Communication Efficient Secure Linear Algebra

Kobbi Nissim* and Enav Weinreb**

Ben Gurion University, Beer-Sheva 84105, Israel.
{kobbi, weinrebe}@cs.bgu.ac.il

Abstract. We present communication efficient secure protocols for a variety of linear algebra problems. Our main building block is a protocol for computing Gaussian Elimination on encrypted data. As input for this protocol, Bob holds a $k \times k$ matrix M, encrypted with Alice's key. At the end of the protocol run, Bob holds an encryption of an upper-triangular matrix M' such that the number of nonzero elements on the diagonal equals the rank of M. The communication complexity of our protocol is roughly $O(k^2)$.
Building on Oblivious Gaussian elimination, we present secure protocols for several problems: deciding the intersection of linear and affine subspaces, picking a random vector from the intersection, and obliviously solving a set of linear equations. Our protocols match known (insecure) communication complexity lower bounds, and improve the communication complexity of both Yao's garbled circuits and that of specific previously published protocols.

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 communication efficient secure protocols for various linear algebraic problems.

The basic linear algebraic problem we focus on is linear subspace intersection. Alice and Bob hold subspaces of F^k for some finite field F, each subspace representing a set of linear equations held by the players. They wish to study different properties of the intersection of their input subspaces, without leaking any information not revealed by the result of the computation. The first variant is a search problem where Alice and Bob wish to *compute* the intersection, while in the second variant they only wish to *decide* whether the intersection is the trivial zero subspace. We also consider the problems of picking a random vector from the intersection, and of affine subspaces intersection.

* Research partially Supported by the Frankel Center for Computer Science.

** Partially supported by a Kreitman Foundation Fellowship and by the Frankel Center for Computer Science.

Cramer and Damgård introduced secure protocols for solving various linear algebraic problems [5]. Their work was done in the information theoretical setup, with the main goal of reducing the round complexity to a constant. The communication complexity of their protocols is $\Omega(k^3)$ while the size of the inputs is merely $O(k^2)$. Another approach for designing secure protocols for these linear algebraic problems is to apply the garbled circuit method of Yao [18]. 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, the communication complexity of the resulting protocol is essentially the circuit complexity of the latter. The best known upper bound for this problem is $O(k^\omega)$ [6] for $\omega \cong 2.38$, which is still larger than the input size.

We introduce a protocol for the subspace intersection problem[1] with communication complexity of roughly $O(k^2)$. Even for *insecure* computation, it is shown in [3] that the deterministic communication complexity of the problem is $\Omega(k^2)$. This result agrees with ours up to a polylogarithmic factor. 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. Our protocol gives rise to communication efficient secure protocols for problems reducible to linear algebra, e.g., perfect matching, and functions with low span program complexity [11]. Unlike the protocol of [5] and [18], our protocols are not constant round. However, using a combination of our techniques and the general techniques of [18], we achieve a round complexity of $O(k^{1-\frac{1}{\omega-1}}) \approx O(k^{0.275})$.

Techniques We use public key homomorphic encryption to put the players in the following situation: Alice holds a private key of a public key homomorphic encryption scheme, while Bob holds an encrypted matrix. Bob wants to perform computations on his matrix, such as checking if it has full rank, without leaking any information on it. Specifically, we show how Bob can use Alice's help to securely perform the Gaussian Elimination procedure on the matrix. As the current state of art in homomorphic encryption does not allow an encryption scheme with both homomorphic addition and multiplication, we use standard techniques to make Alice multiply encrypted elements for Bob.

The Gaussian Elimination procedure requires Bob to find a row with a non-zero element in its leftmost coordinate for the elimination of the leftmost column. As the matrix is encrypted, Bob cannot find such a row on his own. We use Alice's help and randomness to overcome this problem. Alice's help in this case may be interpreted as performing an 'if' statement in Bob's code, although the computation is oblivious. To save communication, we use the paradigm of *lazy evaluation*, in which Bob uses Alice as a storage device as well. Instead of instantly sending Bob the results of the computations, Alice keeps an image of Bob's memory, and sends him only the information he needs for the next round of computation. To conclude, Bob uses Alice for calculations, for flow control, and as a storage device, without enclosing any of his data to her.

[1] All the bounds mentioned in the introduction are for the case where Alice and Bob hold subspaces of dimension $k/2$. Exact bounds are presented later in the text.

The round complexity of our basic protocol is $O(k)$. We use a combination of the garbled circuit method of Yao, and the techniques described above, to reduce the number of rounds to $O(k^{0.275})$. We use randomness to ensure both correctness and security for our protocols. In the context of finding a random vector in the intersection of the input subspaces, we use a technique of adding random constraints to reduce the solution set into only one solution. This is inspired by the "hashing paradigm" [17] that was employed, e.g., by Bellare et al. [2] for uniformly picking an NP witness. Another use of randomness is achieved via the next basic linear algebraic claim. Take a rank r matrix and multiply it from the left and from the right by random full rank matrices. Then, with constant probability, the top-left $r \times r$ sub-matrix of the resulting matrix is of rank r. This fact enables us to reduce problems on non-square matrices to problems on their square counterparts.

Organization. We start in Section 2 with preliminaries and notation. In Section 3 we present secure protocols for computing subspaces intersection and for deciding if the intersection is trivial. Later, in Section 4, we design our main building block, the `Oblivious Gaussian Elimination` protocol. In Section 5 we show how to securely pick a random vector from the intersection, and finally, in Section 6, we design secure protocols for analogous problems on affine subspaces.

2 preliminaries

Notation. Let F be a finite field. We denote by $\boldsymbol{v}$ a row vector in the vector space F^k where $k > 0$ and $\mathbf{0}$ denotes the row vector whose entries are all zero. For a matrix M with entries from F, we denote by $\boldsymbol{M_i}$ the ith row of M. For an encryption scheme, we let λ be its security parameter. W.l.o.g, we assume that the result of encrypting a field element is of length $O(\lambda)$. We use $\mathbf{neg}(k)$ to denote a function that is negligible in k, i.e. $\mathbf{neg}(k) = k^{-\omega(1)}$.

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 $\mathsf{Enc}(m_1)$ and $\mathsf{Enc}(m_2)$, we can efficiently compute $\mathsf{Enc}(m_1 + m_2)$; and (ii) Given an encryption $\mathsf{Enc}(m)$ and $c \in F$, we can efficiently compute $\mathsf{Enc}(cm)$.

Several constructions of homomorphic encryption schemes are known, each with its particular properties (see e.g. [15, 10, 8, 14, 16, 13, 7, 1]). These have been in use in a variety of cryptographic protocols. Over $F = GF(2)$, the encryption scheme of Goldwasser and Micali [10], based on quadratic residuosity, is sufficient for our constructions.

For a vector $\boldsymbol{v} \in F^n$, we denote by $\mathsf{Enc}(\boldsymbol{v})$ the coordinate-wise encryption of $\boldsymbol{v}$. That is, if $\boldsymbol{v} = \langle a_1, \ldots, a_n \rangle$ where $a_1, \ldots, a_n \in F$, then $\mathsf{Enc}(\boldsymbol{v}) = \langle \mathsf{Enc}(a_1), \ldots, \mathsf{Enc}(a_n) \rangle$. Similarly, for a matrix $M \in F^{m \times n}$, we denote by $\mathsf{Enc}(M)$ the $m \times n$ matrix such that $\mathsf{Enc}(M)[i, j] = \mathsf{Enc}(M[i, j])$. 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 $\mathsf{Enc}(\boldsymbol{v_1})$ and $\mathsf{Enc}(\boldsymbol{v_2})$, we can efficiently compute $\mathsf{Enc}(\boldsymbol{v_1}+\boldsymbol{v_2})$, and similarly with matrices. (ii) Given an encryption of a vector $\mathsf{Enc}(\boldsymbol{v})$ and a constant $c \in F$, we can efficiently compute $\mathsf{Enc}(c\boldsymbol{v})$. (iii) Given an encryption of a matrix $\mathsf{Enc}(M)$ and a matrix M' of the appropriate dimensions, we can efficiently compute $\mathsf{Enc}(MM')$ and $\mathsf{Enc}(M'M)$.

