Efficient Disjointness Tests for Private Datasets

Qingsong Ye[1], Huaxiong Wang[1,2], Josef Pieprzyk[1], and Xian-Mo Zhang[1]

[1] Centre for Advanced Computing – Algorithms and Cryptography
Department of Computing, Macquarie University, NSW 2109, Australia
{qingsong,hwang,josef,xianmo}@ics.mq.edu.au
[2] Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore

Abstract. We present efficient protocols for private set disjointness tests. We start from an intuition of our protocols that applies Sylvester matrices. Unfortunately, this simple construction is insecure as it reveals information about the cardinality of the intersection. More specifically, it discloses its lower bound. By using the Lagrange interpolation we provide a protocol for the honest-but-curious case without revealing any additional information. Finally, we describe a protocol that is secure against malicious adversaries. The protocol applies a verification test to detect misbehaving participants. Both protocols require $O(1)$ rounds of communication. Our protocols are more efficient than the previous protocols in terms of communication and computation overhead. Unlike previous protocols whose security relies on computational assumptions, our protocols provide information theoretic security. To our knowledge, our protocols are first ones that have been designed without a generic secure function evaluation. More importantly, they are the most efficient protocols for private disjointness tests for the malicious adversary case.

Keywords: Private Set Disjointness, Private Matching, Secure Multi-Party Computation.

1 Introduction

Suppose two parties, Alice and Bob, each has a private dataset of some items denoted by A and B, respectively. Alice wishes to learn whether these two sets are disjoint, that is, whether $A \cap B = \emptyset$ or not. In doing so, Alice does not want to reveal any information about her set A to Bob, who, in turn, does not wish to reveal any information about his set B, other than whether $A \cap B = \emptyset$ or not. This is called a *private disjointness test* [1].

A private disjointness test is a useful primitive in various online service applications. For example, Bob is a club owner offering a special-status membership called "Super Fun" and Alice would like to know whether she is eligible for membership. Alice has a smart card issued by the state authority containing her resident address, her age band (assuming that 0 for age $0-9$, 1 for the age $11-19$, 2 for the age $20-29$ and so on), her membership status, etc. Bob determines whether Alice is eligible for the special-status membership based on Alice's attribute information. For example, Bob may require that at least one of the following three conditions holds: (1) Alice lives in the same suburb as Bob; (2) Alice's age band is 5; (3) Alice is the member of Good Credit Union.

Bob considers the detail of his policy to be commercial secret and does not want to reveal it to others. Alice is interested in this membership and would like to go forward; however, she wants to reveal as little information about her as possible. On the other hand, Bob wants Alice to know only whether she is eligible for the membership, but nothing else.

There are several protocols to tackle this problem, such as Freedman, Nissim and Pinkas (FNP) [2], Hohenberger and Weis (HW) [3] and Kiayias and Mitrofanova (KM) [1]. The KM protocols have either high round complexity or high communication complexity, while the FNP and HW protocols leak the information about the intersection cardinality. Moreover, both FNP and KM protocols require random oracles and costly sub-protocols that have to be secure in the presence of a malicious adversaries. The HW protocol only considers the malicious Bob and assumes the honest Alice in order to make the protocol efficient. This paper provides efficient protocols for private disjointness tests. The protocols are unconditionally secure against malicious adversaries.

Related Work. Freedman, Nissim and Pinkas (FNP) [2] proposed a protocol for the private computation of set disjointness. The protocol is based on the representation of datasets as roots of a polynomial and applies oblivious polynomial evaluation techniques [4]. The protocol simply lets Alice represent her dataset $A = \{a_1, \ldots, a_m\}$ over a field as a polynomial $\mathcal{F}(y) = \prod_{a_i \in A} (y - a_i) = \sum_{i=0}^{m} \alpha_i y^i$ in that field. Alice then encrypts coefficients of $\mathcal{F}$ with a homomorphic cryptosystem such as Paillier's [5]. Thus, given encrypted coefficients of $\mathcal{F}$, Bob first evaluates $\mathcal{F}(b_i)$ for each elements $b_i \in B$, and then returns encrypted $\gamma\mathcal{F}(b_i)$ where γ is a random non-zero value picked by Bob. Note that any $b_i \in A$ if and only if $\mathcal{F}(b_i) = 0$, which not only indicates the disjointness status but also reveals the information of the intersection cardinality.

The FNP construction leads to a very efficient protocol assuming honest-but-curious adversaries. This construction heavily influences two other related works of Kiayias and Mitrofanova [1] and of Hohenberger and Weis [3]. To cope with malicious adversaries, the FNP protocol employs random oracle and invokes expensive sub-protocols.

Hohenberger and Weis [3] have taken a similar approach to the one given in [2] and designed a protocol using an oblivious polynomial evaluation. The security proof relies on the difficulty of discrete logarithm. Assume $\mathbb{G}$ is a group with the composite order $n = pq$ where $p < q$ are primes. Let g, u be random generators of $\mathbb{G}$ and $h = u^q$. As in the FNP protocol, Alice represents her dataset A by the polynomial $\mathcal{F}(y) = \sum_{i=0}^{|A|} \alpha_i y^i \in \mathbb{Z}_q[y]$, chooses a random polynomial $R(x) = \sum_{i=0}^{|A|} r_i x^i \in \mathbb{Z}_p[x]$ and publishes n and commitments of $\mathcal{F}(y)$, $g^{\alpha_i} h^{r_i}$ for $i \in [0, \ldots, |A|]$. For each $b_j \in B$ selected in random order, Bob obliviously evaluates $v_j = g^{\mathcal{F}(b_j)} h^{R(b_j)}$ and sends $w_j = v_j^{\gamma_j}$ to Alice, where γ_j is a non-zero value randomly picked from $\mathbb{Z}_n^*$. Note that if $b_j \in A$, then $g^{\gamma_j \mathcal{F}(b_j)}$ will have order p. Since h has order p, Alice concludes $A \cap B \neq \emptyset$ if $w_j^p = 1$ with overwhelming probability.

