper and lower triangular Toeplitz matrices $U, L \in \mathbb{F}^{n\times n}$ and a random diagonal matrix $X \in \mathbb{F}^{n\times n}$.
2. Compute $\mathsf{Enc}(M) = XU \cdot \mathsf{Enc}(A) \cdot L$.
3. Run the protocol MINPOLY on M except that in the last step, use a circuit that only outputs the degree of the minimal polynomial minus 1, and not the polynomial itself.

The following theorem is implied by the above two lemmas and summarizes the properties of our RANK protocol.

Theorem 3. *Let $\mathsf{Enc}(A)$ be an encrypted $n \times n$ matrix over a finite field $\mathbb{F}$. Then Protocol* RANK *securely outputs the encrypted rank of A with probability at least $1 - 2n^2/|\mathbb{F}|$, communication complexity $O(n^2 k \log n)$, and round complexity $O(\log n)$, where $k = \log|\mathbb{F}|$.*

5.3 Computing the Determinant

In this section we show how to compute $\mathsf{Enc}(\det(A))$, given an encryption $\mathsf{Enc}(A)$ of a matrix $A \in \mathbb{F}^{n\times n}$. The protocol uses the following fact from linear algebra [24].

Lemma 9. *Let B be an $n \times n$ matrix over $\mathbb{F}$ where all the leading principal submatrices of B, including B itself are nonsingular, and let X be a uniformly chosen diagonal matrix in $\mathbb{F}^{n\times n}$. Then, f_{XB} is square-free with probability greater than $1 - n/|\mathbb{F}|$.*

Protocol DET

Input: $\mathsf{Enc}(A)$ where $A \in \mathbb{F}^{n\times n}$.
Output: $\mathsf{Enc}(d)$ where d is the determinant of A.

1. Generate random unit upper and lower triangular Toeplitz matrices $U, L \in \mathbb{F}^{n\times n}$ and a random diagonal matrix $X \in \mathbb{F}^{n\times n}$.
2. Computes $\mathsf{Enc}(Z) = XU \cdot \mathsf{Enc}(A) \cdot L$.
3. Run the protocol MINPOLY on Z except that in the last step, use a circuit that computes $(-1)^n m_Z(0)/\det(X)$ instead of m_Z (note that Bob knows $\det(X)$, and feeds it to the circuit as part of his input.).

The following theorem summarizes the properties of our DET protocol.

Theorem 4. *Let $\mathsf{Enc}(A)$ be an encrypted $n \times n$ matrix over a finite field $\mathbb{F}$. Then Protocol* DET *securely outputs the encryption of determinant of A with probability at least $1 - 3n^2/|\mathbb{F}|$, communication complexity $O(n^2 k \log n)$, and round complexity $O(\log n)$, where $k = \log|\mathbb{F}|$.*

Proof. If A is singular, $Z = XUAL$ is also singular. Therefore, based on Corollary 2, $m_Z(0) = 0$. Hence, the protocol correctly returns $\mathsf{Enc}(0)$ as the answer. On the other hand, if A is non-singular, Z is also non-singular. Note that from given the determinant of Z, Bob can easily derive the determinant of A, as he has all the other matrices in the clear.

Based on Corollary 3, if Z is square-free, computing the constant coefficient of m_Z is sufficient to compute the $det(Z)$. We now show that the probability that Z is square-free is high. By Lemma 7, the probability that all the leading principals of the matrix UAL are of full rank is $1 - n^2/|\mathbb{F}|$. Conditioned on the latter, Lemma 9 implies that with probability $1 - n/|\mathbb{F}|$ the matrix $Z = XUAL$ is square-free. Hence, with probability greater than $1 - 2n^2/|\mathbb{F}|$, the matrix Z is indeed square free. We also need the minimal polynomial protocol to succeed, which happens with probability $1 - 2n/|\mathbb{F}|$. Hence, the overall success probability of the protocol is at least $1 - 3n^2/|\mathbb{F}|$. The round and communication complexity of the protocol is easy to verify.

5.4 Solving Linear Equations

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

The easy case is where M is a non-singular square matrix. In this case it is enough to compute $\mathsf{Enc}(M^{-1})$ and then execute Protocol MATRIX MULT once to compute $\mathsf{Enc}(M^{-1})\mathsf{Enc}(\mathbf{y}) = \mathsf{Enc}(M^{-1}\mathbf{y})$, which is the unique solution to the system (and hence is also a random solution). To compute $\mathsf{Enc}(M^{-1})$ from $\mathsf{Enc}(M)$ we use Protocol MATRIX INVERT which assume the encrypted input matrix M to be invertible. see Appendix A.1 for an implementation of protocol MATRIX INVERT based on [1].