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.

Remark 1. Our protocols achieve information theoretic security for Bob while Alice's security relies on that of the underlying encryption scheme.

Basic Building Blocks. In our protocols Bob holds data encrypted by a public key homomorphic encryption scheme, while Alice holds the private decryption key. Bob uses Alice's help to perform different calculations, without enclosing his data to her. As a simple example of a protocol where Bob uses Alice's help, assume Bob holds $\mathsf{Enc}(a)$ and $\mathsf{Enc}(b)$ and needs to compute $\mathsf{Enc}(ab)$. Let `Multiply` be the following (folklore) solution: (i) Bob chooses random masks $r_a, r_b \in_R F$ and sends $\mathsf{Enc}(a+r_a)$ and $\mathsf{Enc}(b+r_b)$ to Alice; (ii) Alice deciphers these messages and returns $\mathsf{Enc}((a+r_a)(b+r_b))$; (iii) Given $\mathsf{Enc}((a+r_a)(b+r_b))$, Bob computes $\mathsf{Enc}(ab) = \mathsf{Enc}((a+r_a)(b+r_b) - r_b a - r_a b - r_a r_b)$. It is easy to see that neither Alice nor Bob gain any information about a and b (and ab). The communication complexity of this protocol is $O(\lambda)$. This protocol is easily generalized to vectors of length k, the resulting protocol `Vector Multiply` is of communication complexity $O(\lambda k)$.

Linear Algebra. We need the following simple linear algebraic claim.

Claim 1 ([4]) *Let $k_a < k_b$ be positive integers, F be a finite field, and M be a $k_a \times k_b$ matrix over F. Suppose $r \leq \mathrm{rank}(M)$ and let T_A and T_B be $k_a \times k_a$ and $k_b \times k_b$ randomly chosen full rank matrices over F. Let $M' = T_A M T_B$, and denote the top-left $r \times r$ sub-matrix of M' by N'. Then with constant probability* $\mathrm{rank}(N') = r$.

3 Linear Subspace Intersection

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

The first variant of the problem is of *computing* the subspace V_I itself. The second is of *deciding* whether V_I is the trivial zero subspace. 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 easier as the players gain less information from its output.

A common step in solving both variants is the following reduction of computing V_I into solving a homogeneous linear system. Let $V_B^{\perp}$ be the perpendicular subspace[2] of V_B. Define $k'_b = k - k_b$ and let $B^{\perp}$ be a $k \times k'_b$ matrix whose columns span exactly the subspace $V_B^{\perp}$. Finally define the $k_a \times k'_b$ matrix $M = AB^{\perp}$.

Claim 2 *Let $v \in F^{k_a}$. Then $vA \in V_I$ if and only if $vM = \mathbf{0}$.*

Proof. If $vA \in V_I$ then $vA \in V_B$, and thus $vM = (vA)B^{\perp} = \mathbf{0}$. For the other direction, if $vM = \mathbf{0}$, then $(vA)B^{\perp} = \mathbf{0}$, and thus $vA \in V_B$. As vA is a linear combination of the rows of A, we get that $vA \in V_A$, hence $vA \in V_A \cap V_B = V_I$.

3.1 Computing the Intersection

Protocol `Intersection Computation` securely computes V_I in one round of communication. The communication complexity of the protocol is $O(\lambda k_a k)$. The protocol uses homomorphic encryption to enable a multiplication of an encrypted matrix by an open matrix without the knowledge of the private decryption key.

Protocol `Intersection Computation`

Input: Alice (resp. Bob) holds a $k_a \times k$ (resp. $k_b \times k$) matrix A (resp. B) over a finite field F representing a subspace $V_A \subseteq F^k$ (resp. $V_B \subseteq F^k$).
Output: Alice holds a matrix representing $V_I = V_A \cap V_B$.

1. Bob locally computes a $k \times k'_b$ matrix $B^{\perp}$ that represents the subspace $V_B^{\perp}$.
2. Alice generates keys for a homomorphic public key encryption system, and sends Bob $\mathsf{Enc}(A)$ and the public key.
3. Bob randomly chooses a $k'_b \times k'_b$ full rank matrix T_B, locally computes $\mathsf{Enc}(M)$, where $M \stackrel{\text{def}}{=} AB^{\perp}T_B$, and sends $\mathsf{Enc}(M)$ to Alice.
4. Alice decrypts M and computes the subspace $K = ker(M^T)$, that is, $K = \{v : vM = \mathbf{0}\}$.
5. Alice computes the subspace $V_I = \{vA : v \in K\}$.

Correctness and Security. The correctness of the `Intersection Computation` protocol derives[3] from Claim 2. Alice's security immediately follows from the fact she only sends information encrypted in a semantically-secure encryption scheme. To prove Bob's security, we show a simulator for Alice's view. The simulator and its related security proof appear in Appendix A.

[2] Recall that $V_B^{\perp} \stackrel{\text{def}}{=} \{u : \langle u, v\rangle = 0 \text{ for all } v \in V_B\}$, and is of dimension $k'_b \stackrel{\text{def}}{=} k - k_b$.
[3] Note that although in the protocol $M = AB^{\perp}T_B$, Claim 2 still applies, as the columns of $B^{\perp}$ and the columns of $B^{\perp}T_B$ both span the subspace $V_B^{\perp}$.

3.2 Deciding Whether the Intersection is Trivial

Let V_A, V_B be as above and V_I their intersection[4]. By Claim 2, there is a non trivial intersection between V_A and V_B if and only if there exist a non-zero vector $v \in F^{k_a}$ such that $vAB^{\perp} = \mathbf{0}$. This happens only if $rank(AB^{\perp}) < k_a$, that is, if $AB^{\perp}$ is not a full rank matrix. Hence, computing $AB^{\perp}$ seems useful also in deciding whether $V_I = \{\mathbf{0}\}$. However, unlike in the `Intersection Computation` protocol ,we cannot have Bob sending $AB^{\perp}$ nor any information regarding its dimension to Alice. Such information would compromise the protocol privacy.

As in the `Intersection Computation` protocol, Alice sends an encryption of her input and a public key to Bob, who computes an encryption of $AB^{\perp}$. Here, we are only interested in whether $AB^{\perp}$ is of full rank. Our main building block is a private protocol that transforms the encryption of $AB^{\perp}$ into an encryption of an upper triangular matrix[5]. In particular, there is a 0 on the main diagonal of the resulting matrix if and only if $AB^{\perp}$ is of full rank.

Definition 1 (Oblivious Gaussian Elimination Problem). *Input: Alice holds a private key of a public key homomorphic encryption scheme over a finite field F. Bob holds a $k_a \times k_b$ matrix M encrypted by Alice's public key, where $k_a \leq k_b$.*
Output: Suppose $\operatorname{rank}(M) = r$*. In the end of the protocol Bob holds an encryption of a $k_a \times k_b$ matrix M'. With probability $1 - \mathbf{neg}(k)$, the matrix M' is upper triangular and: (i) There are* at most *r non-zero elements on the main diagonal of M'. (ii) With constant probability there are* exactly *r non-zero elements on the main diagonal of M'.*

The following theorem summarizes the properties of our protocol for solving the Oblivious Gaussian Elimination Problem. The protocol is described in Section 4.