The protocol is efficient and secure without using the random oracle. The security, however, is proven under the assumption that Alice is honest (but Bob can be malicious). If both Alice and Bob are malicious, then the cost of the protocol is the same as in the FNP protocol. Moreover, their security properties are the same as of the FNP protocol and allow Alice to discover the intersection cardinality. In our membership example, if Alice knows the intersection cardinality, she may learn some extra information about Bob's business policy which is against Bob's will.

Kiayias and Mitrofanova [1] provided three protocols for private set disjointness tests. The first protocol assumed that the domain is relatively small, which is not relevant to our work. Our work is related to their second and third protocols, denoted by $\mathrm{KM}-2$ and $\mathrm{KM}-3$, respectively. $\mathrm{KM}-2$ uses a new primitive called *superposed encryption* based on Pedersen commitments [6]. Superposed encryption is closely related to a homomorphic ElGamal variant first used in voting schemes by Cramer et.al. [7]. In the $\mathrm{KM}-2$ protocol, Bob returns to Alice a single ciphertext of $\gamma|A \cap B|$, where γ is a random non-zero value. This protocol needs $|B|$ rounds of communication between two parties. The total communication cost is $|A| \cdot |B|$ if the adversary is honest but curious, but increases by a quadratic factor if either party behaves maliciously. To reduce the high round complexity in $\mathrm{KM}-2$, the authors presented the $\mathrm{KM}-3$ protocol that uses a multivariate polynomial so the task can be done in a single round. The price to be paid is a high communication cost $\Theta(\binom{|A|+|B|}{|B|})$ for the honest-but-curious case. The disadvantage of those two protocols is obvious. It is unlikely for causal clients to use such online services which require either extensive network communication or numerous interactions.

Kissner and Song [8] presented FNP-inspired schemes for various private set operations such as set intersection, set union, threshold cardinality of the set intersection, and multiplicity tests. The problem of secure computation of the subset relation of two private datasets is a variant of the private set intersection problem where the intersection content is one party's whole dataset. This operation can be computed by extending the FNP protocol. The applications of the subset relation were discussed in [9, 10]. Protocols for private equality tests are a special case of the private disjointness tests, where each party has a single element in the dataset. These protocols were proposed in [11, 4, 12]. The distributed case of private equality tests and various private set operations were considered in [13, 14].

Secure determinant computation by multiple parties is discussed in [15]. The secure shares computation and distribution of a matrix are based on the Lagrange interpolation. Using similar technique, Mohassel and Franklin [16] proposed a multi-party computation protocol to securely test whether two shared polynomials are coprime. Their construction applies Sylvester matrices [17] construction.

Our Results: We present two disjointness test protocols. Each protocol takes $O(1)$ rounds. The second protocol that provides verifiability, is secure against malicious adversaries, and the parties learn nothing more than the desired result. In our construction, we build two polynomials g and h whose roots are representing the datasets A and B of the two parties, respectively. The polynomials are next used to form a Sylvester matrix. The determinant of the matrix tells us whether g and h share any root and therefore allows us to ascertain if the intersection of the datasets is empty or not.

We first give an intuition of our protocols that applies a Sylvester matrix directly. However, this simple construction is not secure as Alice can learn the intersection cardinality by computing the rank of the matrix. Note, this is allowed in [2, 3].

To reduce the amount of information leaking about the intersection of sets, we can modify the simple construction as the following. We let two parties cooperate to multiply the Sylvester matrix and its transpose. In such a way that Alice still knows whether the determinant of the related Sylvester matrix is zero. Consequently, this improved version reveals the lower bound of the intersection cardinality only.

To achieve no information leaks apart from the fact that whether $A \cap B = \emptyset$ or not, we utilize a secure determinant evaluation scheme in a multi-party computation setting developed by Cramer and Damgard [15]. In this protocol, Bob randomly picks $|A| + |B| + 1$ distinct indexes and forwards them to Alice along with the shares of his dataset. Alice then constructs the corresponding $|A| + |B| + 1$ shares of the masked Sylvester matrix associated with g and h. Using the Lagrange interpolation, Alice is able to test if the determinant of the masked Sylvester matrix is zero or not. This approach requires $O((|A| + |B|)^2)$ communication cost and $O((|A| + |B|)^{3.697})$ field operations.

We then further employ a verification test to detect misbehaving participants. The test is going to double the communication cost.

The advantage of our solution is that our protocols are conceptually simple. Comparing to the previous work, our protocols are very efficient. In particular, our solution can deal with malicious Bob and malicious Alice at same time. Unlike the previous solutions, our schemes provide unconditional security. Our approach is of a great advantage, where the communication facilities are in a short supply and consequently, protocols with small number of rounds are preferred. Our protocols do not leak any information apart from whether $A \cap B = \emptyset$ or not.

Our paper is organized as follows. In Section 2, we introduce the notations, Sylvester matriices and some techniques that will be used in this paper. In Section 3, we discuss the adversary model and define the problem in hand. A general description of a simple and insecure protocol that is based on Sylvester matrices is presented in Section 4. In Section 5, we show our main protocols for the private disjointness test of two datasets based on the Sylvester matrix construction and demonstrate its security. We also analyze the efficiency of our protocols in this section. Finally, we give concluding remarks in Section 6.

2 Preliminaries

Throughout this paper, let $GL_n(K) \subset K^{n,n}$ denote the group of $n \times n$ non-singular matrices over an arbitrary finite field K. We assume that the number of elements in the field $q = |K|$ is much larger than the dimension n.

2.1 Sylvester Matrix

Given two polynomials $g(x) = \sum_{i=0}^{m} \alpha_i x^i \in \mathbb{Z}_q[x]$ and $h(x) = \sum_{i=0}^{n} \beta_i x^i \in \mathbb{Z}_q[x]$ of degrees m and n, respectively. The Sylvester matrix S associated with g and h is then the $(m + n) \times (m + n)$ matrix obtained as follows:

- The first row is: $(\alpha_m, \alpha_{m-1}, \ldots, \alpha_0, 0, \ldots, 0)$.
- The next row is obtained from the previous one by shifting it one position (column) to the right and putting zero in the first position.

– This process is repeated $n-2$ times.
– The $(n+1)^{\text{th}}$ row is $(\beta_n, \beta_{n-1}, \ldots, \beta_0, 0, \ldots, 0)$.
– Next $m-1$ rows are created in the same way as for the first row. The only difference is the number of rows.