To reduce the general case to the non-singular case, we adapt an algorithm of Kaltofen and Saunders [14]. Their 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, 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 $M'_r\mathbf{x}_r = \mathbf{y}'_r$, and denote the solution by $\mathbf{u}_r$. (iv) Let $\mathbf{u}^* \in \mathbb{F}^n$ be a vector with upper part u_r and lower part 0^{n-r}. 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 [14, Theorem 4]. Note that this algorithm is correct assuming that the system $M\mathbf{x} = \mathbf{y}$ is solvable. An implementation of the first step relies on the following simple linear algebraic lemma.

Lemma 10. *Let M be a matrix in $\mathbb{F}^{m\times n}$ of (unknown) rank r. Let $P \in \mathbb{F}^{m\times m}$ and $Q \in \mathbb{F}^{n\times n}$ randomly chosen full rank matrices, and let $M' = PMQ$. Denote the $r \times r$ leading principal of M' by M'_r. The probability that $det(M'_r) \neq 0$ is greater than $1 - 2n/|\mathbb{F}|$.*

Implementing Kaltofen-Saunders 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 by showing how to implement the Kaltofen-Saunders algorithm using only an encryption of r (computed using Protocol RANK from Section 5.1). 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: $\mathsf{Enc}(M)$ where $M \in \mathbb{F}^{m\times n}$ and $n \leq m$, and $\mathsf{Enc}(\mathbf{y})$ where $\mathbf{y} \in \mathbb{F}^m$. This protocol assumes the system $M\mathbf{x} = \mathbf{y}$ is solvable.
Output: $\mathsf{Enc}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{F}^n$ is a random solution to the system $M\mathbf{x} = \mathbf{y}$.

1. Execute Protocol RANK on $\mathsf{Enc}(M)$ to compute $\mathsf{Enc}(r)$ where $r = \mathrm{rank}(M)$.
2. Locally compute $\mathsf{Enc}(M') = P \cdot \mathsf{Enc}(M) \cdot Q$ and $\mathsf{Enc}(\mathbf{y}') = P \cdot \mathsf{Enc}(\mathbf{y})$ where P and Q are random non-singular $m \times m$ and $n \times n$ matrices respectively.
3. Compute the encrypted matrix $\mathsf{Enc}(N')$, where $N' \in \mathbb{F}^{n\times n}$ consists of the r by r leading principal of M' in the top-left corner and of the unit matrix in the bottom-right corner.
4. Compute $\mathsf{Enc}(N'^{-1})$ using protocol MATRIX INVERT.
5. Pick a random vector $\mathbf{u} \in \mathbb{F}^n$ and set $\mathsf{Enc}(\mathbf{y}'_r)$ for $\mathbf{y}'_r \in \mathbb{F}^n$ to be a vector whose upper r coordinates are the upper r coordinates of $\mathsf{Enc}(\mathbf{y}') + \mathsf{Enc}(M')\mathbf{u}$ and lower $n - r$ coordinates are $\mathsf{Enc}(0)$.
6. Compute $\mathsf{Enc}(\mathbf{u}_r) = \mathsf{Enc}(N'^{-1}) \cdot \mathsf{Enc}(\mathbf{y}'_r)$ and output $\mathsf{Enc}(\mathbf{x}) = Q^{-1} \cdot \mathsf{Enc}(\mathbf{u} - \mathbf{u}_r)$.

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 easily done using Yao's garbled circuit method.

In Step 3 to compute $\mathsf{Enc}(N')$ from $\mathsf{Enc}(M')$ and $\mathsf{Enc}(r)$ one proceeds as follows. First $\mathsf{Enc}(r)$ is converted into unary representation (i.e., $(\mathsf{Enc}(\delta_1), \ldots, \mathsf{Enc}(\delta_n))$ with $\delta_i = 1$ if $i \leq r$ and $\delta_i = 0$ otherwise) using Yao's garbled circuit method. Then create $\mathsf{Enc}(\varDelta)$, where $\varDelta$ is the $n \times n$ matrix $\varDelta = \mathrm{diag}(\delta_1, \delta_2, \ldots, \delta_n)$. Then $\mathsf{Enc}(N')$ is computed as $\mathsf{Enc}(N') = \mathsf{Enc}(M')\mathsf{Enc}(\varDelta) + I_n - \mathsf{Enc}(\varDelta)$, where I_n is the $n \times n$ identity matrix.