Theorem 3. *There is a secure protocol with communication complexity $\tilde{O}(\lambda k_a k_b)$ and round complexity $k_a^{0.275}$, that solves the Oblivious Gaussian Elimination Problem.*

Having a secure protocol for solving the problem above, deciding the intersection of the input subspaces is done in two steps. A procedure for deciding the intersection with *one sided constant error probability* is depicted in Protocol `Intersection Decision` below. To get a protocol with negligible error probability, Alice and Bob run protocol `Intersection Decision` for $m = \omega(\log k)$ times. Alice and Bob then obliviously compute the logical OR of all the executions. The correctness of the protocol is straight forward assuming the correctness of `Oblivious Gaussian Elimination`.

Theorem 4. *Protocol* `Intersection Decision` *is a secure protocol for the subspace intersection decision problem. The communication complexity of the protocol is $\tilde{O}(\lambda k_a k)$ and the round complexity is $\tilde{O}(k_a^{0.275})$.*

[4] W.l.o.g., we assume that $k_a + k_b \leq k$, as otherwise V_A and V_B always have a non-trivial intersection.

[5] For non-square matrices upper triangular means $i > j \Rightarrow M[i, j] = 0$.

Protocol `Intersection Decision`

INPUT: Alice (resp. Bob) holds a $k_a \times k$ (resp. $k_b \times k$) matrix A (resp. B) over a finite field F representing a subspace $V_A \subseteq F^k$ (resp. $V_B \subseteq F^k$). Let $B^\perp$ be a $k \times k'_b$ matrix that represents the subspace $V_B^\perp$.
OUTPUT: If V_I is not the trivial zero subspace, Bob outputs $\mathsf{Enc}(0)$ with probability 1. Else, with constant probability, Bob outputs $\mathsf{Enc}(r)$ for some non-zero $r \in F$.

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 $M \stackrel{\text{def}}{=} AB^\perp$. Note that M is a $k_a \times k'_b$ matrix.
3. Alice and Bob run protocol `Oblivious Gaussian Elimination` on $\mathsf{Enc}(M)$. Denote by M' the resulting $k_a \times k'_b$ matrix that Bob holds at the end of the protocol execution.
4. Bob and Alice use the `Multiply` protocol such that Bob eventually locally outputs $\mathsf{Enc}(r)$ where $r \stackrel{\text{def}}{=} \prod_{i=1}^{k_a} M'[i,i]$.

4 Oblivious Gaussian Elimination

In this section we introduce a protocol for the Oblivious Gaussian Elimination problem (See Definition 1), with parameters matching Theorem 3. We first define the Oblivious Gaussian Elimination problem for square matrices. Then we design a protocol for this special case, and finally we reduce the problem on general matrices to the problem on their square counterparts.

Definition 2 (Oblivious Gaussian Elimination Problem for Square Matrices). *Input: Alice holds a private key of a public key homomorphic encryption scheme over a finite field F. Bob holds a $k \times k$ matrix M encrypted by Alice's public key.*
Output: In the end of the protocol Bob holds an encryption of a $k \times k$ matrix M'. With probability $1 - \mathbf{neg}(k)$, the matrix M' is upper triangular and: (i) If M is full rank then with probability $1 - \mathbf{neg}(k)$ all the diagonal entries of M' are non-zero. (ii) If M is not full rank then there is a 0 entry on the diagonal of M'.

There are two differences between this definition and Definition 1. Here, the diagonal of the resulting matrix M' does not reflect the exact rank of M, but rather only whether M is full rank or not. On the other hand, here we require very high success probability, while in Definition 1, the success probability is constant.

4.1 Gaussian Elimination

The Gaussian Elimination algorithm is a well known method for transforming a matrix into a triangular form, while keeping its rank. Consider the following 'textbook' Gaussian Elimination procedure. To simplify the presentation, we assume the underlying field is the unique finite field with two elements, that is, $F = \mathrm{GF}(2)$. The generalization of all our protocols to other finite fields of fixed size is straight forward[6].

Input: A $k \times k$ matrix M over $F = \mathrm{GF}(2)$:
(1) Find a row $\boldsymbol{M_j}$, such that the leftmost coordinate in $\boldsymbol{M_j}$ is 1, that is, $M[j,1] = 1$.
(2) For every $i \neq j$, if $M[i,1] = 1$, add $\boldsymbol{M_j}$ to $\boldsymbol{M_i}$, so that the result is 0 in the leftmost coordinate.
(3) Swap the first and the jth rows of M.
(4) If $k > 1$, perform steps (1) – (4) on the lower-right $(k-1)\times(k-1)$ sub-matrix of M.

Consider obliviously running Gaussian Elimination on an encrypted $k \times k$ matrix M over GF(2) held by Bob. In step (1) Bob faces the problem of choosing the row $\boldsymbol{M_j}$ as he cannot distinguish a 0 entry from a 1 entry, and letting Bob (or Alice) learn j may compromise privacy. To go around this problem, we let Alice and Bob eliminate the leftmost column using several rows. For each of the rows they use, if the leftmost entry is 1 then we get the desired elimination. On the other hand, if the leftmost entry is 0, the matrix is not changed at all. We use randomization to guarantee that with high probability, the leftmost entry in at least one of the rows used is 1.

4.2 Column Elimination

Protocol `Basic Column Elimination` securely eliminates the leftmost column of a matrix using its jth row.

In the second step of the protocol Bob uses Alice's assistance in computing $\mathsf{Enc}(M[i,1] \cdot M[j,1] \cdot \boldsymbol{M_j})$. Note that if $M[j,1] = 0$ then the result of step 2 is an encryption of $\mathbf{0}$. Therefore, if $M[j,1] = 0$, Bob adds encryptions of $\mathbf{0}$ to every row, and thus $M' = M$. If $M[j,1] = 1$, then Bob adds $\boldsymbol{M_j}$ exactly to the rows $\boldsymbol{M_i}$ with $M[i,1] = 1$, as in the Gaussian Elimination procedure.

The communication complexity of the protocol is $O(\lambda k^2)$, as we run the `Vector Multiply` protocol for $O(k)$ times. However, in all iterations Bob multiplies an encryption of $\boldsymbol{M_j}$. Hence, it is enough for Bob to randomly choose $\boldsymbol{r}_{M_j}$ and send Alice $\mathsf{Enc}(\boldsymbol{M_j} + \boldsymbol{r}_{M_j})$ only once. We get that the communication complexity from Bob to Alice is reduced to $O(\lambda k)$ while the communication from Alice to Bob remains $O(\lambda k^2)$. The communication from Alice to Bob will later be reduced as well.

[6] To generalize our protocols to a field F, use the a sub-protocol for the following problem: Bob holds $\mathsf{Enc}(a)$ for $a \in F$, and Alice holds the private decryption key. In the end of the protocol Bob should hold $\mathsf{Enc}(a^{-1})$ if $a \neq 0$ and $\mathsf{Enc}(0)$ if $a = 0$. If $|F|$ is large, this can be done using the garbled circuit method of Yao, without affecting the asymptotic complexity of the protocol.

