Secure Distributed Linear Algebra in a Constant Number of Rounds

Ronald Cramer[⋆] and Ivan Damgård[⋆⋆]

Abstract. Consider a network of processors among which elements in a finite field K can be verifiably shared in a constant number of rounds. Assume furthermore constant-round protocols are available for generating random shared values, for secure multiplication and for addition of shared values. These requirements can be met by known techniques in all standard models of communication.
In this model we construct protocols allowing the network to securely solve standard computational problems in linear algebra. In particular, we show how the network can securely, efficiently and in constant-round compute determinant, characteristic polynomial, rank, and the solution space of linear systems of equations. Constant round solutions follow for all problems which can be solved by direct application of such linear algebraic methods, such as deciding whether a graph contains a perfect match.
If the basic protocols (for shared random values, addition and multiplication) we start from are unconditionally secure, then so are our protocols. Our results offer solutions that are significantly more efficient than previous techniques for secure linear algebra, they work for arbitrary fields and therefore extend the class of functions previously known to be computable in constant round and with unconditional security. In particular, we obtain an unconditionally secure protocol for computing a function f in constant round, where the protocol has complexity polynomial in the span program size of f over an arbitrary finite field.

1 Introduction

In this paper we consider the problem of secure multiparty computation (MPC), where n players, each holding a secret input, want to compute an agreed function of the inputs, in such a way that the correct result is computed, and no additional information about the inputs is released. This should hold, even in presence of an *adversary* who can *corrupt* some of the players, this means he can see all their internal data and can (if he is *active*) even make them behave as he likes.

A main distinction between different kinds of MPC protocols concerns the model for communication: In the *cryptographic* model (first studied in [25, 14]), the adversary may see all messages sent, and security can then only be guaranteed under a computational assumption. In the *information-theoretic* model

[⋆] Comp. Sc. Dept., Aarhus University & BRICS (Basic Research in Computer Science, center of the Danish National Research Foundation), `cramer@daimi.aau.dk`
[⋆⋆] Comp. Sc. Dept., Aarhus University & BRICS, `ivan@daimi.aau.dk`

(first studied in [5, 7]), one assumes a private channel between every pair of players, and security can then be guaranteed unconditionally.

Two measures of complexity are important for MPC protocols, namely the communication complexity (total number of bits sent) and the round complexity (where a round is a phase where each player is allowed to send one message to each other player).

In this paper, we focus on the round complexity of MPC protocols, in particular on building constant-round protocols. Kilian [19] showed that Boolean formulas can be securely and efficiently evaluated in constant rounds in the two-party case, with secure computations based on Oblivious Transfer. Under a complexity assumption, it was shown in [2] by Beaver, Micali and Rogaway that any function that can be computed in polynomial time can also be securely computed in a constant number of rounds (and polynomial communication). The result works under minimal complexity assumptions, but leads in practice to very inefficient protocols. Thus, for computationally secure MPC in constant round, the question is not which functions can be securely computed, but rather how efficiently it can be done.

The situation is different for *unconditionally* secure MPC: in this model, it is not known which functions can be securely computed in constant-round. However, Bar-Ilan and Beaver [1] showed that it can be done for any arithmetic formula.

Later results by Feige, Kilian and Naor [12] and Ishai and Kushilevitz [16, 17] and Beaver [3] extend this to functions in NL and some related counting classes. More precisely, their protocols are polynomial in the modular branching program size of the function computed. Their methods also apply to the more general arithmetic branching program model of Beimel and Gal[4].

2 Our Work

In this paper, we start from the assumption that we are given an efficient, constant round method to share securely between the players values in a finite field K and to reveal them. For an active adversary, this would be a verifiable secret sharing (VSS). In the following, we write $[a]$ for a sharing of a, i.e. $[a]$ denotes the collection of all information related to a held by the players. When M is a matrix over K, $[M]$ will denote a sharing of each of the coordinates of M. Whenever we say "let $[x]$ be a sharing" we mean that either some processor has distributed shares of his private input x, or that $[x]$ is the result of previous secure computations on certain private inputs of the processors. An expression such as "$[f(x)]$ is securely computed from $[x]$" means that the processors in the network perform secure computations on a sharing of x, as a result of which they obtain a (random) sharing of $f(x)$.

We show how to design efficient constant-round protocols for a number of standard linear algebra problems:

– Given a shared matrix $[A]$ over an arbitrary finite field K, we show how to compute securely a sharing $[\det(A)]$ of the determinant of A. More generally,

[**f**] is computed where **f** denotes the vector containing the coefficients of the characteristic polynomial of A.
– Given a shared (not necessarily square) matrix $[A]$ over a finite field K, we show how to securely compute the rank of A, concretely we can compute $[\mathbf{r}]$ where $\mathbf{r}$ is a unary encoding of the rank of A, padded with zeroes.
– Given also a shared vector $[\mathbf{y}]$ we show how to securely compute $[b]$ where b is a bit that indicates whether the system of equations $A\mathbf{x} = \mathbf{y}$ is solvable. Finally, we show how to solve the system by securely computing $[\mathbf{x}], [B]$, where $\mathbf{x}$ is a solution and $[B]$ generates A's kernel.

Our protocols work for arbitrary fields and do not use any cryptographic assumptions, so if the basic sharing method we start from is unconditionally secure, then so are the protocols we construct.

It is easy to see that our results allow handling all functions computable in constant round and with unconditional security using the most general previous methods [16, 17]: for instance, our protocol for subspace membership immediately implies a constant-round protocol for computing a function f, of complexity polynomial in the span program[18] size of f. By the results from [4] span programs are always at least as powerful as the modular and arithmetic branching programs to which the methods from [16, 17] apply. For fields with fixed characteristic, all three models are equivalent in power. However, since this is not known to hold for arbitrary fields, our results extend the class of function known to be computable in constant round and with unconditional security.

What is equally important, however, is that the standard linear algebra problems we can handle are problems that occur naturally in practice. For instance, deciding if a determinant is non-zero allows to decide if a bipartite graph contains a perfect match. Moreover, privacy is a natural requirement in matching type problems that occur in practice.

We therefore believe it is of interest to be able to do linear algebra securely and efficiently. Our work leads to a protocols with better efficiency compared to solutions based on combinations of known techniques. Please refer to Section 6.5 for more details in the case of determinant and characteristic polynomial.

We note that our results apply to the cryptographic model as well as the information theoretic, the only difference being the implementation of the underlying sharing and multiplication protocols. And because we attack the problems directly rather than going through reductions (to, e.g., Boolean circuits for the problems) we get much more efficient solutions than what one gets from, e.g, [2].

3 Some Basic Protocols

For convenience in describing our main protocols, we assume that secure constant round protocols are available for the following tasks:

– Computing (from scratch) a sharing $[r]$ where $r \in K$ is random and unknown to the adversary.
– Computing from sharings $[a], [b]$ a sharing of $[a + b]$.

– Computing from sharings $[a], [b]$ a sharing of $[ab]$.

