Multi Party Distributed Private Matching, Set Disjointness and Cardinality Set Intersection with Information Theoretic Security

Sathya Narayanan G[1] ⋆, Aishwarya T[1] ⋆⋆, Anugrah Agrawal[2], Arpita Patra[3] ⋆⋆⋆ †, Ashish Choudhary[3] ‡, and C. Pandu Rangan[3] §

[1] `{sathya.phoenix,aishwarya.t}@gmail.com`
National Institute of Technology, Trichy.
[2] `anugrah.agrawal.apm06@itbhu.ac.in`
Institute of Technology, BHU.
[3] `{arpita,ashishc}@cse.iitm.ernet.in` , `rangan@iitm.ernet.in`
Indian Institute of Technology, Madras.

Abstract. In this paper, we focus on the specific problems of Private Matching, Set Disjointness and Cardinality Set Intersection in *information theoretic* settings. Specifically, we give *perfectly secure protocols* for the above problems in n party settings, tolerating a *computationally unbounded semi-honest* adversary, who can passively corrupt at most $t < n/2$ parties. To the best of our knowledge, these are the first such information theoretically secure protocols in a multi-party setting for all three problems. Previous solutions for Distributed Private Matching and Cardinality Set Intersection were *cryptographically* secure and the previous Set Disjointness solution, though information theoretically secure, is in a two party setting. We also propose a new model for Distributed Private matching which is relevant in a multi-party setting.

Keywords: Privacy preserving Set operations, Multiparty Computation.

1 Introduction

Consider the following problem: Alice has a set A of values and there exists an element $a \in A$. Bob also has a set of values B. Alice wants to check if her element a belongs to Bob's set B or not; i.e., if $a \in B$ or not. Alice does not want to reveal her element a to Bob and nor does Bob want Alice to know about any of the elements in his set. Alice should ultimately learn if her element belongs to Bob's set or not and nothing more. And Bob should not learn anything (neither about Alice's value nor about its presence in his set). This is the *private matching problem*. In the distributed private matching problem proposed by Ye et al. [16], Bob's dataset B is distributed across n servers such that t or less servers cannot come together and reconstruct his dataset.

Consider another problem: Alice and Bob have sets A and B respectively and Alice wishes to find out if $A \cap B = \phi$. Alice and Bob also do not want to reveal any other information about their datasets to either party. The only information that Alice should gain is whether $A \cap B = \phi$ or not and Bob learns no new information. This is known as the *private set disjointness* test [6].

Suppose Alice and Bob have the sets A and B respectively and Alice wants to find out the cardinality of the set $A \cap B$. The solution should only reveal $|A \cap B|$ to Alice and should not reveal any more information about Bob's dataset to Alice and at the end of the solution, Bob should not gain any extra information about Alice's dataset. This is the *Cardinality Set-Intersection Problem* [3].

Private Matching has a lot of motivating examples from real life. For example, assume Alice has a highly sensitive information and wants to know if Bob has any record of the same. Bob, concerned about the security of his data and in order to cater to needs across the globe, has distributed all the information he has, in a database over n servers. Bob is willing to help Alice, but at the same time is not ready to reveal any other information that might help Alice get his dataset. Also, Alice does not want to reveal her sensitive information to Bob. For example, Alice could be a credit card service provider and Bob could have the set of all credit faulters from a single service. Alice might want to

⋆ Work supported by IIT-M Summer Fellowship 2009, IIT Madras

⋆⋆ Work supported by Project No. CSE/05-06/075/MICO/CPAN on Foundation Research in Cryptography, sponsored by Microsoft Research, India.

⋆⋆⋆ Financial Support from Microsoft Research India Acknowledged

† Dept. of Computer Science and Engineering, IIT Madras, Chennai India. Email: arpita, ashishc@cse.iitm.ernet.in, rangan@iitm.ernet.in

‡ Financial Support from Infosys Technology India Acknowledged

§ Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation, sponsored by Department of Information Technology, Govt. of India.

check if her customer belongs to bad credit union before agreeing to provide services and Bob would not want Alice to gain information about anyone else on the list. The Distributed Private Matching protocol gives solution to such problems.

As another example, suppose that a community social services centre has a list of drug abusers in the age group 11-19. A school administration in the locality wants to find out if its school is 'clean' or not. Since this is highly sensitive information the centre would not like to reveal any information or names in the list and would be willing to reveal only whether there are report cases of drug abuse from the school. This is an example of private set disjointness test. Again, if the school wants to know the number of students on the list and the centre is willing to reveal the number but does not want to divulge any more information, it becomes an example of the cardinality set intersection.

Existing Literature: Private matching was introduced as a private *two-party* matching problem by Freedman et al.[3], who solved the problem under *cryptographic* assumptions, using oblivious polynomial evaluation with a public-key homomorphic encryption system. In the protocol, Alice has an element a and Bob has a dataset $B = \{b_1, b_2, \cdots, b_m\}$. At the end of the protocol, Alice gets to know if his element a belongs to Bob's Dataset B. The protocol does not reveal Alice's element a to Bob and Alice knows nothing more than whether $a \in B$. Ye, Wang and Pieprzyk [16] extended this problem to a Distributed scenario. In Distributed Private Matching, Alice has a value a and Bob has his dataset $B = \{b_1, b_2, \cdots, b_m\}$ distributed among n servers such that t or less servers cannot discover Bob's Original Dataset while $t + 1$ or more servers can reconstruct Bob's dataset. In [16], a protocol for distributed private matching problem under *cryptographic* assumptions is provided.