Protocol `Basic Column Elimination`

Input: As in Definition 2

Output: At the end of the protocol Bob holds an encryption of a matrix M' with the following properties: If $M[j,1] = 0$ then $M' = M$. Otherwise, $\boldsymbol{M_i'} = \boldsymbol{M_i}$ for every $i \leq j$, and for $i > j$ (i) if $M[i,1] = 0$ then $\boldsymbol{M_i'} = \boldsymbol{M_i}$, and (ii) if $M[i,1] = 1$ then $\boldsymbol{M_i'} = \boldsymbol{M_i} + \boldsymbol{M_j}$.

For every $j < i \leq k$ do the following:

1. Alice and Bob run protocol `Multiply`, with Bob's inputs being $\mathsf{Enc}(M[j,1])$ and $\mathsf{Enc}(M[i,1])$. As a result, Bob holds $\mathsf{Enc}(M[i,1] \cdot M[j,1])$.
2. Alice and Bob run protocol `Vector Multiply`, with Bob's inputs being $\mathsf{Enc}(M[i,1]M[j,1])$ and $\mathsf{Enc}(\boldsymbol{M_j})$. As a result, Bob holds $\mathsf{Enc}(M[i,1] \cdot M[j,1] \cdot \boldsymbol{M_j})$.
3. Bob locally computes $\mathsf{Enc}(\boldsymbol{M_i'}) = \mathsf{Enc}(\boldsymbol{M_i} + M[i,1] \cdot M[j,1] \cdot \boldsymbol{M_j})$.

Oblivious Column Elimination. As we noted above, if the leftmost coordinate of the eliminating row M_j is 0, running `Basic Column Elimination` does not advance the elimination process. Protocol `Oblivious Column Elimination` below uses the upper m rows of M to eliminate the leftmost column. The process is successful if any of these m rows contains 1 in the leftmost coordinate, and the parameter m is chosen such that this happens with high probability. Let $i \in \{1, \ldots, m\}$ be the minimal row index such that $M[i,1]$ is non-zero. Note that (i) the column elimination process using any of the $i-1$ upper rows does not change the matrix; (ii) the ith row $\boldsymbol{M_i}$ eliminates the leftmost column of M; (iii) the column elimination process using rows $i+1$ to m does not effect M anymore. Denote by M' the resulting matrix.

Next, Alice and Bob swap the ith and first rows of M'. However, as the process is run obliviously, Bob does not know what i is. For that, we slightly modify Gaussian Elimination. Note that if the elimination was successful, the ith row in M' is the only row that does not have 0 in the leftmost coordinate. Bob adds the top m rows in M' into the top row of the matrix: $\boldsymbol{M_1'} = \sum_{j=1}^{m} \boldsymbol{M_j'}$. The result is a leftmost 1 entry in at most two rows of M': the first and ith.

To eliminate the non-zero entry in M' we run `Basic Column Elimination` once more using the top row. If M is a full-rank matrix, and there is a 1 entry in the leftmost column of at least one of the top m rows of M, then in the resulting M' satisfies: (i) $M'[1,1] = 1$ and (ii) $M'[j,1] = 0$ for $2 \leq j \leq k$.

We note that Alice and Bob may agree on T by choosing a seed to a pseudorandom generator. Hence, the communication complexity of this protocol is m times that of protocol `Basic Column Elimination`. It is simple to verify that neither Alice nor Bob gain any information about M. Furthermore, $\mathrm{rank}(M') = \mathrm{rank}(M)$ as M is transformed into M' via a sequence of elementary matrix operations. Finally, the following claim shows that the elimination is successful with high probability.

Protocol `Oblivious Column Elimination`

INPUT: As In Definition 2
OUTPUT: At the end of the execution Bob holds an encryption $\mathsf{Enc}(M')$ of a $k \times k$ matrix such that $\mathrm{rank}(M') = \mathrm{rank}(M)$. Furthermore, if the leftmost column of M is non-zero then with high probability $M'[1, 1] = 1$ and $M'[i, 1] = 0$ for $2 \leq i \leq k$.

1. Alice and Bob agree on a random non-singular matrix $T \in_R \mathrm{GF}(2)^{k \times k}$. Bob uses the homomorphic properties of the encryption scheme to compute $\mathsf{Enc}(M')$ where $M' = TM$.
2. For every $1 \leq i \leq m(k)$, Alice and Bob run protocol `Basic Column Elimination` with Bob's inputs being $\mathsf{Enc}(M')$ and i.
3. Bob locally assigns $\boldsymbol{M'_1} = \sum_{j=1}^{m} \boldsymbol{M'_j}$ by adding the m upper encrypted rows of M'.
4. Alice and Bob run protocol `Basic Column Elimination` protocol with Bob's inputs being $\mathsf{Enc}(M')$and 1.

Claim 5 *Let M be a $k \times k$ matrix and T be a random $k \times k$ matrix of full rank, both over* $\mathrm{GF}(2)$ *and let $m = \omega(\log k)$. If the leftmost column of M is non-zero, then with probability $1 - \mathbf{neg}(k)$, at least one entry in the leftmost column of the top m rows of the matrix TM is non-zero.*

Proof. Denote the leftmost non-zero column of M by c, the m top rows of T by $T_1, \ldots, T_m$, and note that $TM[i, 1] = T_i c$. If T was a random matrix, that is $T_1, \ldots, T_m$ were independently randomly chosen vectors, then for every $i \in [m]$ the probability that $TM[i, 1] = 0$ would be exactly $1/2$. Hence the probability that $TM[i, 1] \neq 0$ for at least one value of i would be $1 - \mathbf{neg}(k)$. As a random matrix has full rank with constant probability [4], it follows that for a random non-singular matrix the probability that such an event occurs is also negligible.

4.3 Oblivious Gaussian Elimination

We now have the ingredients to present our `Oblivious Gaussian Elimination` protocol. On a matrix $M \in \mathrm{GF}(2)^{k \times k}$, the protocol first applies `Oblivious Column Elimination`, to eliminate the leftmost column, and then recurses on the lower-right $(k-1) \times (k-1)$ sub-matrix. For clarity of presentation, we first construct a 'naive' protocol, of communication complexity $\tilde{O}(\lambda k^3)$ and round complexity $\tilde{O}(k)$, and then discuss how to reduce the communication complexity to $\tilde{O}(\lambda k^2)$ and the round complexity to $\tilde{O}(k^{0.275})$.

As before, it is easy to verify that the parties gain no information about the matrix M. The following claim asserts the correctness of the protocol.

Claim 6 *At the end of the execution of the* `Oblivious Gaussian Elimination` *Protocol, Bob holds an encryption of an upper triangular matrix M' as required by Definition 2.*

Protocol `Oblivious Gaussian Elimination (for Square matrices)`

Input and Output: As in Definition 2.

1. Alice and Bob run protocol `Oblivious Column Elimination` on M. Let Bob's output be $\mathsf{Enc}(M')$.
2. Alice and Bob recursively run `Oblivious Gaussian Elimination`, on the lower-right $(k-1) \times (k-1)$ submatrix of M'. Let Bob's output be $\mathsf{Enc}(M'')$.
3. Bob locally outputs $\begin{pmatrix} \mathsf{Enc}(M'[1,1]), \mathsf{Enc}(M'[1,2]), \ldots, \mathsf{Enc}(M'[1,k]) & \\ 0 & \\ \vdots & \mathsf{Enc}(M'') \\ 0 & \end{pmatrix}$.