For example, the Sylvester matrix S associated with g and h for $m=4$ and $n=3$ is:

$$S = \begin{pmatrix} \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 & 0 \\ 0 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 & 0 \\ 0 & 0 & \alpha_4 & \alpha_3 & \alpha_2 & \alpha_1 & \alpha_0 \\ \beta_3 & \beta_2 & \beta_1 & \beta_0 & 0 & 0 & 0 \\ 0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 & 0 & 0 \\ 0 & 0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 & 0 \\ 0 & 0 & 0 & \beta_3 & \beta_2 & \beta_1 & \beta_0 \end{pmatrix}.$$

Thus, the determinant of the associated Sylvester matrix is defined by the two associated polynomials g and h. Consequently, two polynomials do not share a common root if and only if the determinant of the Sylvester matrix is non-zero value. If the determinant of the Sylvester matrix is zero, then the rank of the Sylvester matrix determines the degree of the greatest common divisor of g and h. That is:

$$\deg(\gcd(g,h)) = m + n - \operatorname{rank}(S).$$

2.2 Building Blocks

In general, any secret sharing scheme can be used in our protocol. Since there are only two parties involved in our protocol, we assume that (2-out-of-2)-Shamir secret sharing is used. The computations in this paper are carried out over a finite field K. The two parties are Alice and Bob. We frequently use the following building blocks from [18] and [15].

Secure Inversion of Shared Field Elements and Matrices is a protocol that accepts a list of shares of an invertible field element or matrix as its input and generates a list of shares of the inverse. We denote this secure computation of shares of the inverse by $[x^{-1}]_i = [x]_i^{-1}$, and $[M^{-1}]_i = [M]_i^{-1}$ respectively for an element x and a matrix M, where $[x^{-1}]_i$'s are shares of the inverse, $[x]_i^{-1}$'s are the inverse of shares, and $i \in \{A, B\}$ in our protocol. In our protocols, we slightly modify the original protocol to let only one party compute such inverses as the following.

Input: Shares $[x]_A, [x]_B$ of the element x.
Output: Shares $[x^{-1}]_A, [x^{-1}]_B$ of the inverse element x^{-1}.
Protocol:

1. Compute shares $[\rho]_A, [\rho]_B$ of an element $\rho \in K$ that is random and non-zero,
2. Compute $[\sigma]_A = [\rho]_A \cdot [x]_A$ and $[\sigma]_B = [\rho]_B \cdot [x]_B$,
3. Calculate σ from the shares $[\sigma]_A$ and $[\sigma]_B$,
4. Find $[x^{-1}]_A = \sigma^{-1} \cdot [\rho]_A$ and $[x^{-1}]_B = \sigma^{-1} \cdot [\rho]_B$.

Note that the other party i, who receives the pair $[x]_i$ and $[x^{-1}]_i$, cannot find any information about x. This is also true for the matrix M. For simplicity, we denote

$[\sigma]_i = [\rho]_i \cdot [x]_i$ in Step 2. Actually, the computation of $[\sigma]_i$ is not simple and we need to employ an appropriate sub-protocol such as the one presented in Section 1.1 of [18]. Although the secure computation of $[\sigma]_i$ is not required in this protocol, but it is necessary in the next protocols where the appropriate sub-protocol is applied. A constant-round sub-protocol between Alice and Bob might be also needed if a secure computation of $[\sigma]_i$ is expected.

Secure Multiplication of Shared Field Elements is a protocol that produces a share of the product of two shared field elements $[x \cdot y]_A, [x \cdot y]_B$ of x and y. The protocol can be successfully run if all shares are invertible. It proceeds according to the following steps:
Input: Alice and Bob hold their shares of two elements x and y, i.e. Alice has $[x]_A, [y]_A$ and Bob owns $[x]_B$ and $[y]_B$.
Output: Alice gets the shares $[x \cdot y]_A, [x \cdot y]_B$.
Protocol:

1. Alice
   (a) generates shares $[\rho_1]_A, [\rho_1]_B$ of ρ_1, and $[\rho_2]_A, [\rho_2]_B$ of ρ_2 independently at random from all non-zero values.
   (b) computes $[\sigma_1]_A = [x]_A \cdot [\rho_1]_A$, and $[\sigma_2]_A = [\rho_1]_A^{-1} \cdot [y]_A \cdot [\rho_2]_A$,
   (c) sends $[\rho_1]_B, [\rho_2]_B, [\rho_1]_B^{-1}, [\sigma_1]_A, [\sigma_2]_A$ to Bob.
2. Bob
   (a) computes $[\sigma_1]_B = [x]_B \cdot [\rho_1]_B$, and $[\sigma_2]_B = [\rho_1]_B^{-1} \cdot [y]_B \cdot [\rho_2]_B$,
   (b) constructs σ_1, σ_2 from computed shares,
   (c) sends $(\sigma_1 \cdot \sigma_2)$ to Alice.
3. Alice computes $[x \cdot y]_A = \sigma_1 \cdot \sigma_2 \cdot [\rho_2]_A^{-1}$ and $[x \cdot y]_B = \sigma_1 \cdot \sigma_2 \cdot [\rho_2]_B^{-1}$.

Note that only Alice could compute $[x \cdot y]_A$ and $[x \cdot y]_B$. Consequently, Alice learns the result of $x \cdot y$. This is allowed in our protocol. The security requirement of our protocol is that Alice learns $x \cdot y$ without knowing the value of x and/or y.

In general, if one of the inputs is zero, then Ben-Or and Cleve showed in [19] how to modify the protocol given above.

Secure Shared Matrix Multiplication is a protocol that securely generates shares $[M \cdot M']_A, [M \cdot M']_B$ for Alice and Bob respectively, from shares $[M]_A, [M]_B$ of a matrix M, and $[M']_A, [M']_B$ of M', where $[M]_A, [M']_A$ are held by Alice and $[M]_B, [M']_B$ are possessed by Bob. This protocol works in an obvious way following the previous *Secure Multiplication of Shared Field Elements* protocol.