Also, these protocols must remain secure when composed in parallel.

The first two requirements are always met if the sharing method used is *linear* over K, in the sense that from $[a], [b]$ and a known constant c, we can *non-interactively compute* new sharings $[a + b]$ and $[ac]$, and more generally, arbitrary linear functionals. For standard examples of VSS, this just translates to locally adding shares of a to corresponding shares of b, and to multiplying the shares of a by c.

In fact, all three requirements can be met by known techniques in all standard models of communication. We give here a few examples of existing efficient, constant round, linear MPC protocols: The classical unconditionally secure MPC protocols of Ben-Or, Goldwasser and Wigderson [5] and Chaum, Crépeau and Damgaard [7] are examples in the secure channels model satisfying all our requirements, tolerating an active, adaptive threshold adversary that corrupts less than a third of the processors.

MPC protocols secure against general adversaries [15] are given by Cramer, Damgaard and Maurer [9]. Their protocols make no restriction on the field size, as opposed to [7, 5] where this must be larger than the size of the network. [1] For the broadcast model of Rabin and Ben-Or [23], one can take the protocols of [10], tolerating an actively (and adaptively) corrupted minority at the expense of negligible errors and the assumption that a secure broadcast primitive is given. [2] An example in the cryptographic model is given by the protocols of Gennaro, Rabin and Rabin [13]. Here the size of the field is necessarily large. For the binary field an example given in [8]. This protocol, which is based on homomorphic threshold encryption, is quite efficient and tolerates an actively corrupted minority.

Note that parallel composition is not secure in general for all the models of communication mentioned here, unless extra properties are required. Nevertheless, the example protocols considered above are in fact secure under parallel composition.

A final basic protocol (called Π_1 in the following) that we will need is:

– Compute from a sharing $[a]$ a sharing $[h(a)]$ where h is the function on K defined by $h(a) = 0$ if $a = 0$ and $h(a) = 1$ if $a \neq 0$.

Later we show a constant-round realization of this protocol based only on the three requirements above. This realization is efficient if the characteristic of the field is polynomially bounded.

For *arbitrary fields*, we can do the following instead: assume first that $K = GF(q)$ for a (large) prime q. Represent an element $a \in K$ in the natural way as a bit string $a_0, ..., a_k$. Choose a new field $F = GF(p)$ where p is a small prime, all that is required is that p is larger than the number of players, in particular,

[1] In the full version of [9] it is pointed out that their VSS is actually constant round.

[2] One extra level of sub-sharing must be built in (which is no problem) to ensure constant rounds for their multiplication protocol.

p does not depend on the size of the input to the desired computation. Define $[a] = [a_0]_F, ..., [a_k]_F$, i.e., using any of the standard methods described above we share each bit in the representation of a over the field F.

We can now use the well-known fact that for a given, fixed q, there exist NC^1 Boolean circuits for elementary operations in $GF(q)$ (and even for unbounded fan-in addition). This, together with the result from [1] and the fact that Boolean operations can be simulated in a natural way by arithmetic in F immediately implies existence of constant round protocols for this sharing method meeting the three requirements above. Moreover, computing the function h is now trivial since we only have to compute the OR of all bits of the input value.

Finally, we show in Section 4 how the basic protocols for a field K can be lifted to any extension field of K.

Since most of the known MPC protocols are linear, we explain our protocols under this assumption, since it leads to more efficient and easier to explain protocols. At the end of Section 4, we argue that our results also hold in the more general model described earlier.

4 Known Techniques Used

Let a secure linear MPC protocol for elementary arithmetic (i.e., multiplication and addition) over a finite field K be given, that is efficient and constant round. Write $q = |K|$. We frequently use the following constant-round techniques from Bar-Ilan and Beaver [1].

Joint Secret Randomness is a protocol to generate a sharing $[\rho]$ where $\rho \in K$ is random and secret. This is just by letting all players in the network share a random element, and taking the sum as the result. This extends in a natural way to random vectors and matrices. *Secure Matrix Multiplication* is a protocol that starts from sharings $[A], [B]$ of matrices A and B, and generates a sharing $[AB]$ of their product. This protocol works in the obvious way. We denote this secure computation by $[AB] = [A]\cdot[B]$. By our assumptions on the basic MPC, it follows that if any of these matrices, say A, is publicly known, secure multiplication can be performed non-interactively. and we write $[AB] = A \cdot [B]$ instead.

Jointly Random Secret Invertible Elements and Matrices is a protocol that generates a sharing of a secret, random nonzero field element or an invertible matrix. The protocol securely generates two random elements (matrices), securely multiplies them, and reveals the result. If this is non-zero (invertible), one of the secret elements (matrices) is taken as the desired output of the protocol. The probability that a random matrix $A \in K^{n,n}$ is invertible is than $1/4$,[3] and is at least $1 - n/q$. In particular if n is negligible compared to q, almost all $A \in K^{n,n}$ are invertible. This is easy to verify (see also the counting arguments in Section 6.1).

Secure Inversion of Field Elements and Matrices is a protocol that starts from a sharing of an invertible field element or matrix, and results in a sharing of its

[3] For instance, by simple counting and induction it follows that this probability is at least $1/4 + (1/2)^{n+1}$. Also, there are better estimates known from the literature.

inverse. We denote this secure computation by $[x^{-1}] = [x]^{-1}$, and $[A^{-1}] = [A]^{-1}$ respectively. This protocol first generates $[\rho]$ with $\rho \in K$ random and non-zero, securely computes $[\sigma] = [\rho] \cdot [x]$, and finally reveals σ. The result $[x^{-1}]$ is then non-interactively computed as $\sigma^{-1} \cdot [\rho]$. The same approach applies to the case of an invertible matrix.

Securely Solving Regular Systems is a protocol that starts from sharings of an invertible matrix and a vector, and generates a sharing of the unique pre-image of that vector under the given invertible matrix. This protocol follows immediately from the above protocols.

Secure Unbounded Fan-In Multiplication is a protocol that produces a sharing of the product of an unbounded number n of shared field elements $[x_i]$. First consider the case where all elements $[x_1], \dots, [x_m]$ are invertible. The network generates sharings $[\rho_1], \dots, [\rho_m]$ of independently random non-zero values, and subsequently sharings of their multiplicative inverses. Next, they compute $[\sigma_1] = [x_1] \cdot [\rho_1]$, and, for $i = 2 \dots m$, $[\sigma_i] = [\rho_{i-1}^{-1}] \cdot [x_i] \cdot [\rho_i]$. Finally, they publicly reconstruct σ_i for $i = 1 \dots m$, compute the product of the σ_i's, and multiply the result into the sharing of ρ_m^{-1} to get a sharing of the product of the x_i's. See [3] for a more efficient solution. Using a result by Ben-Or and Cleve, the general case (i.e., x_i's may be equal to 0) is reduced to the previous case. The resulting protocol comes down to unbounded secure multiplication of certain invertible 3×3-matrices. The overhead is essentially a multiplicative factor n. See Section 6.4 for an alternative approach.