4.4 Reducing Communication Complexity via Lazy Evaluation

Informally, in the above protocol, Bob uses Alice as a 'calculator' for performing multiplications of encrypted field elements. The communication complexity of protocol `Oblivious Gaussian Elimination` is $O(\lambda m k^3) = \tilde{O}(\lambda k^3)$, by picking $m = \text{polylog}(k)$. We now show that Bob can also use Alice as a storage device, and by this to reduce the communication complexity by a factor of k. Note that in each round of the protocol, Bob sends to Alice one row and one column of $\mathsf{Enc}(M)$, (masked with random vectors). In return, Alice sends $O(k)$ vectors that Bob adds to the matrix M. Each of these vectors is of size k, resulting in $\tilde{O}(\lambda k^2)$ communication per round.

However, as Bob is not using all the matrix entries in the following round, we can have Alice send him only the single row and column that are needed for completing the next round. We make a simple modification to the protocol, and let Alice maintain a matrix L, where $L[i,j]$ equals the sum of elements Bob needs to add to the entry $M[i,j]$. Alice would then send $\mathsf{Enc}(L[i,j])$ just before the ith row, or the jth column is needed for Bob. Moreover, whenever Bob multiplies his matrix by a full-rank matrix, Alice needs to multiply L by the same matrix, and this is the reason why Alice and Bob choose the random matrices together. This reduces the communication complexity of each round to $\tilde{O}(\lambda k)$, and hence the communication of the entire protocol to $\tilde{O}(\lambda k^2)$.

4.5 Reducing the Round Complexity

The round complexity of our protocol is linear in the matrix dimension, that is $\Omega(k)$. In this section we show how to reduce the round complexity to sub-linear while preserving the low communication complexity. The idea is to combine our communication efficient protocol with the general purpose round efficient protocol of Yao [18]. This idea was used before, in, e.g., [12].

The protocol is still based on Gaussian Elimination, only that here we eliminate a number of columns together in the same round. Let $\ell = k^\epsilon$ where $0 < \epsilon < 1$ is a parameter to be specified later. The first modification we make to `Oblivious`

`Gaussian Elimination` is that Bob multiplies the matrix M by full rank matrices from *both* sides and not only from the left. By Claim 1, if M is a full rank matrix then with constant probability, the top-left $\ell \times \ell$ sub-matrix of M, denoted by N, is of full rank as well.

In this stage Alice and Bob execute a secure sub-protocol base on [18], such that at the end of the protocol Bob holds an encryption of N^{-1} if N is invertible, and an encryption of the 0 matrix if N is not full rank. Following this stage, the protocol is very similar to the original `Oblivious Gaussian Elimination` protocol. We divide the k rows of M into k/ℓ blocks of ℓ rows each. Denote the block of the top ℓ rows of M by K. The notations are depicted in Figure 1. For every other ℓ-rows block L, Alice and Bob perform the following:

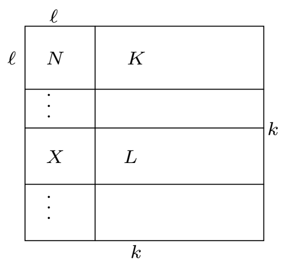

Fig. 1. Notations for the round efficient protocol.

Denote the $\ell \times \ell$ left sub-matrix of L by X. Bob uses the help of Alice to compute $L \leftarrow L - XN^{-1}K$. If N is not invertible, then Bob has an encryption of the 0 matrix as N^{-1}, and thus the matrix is left unchanged. Otherwise, this procedure zeros the ℓ leftmost columns of L. As this process succeeds with constant probability, we repeat it a polylogarithmic number of times. Using basic techniques Alice and Bob can make sure that after finding a non-invertible N, no changes are done to the matrix till working on the next block of columns.

We first analyze the communication complexity of the protocol *excluding* the sub-protocol for computing N^{-1}. The communication complexity from Bob to Alice in each round is $\tilde{O}(\lambda \ell k)$ as Bob sends a masking of the top ℓ rows and the leftmost ℓ columns to Alice. Therefore, as there are k/ℓ rounds, the overall communication from Bob to Alice is $\tilde{O}(\lambda k^2)$. The communication complexity from Alice to Bob in each round is large as she needs to send $O(\lambda \ell k)$ bits for every ℓ-rows block. However, as before, we use lazy evaluation. Alice only sends Bob the $O(\lambda \ell k)$ bits he needs for the next block of columns, and keeps a matrix with the changes needed to be made to all the other entries in the matrix of Bob. This makes the overall communication complexity from Alice to Bob $O(\lambda k^2)$, excluding the protocol for computing N^{-1}.

We now analyze the communication complexity of the secure sub-protocol for computing N^{-1}. The communication complexity of securely inverting a matrix using Yao's garbled circuit method is related to the circuit complexity of matrix inversion. As matrix inversion is reducible to matrix multiplication, this can be done using a circuit of size $O(\ell^\omega)$, where the best known upper bound [6] for ω is approximately 2.38.

Therefore, the communication complexity of the sub-protocol is $O(\lambda\ell^\omega)$. As it is executed $\tilde{O}(k/\ell)$ times through the protocol we get that the overall complexity of executing the sub-protocol is:

$$(\lambda k/\ell)\ell^\omega = \lambda k\ell^{\omega-1} = kk^{\epsilon(\omega-1)} = \lambda k^{1+\epsilon\omega-\epsilon}.$$

To get a communication complexity of $\tilde{O}(\lambda k^2)$, we set the value of ϵ such that $1+\epsilon\omega-\epsilon = 2$, i.e., $\epsilon = 1/(\omega-1) = 1/1.38 \cong 0.725$. The round complexity of the protocol is $\tilde{O}(k^{1-\epsilon}) = \tilde{O}(k^{0.275})$. Choosing different values for ℓ, one gets a tradeoff between the communication complexity and the round complexity.

Theorem 7. *There is a protocol for the Oblivious Gaussian Elimination Problem for Square Matrices (See Definition 2) over* $\mathrm{GF}(2)$ *with communication complexity* $\tilde{O}(\lambda k^2)$ *and round complexity* $\tilde{O}(k^{0.275})$*.*

4.6 Handling Non-Square Matrices

Protocol `Oblivious Gaussian Elimination` as described above works with very high probability for square matrices. We now show how to generalize the protocol to non-square matrices using a reduction. On a non-square matrix M of dimensions $k_a \times k_b$, Bob first randomly chooses a $k_a \times k_a$ full-rank matrix T_A and a $k_b \times k_b$ full-rank matrix T_B and computes $M^* = T_A M T_B$. Suppose w.l.o.g., that $k_a < k_b$ (otherwise perform the elimination on M^T). Alice and Bob execute the `Oblivious Gaussian Elimination` protocol on the top left $k_a \times k_a$ of M^*, denoted by N^*. The $k_b - k_a$ right columns of M are updated during the protocol, but are not eliminated. By Claim 2, if $\mathrm{rank}(M) \geq r$ then with constant probability, $\mathrm{rank}(N^*) = r$, and thus after executing the `Oblivious Gaussian Elimination` protocol on N^*, Bob holds an encrypted matrix $\mathsf{Enc}(M')$ such that M' is upper triangular, and with constant probability M' has exactly r non-zero entries on its diagonal. The communication complexity of the protocol is $\tilde{O}(\lambda k_a k_b)$ and the round complexity remains $\tilde{O}(k_a^{0.275})$. This completes the proof of Theorem 3.