Secure Determinant Evaluation *(SDE)* computes the determinant of a matrix $M \in K^{n,n}$ from a list of related non-singular matrices. Let $z_0, \ldots, z_n$ are distinct and random integers selected from K. We simplify the technique of secure determinant evaluation in the multiparty computation model introduced by Cramer and Damgard [15], and we use the following equation

$$\det(M) = (-1)^n \cdot \sum_{i=0}^{n} \left(\left(\prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{z_j}{z_i - z_j} \right) \cdot \det(z_i I_{m+n} - M) \right),$$

where I_n denotes the $n \times n$ identity matrix. For each $z_i \in K$, it holds that $(z_i I_n - M) \in GL_n(K)$ if and only if z_i is not an eigenvalue of M. Since M has at most n eigenvalues, each matrix $z_i I_n - M$ is invertible, when z_i is randomly and independently chosen, except with the probability at most $\frac{n}{q}$.

3 Model and Definition

This section formally defines our verifiable disjointness test of two private datasets. Our construction can be described as follows. Let Alice $\mathcal{P}_{\mathcal{A}}$ and Bob $\mathcal{P}_{\mathcal{B}}$ be two probabilistic polynomial time interactive Turing machines. Let $A = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_n\}$ be datasets owned by $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{B}}$, respectively. We assume that the set cardinalities $|A|$ and $|B|$ are not secret. The private disjointness test checks whether $A \cap B = \emptyset$ or not. For sets $A, B \subset K$, define the disjointness predicate $\mathcal{D}(A, B) = (A \cap B = \emptyset)$, that is, $\mathcal{D}(A, B)$ will have value 1 if and only if A and B are disjoint otherwise, the predicate is equal zero. The interaction between $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{B}}$ yields a result that is known to $\mathcal{P}_{\mathcal{A}}$ only.

In our model, an adversary can be misbehaving Bob, misbehaving Alice or both. In particular, we cannot hope to avoid parties that (i) refuse to participate in the protocol, (ii) substitute a correct input by an arbitrary value, and (iii) abort the protocol any time. In our work, we do not address these issues. The way that security is dealt in this case is by comparing the player's views with respect to an "ideal" protocol implementation, using a trusted third party. The reader is referred to [20] for a more complete discussion.

Definition 1. (Private Disjointness Testing) *Two probabilistic polynomial time interactive Turing machines, $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{B}}$, define a Private Disjointness Testing protocol if the following conditions hold:*

Completeness. *If both parties are honest, the protocol works and $\mathcal{P}_{\mathcal{A}}$ learns the disjointness predicate, that is whether $A \cap B = \emptyset$.*

Soundness. *For an unknown $\mathcal{P}_{\mathcal{A}}$'s set $A \subset K$, the probability that $\mathcal{P}_{\mathcal{B}}$ will convince $\mathcal{P}_{\mathcal{A}}$ to accept $A \cap B \neq \emptyset$ is negligible.*

Security. *Assume that the size of both datasets are public. With an overwhelming probability, $\mathcal{P}_{\mathcal{A}}$ does not get any extra information about $\mathcal{P}_{\mathcal{B}}$'s dataset beyond the knowledge of the disjointness predicate. $\mathcal{P}_{\mathcal{B}}$ learns nothing about $\mathcal{P}_{\mathcal{A}}$'s set.*

Informally, completeness means that a correct execution between two honest parties will return the correct value of the disjointness predicate to $\mathcal{P}_{\mathcal{A}}$. The soundness implies that on an unknown input set $A \subset K$ for $\mathcal{P}_{\mathcal{B}}$, $\mathcal{P}_{\mathcal{A}}$ has no chance of obtaining a non-zero result when interacting with any malicious Bob $\mathcal{P}^*_{\mathcal{B}}$. That is, unless $\mathcal{P}^*_{\mathcal{B}}$ actually knows a value in $\mathcal{P}_{\mathcal{A}}$'s set, $\mathcal{P}_{\mathcal{A}}$ will not be fooled into thinking otherwise. As pointed out in [3], both FNP and KM protocols are not sound according to this definition. In those schemes, $\mathcal{P}_{\mathcal{A}}$ will believe that there is an intersection if it receives the value zero encrypted under a public-key. $\mathcal{P}^*_{\mathcal{B}}$ could trivially violate the soundness property by encrypting a zero value itself.

In a verifiable protocol, $\mathcal{P}_\mathcal{A}$'s privacy requires that no malicious Bob $\mathcal{P}_\mathcal{B}^*$ can learn anything about the set A beyond $|A|$ from an interaction with $\mathcal{P}_\mathcal{A}$. Using the same argument for a malicious Alice $\mathcal{P}_\mathcal{A}^*$, $\mathcal{P}_\mathcal{B}$'s privacy ensures that $\mathcal{P}_\mathcal{A}^*$ does not learn anything about B beyond the set cardinality.

4 Intuition of Set Disjointness Test from Sylvester Matrix Construction

Our solution is based on the Sylvester matrix construction. To test if $\mathcal{P}_\mathcal{A}$'s dataset $A = \{a_1, \ldots, a_m\}$ and $\mathcal{P}_\mathcal{B}$'s dataset $B = \{b_1, \ldots, b_n\}$ are disjoint, we represent two datasets as two polynomials $g(x) = \prod_{a_i \in A}(x - a_i) = \sum_{i=0}^{m} \alpha_i x^i$ and $h(x) = \prod_{b_j \in B}(x - b_j) = \sum_{j=0}^{n} \beta_j x^j$, respectively. As in Section 2.1, we can build a Sylvester matrix S from the polynomials g and h. Then, the determinant of S indicates whether A and B are disjoint.