Note also that the MPC protocol over K is easily "lifted" to an extension field L of K, as we show below. If the original is efficient and constant round, then so is the lifted protocol.

L may be viewed as a K-vectorspace, and let $b_0, \dots, b_{d-1}$ be a fixed K-basis for L, where d is the degree of L over K. More precisely, let α be a root of an irreducible polynomial $f(X) \in K[X]$ of degree d. We set $b_i = \alpha^i$ for $i = 0 \dots d-1$. Elements of L are represented by coordinate vectors with respect to this chosen basis. In particular, the vectors that are everywhere zero except possibly in the first coordinate correspond to the elements of K.

If $[x_0]_K, \dots, [x_{d-1}]_K$ are sharings, with the x_i's in K, it is interpreted as $[x]_L$, where $x = \sum_{i=0}^{d-1} x_i \cdot b_i \in L$. Let $[x]_L = ([x_0]_K, \dots, [x_{d-1}]_K)$, $[y]_L = ([y_0]_K, \dots, [y_{d-1}]_K)$ be sharings of $x, y \in L$. Securely computing $[x+y]_L = [x]_L + [y]_L$ amounts to computing the sum of the vectors, which is be done by local operations. So we have the correspondence $[x+y]_L \leftrightarrow ([x_0+y_0]_K, \dots, [x_{d-1}+y_{d-1}]_K)$.

Now consider multiplication. For $i = 0, \dots, d-1$ let B_i be the matrix whose j-th column is the vector representation of the element $e_i \cdot e_j \in L$, and let $B = B_0 || \dots || B_{d-1}$ (concatenation from left to right). Let $\mathbf{x} \otimes \mathbf{y} \in K^{n^2}$ be the (column-) vector whose j-th "block" is $x_j \cdot y_0, \dots, x_j \cdot y_{d-1}$. Then: $x \cdot y = \sum_{i,j=0}^{d-1} x_i \cdot y_j \cdot e_i \cdot e_j = \sum_{i=0}^{d-1} \lambda_i \cdot e_i$, where $B(\mathbf{x} \otimes \mathbf{y}) = (\lambda_0, \dots, \lambda_{d-1})^T \in K^d$, and so we have the correspondence: $[x \cdot y]_L \leftrightarrow [B(\mathbf{x} \otimes \mathbf{y})]_K$.

Over K, it is straightforward for the network to first securely compute $[\mathbf{x} \otimes \mathbf{y}] = [\mathbf{x}] \otimes [\mathbf{y}]$ efficiently in constant rounds. Since B is public, secure computation of $[B(\mathbf{x} \otimes \mathbf{y})]_K$ is then by local operations only. Hence, secure multiplication in the extension field can be carried out efficiently in constant rounds. Note that securely multiplying in a known constant is a special case, which is handled completely by local operations.

Finally, we note that if the linearity assumption on the MPC protocol is dropped, and instead we work with the model also described in Section 2, where more generally secure constant round protocols for generation of shared random element, addition and multiplication are assumed, the above sub-protocols still work. It is sufficient to argue that unbounded addition can be securely and efficiently performed in constant rounds.

Although this does not directly follow from the model, as is the case with linearity, it can be done in similar style as unbounded multiplication of non-zero field elements using the ideas of Bar-Ilan and Beaver. This is actually where the assumption on secure generation of a shared random value comes into play. It is easy to verify that all protocols to follow also work in this more general model.

5 Overview and Conventions

Throughout we assume efficient, constant round, secure linear MPC protocols over a finite field K. In the analysis we assume that the required properties are perfectly satisfied.

The linear algebraic problems of our interest are determinant, characteristic polynomial, rank, sub-space membership, random sampling and general linear systems. We first explain secure solutions with negligible error probabilities.

We will assume that K ("the field of interest") is "large enough", i.e., n ("dimension" of the linear algebraic problems) is negligible compared to $q = |K|$. Without loss of generality we may use the lifting technique to achieve this.

In all cases, solutions of the original problems defined over K can be recovered from the solutions of the lifted problem.

In Section 9 we argue how to obtain zero-error modifications of our protocols.

6 Secure MPC of Determinant

Let $[A]$ be a sharing, where $A \in K^{n,n}$. The goal of the network is to securely compute a sharing $[\det(A)]$, where $\det(A)$ is the determinant of A, efficiently in constant rounds.

Secure computation via the standard definition of determinant is inefficient, and a secure version of Gaussian elimination for instance, seems inherently sequential. After we give our efficient and constant round solution, we discuss some less efficient alternatives based on combinations of known techniques.

6.1 The Case of Invertible Matrices

We start by solving the problem under the assumption that the shared matrix A is promised to be invertible and that there exists an efficient constant round protocol Π_0 allowing the network to securely generate a pair $([R], [\det(R)])$ where

$R \in K^{n,n}$ is an (almost) random invertible matrix and $\det(R)$ is its determinant. Note that we do *not* require that the network can securely compute the determinant of a random invertible matrix; we merely require that Π_0 securely constructs a sharing of a random invertible matrix *together* with its determinant.

In the following, let $\mathrm{GL}_n(K) \subset K^{n,n}$ denote the group of invertible matrices. Let $[A]$ be a sharing, with $A \in \mathrm{GL}_n(K)$.

1. Under the assumption that protocol Π_0 is given, the network securely generates $([R],[d])$, where $R \in \mathrm{GL}_n(K)$ is random and $d = \det(R)$.
2. By the method of Bar-Ilan and Beaver for secure inversion, the network securely computes $[d^{-1}] = [d]^{-1}$.
3. The network securely computes $[S] = [R] \cdot [A]$, and *reveals* S.
4. All compute $e = \det(S)$, and by local operations they securely compute $[\det(A)] = e \cdot [d^{-1}]$.

Note that (S, e) gives no information on A. Also note that the protocol is not private if A is not invertible, since $e = 0$ exactly when that is the case. A realization for Protocol Π_0 is shown below.

Realization of Protocol $\boldsymbol{\Pi_0}$ We show an efficient, constant round protocol for securely generating pairs $([R],[d])$, where $R \in \mathrm{GL}_n(K)$ is random and $d = \det(R)$. It achieves perfect correctness. The distribution of (R, d) has a negligible bias.

Our solution is based on the idea of securely multiplying random matrices of a special form, and requires that n is negligible compared to q. The protocol goes as follows.