5 Finding a Random Element in The Intersection

As in the previous sections, Alice holds a k_a-dimensional subspace $V_A \subseteq F^k$ represented by a $k_a \times k$ matrix A, while Bob holds holds a k_b-dimensional subspace $V_B \subseteq F^k$ represented by B. Alice and Bob wish to securely compute a uniformly distributed random vector in the subspace $V_A \cap V_B$. The main step in the design of our protocol is the addition of random linear constraints to the linear system created by the input subspaces, to reduce the number of solutions into only *one* random uniformly distributed solution.

We start with a definition of the Oblivious Linear Solve Problem.

Definition 3 (Oblivious Linear Solve Problem). *Input: Alice holds a private key of a public key homomorphic encryption scheme over a finite field F. Bob holds a $k_a \times k_b$ matrix M and a vector $v \in F^{k_b}$, encrypted by Alice's public key.*
Output: (i) If there exists a vector x such that $xM = v$, then with constant probability, Bob holds an encryption of an arbitrary *such vector, and with constant probability Bob holds an encryption of $\mathbf{0}$. (ii) Otherwise Bob holds an encryption of $\mathbf{0}$.*

In Appendix B we modify protocol `Oblivious Gaussian Elimination` to get protocol `Oblivious Linear Solve` whose properties are summarizes in the following claim.

Claim 8 *Protocol* `Oblivious Linear Solve` *is a secure protocol for the Oblivious Linear Solve Problem. The communication complexity of the protocol is $\tilde{O}(\lambda k_a k)$ and the round complexity is $\tilde{O}(k_a^{0.275})$.*

As in our previous protocols, Alice sends Bob $\mathsf{Enc}(A)$, and Bob computes $\mathsf{Enc}(M)$ for $M = AB^{\perp}$. By Claim 2, it is enough for Alice and Bob to find a random solution vector x to the linear system $xM = \mathbf{0}$. However, this linear system may have many solutions and picking an *arbitrary* solution is not satisfactory for our purpose. Therefore, we add random linear constrains to the linear system. That is, we concatenate a matrix R to M from the left, and a vector u to $\mathbf{0}$ and solve the linear system $x(R|M) = (u|\mathbf{0})$. We want to choose R and u so that with high probability, the system has a unique uniformly distributed solution.

The number of constraints needed to be added to the linear system depends on the dimension of the solution space of $xM = \mathbf{0}$. To this end, Alice and Bob first execute the `Oblivious Gaussian Elimination` protocol on M. By Theorem 3, with constant probability, the number of non-zero elements on the main diagonal of the result matrix M' equals the rank of M. Thus, Alice and Bob add a random linear constraint to R and u for every 0 on the main diagonal of M' and a trivial $x\mathbf{0} = 0$ constraint for every non-zero element on the diagonal of M'. Alice and Bob pick each random constraint by Alice sending the encryption of a random vector to Bob, who adds to it a second random vector. This way neither Alice nor Bob have information regarding the random constraints used. The technical method to add the constraints is depicted in Protocol `Random Intersection Vector`.

After adding the random constraints, Alice and Bob run the `Oblivious Linear Solve` protocol to get an encryption of a solution to the system $x(R|M) = (u|\mathbf{0})$. There are three possible cases: (i) The vector $(u|\mathbf{0})$ is not in the row span of the matrix $(R|M)$. In this case we get $x = \mathbf{0}$. (ii) There exists a non-zero vector x such that $x(R|M) = (u|\mathbf{0})$, but x is not unique. In this case it holds that $xM = \mathbf{0}$ but we do not argue that x is a random vector satisfying this requirement. (iii) There exist a unique non-zero vector x such that $x(R|M) = (u|\mathbf{0})$. In this case, by a symmetry argument, the vector x is a random vector satisfying $xM = 0$.

Protocol `Random Intersection Vector`

INPUT: Alice (resp. Bob) holds a $k_a \times k$ (resp. $k_b \times k$) matrix A (resp. B) over $GF(2)$ representing a subspace $V_A \subseteq \mathrm{GF}(2)^k$ (resp. $V_B \subseteq \mathrm{GF}(2)^k$).
OUTPUT: Alice locally outputs a random vector v satisfying $v \in V_A \cap V_B$.

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 $M \stackrel{\text{def}}{=} AB^{\perp}$.
3. For every $j \in \{1, \ldots, \ell\}$:
   (a) Alice and Bob execute Protocol `Oblivious Gaussian Elimination` on M. Let Bob's output be $\mathsf{Enc}(M')$.
   (b) For every $1 \leq i \leq k_a$, Alice and Bob choose a random vector w_i and set the ith column of the matrix R to be $c_i = (1 - M'[i,i])w_i$. That is, For every 0 on the diagonal of M', the vector c_i is a random vector, and for every 1 on the diagonal it is an encryption of $\mathbf{0}$.
   (c) Bob generates the vector $u \in \mathrm{GF}(2)^{k_a}$ in the following way. For $1 \leq i \leq k_a$, if $M'[i,i] = 1$ Bob assigns $u[i] = 0$, while if $M'[i,i] = 0$ Bob randomly assigns $u[i] \in_R \{0,1\}$. That is, Bob adds a random constraint for every 0 on the diagonal of M'.
   (d) Alice and Bob execute protocol `Linear Equations Solve` on $(R|M)$ and $(u|\mathbf{0})$ to get an encryption of a vector x_j such that $x_j(R|M) = (u|\mathbf{0})$, or $\mathsf{Enc}(\mathbf{0})$ if no such vector exists.
4. Bob's computes $\mathsf{Enc}(x) = \sum_{j=1}^{\ell} \mathsf{Enc}(x_j)$ and sends $\mathsf{Enc}(x)$ to Alice.
5. Alice outputs $v = xA$.

Alice and bob run `Linear Equations Solve` ℓ times and finally use the sum of the vectors x_j computed in these ℓ executions. The vectors x satisfying $xM = \mathbf{0}$ form a subspace, and hence are closed for addition. Thus, it is enough for one execution of `Linear Equations Solve` to yield a random solution, as in case (iii) above. To get to case (iii) we need the `Oblivious Gaussian Elimination` protocol to succeed and we need the linear system $x(R|M) = (u|\mathbf{0})$ to have a unique solution. The first event succeeds with constant probability. The success probability of the second event equals the probability that the sum of two random subspaces $V_1, V_2 \subseteq F^n$ of dimensions s and $n-s$ satisfy $V_1 \oplus V_2 = F^n$. The probability for this event is a constant as well. As both events occur with constant probability, case (iii) occurs with constant probability, and thus it is enough to run `Linear Equations Solve` $\omega(\log k)$ times, to get a negligible error probability.

Theorem 9. *Protocol* `Random Intersection Vector` *is a secure protocol for computing a random intersection vector. The communication complexity of the protocol is $\tilde{O}(\lambda k_a k)$ and the round complexity is $\tilde{O}(k_a^{0.275})$.*

6 Intersection of Affine Subspaces

In the affine subspace intersection problem Alice's input is an affine subspace $v_a + V_A$ where $v_a \in F^k$ and $V_A \subseteq F^k$ is a k_a dimensional linear subspace. Similarly, Bob's input is $v_b + V_B$, where $k_b = \dim(V_B)$. We design secure protocols for several problems concerning $(v_a + V_A) \cap (v_b + V_B)$. Our protocols are based on reductions to problems on linear subspaces. For example, to *compute* the intersection of two affine subspaces, we need both the `Intersection Computation` and the `Random Intersection Vector` protocols on linear subspaces. The following simple claims reduces the problem into computing whether a vector is contained in a subspace.