In order to protect datasets privacy, we can let $\mathcal{P}_\mathcal{A}$ send encrypted g to $\mathcal{P}_\mathcal{B}$ by using a public-key homomorphic cryptosystem, such as Paillier's [5], where the encrypted g is denoted as the encryption of g's coefficients with $\mathcal{P}_\mathcal{A}$'s public key. $\mathcal{P}_\mathcal{B}$ then constructs the Sylvester matrix based on the polynomial h and encrypted polynomial g. To protect the privacy of the polynomial h, $\mathcal{P}_\mathcal{B}$ randomly selects $R_1 \in GL_{m+n}(K)$ and obliviously computes $R_1 \cdot S$ by using the homomorphic properties of the encryption applied. After receiving the cryptogram of $R_1 \cdot S$, $\mathcal{P}_\mathcal{A}$ decrypts it and is able to compute $\det(R_1 \cdot S)$. In such a way, $\mathcal{P}_\mathcal{A}$ learns $\det(R_1 \cdot S) = 0$ if and only if $\det(S) = 0$ without leaking any information about the polynomial g and gaining no other information apart from the disjointness of two datasets.

However, if we apply this idea directly to construct a protocol, then $\mathcal{P}_\mathcal{A}$ can learn the intersection cardinality. This is because $\text{rank}(R_1 \cdot S) = \text{rank}(S)$. Thus, $\deg(\gcd(g, h)) = \deg(g) + \deg(h) - \text{rank}((R_1 \cdot S))$ which reveals $|A \cap B|$. However, with slightly bigger communication cost, we could let $\mathcal{P}_\mathcal{A}$ learn only the lower bound of the intersection by securely computing $\det(S^T \cdot S)$. It is easy to see that $\mathcal{P}_\mathcal{A}$ is still able to determine whether $\det(S)$ is zero or not from the computation. The fact is that $\text{rank}(S^T \cdot S) \le \text{rank}(S)$. Denote $S = \begin{pmatrix} M_B \\ M_A \end{pmatrix}$, then,

$$S^T \cdot S = M_A^T \cdot M_A + M_B^T \cdot M_B$$

where $M_A^T \cdot M_A$ and $M_B^T \cdot M_B$ can be computed independently by $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{B}$. The secure computation of $\det(S^T \cdot S)$ works in the same way as the one discussed above.

5 Private Disjointness Test

In this section, we propose a solution to test the disjointness without releasing any extra information beyond $|A|$ and $|B|$. Our private computation is based on the Sylvester matrix construction and the technique of secure determinant evaluation in the multi-party computation model introduced by Cramer and Damgard [15]. Let polynomials g and h represents the datasets A and B.

5.1 Protocol without Bob-Verifiability

To secure construct a Sylvester matrix S from the polynomials g and h, and accordingly evaluate if $\det(S) = 0$, we employ the SDE technique. We form a list of $(\deg(g) + \deg(h) + 1)$ shares of S held by two parties in a certain way to let one party to compute $\det(S)$ without knowing the $\mathrm{rank}(S)$. The protocol runs according to the following steps.

Input: $\mathcal{P}_\mathcal{A}$ and $\mathcal{P}_\mathcal{B}$ hold the datasets A and B, respectively.
Output: $\mathcal{P}_\mathcal{A}$ learns if $A \cap B = \emptyset$.
Protocol Π_1

1. $\mathcal{P}_\mathcal{A}$ constructs the polynomial g from the dataset A, computes shares $[g]_A, [g]_B$ of g, and sends $[g]_B$ to $\mathcal{P}_\mathcal{B}$.
2. $\mathcal{P}_\mathcal{B}$
   (a) constructs the polynomial h from the dataset B, computes shares $[h]_A, [h]_B$ of h, and forms an $m \times (m+n)$ half Sylvester matrix $[M_B]_B$ related to $[h]_B$ as

$$\begin{pmatrix} [\beta_n]_B & [\beta_{n-1}]_B & \dots & [\beta_0]_B & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & [\beta_n]_B & \dots & [\beta_1]_B & [\beta_0]_B & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & [\beta_n]_B & [\beta_{n-1}]_B & [\beta_{n-2}]_B & \dots & [\beta_0]_B & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & [\beta_n]_B & \dots & [\beta_2]_B & [\beta_1]_B & [\beta_0]_B \end{pmatrix},$$

   (b) generates shares $[R]_A, [R]_B$ for a random matrix $R \in GL_{m+n}(K)$ in a certain way that both $[R]_A$ and $[R]_B$ are invertible (the reader is refered to [15] for more information). Let $d = \det(R)$,
   (c) forms an $n \times (m+n)$ half Sylvester matrix $[M_A]_B$ from received $[g]_B$ as in Step $2(a)$,
   (d) randomly selects distinct non-zero $z_0, \dots, z_{m+n}$ from the field K, and assigns $[z_i]_A = [z_i]_B$ for each z_i,
   (e) sends $[h]_A, [R]_A, d^{-1}, [z_0]_A, \dots, [z_{m+n}]_A$ to $\mathcal{P}_\mathcal{A}$.
3. $\mathcal{P}_\mathcal{B}$ assists $\mathcal{P}_\mathcal{A}$ in computing $[S_i']_A = [R]_A \cdot ([z_i]_A \cdot I_{m+n} - \begin{pmatrix} [M_B]_A \\ [M_A]_A \end{pmatrix})$, $[S_i']_B = [R]_B \cdot ([z_i]_B \cdot I_{m+n} - \begin{pmatrix} [M_B]_B \\ [M_A]_B \end{pmatrix})$ separately as in Sect. 2.2, where the matrices $[M_A]_A, [M_B]_A$ are constructed by $\mathcal{P}_\mathcal{A}$ in the same way as $[M_A]_B, [M_B]_B$.
4. $\mathcal{P}_\mathcal{A}$
   (a) computes S_i' from shares $[S_i']_A, [S_i']_B$ and further computes $\det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix}) = \det(S_i') \cdot d^{-1}$, where $S_i' = R \cdot (z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix})$,

(b) concludes $A \cap B \neq \emptyset$ if and only if

$$\sum_{i=0}^{m+n}\left(\left(\prod_{\substack{0\leq j\leq m+n\\ j\neq i}}\frac{z_j}{z_i - z_j}\right)\cdot \det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix})\right) = 0.$$

Theorem 1. *The construction of Protocol Π_1 is correct and secure with no other information revealed beyond $|A|$ and $|B|$ if both parties follow the protocol faithfully.*

Poof. The soundness proof is irrelevant to this protocol based on Definition 1, since $\mathcal{P}_\mathcal{B}$ is honest-but-curious and follows the protocol faithfully.