Quite a few protocols exist for the set disjointness problem. Freedman et al. [3] proposed a protocol for two-party set disjointness under *cryptographic* assumptions, based on the representation of datasets as roots of a polynomial and oblivious polynomial evaluation techniques. The protocol reveals information about the cardinality of set intersection. It is very efficient against honest-but-curious adversaries but invokes expensive sub-protocols to work against malicious adversaries. Hohenberger and Weis [5] had used similar construction as in [3] and proposed a protocol in cryptographic setting (the security proof relies on the hardness of the discrete logarithm problem). Their protocol assumes an honest Alice while Bob can be malicious and again reveals information about the cardinality of the set intersection. Kiayias and Mitrofanova [7] proposed three protocols for set disjointness. The first protocol works on a relatively smaller domain for set disjointness, the second uses a new primitive called superposed encryption based on Pedersen commitments [13], the third uses a multi variate polynomial to reduce the high round complexity of the second protocol. Both [5] and [7] work in the two party setting. [17] provided the first information theoretic solution to the private set disjointness problem. [17] presented two protocols using Sylvester matrices technique for two-party set disjointness with round complexity $O(1)$. While the first protocol is secure against honest-but-curious adversaries, the second protocol is secure against malicious adversaries.

The Cardinality Set-Intersection problem was previously studied in [3] in the two-party setting. In [7], Kiayias et al. studied private set disjointness as mentioned above which can be looked at as a restricted version of Cardinality Set Intersection. In [8], Kisner et al. studied the problem in the multi-party setting and proposed efficient solutions for both honest-but-curious and malicious adversary under *cryptographic* assumptions, using zero knowledge proofs. Vaidya and Clifton [15] presented a protocol for cardinality set intersection that is scalable and efficient in cryptographic settings and hence suitable for data mining applications.

Also, multi-party set intersection problems in information theoretic settings have been studied in [9] and [12]. Though there exist protocols for set intersection and in general for Multi-Party Computations, using them to solve set disjointness or cardinality set intersection will be an overkill and highly inefficient. Our goal is to design efficient customised protocols as opposed to using generic abstract protocols.

Our Motivation and Contribution: From the literature, we find that existing solutions for private matching are in cryptographic settings. Also, the Distributed Private Matching proposed in [16] is essentially between two parties, where the data set of the second party is distributed among n servers. In this paper, we propose the first information theoretically secure protocol for Distributed Private Matching in the model proposed in [16]. We then propose a new model for Distributed Private Matching in a multi-party setting. The distributed private matching in our new model can be looked at as a general n-party Private Matching, where each party has a dataset and Alice has a value a and wants to know if a belongs to any of the n datasets. Here, the parties distribute their datasets among themselves such that t or less parties cannot come together and gain information about any honest party's dataset. Thus the parties themselves act as the servers used in the 2-party setting. The n-party Distributed Private Matching is useful in many scenarios. For example, suppose there are n trading agencies who store information about the available resources in a region. This information is sensitive and to ensure its safety, they share it among each other, so that any set of t or less agencies cannot get the information of any other agency. Now assume that Alice is a trader interested in setting up a factory over this region but needs to know if she can get the necessary resources for her production from any of the trading agencies. But Alice does not want to reveal her requirements to the agencies till she can get a confirmation that they will be of help. In such a case, n-party Distributed Private Matching is helpful. We also propose an information theoretically secure protocol for n-party Distributed Private Matching, secure against a semi-honest adversary.

Set Disjointness has been handled in information theoretic setting previously [17], but only in a 2-party setting. We provide the first multi party information theoretically secure protocol for set disjointness, secure against a semi-honest adversary. In our model, there are n parties where each party's dataset is distributed among the n parties, such that $t + 1$ or more parties need to come together to reconstruct the entire dataset, similar to our proposed model for n-party Distributed Private Matching.

Privacy law is the area of law concerned with the protection and preservation of the privacy rights of individuals. The law of privacy regulates the type of information which may be collected and how this information may be used. Many privacy rules and regulations like HIPAA, GLBA and SOX [10] exist that restrict companies from sharing their data as it is to other parties. For example, there could be a hospital database and there could be a vendor who wants to check if the technology used by his mobile results in complaints such as ear ache, headache etc. Also there could be a vendor who wants to check for multiple ailments which could result from the use of his product. So, he would like to find out if a threshold number of customers have complained of these ailments by checking with the hospital database. Also, the hospital's database would be governed by privacy rules like the ones mentioned above. Hence, this problem is an example for multi-party cardinality set intersection.

The existing solutions for Cardinality Set Intersection problem (both in 2-party and n-party setting) are in *cryptographic* settings. We provide the first multi party information theoretically secure protocol for Cardinality Set Intersection problem, secure against a semi-honest adversary.

Hence our contribution in this paper is to provide information theoretically secure protocols for Private Matching, Set Disjointness and Cardinality Set-Intersection in a multi-party setting against a semi-honest adversary. We also show how to adapt our protocols to work against an active adversary in the same model. To the best of our knowledge, this is the first work to address these problems in in a multi-party scenario, in information theoretic settings.

2 Model Definitions and Preliminaries

In this paper, we will be considering two different models. The first model is adapted from [16], while the second model is proposed by us. We provide a perfectly secure protocol for Two party distributed private matching problem in the first model, while we propose perfectly secure protocols for n-party distributed private matching, Cardinality Set Intersection and Set Disjointness in the second model. We now briefly discuss these models. We also give the details of various existing sub-protocol, used in this paper.