1. The network securely generates the pair of shared vectors $[\mathbf{x}_L], [\mathbf{x}_U]$, where $\mathbf{x_L}, \mathbf{x_U} \in K^n$ both consist of random non-zero entries, and securely computes $[d] = (\prod_{i=1}^{n} [x_L(i)]) \cdot (\prod_{i=1}^{n} [x_U(i)])$.
   This is done using the methods of Bar-Ilan and Beaver for secure unbounded fan-in multiplication of *non-zero* values.
2. The network securely generates $n^2 - n$ elements $[r_i]$, where the $r_i \in K$ are random.
   Next, the network defines $[L]$ such that $L \in K^{n,n}$ has $\mathbf{x}_L$ on its diagonal, while the elements below the diagonal are formed by the first $\frac{1}{2}(n^2 - n)$ of the $[r_i]$'s. The elements above the diagonal are set to 0.
   Similarly for the matrix $[U]$, but with $\mathbf{x}_U$ on its diagonal, and the remaining $[r_i]$'s placed above the diagonal. The elements below the diagonal are set to 0.
   Finally, the network securely computes $[R] = [L] \cdot [U]$, Note that $\det(R) = d \neq 0$. The result of the protocol is set to $([R],[d])$.

Correctness is clear. We now discuss privacy. Define $\mathcal{L}, \mathcal{U}$ as the sub-groups of $\mathrm{GL}_n(K)$ consisting of the invertible lower- and upper-triangular matrices, i.e., the matrices with non-zero diagonal elements, and zeroes above (resp. below) the diagonal. For $n > 1$ these groups are non-abelian. Let $\mathcal{D}$ denote the invertible diagonal matrices, i.e., the matrices with non-zero diagonal elements and zeroes elsewhere. We have $\mathcal{L} \cap \mathcal{U} = \mathcal{D}$, $|\mathcal{L}| = |\mathcal{U}| = q^{\frac{n^2-n}{2}}(q-1)^n$, $|\mathcal{D}| = (q-1)^n$.

Define the map $h : \mathcal{L} \times \mathcal{U} \longrightarrow \mathrm{GL}_n(K)$, $(L, U) \mapsto LU$, and write $\mathcal{R} = h(\mathcal{L} \times \mathcal{U})$ for the range of h, i.e., $\mathcal{R}$ consists of all invertible matrices that can be written as the product of a lower- and an upper-triangular matrix.

For each $R \in \mathcal{R}$, it holds that $|h^{-1}(R)| = |\mathcal{D}|$. Using the fact that $\mathcal{L}$, $\mathcal{U}$ and $\mathcal{D}$ are groups and that $\mathcal{L} \cap \mathcal{U} = \mathcal{D}$, this claim is easily proved as follows. Let $R = LU$, and let $D \in \mathcal{D}$. Then $LD^{-1} \in \mathcal{L}$ and $DU \in \mathcal{U}$, and $R = (LD^{-1})(DU)$. This shows that R has at least $|\mathcal{D}|$ pre-images under h. On the other hand, if $R = L_0U_0 = L_1U_1$, then $L_1^{-1}L_0 = U_1U_0^{-1}$. Since $L_1^{-1}L_0 \in \mathcal{L}$ and $U_1U_0^{-1} \in \mathcal{U}$, both are equal to D for some $D \in \mathcal{D}$, and so we can write $L_1 = L_0D^{-1}$ and $U_1 = DU_0$.

As a consequence, $|\mathcal{R}| = \frac{|\mathcal{L}|\cdot|\mathcal{U}|}{|\mathcal{D}|}$. Thus we have $\frac{|\mathcal{R}|}{|K^{n,n}|} = (1 - \frac{1}{q})^n$, and hence, $\frac{|\mathcal{R}|}{|\mathrm{GL}_n(K)|} > (1 - \frac{1}{q})^n \geq 1 - \frac{n}{q}$.

These facts imply that if (L, U) is chosen uniformly at random from $\mathcal{L} \times \mathcal{U}$, then $R = LU$ is distributed uniformly in $\mathcal{R}$, which is almost all of the invertible matrices when n is negligible compared to q. [4]

We note that it is possible to devise an alternative for protocol Π_0, where each player in the network shares a random invertible matrix and a value he claims is the determinant. Invertibility is proved using Bar-Ilan and Beaver's techniques. Using cut-and-choose techniques it can be established that this value is indeed the determinant. The desired output is obtained by taking products. However, this method introduces correctness errors, and is less efficient compared to the above solution.

6.2 The General Case of Determinant

If $A \in K^{n,n}$ is no longer guaranteed to be invertible the situation becomes slightly more involved. Although the protocol would still compute the determinant correctly, security is not provided if the matrix is singular: by inspection of the previous protocol, the publicly available value e is equal to 0 exactly when A is singular. Moreover, any blinding technique in which a product of A with randomizing matrices is revealed, provides a lower-bound on A's rank. [5]

We now propose our solution for secure computation of determinant. Let $[A]$ be a sharing, where $A \in K^{n,n}$ is an arbitrary matrix. The purpose of the network is to securely compute a sharing $[\det(A)]$ efficiently in constant rounds.

[4] We note that all invertible matrices can be brought into "LUP" form, where L and U are invertible matrices in lower-, resp. upper-triangular form, and P is a permutation matrix. However, choosing each of these at random, LUP does not have the uniform distribution on $\mathrm{GL}_n(K)$. Moreover, securely computing the sign of the permutation would pose a separate problem at this point.

[5] The rank of the product of matrices is at most equal to the smallest rank among them.

Let $f_A(X) = \det(X \cdot I_n - A) \in K[X]$ denote the characteristic polynomial of A, where I_n denotes the $n \times n$ identity matrix. Then $f_A(0) = (-1)^n \cdot \det(A)$ and $\deg f = n$. By Lagrange Interpolation, for distinct $z_0, \ldots, z_n \in K$, there are $l_0, \ldots, l_n \in K$, only depending on the z_i's, such that $\det(A) = (-1)^n \cdot f_A(0) = (-1)^n \cdot \sum_{i=0}^{n} l_i \cdot f_A(z_i) = (-1)^n \cdot \sum_{i=0}^{n} l_i \cdot \det(z_i I_n - A)$.

Now, for $z \in K$, it holds that $zI_n - A \in \mathrm{GL}_n(K)$ if and only if $f_A(z) \neq 0$, i.e., z is not an eigenvalue of A.

Since A has at most n eigenvalues, the matrix $zI_n - A$ is invertible when z is randomly and independently chosen, except with probability at most $1/q$.

This enables a reduction of the problem of securely computing $[\det(A)]$ to that of secure computation of the determinant of a number of invertible matrices, which now we know how to do, and proceed as before.

1. In parallel, the network securely generates $[z_0], \ldots, [z_n]$, where the z_i are randomly distributed in K. They *reveal* the z_i's. Except with negligible probability, the z_i's are distinct (which can be checked of course) and all matrices $z_i I_n - A$ are invertible. For $i = 0, \ldots, n$, the network securely computes by local computations $[z_i I_n - A] = z_i I_n - [A]$
2. Using our protocol for securely computing the determinant of an invertible matrix, they securely compute in parallel $[\det(z_0 I_n - A)], \ldots, [\det(z_n I_n - A)]$.
3. Finally, the network securely computes $[\det(A)] = (-1)^n \cdot \sum_{i=0}^{n} l_i \cdot [\det(z_i I_n - A)]$, where the l_i's are the interpolation coefficients.