Completeness. The completeness of this protocol is clear. This is ensured by the Sylvester matrix construction. $\det(\begin{pmatrix} M_B \\ M_A \end{pmatrix}) = 0$ if and only if related polynomials g and h share common root(s), in other word $A \cap B \neq \emptyset$. The correct computation of $\det(\begin{pmatrix} M_B \\ M_A \end{pmatrix})$ from related $m + n + 1$ matrices is provided by Cramel and Damgard's SDE scheme. The associated shares construction and computation are guaranteed by the Shamir secret sharing scheme.

Security. The privacy of $\mathcal{P}_\mathcal{A}$'s g is unconditional. It is guaranteed by the perfectness of Shamir secret sharing, since $\mathcal{P}_\mathcal{B}$ only knows partial share of g owned by $\mathcal{P}_\mathcal{A}$.

$\mathcal{P}_\mathcal{B}$'s security ensures that $\mathcal{P}_\mathcal{A}$ given $S'_i = R \cdot (z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix})$ cannot learn anything about B beyond $|B|$.

The proof of $\mathcal{P}_\mathcal{B}$'s security is that an honest-but-curious $\mathcal{P}^*_\mathcal{A}$ is not able to glean any information about B from the result of $R \cdot (z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix})$ with unknown matrices R and M_B , where $R \in GL_{m+n}(K)$ is random, M_B is an $m \times (m + n)$ matrix with a half Sylvester matrix form. $\mathcal{P}^*_\mathcal{A}$ can launch an attack on M_B with

$$S'_i = R \cdot (z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix}) \tag{1}$$

Denote $\begin{pmatrix} \hat{M}_B \\ \hat{M}_A \end{pmatrix} = (z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix})$ where M_B and $\hat{M}_B$ are same size. $\mathcal{P}^*_\mathcal{A}$ knows S'_i and $\hat{M}_A$, and tries to find out the matrix $\hat{M}_B$ (really just one row of the entry, the polynomial h). Note that $\begin{pmatrix} \hat{M}_B \\ \hat{M}_A \end{pmatrix}$ is non-singular, and $R \in GL_{m+n}(K)$. Therefore, S'_i must be non-singular. By only knowing $\hat{M}_A$ and with no knowledge about h, $\mathcal{P}^*_\mathcal{A}$ can search possible candidate polynomial, which can assure $\begin{pmatrix} \hat{M}_B \\ \hat{M}_A \end{pmatrix}$ be non-singular (in other words, can satisfy Equation 1).

Non-singular $\begin{pmatrix} \hat{M}_B \\ \hat{M}_A \end{pmatrix}$ means that $\det \begin{pmatrix} \hat{M}_B \\ \hat{M}_A \end{pmatrix} \neq 0$. Let $\det \begin{pmatrix} \hat{M}_B \\ \hat{M}_A \end{pmatrix} = f(\beta_0, \beta_1 \ldots, \beta_n)$ where f is a polynomial with $n+1$ unknowns. For any β_j by fixing $\beta_i, 0 \leq i \leq n$ and $i \neq j$, $\deg(f) = n$ and there are at most n solutions for $f(\ldots, \beta_j, \ldots) = 0$. We know that there are q possible selections for β_j in the field. Therefore, there must exists at least $q - n$ possible choices for β_j, such that $f(\ldots, \beta_j, \ldots) \neq 0$. Since the polynomial f has $n+1$ unknowns, the total possible candidates for $\hat{M}_B$ are $(q-n)^{n+1}$. If q is large enough, $\mathcal{P}_{\mathcal{A}}^*$ only has a negligible probability to guess h correctly.

5.2 Verifiable Disjointness Test Protocol

In order to deal with a malicious $\mathcal{P}_{\mathcal{B}}$, $\mathcal{P}_{\mathcal{A}}$ needs to verify whether the matrix M_B associated with the shared polynomial h has the full rank as he claims to prevent the malicious $\mathcal{P}_{\mathcal{B}}$ inserting one row zeros or two dependent rows in the matrix. In the following, we show how to modify our previous protocol to gain security against malicious $\mathcal{P}_{\mathcal{B}}$ with a verification test. Assume that $\deg(h)$ is known by $\mathcal{P}_{\mathcal{A}}$. Otherwise, $\mathcal{P}_{\mathcal{B}}$ needs to send a single value $\deg(h)$ to $\mathcal{P}_{\mathcal{A}}$ at the beginning of the protocol. Suppose that $\mathcal{P}_{\mathcal{A}}$ has a private and random permutation function π, which permutes each of $m+n$ tuples.

Input: $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{B}}$ hold the datasets A and B, respectively.
Output: $\mathcal{P}_{\mathcal{A}}$ learns if $A \cap B = \emptyset$.
Protocol Π_2

1. $\mathcal{P}_{\mathcal{A}}$
   (a) constructs the polynomial g from the dataset A, and computes n pairs of shares $\{([g]_{1_A}, [g]_{1_B}), \ldots, ([g]_{(m+n)_A}, [g]_{(m+n)_B})\}$, where the combination of two shares in any pair can find g,
   (b) sets an constant polynomial $g' = 1$, and computes n pairs of shares as in previous step, so she gets $\{([g']_{1_A}, [g']_{1_B}), \ldots, ([g']_{(m+n)_A}, [g']_{(m+n)_B})\}$,
   (c) obtains $\{(e_{1_{\pi_1(1)}}, e_{1_{\pi_1(2)}}), \ldots, (e_{(m+n)_{\pi_{m+n}(1)}}, e_{(m+n)_{\pi_{m+n}(2)}})\}$ by performing $\pi\{(e_{1_1}, e_{1_2}), \ldots, (e_{(m+n)_1}, e_{(m+n)_2})\}$, where $\{(e_{1_1}, e_{1_2}), \ldots, (e_{(m+n)_1}, e_{(m+n)_2})\} = \{([g]_{1_B}, [g']_{1_B}), \ldots, ([g]_{(m+n)_B}, [g']_{(m+n)_B})\}$,
   (d) sends $\{(e_{1_{\pi_1(1)}}, e_{1_{\pi_1(2)}}), \ldots, (e_{(m+n)_{\pi_{m+n}(1)}}, e_{(m+n)_{\pi_{m+n}(2)}})\}$ to $\mathcal{P}_{\mathcal{B}}$;
2. For each pair $(e_{i_{\pi_j(1)}}, e_{i_{\pi_j(2)}})$, the protocol runs step 2 and 3 of Protocol Π_1 parallel with the same parameters and computes