2.1 Model for 2-Party Distributed Private Matching [16]

Here Alice and Bob are two parties. Alice has a secret value $a \in \mathbb{F}$ and Bob has a private dataset $B = \{b^{(1)}, \ldots, b^{(m)}\}$, consisting of m elements from a finite prime field $\mathbb{F}$, where $|\mathbb{F}| > n$. The dataset of Bob is distributed among n servers in a manner as explained in section 2.5, where $n \geq 2t+1$. There exists a *passive adversary* with *unbounded computing power*, who can control at most t servers out of the n servers. We assume that Alice does not interact with Bob directly. Instead Alice contacts the set of n servers to perform the private-matching operations. We assume also that no server colludes with Alice to cheat and only Alice learns the output of any operation. More precisely, the following conditions should hold [16]:

1. CORRECTNESS: *If Alice and the servers honestly follow the steps of the protocol, then protocol works and Alice learns the correct result of the operation specified in the protocol.*

2. ALICE'S SECURITY : *If Alice is honest, then at the end of the protocol, the adversary controlling t servers should not get any information whatsoever about a.*

3. BOB'S SECURITY : *Provided that no server colludes with Alice, the protocol ensures that Alice does not get any extra information other than the output of the operation. In addition, any t or less servers should not able to find out any information about Bob's dataset.*

2.2 Model for n-party Distributed Private Matching, Set Disjointness and Cardinality Set Intersection

Here we consider a complete synchronous network of n parties, denoted as $\mathcal{P} = \{P_1, \ldots, P_n\}$, who are pairwise connected by a secure channel. There exists a centralized adversary, having *unbounded computing power*, who can passively control at most $t < n/2$ parties. This is a valid assumption as information theoretic MPC against a computationally unbounded t-active passive adversary is possible iff $n \geq 2t + 1$ [2]. By passive adversary, we mean that all the parties under the control of adversary follow the prescribed steps of the protocol, but may try to learn *something extra* from the messages seen during the execution of the protocol. Each party P_i has a private data set $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$, consisting of m elements from a finite prime field $\mathbb{F}$ where $|\mathbb{F}| > n$ (the protocols presented in this paper will also work if the number of elements in each data set is different). All computation and communication in our protocols are done over $\mathbb{F}$. To ensure the secrecy and distributed nature of datasets, each party P_i distributes his dataset among all other parties, as shown in Section 2.5 and Section 2.5. We now state the security definition, associated with n-party distributed private matching, Cardinality Set Intersection and Set Disjointness.

Security Definition for n-Party Distributed Private Matching Here Alice has an element $a \in \mathbb{F}$, whose presence she wants to check for in any of the n datasets. For this, she interacts with the n parties. As in [16], we assume that no party colludes with Alice and only Alice learns the output of any operation. More precisely, the following should hold as in [16]:

1. CORRECTNESS: *If Alice and the parties honestly follow the steps of the protocol, then protocol works and Alice learns the correct result of the operation specified in the protocol.*

2. ALICE'S SECURITY : *If Alice is honest, then at the end of the protocol, the adversary controlling t parties should not get any information whatsoever about a.*

3. PARTY'S SECURITY : *Provided that no party colludes with Alice, the protocol ensures that Alice does not get any extra information other than the output of the operation. In addition, if P_i is honest, then his dataset B_i is secure against a passive adversary controlling at most t parties.*

Security Definition for n-Party Set Disjointness Here the n parties want to know whether $(B_1 \cap B_2 \cap \cdots \cap B_n) = \phi$ or not and nothing more. More specifically, the following conditions should be satisfied at the end of the protocol, even if $t < n/2$ parties are passively corrupted by a computationally unbounded adversary:

1. CORRECTNESS: *If the parties honestly follow the steps of the protocol, then they learn if $(B_1 \cap B_2 \cap \cdots \cap B_n) = \phi$ or not.*

2. PARTY'S SECURITY : *The adversary should not get any extra information about the input and output of honest parties, other than what can be inferred by the input of t corrupted parties (i.e., the dataset of these parties) and the output of t corrupted parties (which is $(B_1 \cap B_2 \cap \cdots \cap B_n) \overset{?}{=} \phi$).*

Security Definition for n-Party Cardinality Set Intersection Here the parties want to know $|B_1 \cap B_2 \cap \cdots \cap B_n|$ and nothing more. More specifically, the following conditions should be satisfied at the end of the protocol, even if $t < n/2$ parties are passively corrupted by a computationally unbounded adversary:

1. CORRECTNESS: *If the parties honestly follow the steps of the protocol, then the parties learn $|B_1 \cap B_2 \cap \cdots \cap B_n|$.*

2. PARTY'S SECURITY : *The adversary should not get any extra information about the input and output of honest parties, other than what can be inferred by the input of t corrupted parties (i.e., the dataset of these parties) and the output of t corrupted parties (which is $|B_1 \cap B_2 \cap \cdots \cap B_n|$).*

2.3 Sharing a Value s Among n Parties

Consider the following problem: there exists a *dealer* $D \in \mathcal{P}$. D has a secret $s \in \mathbb{F}$, which he wants to share among $P_1, \ldots, P_n$, such that if t or less parties pool their shares, then they will know nothing about s. On the other hand, if $t+1$ or more parties pool their shares, then they can reconstruct s. This problem is called *secret sharing*. One of the methods to solve this problem is Shamir Secret Sharing [14], where to share s, D chooses a random polynomial $f(x)$ of degree t, such that $f(0) = s$. D then gives P_i his share $s_i = f(\alpha_i)$, where each α_i is a publicly known distinct element from $\mathbb{F}$. To reconstruct s, each party produces his share s_i. Once all the n shares are available, anyone can interpolate the t degree polynomial $f(x)$ passing through (α_i, s_i)'s and hence reconstruct $s = f(0)$. It is easy to see that if t parties pool their shares, then they will know nothing about s [14].

d-Sharing and its Properties [1] We say a value $s \in \mathbb{F}$ is d-shared among the parties in $\mathcal{P}$, if every (honest) party $P_i \in \mathcal{P}$ is holding a share s_i of s, such that there exists a degree-d polynomial $p(\cdot)$ with $p(0) = s$ and $p(\alpha_i) = s_i$ for every $P_i \in \mathcal{P}$. The vector of shares $(s_1, \ldots, s_n)$ is called a d-sharing of s, and is denoted by $[s]_d$. *In the rest of the paper, whenever we say that the parties have $[s]_d$ for some $s \in \mathbb{F}$, we mean to say that each party is holding his share corresponding to d-sharing of s.*

Shamir sharing is a t-sharing scheme and generates t-sharing $[s]_t$ of secret s. Notice that Shamir sharing satisfies the following properties:

1. $[a]_t + [b]_t = [a+b]_t$, for any $a, b \in \mathbb{F}$.
2. $[a]_t[b]_t = [ab]_{2t}$, for any $a, b \in \mathbb{F}$.