Claim 10 *There exists a vector $v \in (v_a + V_A) \cap (v_b + V_B)$ if and only if $v_a - v_b \in V_A + V_B$.*

Proof. Assume $v \in (v_a + V_A) \cap (v_b + V_B)$. Then $v = v_a + w_a$ for some $w_a \in V_A$ and $v = v_b + w_b$ for some $w_b \in V_B$. Hence $v_a + w_a = v_b + w_b$, and therefore, $v_a - v_b = w_b - w_a$, which means that $v_a - v_b \in V_A + V_B$. Now assume $v_a - v_b \in V_A + V_B$. Then there exist $w_a \in V_A$ and $w_b \in V_B$ such that $v_a - v_b = w_a + w_b$. Then $z \stackrel{\text{def}}{=} v_a - w_a = v_b + w_b$ is in the intersection $(v_a + V_A) \cap (v_b + V_B)$.

Claim 11 *Suppose $v_a + w_a = v_b + w_b$ for some $w_a \in V_A$ and $w_b \in V_B$. Then $(v_a + V_A) \cap (v_b + V_B) = (v_a + w_a) + (V_A \cap V_B)$.*

Proof. Let $v \in (v_a + V_A) \cap (v_b + V_B)$. Then there exist $z_a \in V_A$ and $z_b \in V_B$ such that $v = v_a + z_a = v_b + z_b$. As $v_a + w_a = v_b + w_b$, by subtracting equations we get $w_a - z_a = w_b - z_b$. Since $w_a - z_a \in V_A$ and $w_b - z_b \in V_B$, we get that $w_a - z_a \in V_A \cap V_B$. Thus $v = (v_a + w_a) - (w_a - z_a) \in (v_a + w_a) + (V_A \cap V_B)$.

For the other direction, let $v \in (v_a + w_a) + (V_A \cap V_B)$. Thus, $v = v_a + w_a + z = v_b + w_b + z$ where $z \in (V_A \cap V_B)$. As $w_a + z \in V_A$ and $w_b + z \in V_B$ we get $v \in (v_a + V_A) \cap (v_b + V_B)$.

Deciding if $(v_a + V_A) \cap (v_b + V_B)$ is empty. Protocol `Affine Intersection Decision` below is based on Claim 10. I.e., it checks whether $v = v_a - v_b \in \mathrm{span}(V_A + V_B)$. The privacy of the protocol follows from that of protocol `Linear Equations Feasibility`, and the communication complexity is $\tilde{O}(\lambda k(k_a + k_b))$.

Computing $(v_a + V_A) \cap (v_b + V_B)$. We describe a protocol for computing $(v_a + V_A) \cap (v_b + V_B)$, assuming the intersection is not empty. By Claim 11, it is enough for Alice and Bob to compute $V_A \cap V_B$, and find $w_a \in V_A$ and $w_b \in V_B$ such that $v_a + w_a = v_b + w_b$. We use Protocol `Linear Equation Solve` on the matrix $\begin{pmatrix} \mathsf{Enc}(A) \\ \mathsf{Enc}(B) \end{pmatrix}$ and the vector $\mathsf{Enc}(v_a + v_b)$. In the end of Protocol `Linear Equation Solve` Bob holds an encryption of a vector $\boldsymbol{c} \in \mathrm{GF}(2)^{k_a + k_b}$. Bob denotes the k_a leftmost coordinates of $\boldsymbol{c}$ by w_a. Alice and Bob now execute Protocol `Random Intersection Element` on V_A and V_B, such that Bob holds an encryption of a vector $r \in_R V_A \cap V_B$. Bob sends $\mathsf{Enc}(v_i) \stackrel{\text{def}}{=} \mathsf{Enc}(v_a + w_a + r)$ to Alice. Now Alice

Protocol `Affine Intersection Decision`

Input: Alice holds a k_a dimensional affine subspace $v_a + V_A$ of $\mathrm{GF}(2)^k$. Bob holds a k_b dimensional affine subspace $v_b + V_B$ of $\mathrm{GF}(2)^k$.
Output: The output is 1 if and only if $v_a + V_A \cap v_b + V_B \neq \emptyset$.

1. Alice sends Bob $\mathsf{Enc}(A)$ and $\mathsf{Enc}(v_a)$, where A is a $k_a \times k$ matrix representing V_A.
2. Bob computes $\mathsf{Enc}(B)$ and $\mathsf{Enc}(v_b)$, where B is a $k_b \times k$ matrix representing V_B.
3. Alice and Bob execute Protocol `Linear Equations Feasibility` on the matrix $\begin{pmatrix} \mathsf{Enc}(A) \\ \mathsf{Enc}(B) \end{pmatrix}$ and the vector $\mathsf{Enc}(v_a + v_b)$. Bob sends the outcome of Protocol `Linear Equations Feasibility` to Alice, that decrypts it as the output.

and Bob execute Protocol `Intersection Computation` such that Alice learns $V_I \stackrel{\text{def}}{=} V_A \cap V_B$. Alice outputs $v_i + V_I$.

Random Intersection Vector. Note that if instead of computing V_I in the protocol above, Alice simply outputs v_i, we get a protocol for computing a random intersection element.

Acknowledgments. We thank Amos Beimel, Yinnon Haviv, Benny Pinkas and Lior Zolf for helpful conversations.

References

1. D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. TCC 2005 pages 325–341.
2. M. Bellare, O. Goldreich, and E. Petrank. Uniform Generation of NP-Witnesses Using an NP-Oracle. In Inf. Comput. 163(2): 510-526 (2000).
3. A. Beimel, and E. Weinreb. Separating the Power of Monotone Span Programs over Different Fields. In FOCS 2003: 428-437.
4. A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and gcd computations. In Information and Control, 52(3):241-256, March 1982.
5. R. Cramer, and I. Damgård. Secure Distributed Linear Algebra in a Constant Number of Rounds. In CRYPTO 2001: 119-136.
6. D. Coppersmith, and S. Winograd. Matrix Multiplication via Arithmetic Progressions. In Proc. 19th ACN Symp. on Theory of Computing, pp. 1–6, 1987.
7. I. Damgård and M. Jurik. A generalization, a simplification and some applications of Paillier's probabilistic public-key system. In K. Kim, editor, *Proceedings of Public Key Cryptography 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, 2001.
8. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, Jul 1985.
9. O. Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, 2004.

10. S. Goldwasser and S. Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 365–377. ACM Press, 1982.
11. M. Karchmer and A. Wigderson. On Span Programs In *Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111, 1993.
12. Y. Lindell and B. Pinkas. Privacy Preserving Data Mining In *J. Cryptology* 15(3):177–206, 2002.
13. P. Pallier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Proceedings of Eurocrypt 1999*, volume 1592 of *LNCS*, pages 223–238. Springer-Verlag, May 1999.
14. T. P. Pedersen. A threshold cryptosystem without a trusted party. In D. Davies, editor, *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 522–526. Springer, 1991.
15. R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM 21(2): 120–126 (1978).
16. T. Sander, A. Young, and M. Yung. Non-interactive CryptoComputing for NC^1. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*, pages 554–567, New York, NY, USA, Oct. 1999. IEEE Computer Society Press.
17. M. Sipser. A Complexity Theoretic Approach to Randomness. In Proc. of the 15th Annual Symp. on the Theory of Computing, 1983.
18. A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Symposium on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE Computer Society Press, 1982.

A Security proof for the Intersection Computation Protocol

In this section we prove Bob's security in Protocol `Intersection Computation`. Note that the only information Bob sends to Alice is $\mathsf{Enc}(M)$, from which she learns M.