Note that if some z_i happens to be an eigenvalue of A, this becomes publicly known, since the sub-protocol for securely computing the determinant of an invertible matrix noticeably fails in case it is not invertible. On the other hand, it also means that if z_i is *not* an eigenvalue of A, this also becomes known, and the adversary can rule out all matrices A' of which z_i is an eigenvalue.

However, it is only with negligible probability that the adversary learns an eigenvalue. The actual probability depends on A, but this poses no privacy problems since it is negligible anyway.

Also, the adversary could predict with almost complete certainty in advance that z_i is not an eigenvalue. Hence we have almost perfect privacy, and perfect correctness.

6.3 Secure MPC of Characteristic Polynomial

Let $M \in K^{n+1,n+1}$ be the Vandermonde matrix whose i-th row is $(1, z_i, \ldots, z_i^n)$, and write $\mathbf{y}$ and $\mathbf{f}$ for the (column) vectors whose i-th coordinates are equal to y_i and to the coefficient of X^i in $f_A(X)$, respectively. Then $\mathbf{f} = M^{-1}\mathbf{y}$.

The protocol above not only securely computes the determinant $[\det(A)]$, but in fact the coefficient vector $\mathbf{f}$ of the characteristic polynomial, if we replace the last step by $[\mathbf{f}] = M^{-1} \cdot ([\det(z_0 I_n - A)], \ldots, [\det(z_n I_n - A)])^T$. Note that we might as well omit computation of the leading coefficient of $f_A(X)$ since it is equal to 1 anyway.

6.4 Alternative Protocol For Unbounded Multiplication

As an aside, we note that a similar reduction via interpolation yields an alternative protocol for unbounded multiplication. Namely, consider $[a_1], \ldots, [a_n]$ with the a_i's in K, and define $f(X) = \prod_{i=1}^{n}(X - a_i)$. By applying interpolation through random points on $f(X)$, we get a similar reduction to the much simpler case of unbounded multiplication of non-zero field elements. Zero-error can be obtained by a method described in Section 9.

6.5 Other Approaches

We discuss some interesting but less efficient alternatives based on combinations of known results, in particular from Parallel Computing.

First we consider a combination of techniques due to Mahajan and Vinay [21], Ishai and Kushilevitz [16, 17], and Beimel and Gál [4].

For our purposes, an Arithmetic Program (AP) [4] is a weighted directed acyclic graph with two distinguished vertices s, t. Each edge is labelled by a variable that can take on a value in a finite field K. The function computed by an AP is defined by taking a path from s to t, multiplying the weights, and summing up over all such paths to finally obtain the function value. The computations take place in the finite field K. By elementary algebraic graph theory, the function value shows up as the (s, t)-entry in the matrix $(I - H)^{-1}$, where H is the adjacency matrix of the weighted graph. Ishai and Kushilevitz [16, 17] nicely exploit this fact in their construction of representations of functions in terms of certain degree-3 randomized polynomials obtained from branching programs.

The result of [21] in particular says that there is an AP with roughly n^3 vertices for computing determinant. The weights are entries from the matrix of interest, where the correspondence does not depend on the actual matrix.

Therefore, determinant can in principle be securely computed using a single secure matrix inversion. Unfortunately, this matrix has dimension greater than n^3. Bar-Ilan and Beaver's inversion applied to the matrix $I - H$, essentially requires secure multiplication of two $n^3 \times n^3$ matrices. Methods for securely computing a sharing of just the (s, t)-entry of $(I - H)^{-1}$ rather than the whole matrix (via a classical identity relating inverse with determinants) seem to require secure computations of determinant in the first place.

Another approach can be based on Leverier's Lemma (see e.g. [20]), which retrieves the coefficients of the characteristic polynomial by inverting a certain lower-triangular matrix, where each entry below the diagonal is the trace of a power of the matrix of interest. This lemma is obtained by combining Newton's identities with the fact that these traces correspond to sums of powers of the characteristic roots.

If $K = p$, with p a prime greater that the dimension n of the matrix, it is possible to devise a secure protocol for characteristic polynomial whose complexity is dominated by securely computing all i-th powers of the matrix, for

$i = 1 \ldots n$. These terms can be computed separately using techniques of Bar-Ilan and Beaver, or by using the observation that obtaining the n powers of an $n \times n$-matrix is no harder than inverting an $n^2 \times n^2$-matrix (see e.g. [20] for more details).

Note that our solution for large fields essentially just requires secure multiplication of two $n \times n$-matrices (due to Bar-Ilan and Beaver's matrix inversion) if the matrix is promised to be invertible, and n times that amount in the general case.

7 Secure MPC of Rank

The purpose of the network is to securely compute the *rank* of a matrix A efficiently in constant rounds. An important feature of our solution is that the network in fact securely computes a sharing $[\mathbf{r}]$, where $\mathbf{r} \in K^n$ encodes the rank of A in *unary*. This means that, when viewed as a column vector, all non-zero entries of $\mathbf{r}$ are all equal to 1 and occur in the bottom r positions. Rank encoded this way facilitates an easy way to securely compare the ranks of given matrices, as we show in an application to the subspace membership problem later on.

We note that Ishai and Kushilevitz [16, 17] have proposed an elegant and efficient protocol for secure computation of rank. Their protocol produces a random shared matrix with the same rank as the shared input matrix. This particular way of encoding rank, however, seems to limit applicability in a scenario of ongoing secure multi-party computations.

In some special cases, such as when a square matrix A is in triangular form, its rank $\mathrm{r}(A)$ can be read off its characteristic polynomial $f_A(X)$, as $n - t$, where X^t is largest such that it divides $f_A(X)$, and n is its degree. This is not always the case.

Mulmuley [22] proved the following result. Let $S \in K^{m,m}$ be symmetric. Let Y be an indeterminate, and define the diagonal matrix $D = (d_{ii}) \in K[Y]^{m,m}$ with $d_{ii} = Y^{i-1}$. Let $f_{DS}(X) \in K[X,Y]$ denote the characteristic polynomial of $DS \in K[Y]^{m,m}$. Then $\mathrm{r}(S) = m - t$ where t is maximal such that X^t divides the characteristic polynomial $f_{DS}(X) \in K[X,Y]$ of DS. In other words, $f_{DS}(X) = X^{m-\mathrm{r}(S)} \cdot \sum_{i=0}^{\mathrm{r}(S)} f_i(Y)X^i$, where $f_0(Y) \neq 0$ and $f_{\mathrm{r}(S)}(Y) = (-1)^m$.

If S is not symmetric, it can be replaced by the symmetric matrix S^*, which has S^T in its lower-left corner and S in its upper-right corner, while the rest is set to 0. Both dimension and rank of S^* are twice that of S.

We exploit this result as follows. Let $[A]$ be a sharing with $A \in K^{n,n}$. [6].