$$[S_i']_{\pi_j(1)_A} = [R]_A \cdot ([z_i]_A \cdot I_{m+n} - \begin{pmatrix} [M_B]_A \\ [M_A]_{i_{\pi_j(1)_A}} \end{pmatrix})$$

$$[S_i']_{\pi_j(1)_B} = [R]_B \cdot ([z_i]_B \cdot I_{m+n} - \begin{pmatrix} [M_B]_B \\ [M_A]_{i_{\pi_j(1)_B}} \end{pmatrix})$$

$$[S_i']_{\pi_j(2)_A} = [R]_A \cdot ([z_i]_A \cdot I_{m+n} - \begin{pmatrix} [M_B]_A \\ [M_A]_{i_{\pi_j(2)_A}} \end{pmatrix})$$

$$[S'_i]_{\pi_j(2)_B} = [R]_B \cdot ([z_i]_B \cdot I_{m+n} - \begin{pmatrix} [M_B]_B \\ [M_A]_{i_{\pi_j(2)_B}} \end{pmatrix})$$

where $[M_A]_{i_{\pi_j(1)}}, [M_A]_{i_{\pi_j(2)}}$ are constructed from $[g]_i$ and $[g']_i$ with the order determined by the permutation π_j, which is unknown to $\mathcal{P_B}$.

3. $\mathcal{P_A}$
   (a) computes $S'_{i_{\pi_j(1)}}$ from shares $[S'_i]_{\pi_j(1)_A}, [S'_i]_{\pi_j(1)_B}$, and $S'_{i_{\pi_j(2)}}$ from shares $[S'_i]_{\pi_j(2)_A}, [S'_i]_{\pi_j(2)_B}$,
   (b) obtains $\{(S'_{1_1}, S'_{1_2}), \ldots, (S'_{(m+n)_1}, S'_{(m+n)_2})\}$ by performing $\pi^{-1}\{(S'_{1_{\pi_1(1)}}, S'_{1_{\pi_1(2)}}), \ldots, (S'_{(m+n)_{\pi_{m+n}(1)}}, S'_{(m+n)_{\pi_{m+n}(2)}})\}$
   (c) computes $f_{i_1} = \det(S'_{i_1})$ and $f_{i_2} = \det(S'_{i_2})$ for $i \in \{0, \ldots, m+n\}$ as in Protocol Π_1,
   (d) computes $\det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M'_A \end{pmatrix}) = f_{i_2} \cdot d^{-1}$ for $i \in \{0, \ldots, m+n\}$, where M'_A denotes a half Sylvester matrix constructed from the polynomial g',
   (e) halts if

$$(-1)^{m+n} \cdot \sum_{i=0}^{m+n} \left(\left(\prod_{\substack{0 \le j \le m+n \\ j \ne i}} \frac{z_j}{z_i - z_j} \right) \cdot \det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M'_A \end{pmatrix}) \right) \neq 1.$$

   (f) computes $\det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix}) = f_{i_1} \cdot d^{-1}$ for $i \in \{0, \ldots, m+n\}$,
   (g) concludes $A \cap B \neq \emptyset$ if and only if

$$\sum_{i=0}^{m+n} \left(\left(\prod_{\substack{0 \le j \le m+n \\ j \ne i}} \frac{z_j}{z_i - z_j} \right) \cdot \det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M_A \end{pmatrix}) \right) = 0.$$

Theorem 2. *The construction of the Protocol Π_2 is complete and sound against a malicious adversaries. With overwhelming probability, a malicious $\mathcal{P}^*_\mathcal{B}$ will be caught. In other word, unless $\mathcal{P}^*_\mathcal{B}$ actually knows a value in $\mathcal{P_A}$'s set, $\mathcal{P_A}$ will not be fooled into thinking otherwise. The security of both $\mathcal{P_A}$ and $\mathcal{P_B}$ is also protected based on the shares of each polynomial are randomly selected from field, and the* Secure Determinant Evaluation.

Proof. The correctness proof is the same as for the Protocol Π_1. The only difference is that we use $m+n$ pairs of shares $[g]_{i_A}, [g]_{i_B}$ for g. The reason for doing this is to ensure the soundness of this protocol, and will be discussed shortly. The security proof is the similar as the one in the Protocol Π_1. An adversary does not gain any extra information. This is because of the perfectness of Shamir secret sharing and the SDE we used.

Soundness. In the given soundness definition, $\mathcal{P}^*_\mathcal{B}$ is operating with an unknown dataset $A \subset K$. From our construction, $\mathcal{P_B}$ does not know anything about A beyond a share of A and $\mathcal{P_A}$ knows a share of B. $\mathcal{P_A}$ will only accept $A \cap B \neq \emptyset$ when $\det(\begin{pmatrix} M_B \\ M_A \end{pmatrix}) = 0$.

Note that $[g]_{i_A}$ is sent along with $[g']_{i_A}$ for $i \in \{1, \dots, m+n\}$. In the setting, $[g]_{i_A} \neq [g]_{j_A}$ and $[g']_{i'_A} \neq [g']_{j'_A}$ for $i \neq j$ and $i' \neq j'$. With the randomness of $[g]_{i_A}$'s and $[g']_{i_A}$'s, $\mathcal{P}_{\mathcal{B}}^*$ could deliberately set $\mathrm{rank}(M_B) < n$. This way $\mathcal{P}_{\mathcal{A}}$ will accept $A \cap B \neq \emptyset$ since $\det(\begin{pmatrix} M_B \\ M_A \end{pmatrix}) = 0$. The only way for letting $\mathrm{rank}(M_B) < n$ is to set h be a zero polynomial in our setting. But this will be challenged by our verification test, which $\mathcal{P}_{\mathcal{A}}$ only accepts h when

$$(-1)^{m+n} \cdot \sum_{i=0}^{m+n} \left(\left(\prod_{\substack{0 \le j \le m+n \\ j \neq i}} \frac{z_j}{z_i - z_j} \right) \cdot \det(z_i \cdot I_{m+n} - \begin{pmatrix} M_B \\ M'_A \end{pmatrix}) \right) = 1.$$