Simulator Alice

INPUT: A $k_i \times k$ matrix C representing V_I, a $k_a \times k$ matrix A representing V_A and an integer $k_b \leq k$.
OUTPUT: A matrix M in the same distribution of M in protocol `Intersection Computation`.

1. Compute a $k_i \times k_a$ matrix D, satisfying $DA = C$. As $V_I \subseteq V_A$ such a matrix exists and is easy to compute.
2. Compute a $k_b \times k_a$ matrix E by adding $k_b - k_i$ zero rows to D.
3. Compute the $k_a \times k_{b'}$ matrix $E^\perp$ whose columns represent the kernel of the matrix E.
4. Randomly choose a $k'_b \times k'_b$ full rank matrix T_E and output $M = E^\perp T_E$.

Claim 12 *M_S is distributed identically to M in* `Intersection Computation`.

Proof. Define the subspace $W = \{v : vA \in A \cap B\}$. The rows of the matrices D and E in the simulator span W. Therefore the columns of the matrix $E^{\perp}$ span the subspace $W^{\perp}$. Moreover, according to Claim 2, the columns of the matrix $AB^{\perp}$ from protocol `Intersection Computation` also span $W^{\perp}$.

Thus, there exists a full rank matrix T_0 of dimensions $k'_b \times k'_b$ such that $E^{\perp}T_0 = AB^{\perp}$. The probability that a matrix M_S is the simulator output is $Pr_{T_E}[M_S = E^{\perp}T_E]$. For every such choice of T_E take $T_B = T_0^{-1}T_E$ to be the choice of the protocol, to get $M = AB^{\perp}T_B = AB^{\perp}T_0^{-1}T_E = E^{\perp}T_E$. Conversely, for every random choice T_B of the protocol, set $T_E = T_0T_B$ to get $M_S = E^{\perp}T_E = E^{\perp}T_0T_B = AB^{\perp}T_B$. Therefore, the distributions are identical.

B Obliviously Solving Sets of Linear Equations

Let Bob hold an encrypted matrix $\mathsf{Enc}(M)$ and an encrypted vector $\mathsf{Enc}(v)$. We consider the decisional and functional versions of solving the linear system $cM = v$ (i.e., deciding whether exists a vector c satisfying $cM = v$, and finding such c).

Protocol `Linear Equations Feasibility`

Input: Alice holds a private key for a public-key homomorphic encryption scheme over GF(2). Bob holds an encryption $\mathsf{Enc}(M)$ of a $k_a \times k_b$ matrix over $GF(2)$ (we assume $k_a \leq k_b$; the general case is analogous), and an encryption $\mathsf{Enc}(v)$ of a vector $v \in \mathrm{GF}(2)^{k_b}$.
Output: If a vector $c \in \mathrm{GF}(2)^{k_a}$ exists such that $cM = v$ then Bob locally outputs $\mathsf{Enc}(1)$; Otherwise, he outputs $\mathsf{Enc}(0)$.

1. Bob randomly chooses a non-singular $k_a \times k_a$ matrix T_R, and a non-singular $k_b \times k_b$ matrix T_C, and computes $M' = T_RMT_C$, and $v' = vT_C$.
2. Alice and Bob run protocol `Oblivious Gaussian Elimination`, on the $(k_a + 1) \times k_b$ matrix $\begin{pmatrix} M' \\ v' \end{pmatrix}$, with the following exception: when multiplying the matrix M' by random matrices from the left, Alice and Bob pick a matrix that does not change the lower row of M'. Let Bob's output be $\mathsf{Enc}(M'')$.
3. Alice and Bob use the `Multiply` protocol to compute an encryption of $\prod_{i=1}^{k_b}(1 - M''[k_a + 1, i])$. This product is 1 if and only if the $k_a + 1$ row of M'' is $\mathbf{0}$.

In the first step of the protocol Bob multiplies M by random operators from the left and from the right to get $M' = T_RMT_C$. The following simple claim shows that it is enough to check if there exists a vector c' such that $c'M' = v'$ to solve the original $cM = v$ system.

Claim 13 *There exists a vector $c \in \mathrm{GF}(2)^{k_a}$ such that $cM = v$ if and only if there exists a vector $c'\mathrm{GF}(2)^{k_a}$ such that $c'M' = v'$.*

Proof. If there exists a vector $c \in \mathrm{GF}(2)^{k_a}$ such that $cM = v$, then the rows of M span v. Multiplying M by T_R from the left does not change the row space of M. Thus, there exists a vector c^* such that $c^* T_R M = v$. Multiplying both sides by T_C from the right results in $c^* M' = c^* T_R M T_C = v T_C = v'$. The other direction follows similarly.

By Claim 1, if M is a rank r matrix, then with constant probability the $r \times r$ top left sub-matrix of M' is of full rank. In the second step, Alice and Bob jointly perform Gaussian Elimination on the matrix $\begin{pmatrix} M' \\ \boldsymbol{v'} \end{pmatrix}$. We run the protocol on the $k_a \times k_a$ top left sub-matrix, letting `Basic Column Elimination` update the entire matrix. If the rows of the matrix M' span the row v', then by the end of the Gaussian Elimination protocol, the bottom row will be $\mathbf{0}$. Otherwise, the bottom row will not be $\mathbf{0}$. In step 3. we translate a zero vector in the last column to $\mathsf{Enc}(1)$, and a non-zero vector to $\mathsf{Enc}(0)$.

The protocol has a one-sided error. If the answer is NO then Bob will always hold an encryption of 0. If the answer is YES, then if in step (1) the rank of the top-left $k_a \times k_a$ sub-matrix of M' is that of M, Bob will hold an encryption of 1. As this happens with constant probability, Alice and Bob can execute the protocol a polylogarithmic number of times, and OR the results in order to make the error probability negligible.

Solving the Linear System. Note that the computation done in the Gaussian Elimination protocol may be viewed as multiplying M by non-singular matrices from the right and from the left (for column elimination, and for randomizing). That is, at the end of the protocol we get an encryption of M' where $M' = T_1 M T_2$ for some non-singular matrices T_1 and T_2.

To have Bob hold an encryption of a vector c such that $cM = v$, we need Bob to hold an encryption of T_1. We modify the Gaussian Elimination protocol such that any operation done on the rows of the input matrix M is simultaneously performed on the rows of a unit matrix I_{k_a+1}. At the end of this process Bob holds an encryption $T_1 I_{k_a+1} = T_1$.

We now describe protocol `Linear Equations Solve`. We assume that $cM = v$ is feasible, and compute such a solution c. Alice and Bob execute protocol `Linear Equations Feasibility`, using the modified `Oblivious Gaussian elimination protocol`. As a result, Bob holds a matrix T_1 such that $T_1 \begin{pmatrix} M' \\ \boldsymbol{v'} \end{pmatrix} = \begin{pmatrix} M'' \\ \mathbf{0} \end{pmatrix}$. Denote the lower row of T_1 by t_1. The vector t_1 gives a linear combination of M' and v' that gives the vector $\mathbf{0}$. Moreover, as we modified the Gaussian Elimination protocol not to use the bottom row in the elimination process, the rightmost entry of t_1 must be 1. Thus, denoting the k_a left entries of t_1 by c_1, we get that $c_1 M' + v' = 0$, that is, over $GF(2)$, $c_1 M' = v'$. Recall that $M' = T_R M T_C$ and $v' = v T_C$, and thus $c_1 T_R M T_C = v T_C$. Therefore, $c_1 T_R M = v$, and having an encryption of c_1, Bob can output $\mathsf{Enc}(c_1 T_R)$ as the output of the protocol.