The network first constructs a sharing $[A^*]$ of the symmetric matrix $A^* \in K^{2n,2n}$, which is done locally in a trivial manner. Next, they securely generate $[y_0]$ with y_0 random in K, and reveal it. Define $D_0 \in K^{2n,2n}$ as the matrix D from above, with the substitution $Y = y_0$.

[6] Note that if A is not square, say $A \in K^{n,m}$, then we can easily extend A to a square matrix whose rank is the same, by appending all-zero rows or columns. This leads to an $s \times s$-matrix where $s = \max(n, m)$.

If $f_0(y_0) \neq 0$, then $2 \cdot \mathrm{r}(A) = 2n - t$, with X^t largest such that it divides the characteristic polynomial $f_{D_0 A^*}(X) \in K[X]$ of the matrix $D_0 A^*$. Since the degree of $f_0(Y)$ is at most $n(2n-1)$ (as follows from simple inspection), $f_0(y_0) \neq 0$, except with probability $n(2n-1)/q$.

The next step for the network is to securely compute $f_{D_0 A^*}(X)$. To this end, they publicly compute D_0 from y_0, and finally by local computations $[D_0 A^*] = D_0[A^*]$. Using our Characteristic Polynomial Protocol they securely compute a sharing of the coefficient vector of the polynomial.

Viewing this as a column vector whose i-th entry is the coefficient of X^i in the polynomial, $i = 0 \dots 2n-1$, and neglecting the coefficient of X^{2n}, it has its top t entries equal to zero, while the $t+1$-st is non-zero. By discarding "every second" entry we obtain a vector $\mathbf{f} \in K^n$ whose top $n - r$ entries are zero, while its $n - r + 1$-st entry is non-zero, where $r = \mathrm{r}(A)$.

Definition 1 *Let* $\mathbf{r} \in K^n$ *be a column-vector, and let* $0 \leq r \leq n$ *be an integer. We say that* $\mathbf{r}$ *is an* almost-unary *encoding of* r *if its bottom* r *entries are non-zero, while it has zeroes elsewhere. If the non-zero entries are all equal to 1, we say that* $\mathbf{r}$ *is a* unary *encoding of* r.

If $H \in K^{n,n}$ is a random lower-triangular matrix, then $H\mathbf{f}$ has its top $n - r$ entries equal to 0, while its bottom r entries are randomly and independently distributed in K. Hence, except with probability at most $r/q \leq n/q$, $H\mathbf{f}$ is a random almost-unary encoding of A's rank r. The actual probability depends on the rank, but it is negligible anyway.

The network now simply securely generates a sharing $[H]$ of a random lower-triangular, reveals it, and non-interactively computes the almost-unary encoding of A's rank as $[H\mathbf{f}] = H[\mathbf{f}]$.

A unary encoding $[\mathbf{r}]$ of A's rank r can be securely computed from $[H\mathbf{f}]$ by applying the protocol Π_1 mentioned earlier. This protocol starts from a sharing $[x]$ with $x \in K$, and securely computes $[h(x)]$ in constant rounds, where $h(x) = 1$ if $x \neq 0$ and $h(x) = 0$ if $x = 0$. [7]

Applying protocol Π_1 in parallel to each of the entries of the almost-unary encoding $H\mathbf{f}$, we get the desired result. We show one realization of such a protocol below. A less efficient, but more general method was shown in Section 3.

7.1 Protocol Π_1 Based On Secure Exponentiation

We assume that the field K has "small" characteristic p, and that the MPC protocol over K run by the network can be viewed as a lifting from protocols over $\mathrm{GF}(p)$ to K.

Let $[x]$ with $x \in K$ be a sharing. Note that $h(x) = x^{q-1}$, where $q = |K|$.

The first idea that comes to mind is to securely perform repeated squaring. This requires $O(\log q)$ rounds of communication however. Applying the constant round protocol of Bar-Ilan and Beaver for unbounded fan-in secure multiplication

[7] As an aside, note that $h(x)$ is the rank of the 1×1-matrix x.

to our problem is no option either, since in this case the communication overhead will be polynomial in q instead of $\log q$.

Another idea is to apply Bar-Ilan and Beaver's protocol for secure inversion. Namely, the network would securely compute $[y]$, where $y = x^{-1}$ if $x \neq 0$ and $y = 0$ if $x = 0$, and finally compute $[h(x)] = [x] \cdot [y]$. Unfortunately, the network would learn that $x = 0$ in the first step, as can be seen by inspection of the Bar-Ilan and Beaver method. Hence, the security requirements are contradicted. We note that the function h defined above is closely related to the Normalization Function defined in [1], which tells whether two elements are equal or not. They show how this function (and hence h as well) can be securely computed in constant rounds if the field K is small.

We need an alternative approach which works for exponentially large fields. Our solution comes at the expense of assuming small characteristic. Write d for the degree of K over $\mathrm{GF}(p)$. So $q = p^d = |K|$. Let $1 \leq s \leq q - 1$ be a given, public integer, and let $[x]$ with $x \in K$ be a sharing. This is how they can securely compute $[x]^s$, efficiently in constant rounds. Setting $s = q - 1$, we get the desired protocol Π_1.

Taking p-th powers in K is a field automorphism of K that leaves $\mathrm{GF}(p)$ fixed. In particular this means that this map can be viewed as an automorphism of K as a $\mathrm{GF}(p)$-vectorspace. Let $B \in \mathrm{GL}_d(\mathrm{GF}(p))$ denote the (public) matrix representing this map, with respect to the chosen basis. Then for $i \geq 1$, the matrix $B^i \in \mathrm{GL}_d(\mathrm{GF}(p))$ represents taking p^i-th powers.

For $i = 0 \ldots d - 1$, write $z_i = x^{p^i}$. Let $s = \sum_{i=0}^{d-1} s_i p^i$ be the p-ary representation of s. Then we have $x^s = x^{\sum_{i=0}^{d-1} s_i p^i} = \prod_{i=0}^{d-1} \left(x^{p^i}\right)^{s_i} = \prod_{i=0}^{d-1} z_i^{s_i}$. If $(x_0, \ldots, x_{d-1}) \in \mathrm{GF}(p)^d$ is the vector representation of x, then the vector representation of z_i is $B^i(x_0, \ldots, x_{d-1})^T$. Since B^i is public and since the vector representation of x are available as sharings, the network can securely compute the vector representation of $[z_i]$ by local computations. Next, the network securely computes the s_i-th powers of the z_i, running Bar-Ilan's and Beaver's unbounded fan-in secure multiplication protocols in parallel. Each of these steps costs $O(p^2)$ secure multiplications, so the total number is $O(\log q \cdot p^2)$. But since p is "small" (for instance, constant or polynomial in $\log q$)) this is efficient. The protocol is finalized by securely multiplying the $d = O(\log q)$ results together using the same technique.

7.2 Application to Sub-Space Membership Decisions

Using our Rank Protocol, the network can securely compute a shared decision bit $[b]$ from $[A]$ and $[\mathbf{y}]$, where $b = 1$ if the linear system $A\mathbf{x} = \mathbf{y}$ is solvable and $b = 0$ otherwise.