The malicious Bob $\mathcal{P}_{\mathcal{B}}^*$ can find the shares $[h]_A, [h']_B, [h]_B$ for the polynomial h, such that h can be reconstructed through the shares $[h]_A$ and $[h]_B$, but the combination of $[h]_A$ and $[h']_B$ corresponds to a zero polynomial. $\mathcal{P}_{\mathcal{B}}^*$ can then use $[h]_B$ for the verification test and $[h']_B$ for disjointness test if he can guess which one of $(e_{i_{\pi(1)}}, e_{i_{\pi(2)}})$ corresponds to $[g']_{i_A}$. But the chance $\mathcal{P}_{\mathcal{B}}^*$ guesses correctly in each pair is $\frac{1}{2}$. Thus, the chance $\mathcal{P}_{\mathcal{B}}^*$ can guess correctly for all $m+n$ pairs is $\frac{1}{2^{m+n}}$. If m and n are reasonable sizes, $\mathcal{P}_{\mathcal{B}}^*$ will be caught with an overwhelming probability.

5.3 Computation and Communication Complexity

Two protocols proposed in this paper are very simple and require only $O(1)$ rounds of communication. The communication cost is in terms of number of $\lceil \log_2 q \rceil$ bits that are transmitted. The computation cost is measured in number of field operations. In our calculation, the complexity of matrix multiplication is $O((m+n)^{2.375})$ [21]; the complexity of determinant computation is $O((m+n)^{2.697})$ [22].

The communication complexity of the Protocol Π_1 is $O((m+n)^2)$. The protocol requires $2(m+n)$ matrix multiplication, and $m+n$ determinant computations. The overall computation complexity is $O((m+n)^{3.697})$ field operations.

There is slightly more commnication cost for Protocol Π_2, but complexity is still $O((m+n)^2)$. The computation cost is only double the cost of the Protocol Π_1.

6 Conclusion

We proposed protocols for private disjointness tests that are based on the polynomial representation of datasets and Sylvester matrix construction. We first introduced the structure of Sylvester matrices and the intuition of our protocols that applies Sylvester matrices. To avoid revealing the intersection cardinality by directly applying Sylvester matrices, we provided a protocol to test the set disjointness without revealing any additional information in the honest-but-curious case. Finally, we described a protocol to against malicious adversaries by applying a verification test.

The protocols constructed in this paper are more efficient than previous protocols with respect to communication and computation complexity. They are all $O(1)$ rounds, and do not require the parties to compute exponentiations or any other kind of public key operations. Our protocols also provide information theoretic security, and do not rely on any computational assumption.

Acknowledgment

The authors are grateful to the anonymous reviewers for their comments to improve the quality of this paper. We also like to thank C. Pandu Rangan for some helpful discussions. This work was supported by the Australian Research Council under ARC Discovery Projects DP0558773, DP0665035 and DP0663452. Qingsong Ye's work was funded by an iMURS scholarship provided by Macquarie University. The research of Huaxiong Wang is partially supported by the Ministry of Education of Singapore under grant T206B2204.

References

[1] Kiayias, A., Mitrofanova, A.: Testing disjointness and private datasets. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 109–124. Springer, Heidelberg (2005)
[2] Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–9. Springer, Heidelberg (2004)
[3] Hohenberger, S., Weis, S.A.: Honest-verifier private disjointness testing without random oracles. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 277–294. Springer, Heidelberg (2006)
[4] Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: 31st annual ACM Symposium on Theory of Computing (STOC 1999), Atlanta, Georgia, May 1999, pp. 245–254 (1999)
[5] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
[6] Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
[7] Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
[8] Kissner, L., Song, D.: Privacy-preserving set operaitons. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
[9] Laur, S., Lipmaa, H., Mielikainen, T.: Private itemset support counting. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 97–111. Springer, Heidelberg (2005)
[10] Kiayias, A., Mitrofanova, A.: Syntax-driven private evaluation of quantified membership queries. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 470–485. Springer, Heidelberg (2006)
[11] Fagin, R., Naor, M., Winkler, P.: Comparing information without leaking it. Communications of the ACM 39(5), 77–85 (1996)
[12] Lipmaa, H.: Verifiable homomorphic oblivious transfer and private equality test. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 416–433. Springer, Heidelberg (2003)

[13] Ye, Q., Wang, H., Tartary, C.: Privacy-preserving distributed set intersection. In: The 2nd Workshop on Advances in Information Security (conjuncted with ARES 2008), Barcelona, Spain, March 2008, pp. 1332–1339. IEEE Computer Society Press, Los Alamitos (2008)

[14] Ye, Q., Wang, H., Pieprzyk, J.: Distributed private matching and set operations. In: ISPEC 2008, April 2008. LNCS, vol. 4991, pp. 347–360. Springer, Heidelberg (2008)

[15] Cramer, R., Damgard, I.: Secure distributed linear algebra in a constant number of rounds. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 119–136. Springer, Heidelberg (2001)

[16] Mohassel, P., Franklin, M.: Efficient polynomial operations in the shared-coefficients setting. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 44–57. Springer, Heidelberg (2006)

[17] von zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, Cambridge (2003)

[18] Bar-Ilan, J., Beaver, D.: Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In: 8th ACM Annual Symposium on Principles of Distributed Computing (PODC 1989), pp. 201–209. ACM Press, New York (1989)

[19] Ben-Or, M., Cleve, R.: Computing algebraic formulas using a constant number of registers. In: 20th annual ACM Symposium on Theory of Computing (STOC 1988), pp. 254–257. ACM Press, New York (1988)

[20] Goldreich, O.: The Foundations of Cryptography, vol. 2. Cambridge University Press, Cambridge (2004)

[21] Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. Journal of Symbolic Computing 9, 251–280 (1990)

[22] Kaltofen, E., Villard, G.: On the complexity of computing determinants. Computational Complexity 13(3-4), 91–130 (2005)