Thus if the parties hold $[a]_t$ and $[b]_t$, then they can locally generate $[a+b]_t$, without doing any communication, by simply adding their respective shares of a and b. On the other hand, if the parties simply multiply their respective shares of a and b, then this will generate $[ab]_{2t}$. To generate $[ab]_t$ from $[a]_t$ and $[b]_t$, we need to use the multiplication protocol specified in section 2.4.

2.4 Multiplying Shared Values

Let a and b be two values, which are Shamir shared (i.e., t-shared) among $P_1, \ldots, P_n$ using degree-t polynomials $f(x)$ and $g(x)$ respectively. Thus party P_i has shares a_i and b_i of a and b respectively. Then the parties $P_1, \ldots, P_n$ can generate the Shamir shares of $c = ab$ by using the multiplication protocol of [4] as follows: Let $d_i = a_i b_i$. Each party P_i Shamir share d_i, say using degree-t polynomial $h_i(x)$. This results in party P_i holding the share-share $d_{1i}, \cdots, d_{ni}$ of $d_1, \ldots, d_n$ respectively. Then from Lagrange's interpolation, the degree-t polynomial $h(x) = \sum_{i=1}^{n} w_i h_i(x)$ is the polynomial that Shamir shares c, where

$$w_i = \prod_{j=1, j\neq i}^{n} \alpha_j / (\alpha_j - \alpha_i) \tag{1}$$

To get j^{th} share of c, party P_j computes $c_j = h(\alpha_j) = \sum_{i=1}^{n} w_i h_i(\alpha_j) = \sum_{i=1}^{n} w_i d_{ij}$. It is easy to see that during this process, an adversary passively controlling at most t parties does not get any information about a, b and c [4]. Also, the method works only if $n > 2t$ which holds in our case.

Lemma 1. *The above multiplication protocol communicates $\mathcal{O}(n^2)$ field elements and takes one round of communication.*

PROOF: In the protocol, each party Shamir shares a value, which involves a communication complexity of $O(n)$ field elements and one round of communication. Hence the lemma. $\square$

Lemma 2. *Suppose the parties have $[a^{(1)}]_t, \ldots, [a^{(\ell)}]_t$ and $[b^{(1)}]_t, \ldots, [b^{(\ell)}]_t$, where each $a^{(l)}$ and $b^{(l)}$ belongs to $\mathbb{F}$ and $\ell \geq 1$. Then the parties can generate $[a^{(l)}b^{(l)}]_t$, for $l = 1, \ldots, \ell$ in 1 communication round using the above multiplication protocol. On the other hand the parties can generate $[a^{(1)} \ldots a^{(\ell)}]_t$ in $\log_2 \ell$ communication rounds.*

PROOF: Computing $[a^{(l)}b^{(l)}]_t$ for $l = 1, \ldots, \ell$ will require 1 round because each $a^{(l)}b^{(l)}$ is independent of the other and we can generate these products in parallel. On the other hand, generating $[a^{(1)} \ldots a^{(\ell)}]_t$ requires $\log_2 \ell$ communication rounds because we can multiply two operands at a time, say a_i and $a_{i+\lfloor \frac{\ell}{2} \rfloor}$ for $i = 1, \cdots, \lfloor \frac{\ell}{2} \rfloor$, and find their products in the first round and then make pairs among the resulting products(after the first round) and multiply them in the next round and so on. $\square$

2.5 Dataset Distribution of Parties

In both the models, namely the one presented in section 2.1 and section 2.2, the parties distribute their dataset in a specif manner. We now give the details of how this is done.

Dataset Distribution for Two Party Distributed Private Matching Here Bob on having the data set $B = \{b^{(1)}, \ldots, b^{(m)}\}$ does the following: Bob forms a polynomial $F(x)$ such that the elements of his set B are roots of the polynomial (i.e.,) $F(x) = \prod_{i=1}^{m}(x - b^{(i)}) = \sum_{i=0}^{m} C_i x^i$, such that $C_m = 1$. Bob then Shamir shares each C_i among the n servers. It is easy to see that even if t or less servers combine their shares, they will have no information about B. On the other hand, B can be reconstructed by pooling the shares of any $t + 1$ or more servers.

Dataset Distribution for n-party Distributed Private Matching Here each party P_i on having a dataset $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$ distributes it in the following way: P_i forms a polynomial $F_i(x)$ such that the elements of his set B_i are roots of the polynomial (i.e.,) $F_i(x) = \prod_{j=1}^{m}(x - b^{(i,j)}) = \sum_{j=0}^{m} C^{(i,j)} x^j$, such that $C^{(i,m)} = 1$. Party P_i then Shamir shares each $C^{(i,j)}$ among the n parties. It is easy to see that even if t or less parties combine their shares, they will have no information about B_i. On the other hand, B_i can be reconstructed by pooling the shares of any $t + 1$ or more servers.

Dataset Distribution for n-party Set Disjointness and Cardinality Set Intersection Here each party P_i on having dataset $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$, distributes it in the following way: for $j = 1, \ldots, m$, party P_i Shamir shares $b^{(i,j)}$ among the parties in $\mathcal{P}$. Since each element in the dataset is individually shared using a t-degree polynomial, it implies that if P_i is honest, then each element of his dataset B_i is secure against a passive adversary controlling at most t parties. Moreover, any set of $t + 1$ or more parties can reconstruct B_i by pooling their shares.