Defining $A_{\mathbf{y}}$ by concatenating $\mathbf{y}$ to A as the last column, we have $1 - b = \mathrm{r}(A_{\mathbf{y}}) - \mathrm{r}(A)$, where $\mathrm{r}(\cdot)$ denotes the rank of a matrix and $b = 1$ if the system is solvable, and $b = 0$ otherwise.

Suitably padding both matrices with zeroes, we make them both square of the same dimension, while their respective ranks are unchanged. Running the

Rank Protocol in parallel, the network now securely computes unary encodings $[\mathbf{r}]$, $[\mathbf{r_y}]$ of the ranks of A and $A_{\mathbf{y}}$, respectively. It holds that $\mathrm{r}(A) = \mathrm{r}(A_{\mathbf{y}})$ exactly when $\mathbf{r} = \mathbf{r_y}$.

Next, the network securely computes $[\mathbf{u}] = [\mathbf{r}] - [\mathbf{r_y}]$ by local computations, securely generates $[\mathbf{v}]$ with $\mathbf{v}$ random in K^n, and finally securely computes $[v] = [\mathbf{u}] \cdot [\mathbf{v}^T]$. Except with negligible probability $1/q$, it holds that $v = 0$ if $b = 1$ and $v \neq 0$ if $b = 0$. The network securely computes $[b] = 1 - [h(v)]$, using protocol Π_1.

8 General Linear Systems

Let $[A]$ and $[\mathbf{y}]$ be sharings, where A is a square matrix, [8] say $A \in K^{n,n}$, and $\mathbf{y} \in K^n$. The purpose of the network is to securely compute $[b]$, $[\mathbf{x}]$ and $[B]$, efficiently and in constant rounds, with the following properties. If the system is solvable, $b = 1$ and $\mathbf{x} \in K^n$ and $B \in K^{n,n}$ are such that $A\mathbf{x} = \mathbf{y}$ and the columns of B generate the null-space Ker A (optionally, the non-zero columns form a basis). If the system is not solvable, $b = 0$, and $[\mathbf{x}]$, $[B]$ are both all-zero.

Our solution is based on a Random Sampling Protocol which we describe first.

8.1 Secure Random Sampling

Let $\mathbf{y}$ be given to the network, and assume for the moment that the linear system $A\mathbf{x} = \mathbf{y}$ has a solution. The purpose of the network is to securely compute $[\mathbf{x}]$, where $\mathbf{x}$ is a random solution of $A\mathbf{x} = \mathbf{y}$, efficiently and in constant rounds. Note that in particular this implies a means for the network to securely sample random elements from Ker A by setting $\mathbf{y} = \mathbf{0}$.

Our approach is to reduce the problem to that of solving a regular system, since this can be handled by the methods of Bar-Ilan and Beaver. Using the Sub-Space Membership Protocol the network first securely computes the shared decision bit $[b]$ on whether the system has a solution at all. Applying that same protocol in an appropriate way, they are able to select a linearly independent generating subset of the columns of A, and to replace the other columns by random ones. With high probability θ, this new matrix T is invertible: if r is the rank of A, then $\theta = \frac{(q^n - q^r)\cdots(q^n - q^{n-1})}{q^n} \geq \left(1 - \frac{1}{q}\right)^{n-r} \geq 1 - \frac{n}{q}$, which differs from 1 only negligibly.

This means that, with high probability, the methods of Bar-Ilan and Beaver can be applied to the system $T\mathbf{x}_1 = \mathbf{y}$ in the unknown $\mathbf{x}_1$. More precisely, they are applied to the system $T\mathbf{x}_1 = \mathbf{y} - \mathbf{y}_0$, where $\mathbf{y}_0$ is a random linear combination over the columns of A that were replaced by columns of R in the construction of T. In other words, $\mathbf{y}_0 = A\mathbf{x}_0$ for $\mathbf{x}_0$ chosen randomly such that its i-th coordinate equals 0 if $c_i = 1$.

[8] As in the Rank Protocol, the assumption that A is a square matrix is not a limitation.

If T is indeed invertible and if $A\mathbf{x} = \mathbf{y}$ has a solution at all, then the coordinates of $\mathbf{x}_1$ corresponding to the "random columns" in T must be equal to 0. Then $\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1$ is a solution of $A\mathbf{x} = \mathbf{y}$, since $A\mathbf{x} = A\mathbf{x}_0 + A\mathbf{x}_1 = \mathbf{y}_0 + T\mathbf{x}_1 = \mathbf{y}_0 + (\mathbf{y} - \mathbf{y}_0) = \mathbf{y}$. It is also clearly random, since $\mathbf{x}_1$ is unique given $\mathbf{y}$ and random $\mathbf{x}_0$.

The result of the protocol is computed as $([b], [b] \cdot [\mathbf{x}])$, where b is the decision bit computed at the beginning.

Here are the details. Write $\mathbf{k}_1, \ldots, \mathbf{k}_n$ to denote the columns of A, and set $\mathbf{k}_0 = \mathbf{0}$. Define the vector $\mathbf{c} \in K^n$ by $c_i = 1$ if $\mathbf{k}_i$ is not a linear combination of $\mathbf{k}_0, \ldots, \mathbf{k}_{i-1}$, and $c_i = 0$ otherwise. Note that $\mathcal{B} = \{\mathbf{k}_i : c_i = 1\}$ is a basis for the space generated by the columns of A.

The shared vector $[\mathbf{c}]$ is securely computed by applying the Sub-Space Membership Protocol in parallel to the pairs $([A_{i-1}], [\mathbf{k}_i])$, where A_i is the matrix consisting of the columns $\mathbf{k}_0, \ldots, \mathbf{k}_{i-1}$, and "negating" the resulting shared decision bits.

Write $[C]$ for the shared diagonal matrix with $\mathbf{c}$ on its diagonal. We 'll use this matrix as a selector as follows. After generating a random shared matrix $[R]$, the network replaces the columns in A that do not belong to the basis $\mathcal{B}$ by corresponding columns from R, by securely computing $[T] = [A] \cdot [C] + [R] \cdot ([I - C])$. As argued before, T is invertible with high probability.

Next, they securely generate a random shared vector $[\mathbf{x}_0]$ with zeroes at the coordinates i with $\mathbf{k}_i \in \mathcal{B}$, by generating $[\mathbf{x}_0']$ randomly, and securely multiplying its i-th coordinate by $[1 - c_i]$, $i = 1, \ldots, n$. The shared vector $[\mathbf{y}_0]$ is now securely computed as $[\mathbf{y}_0] = [A] \cdot [\mathbf{x}_0]$.

Using Bar-Ilan and Beaver's method for securely solving a regular system, the network computes $[\mathbf{x}_1] = [T]^{-1} \cdot ([\mathbf{y}] - [\mathbf{y}_0])$, and finally $[\mathbf{x}] = [\mathbf{x}_0] + [\mathbf{x}_1]$ and $[b] \cdot [\mathbf{x}]$. They take $([b], [b] \cdot [\mathbf{x}])$ as the result.