As a final note, we stress that the requirement that $n \leq m$ is made only for simplicity of presentation. Otherwise, N' 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 5. *Let $\mathsf{Enc}(M)$ be an encrypted $m\times n$ matrix over a finite field $\mathbb{F}$, and let $\mathsf{Enc}(\mathbf{y})$ be an encrypted vector $\mathbf{y} \in \mathbb{F}^m$. Protocol* LINEAR SOLVE *securely computes $\mathsf{Enc}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{F}^n$ is a random solution of $M\mathbf{x} = \mathbf{y}$, with probability $1 - 3n^2/|\mathbb{F}|$, communication complexity $O(n^2 k \log n)$ and round complexity $O(\log n)$, where $k = \log |\mathbb{F}|$.*

We now discuss the success probability of Protocol LINEAR SOLVE. In Step 1, we compute an encryption of the rank of M which is by Theorem 3 correct with probability $1 - 2n^2/|\mathbb{F}|$. In Step 2, we multiply the matrix M from the right and from the left by random non-singular matrices to get the matrix

M'. By Lemma 10, the top left $r \times r$ sub-matrix of M' is of rank r with probability $1 - 2n/|\mathbb{F}|$. If this is the case, then the rest of the protocol follows the Kaltofen-Saunders algorithm, and thus its correctness is implied by [14, Theorem 4].

Acknowledgments We thank Amos Beimel for valuable comments on a previous version of this paper. The first author was 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.

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. P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*. Springer-Verlag, Berlin, 1997.
5. 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.
6. 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.
7. 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.
8. M. Curtis. *Abstract Linear Algebra*. Springer-Verlag, 1990.
9. J. L. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inf. Theory*, it-33(3):428–431, 1987.
10. 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.
11. J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, New York, 1999.
12. O. Goldreich. *Foundations of Cryptography, Voume II Basic Applications*. Cambridge University Press, 2004.
13. 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.

14. 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.
15. M. Karchmer and A. Wigderson. On span programs. In *Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111, 1993.
16. Y. Lindell and B. Pinkas. A proof of yao's protocol for secure two-party computation. *eprint archive*, 2004.
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*, pages 522–541, 2006.
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 More Protocols

A.1 Matrix Inversion

Bob holds an encrypted matrix $\mathsf{Enc}(M)$ such that $M \in \mathbb{F}^{n\times n}$ is guaranteed to be invertible. Alice holds the private decryption key. Based on the shared field inversion protocol from Bar-Ilan and Beaver [1] we design a protocol for computing $\mathsf{Enc}(M^{-1})$.

Protocol MATRIX INVERT

Input: $\mathsf{Enc}(M)$ where $M \in \mathbb{F}^{n\times n}$.
Output: $\mathsf{Enc}(M^{-1})$

1. Bob picks an $n \times n$ random non-singular matrix Q.
2. Bob computes the encrypted matrix $\mathsf{Enc}(QM)$ by multiplying $\mathsf{Enc}(M)$ from the left by the matrix Q, and sends $\mathsf{Enc}(QM)$ to Alice.
3. Alice decrypts $\mathsf{Enc}(QM)$ and compute $(QM)^{-1} = M^{-1}Q^{-1}$. Alice encrypts $M^{-1}Q^{-1}$ and sends Bob $\mathsf{Enc}(M^{-1}Q^{-1})$.
4. Bob computes $\mathsf{Enc}(M^{-1}) = \mathsf{Enc}(M^{-1}Q^{-1})Q$.
5. Bob locally outputs $\mathsf{Enc}(M^{-1})$.

It is easy to see that Alice gets a random non-singular matrix QM, and thus learns no information in the protocol. Since Bob only learns encrypted values from the protocol, he gets no information on the value of M.

A.2 Minimal Polynomial

We demonstrate an algorithm from [9] 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

$$m_{\mathbf{a}} \cdot (a_1 + a_2 x + \ldots + a_{2n} x^{2n-1}) - q(x) \cdot x^{2n} = r(x), \tag{1}$$

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 $r(x) = r_i(x)$.

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

Lemma 11 ([19]). *Define $M = AB^\top$. Then $V_I \neq \{\mathbf{0}\}$ if and only if the matrix M is not of full row 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 $V_B^\top$, where $n'_b \stackrel{\text{def}}{=} n - n_b$.
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 $\mathsf{Enc}(A)$ and the public key.
2. Bob locally computes $\mathsf{Enc}(M)$, 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 $\mathsf{Enc}(M)$. Denote by $\mathsf{Enc}(r)$ the output of the protocol held by Bob.
4. Alice and Bob execute protocol EQUAL on $\min n_a, n'_b$ and $\mathsf{Enc}(r)$. 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]. However, the round complexity of this protocol, which is $O(\log n)$ is substantially better 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 $n_a \times n$ 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 $O(n^2 k \log n)$ and round complexity $\Omega(n^{0.275})$. We show a protocol which improves the round complexity to $O(\log n)$ while keeping the communication complexity roughly $O(n^2)$.

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

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

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