2.6 Checking If a Shared Value is Zero

Nishide and Ohta [11] present an efficient and deterministic protocol to check if a shared value is zero or not. More specifically, the protocol takes $[s]_t$ as input, where s is shared using Shamir sharing and outputs the following:

1. If $s = 0$, then the protocol generates $[1]_t$.
2. If $s \neq 0$, then the protocol generates $[0]_t$.

The protocol performs $81l$ multiplications of shared values, where $l = \log(|\mathbb{F}|)$ and takes 8 rounds. In the rest of the paper, we use this protocol for testing if a shared value is zero. We shall henceforth refer to this protocol as TEST-IF-ZERO.

Remark 1. The TEST-IF-ZERO protocol of [11] is a deterministic protocol, without any error, which we use in the rest of this paper. A drawback of this protocol is that it performs very large number of multiplications. In the last section of this paper, we present a simple protocol for testing if a shared value is zero, involving significantly less number of multiplications. However, this protocol is probabilistic and gives the correct output, except with an error probability of $\frac{1}{|\mathbb{F}|}$.

3 Two-Party Distributed Private Matching Protocol

Recall that in the 2-party distributed private matching, Bob has a private data set B of m elements, which he has distributed among n servers, say $S_1, \ldots, S_n$, as explained in section 2.5. Alice has a secret element $a \in \mathbb{F}$, whose presence she wants to check in Bob's dataset. For this she interacts with the servers. We now present a perfectly secure protocol for this problem. Before proceeding further, we give the following trivial lemma:

Lemma 3. *The value a belongs to B iff $F(a) = 0$, where $F(x) = \prod_{i=1}^{m}(x - b^{(i)}) = \sum_{i=0}^{m} C_i x^i$.*

Proof: The proof is obvious and it follows from the definition of $F(x)$. □

The high level idea of the protocol is as follows: Alice first Shamir shares the values $a, \ldots, a^m$ among n servers. Hence all the servers, apart from having the shares of the coefficients of $F(x)$, now also have the shares of $a, \ldots, a^m$. The servers then compute the Shamir shares of $V_j = C_j a^j$ for $1 \leq j \leq m$, using the multiplication protocol (see section 2.4). Since $F(a) = \sum_{j=0}^{m} V_j$, by a linear combination of all the shares that a server has, each server gets his share of $F(a)$. Till this point, the servers have generated the Shamir shares of $F(a)$. Now if the servers give their shares of $F(a)$ to Alice, then Alice could reconstruct $F(a)$ and find whether a belongs to B. But directly revealing $F(a)$ to Alice will violate Bob's security, as Alice would come to know about one point on $F(x)$.

Since Alice wants to know only if $a \in B$ or not, all we need to find out is if $F(a) = 0$ or not. For this, all the servers run the TEST-IF-ZERO protocol on the shares of $F(a)$ and reconstruct the output towards Alice. If the output is one then Alice concludes that $F(a) = 0$, otherwise $F(a) \neq 0$. Accordingly, Alice concludes that a belongs (does not belong) to B. The protocol called 2-party DPMP is formally given in the following table:

2-Party DPMP

Setup Phase:

Alice Shamir shares $a, \ldots, a^m$ among n servers. Bob distributes his dataset $B = \{b^{(1)}, \ldots, b^{(m)}\}$ among n servers as explained in section 2.5. Let $F(x) = \prod_{j=1}^{m}(x - b^{(j)}) = \sum_{j=0}^{m} C_j x^j$. Thus the parties have $[C_j]_t$ for $j = 0, \ldots, m$.

Computation (by each server):

1. The servers compute $[V_j]_t = [C_j a^j]_t$ for $1 \leq j \leq m$ using the multiplication protocol described in section 2.4.
2. The servers then compute $[F(a)]_t = [V_1]_t + \ldots + [V_m]_t$.
3. Finally the servers run the protocol TEST-IF-ZERO on $[F(a)]_t$ to generate $[v]_t$, where $v = 1(0)$, if $F(a) = 0(\neq 0)$.

Reconstruction Phase:

The servers give their shares of v to Alice. Alice reconstruct v and checks if $v = 1$. If $v = 1$, then $a \in B$ else $a \notin B$.

Before proceeding further to prove the properties of 2-Party DPMP, we make the following claim.

Claim. In protocol 2-Party DPMP if Alice is honest, then a passive adversary controlling at most t servers does not get any information about a even after knowing t shares of $a, \ldots, a^m$.

Proof: The proof follows easily from the properties of Shamir sharing and simple linear algebra. For a complete proof, see **APPENDIX A**. □

Lemma 4. *Protocol 2-party DPMP satisfies the properties of 2-party distributed private matching.*

Proof: The correctness property is trivial. The secrecy of Bob's dataset against a passive adversary controlling at most t servers follows from the properties of Shamir sharing. The secrecy of Bob's data set against a passive Alice follows from the secrecy of TEST-IF-ZERO. Finally, secrecy of Alice's a follows from Claim 3. □

Lemma 5. *Protocol 2-party DPMP communicates $\mathcal{O}(n^2m)$ field elements and involves one invocation of* TEST-IF-ZERO*. The protocol takes two rounds.*

Proof: In the setup phase, the parties communicates $\mathcal{O}(nm)$ field elements for data set distribution. In the computation phase, there are m multiplications and hence it communicates $\mathcal{O}(n^2m)$ field elements. Since all the multiplications are independent, by lemma 2 it can be done in parallel in one round. Moreover, setup phase takes one round. □

4 n-Party Distributed Private Matching Protocol

We now present a perfectly secure protocol called n-party DPMP for distributed private matching in n-party settings. For this, we use the model presented in section 2.2. Recall that in this model, there are n parties denoted as $\mathcal{P} = \{P_1, \ldots, P_n\}$, where each P_i has a private dataset $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$ represented using $F_i(x) = \prod_{j=1}^{m}(x - b^{(i,j)}) = \sum_{j=0}^{m} C^{(i,j)}x^j$. Moreover, the dataset B_i is Shamir shared among the n parties; i.e., the parties hold $[C^{(i,j)}]_t$, for $i = 1, \ldots, n$ and $j = 0, \ldots, m$ (see Section 2.5). Alice has a secret element a. Alice wants to know if $a \in (B_1 \cup \ldots \cup B_n)$. Before proceeding further, we give the following lemma.