8.2 General Linear Systems Protocol

Let $[A]$, $[\mathbf{y}]$ be sharings, where $A \in K^{n,n}$ and $\mathbf{y} \in K^n$. If $\mathbf{x}$ is a solution of $A\mathbf{x} = \mathbf{y}$, then the complete set of solutions is given by $\mathbf{x} + \mathrm{Ker}(A)$.

Using the Random Sampling Protocol it is now an easy task for the network to securely solve a system of linear equations efficiently in constant rounds.

Assume for the moment that the system is solvable. The network first securely generates $[\mathbf{u}_1], \ldots, [\mathbf{u}_n]$, where the $\mathbf{u}_i$ are independently random samples from Ker A. With high probability, these actually generate Ker A. The network defines $[B]$ such that the i-th column of B is $\mathbf{u}_i$. Next, they securely compute $[\mathbf{x}]$, where $\mathbf{x}$ is an arbitrary solution of the linear system. The result of the protocol is $([\mathbf{x}], [B])$.

To deal with the general case, where the system may not be solvable, we first have the network securely compute $[b]$ using the Sub-Space Membership Protocol, where b is the bit that indicates whether it is solvable. After $[\mathbf{x}], [B]$ has been securely computed, they securely compute $([b] \cdot [\mathbf{x}], [b] \cdot [B])$, and take $([b], ([b] \cdot [\mathbf{x}], [b] \cdot [B]))$ as the result.

9 Achieving Perfect Correctness and Privacy

By inspection of our protocols, non-zero error probabilities arise when the network happens to select zeroes of "hidden" polynomials. Since upper-bounds on their degree are known, such errors can be avoided altogether by passing to an extension field and having the network select elements with sufficiently large algebraic degree instead. This, together with some other minor modifications, leads to protocols with perfect correctness in all cases. In [11] we study efficient alternatives with perfect privacy, thereby avoiding the need for large fields.

10 Acknowledgements

We 'd like to thank Donald Beaver, Vanessa Daza, Yuval Ishai, Erich Kaltofen, Jesper Buus Nielsen, and the anonymous referees for useful comments. Thanks to Michael Rabin for suggesting an approach based on Leverier's Lemma in Section 6.5.

References

1. J. Bar-Ilan, D. Beaver: *Non-cryptographic fault-tolerant computing in constant number of rounds of interaction*, Proc. ACM PODC '89, pp. 201-209, 1989.
2. D. Beaver, S. Micali, P. Rogaway: *The Round Complexity of Secure Protocols*, Proc. 22nd ACM STOC, pp. 503–513, 1990.
3. D. Beaver: *Minimal Latency Secure Function Evaluation*, Proc. EUROCRYPT '00, Springer Verlag LNCS, vol. 1807, pp. 335–350.
4. A. Beimel, A. Gál: *On Arithmetic Branching Programs*, J. Comp. & Syst. Sc., 59, pp. 195–220, 1999.
5. M. Ben-Or, S. Goldwasser, A. Wigderson: *Completeness theorems for non-cryptographic fault-tolerant distributed computation*, Proc. ACM STOC '88, pp. 1–10, 1988.
6. R. Canetti, U. Feige, O. Goldreich, M. Naor: *Adaptively secure multi-party computation*, Proc. ACM STOC '96, pp. 639–648, 1996.
7. D. Chaum, C. Crépeau, I. Damgård: *Multi-party unconditionally secure protocols*, Proc. ACM STOC '88, pp. 11–19, 1988.
8. R. Cramer, I. Damgård, J. Buus Nielsen: *Multiparty Computation from Threshold Homomorphic Encryption*, Proc. EUROCRYPT '01, Springer Verlag LNCS, vol. 2045, pp. 280–300., 2001.
9. R. Cramer, I. Damgård, U. Maurer: *General Secure Multi-Party Computation from any Linear Secret-Sharing Scheme*, Proc. EUROCRYPT '00, Springer Verlag LNCS, vol 1807, pp. 316–334. Full version available from IACR eprint archive, 2000.
10. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt and T. Rabin: *Efficient multiparty computations secure against an adaptive adversary*, Proc. EUROCRYPT '99, Springer Verlag LNCS, vol. 1592, pp. 311–326, 1999.
11. R. Cramer, I. Damgård, V. Daza: work in progress, 2001.
12. U. Feige, J. Kilian, M. Naor: *A Minimal Model for Secure Computation*, Proc. ACM STOC '94, pp. 554–563, 1994.

13. R. Gennaro, M. Rabin, T. Rabin: *Simplified VSS and fast-track multi-party computations with applications to threshold cryptography*, Proc. ACM PODC'98, pp. 101–111, 1998.
14. O. Goldreich, S. Micali and A. Wigderson: *How to play any mental game or a completeness theorem for protocols with honest majority*, Proc. ACM STOC '87, pp. 218–229, 1987.
15. M. Hirt, U. Maurer: *Player simulation and general adversary structures in perfect multi-party computation*, Journal of Cryptology, vol. 13, no. 1, pp. 31–60, 2000. (Preliminary version in Proc. ACM PODC'97, pp. 25–34, 1997)
16. Y. Ishai, E. Kushilevitz: *Private Simultaneous Messages Protocols with Applications*, Proc. 5th Israel Symposium on Theoretical Comp. Sc. (ISTCS '97), pp. 174–183, 1997.
17. Y. Ishai, E. Kushilevitz: *Randomizing Polynomials: A New Paradigm for Round-Efficient Secure Computation*, Proc. of FOCS '00, 2000.
18. M. Karchmer, A. Wigderson: *On span programs*, Proc. of Structure in Complexity '93, pp. 102–111, 1993.
19. J. Kilian: *Founding Cryptography on Oblivious Transfer*, Proc. ACM STOC '88, pp. 20-31, 1988.
20. F. T. Leighton: *Introduction to Parallel Algorithms and Architectures: Arrays–Trees–Hypercubes*, Morgan Kaufmann Publishers, 1992.
21. M. Mahajan and V. Vinay: *Determinant: combinatorics, algorithms and complexity*, Chicago J. Theoret. Comput. Sci., Article 5, 1997.
22. K. Mulmuley: *A Fast Parallel Algorithm to Compute the Rank of a Matrix over an Arbitrary Field*, Combinatorica, vol. 7, pp. 101–104, 1987.
23. T. Rabin, M. Ben-Or: *Verifiable secret sharing and multi-party protocols with honest majority*, Proc. ACM STOC '89, pp. 73–85, 1989.
24. T. Rabin: *Robust sharing of secrets when the dealer is honest or cheating*, J. ACM, 41(6):1089-1109, November 1994.
25. A. Yao: *Protocols for Secure Computation*, Proc. IEEE FOCS '82, pp. 160–164, 1982.