Lemma 6. $a \in (B_1 \cup B_2 \cup \cdots \cup B_n)$ *iff atleast one of the* $F_i(a) = 0$.

The high level idea of protocol n-party DPMP is as follows: Alice first Shamir shares the values $a, \ldots, a^m$ among n servers. Thus parties hold $[a^j]_t$ for $j = 1, \ldots, m$. The parties then compute $[V^{(i,j)}]_t = [C^{(i,j)}a^j]_t$ using the multiplication protocol. The parties then compute the Shamir shares of $[F_i(a)]_t = \sum_{j=0}^{m}[V^{(i,j)}]_t$. Since shamir sharing is linear, this can be done locally. The parties then compute $[F(a)]_t$ where

$$F(a) = \prod_{i=0}^{n} F_i(a) \tag{2}$$

After this step, the parties have $[F(a)]_t$. To ensure that no more information is revealed to Alice than what is necessary, the parties run the TEST-IF-ZERO protocol on $[F(a)]_t$ and reconstruct the output towards Alice, so that Alice gets to know only if $F(a)$ is zero or not and nothing more. Alice checks if the reconstructed value is 0 or not to find if $a \in (B_1 \cup B_2 \cup \cdots \cup B_n)$. Protocol n-party DPMP is formally given in the following table.

n-Party DPMP

Setup Phase:
Alice shamir shares $a, a^2, a^3, \cdots, a^m$ among the n parties and each party P_i for $1 \le i \le n$ distributes his dataset B_i using the polynomial $F_i(x) = \sum_{j=0}^{m} C^{(i,j)}x^j$ among n parties using Dataset Distribution scheme described in section 2.5. Thus the parties have $[a^j]_t$ for $1 \le j \le m$ and $[C^{(i,j)}]_t$ for $1 \le i \le n$ and $0 \le j \le m$.

Local Computation (by each party):

1. The parties compute $[V^{(i,j)}]_t = [C^{(i,j)}a^j]_t$ for $0 \le j \le m$ and $1 \le i \le n$ and compute $[F_i(a)]_t = \sum_{j=0}^{m}[V^{(i,j)}]_t$.

2. The parties compute $[F(a)]_t = \prod_{i=1}^{n}[F_i(a)]_t$ by running the multiplication protocol specified in section 2.4.

3. The parties now run the protocol TEST-IF-ZERO on $[F(a)]_t$ to generate $[v]_t$, where $v = 1(0)$, if $F(a) = 0(\neq 0)$.

Reconstruction Phase: The parties give their shares of v to Alice. Alice reconstructs v and checks if $v = 1$. If $v = 1$, then $a \in B_1 \cup B_2 \cup \cdots \cup B_n$ else $a \notin B_1 \cup B_2 \cup \cdots \cup B_n$.

Lemma 7. *Protocol n-party DPMP satisfies the properties of n-party distributed private matching.*

Proof: Follows directly from the protocol steps and properties of Shamir sharing and TEST-IF-ZERO protocol. □

Lemma 8. *Protocol n-party DPMP communicates $\mathcal{O}(n^3m)$ field elements and executes one instance of* TEST-IF-ZERO*. The protocol involves $\mathcal{O}(\log(n))$ communication rounds.*

Proof: The communication complexity is easy to analyze. In setup phase, $\mathcal{O}(nm)$ values are t shared and it communicates $\mathcal{O}(n^2m)$ field elements. During computation phase, in step 1, we first do nm multiplications simultaneously which can be done in 1 round. In step 2, n shared values need to be multiplied. Since these values are dependant, the multiplications totally take $\mathcal{O}(\log(n))$ communication rounds and communicates $\mathcal{O}(n^3m)$ field elements by lemma 1 and lemma 2. □

5 n-Party Set Disjointness Protocol

We now present a perfectly secure protocol called n-party Set Disjointness for n-party set disjointness problem. Recall that in this problem, there are n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where each P_i has a private dataset $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$. Moreover, each $b^{(i,j)}$ is Shamir shared among the n parties for $j = 1, \ldots, m$. Thus the parties hold $[b^{(i,j)}]_t$. The parties want to know if $B_1 \cap \ldots \cap B_n = \phi$ or not. We first present a protocol called Gen-E^l, which we will later use in solving the n-party set disjointness as well as cardinality set intersection problem.

5.1 Protocol Gen-E^l

In this section we give a protocol that helps in solving the set cardinality and disjointness problems. We call this protocol as Gen-E^l, which generates shares of E^l (which we will define subsequently). We first observe that, $(B_1 \cap \cdots \cap B_n) \subseteq B_r$ for any $1 \leq r \leq n$. Hence we fix a party $P_r \in \mathcal{P}$ as reference and refer to the elements of his dataset as $\{a^1, \ldots, a^m\}$ for convenience. Now we check if some element of B_r is present in the the dataset of each party in $P \setminus P_r$. If there exists any such element $a^l \in \{a^1, \ldots, a^m\}$, then the sets B_i's are not disjoint. Protocol Gen-E^l checks for the presence of any such element a^l.

Gen-E^l

Setup Phase:

1. Each party P_i on having dataset $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$ Shamir shares each $b^{(i,j)}$. Thus the parties hold $[b^{(i,j)}]_t$, for $1 \leq i \leq n$ and $1 \leq j \leq m$.

2. The parties fix one arbitrary party among them as the reference, say, the lowest index party, P_1. For the ease of presentation, let $\{a^1, \ldots, a^m\}$ denote the element of P_1's dataset.

Computation (by each party):

1. The parties first compute $[V^{(l,i,j)}]_t = [b^{(i,j)}]_t - [a^l]_t$ for $i = 2, \ldots, n$, $j = 1 \ldots, m$ and $l = 1 \ldots, m$.

2. The parties use the multiplication protocol to generate $[C^{(i,l)}]_t = \prod_{j=1}^{j=m}[V^{(l,i,j)}]_t$ for $i = 2, \ldots, n$ and $l = 1 \ldots, m$.

3. The parties then compute $[E^l]_t = \sum_{i=2}^{i=n}[C^{(i,l)}]_t$.

Lemma 9. *If $E^l = 0$ is zero, then the element $a_l \in B_1$ is also present in $(B_1 \cap \cdots \cap B_n)$.*

Proof : The proof follows easily from the protocol steps. □

Lemma 10. *Protocol Gen-E^l communicates $\mathcal{O}(n^3m^2)$ field elements in setup phase. The protocol takes $\mathcal{O}(\log(nm^2))$ communication rounds.*

Proof: The communication complexity and the number of multiplications is easy to analyze. Since there are nm^2 multiplications and all of them are dependent, from Lemma 2, the number of rounds needed to perform these multiplications is $\mathcal{O}(\log(nm^2))$. □

5.2 Protocol for Multi Party Set Disjointness

To compute multi party set disjointness, the parties first execute protocol Gen-E^l to generate $[E^l]_t$ for $l = 1, \ldots, m$. The parties then compute $[E]_t = \prod_{l=0}^{l=m}(E^l)$ by using the multiplication protocol. The parties then execute TEST-IF-ZERO on $[E]_t$ to generate $[v]_t$. Finally v is reconstructed by each party, after which the parties conclude whether the sets are disjoint or not.

n-party Set Disjointness

1. The parties run protocol Gen-E^l to generate $[E^l]_t$ for $l = 1, \ldots, m$.
2. The parties now run multiplication protocol to compute $[E]_t = \prod_{l=1}^{l=m}([E^l]_t)$.
3. The parties then run the TEST-IF-ZERO protocol on $[E]_t$ to generate $[v]_t$.

Reconstruction Phase:

The parties produce their respective share of v. Once v is reconstructed, the parties check if $v = 0$ or 1. If $v = 0$, then $(B_1 \cap \cdots \cap B_n) = \phi$ else $(B_1 \cap \cdots \cap B_n) \neq \phi$.

Lemma 11. *Protocol n-party Set Disjointness satisfies the properties of n-party set disjointness.*

Proof: The proof follows easily from the protocol steps, properties of protocol Gen-E^l and Shamir secret sharing. □

Lemma 12. *Protocol n-party Set Disjointness communicates $\mathcal{O}(n^3m^2 + n^2m)$ field elements, and executes one instance of* TEST-IF-ZERO. *The protocol takes $\mathcal{O}(\log(nm^2) + \log(m))$ communication rounds.*

Proof: Communication complexity is easy to analyze. In the protocol, Gen-E^l performs nm^2 multiplications. Computing $[E]_t$ require m multiplications. Hence the total number of multiplications done is $(nm^2 + m)$. Since these multiplications are dependent, we require $\mathcal{O}(\log(nm^2) + \log(m))$ rounds to perform them. It is easy to see that protocol executes one instance of TEST-IF-ZERO. □

6 n-Party Cardinality of Set Intersection

We now present a perfectly secure protocol called n-party Cardinality Set Intersection for n-party cardinality set intersection. Recall that in this problem, there are n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where each P_i has a private dataset $B_i = \{b^{(i,1)}, \ldots, b^{(i,m)}\}$. Moreover, each $b^{(i,j)}$ is Shamir shared among the n parties for $j = 1, \ldots, m$. Thus the parties hold $[b^{(i,j)}]_t$. The parties want to only know the value $|B_1 \cap \cdots \cap B_n|$ and nothing more. Protocol n-party Cardinality Set Intersection is formally given in the following table.

n-party Cardinality Set Intersection
1. The parties run the protocol Gen-E^l to generate the $[E^l]_t$ for $l = 1, \ldots, m$.
2. For $l = 1, \ldots, m$, the parties run TEST-IF-ZERO on $[E^l]_t$ to generate $[v^l]_t$.
3. The parties then compute $[v]_t = \sum_{l=1}^{l=m} [v^l]_t$.
<u>Reconstruction Phase:</u> All the parties produce their respective shares of v to reconstruct v. The value v is $\|B_1 \cap \cdots \cap B_n\|$.

Lemma 13. *Protocol n-party Cardinality Set Intersection satisfies the properties of n-party cardinality set intersection.*

Proof: The security of honest parties' dataset is satisfied because the elements of the dataset of honest parties are Shamir and hence are safe against a passive adversary having control over t parties. Also the final outcome of the protocol is only the cardinality of $(B_1 \cap \cdots \cap B_n)$ and not anything more. The correctness property follows from Lemma 9 and protocol steps. □

Lemma 14. *Protocol n-party Cardinality Set Intersection communicates $\mathcal{O}(n^3m^2)$ field elements and invokes m instances of* TEST-IF-ZERO *protocol. The protocol has a round complexity of $\mathcal{O}(\log(nm^2))$.*

Proof: The protocol communicates $\mathcal{O}(n^2m)$ field elements for sharing the datasets and does nm^2 multiplications. Since all the multiplications are dependent, from Lemma 2, it takes $\mathcal{O}(\log(nm^2))$ communication rounds to perform them. From protocol, we can clearly see that it involves m invocations of TEST-IF-ZERO protocol. □

7 A Simple Protocol for Checking if a Shared Value is Zero

In section 2.6, we gave the details of protocol TEST-IF-ZERO, which checks whether a shared value is zero or not. The protocol is used in all our protocols as a black-box. However, protocol TEST-IF-ZERO involves a large number of multiplications. We now present a protocol NEW-TEST-IF-ZERO, which takes $[a]_t$ as an input and produces $[V]_t$ as the output, where $a \in \mathbb{F}$ is a random value. The protocol has the following properties:

1. If $a = 0$, then $V = 0$.
2. If $V = 0$, then except with probability $\frac{1}{|\mathbb{F}|}$, $a = 0$.
3. If $a \neq 0$ then except with probability $\frac{1}{|\mathbb{F}|}$, V can be any random non-zero value.

4. Even if V is reconstructed by each party, a passive adversary controlling at most t parties/servers will have no information about a.
5. The protocol performs significantly less number of multiplications in comparison to protocol TEST-IF-ZERO.

The protocol is formally given in the following table.

NEW-TEST-IF-ZERO

Setup Phase:

1. Each party P_i chooses a random value $r^{(i)} \in \mathbb{F}$ and Shamir shares $r^{(i)}$. So now the parties have $[a]_t$ and $[r^{(i)}]_t$ for $1 \le i \le n$.

Computation (by each party):

1. The parties compute $[R]_t = \sum_{i=1}^{n}[r^{(i)}]_t$.

2. The parties compute $[V]_t = [R]_t[a]_t$ by using the multiplication protocol.

Reconstruction Phase:

The parties reconstruct the value V. If V is non-zero, then the parties conclude that a is also non-zero. If V is 0, then the parties conclude that a is 0 with very high probability.

Lemma 15. *Protocol* NEW-TEST-IF-ZERO *satisfies all the properties mentioned above.*

PROOF: Before proceeding further, we first note that $R = \sum_{i=1}^{n} r^{(i)}$ is completely random. This is because at least one $r^{(i)}$ in R is shared by an honest P_i and hence $r^{(i)}$ is random, implying that R is also random. It is easy to see that if $a = 0$ then $V = Ra$ will be also zero. Thus if $V = 0$, the probability that $a \neq 0$ is same as the probability $R = 0$, which is $\frac{1}{|\mathbb{F}|}$. Moreover, if $a \neq 0$, then except with probability $\frac{1}{|\mathbb{F}|}$, V can be any random value. This is because R can be any random value. If a passive adversary knows t shares of a, then even after knowing $V = Ra$, the value a remains information theoretically secure due to the random R. Finally it is easy to see that the protocol performs only one multiplication. □

7.1 Application of Protocol NEW-TEST-IF-ZERO

By seeing the properties of protocol NEW-TEST-IF-ZERO, we find that it be used as a substitute of TEST-IF-ZERO in any protocol, where we just want to know whether the shared value a is zero or not. In protocols 2-Party DPMP, n-party DPMP and n-party Set Disjointness, protocol TEST-IF-ZERO was used to just check whether a shared value a is zero or not. So we can replace TEST-IF-ZERO with our new protocol NEW-TEST-IF-ZERO in 2-Party DPMP, n-party DPMP and n-party Set Disjointness. Since NEW-TEST-IF-ZERO requires less number of multiplications than TEST-IF-ZERO, the resultant protocols for 2-Party DPMP, n-party DPMP and n-party Set Disjointness becomes more efficient. However, since protocol NEW-TEST-IF-ZERO involves a negligible error probability, the resultant protocols for 2-Party DPMP, n-party DPMP and n-party Set Disjointness will also now involve a negligible error probability in correctness.

Notice that we cannot use our new protocol NEW-TEST-IF-ZERO as a substitute of TEST-IF-ZERO in protocol n-party Cardinality Set Intersection. This is because if $a \neq 0$, then NEW-TEST-IF-ZERO outputs $[V]_t$, where except with probability $\frac{1}{|\mathbb{F}|}$, V can be any random non-zero value. On the other hand, protocol TEST-IF-ZERO would output $[0]_t$ in this case. Now in protocol n-party Cardinality Set Intersection, the parties added the outcome of each instance of TEST-IF-ZERO to count the number of elements in the intersection of n sets. However, the parties cannot do so if TEST-IF-ZERO is replaced by NEW-TEST-IF-ZERO.

8 Some Optimization Issues

8.1 Further Reduction of Communication Complexities

In all our protocols, we have used the multiplication protocol of [4], which communicates $\mathcal{O}(n^2)$ field elements to generate the t-sharing of the product of two t-shared values, in the presence of a computationally unbounded passive adversary, controlling $t < n/2$ parties/servers. To further optimise the communication complexity of our protocols, we can use the multiplication protocol in [1]. In [1] the authors have presented a perfectly secure multiplication protocol tolerating a t-active *malicious* adversary. The MPC protocol of [1], when executed with $n = 2t + 1$ parties, in the presence of a *passive* adversary, controlling at most t parties, will communicate $\mathcal{O}(n)$ field elements to generate the t-sharing of the product of two t-shared values. Moreover, the protocol will take $\mathcal{O}(1)$ communication rounds.

8.2 Adapting our Protocols to Work Against Malicious Adversary

All our protocols can be extended to work against a t-active malicious adversary [4], having unbounded computing power, by doing the following steps:

1. Taking $n = 3t + 1$, instead of $n = 2t + 1$. This is required because from [2], secure computation tolerating an all powerful, t-active malicious adversary is possible iff $n \geq 3t + 1$.
2. Using *Verifiable Secret Sharing* (VSS) [2] instead of Shamir's secret sharing scheme for $n = 3t + 1$.
3. Using multiplication protocol of [1] secure against a malicious adversary.
4. Modifying the TEST-IF-ZERO protocol as suggested in [11] to work against a malicious adversary.

9 Conclusion and Open Problems

In this paper, we have given perfectly secure protocols for private matching, set disjointness and cardinality set intersection problems in information theoretic settings, secure against a computationally unbounded passive adversary. Future work would be to come up with efficient protocols that can work against more powerful adversaries such as byzantine and mixed adversaries. Also, improving the communication complexity of the protocols presented in this paper is another interesting future direction.

References

1. Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure mpc with linear communication complexity. In *TCC*, pages 213–230, 2008.
2. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.
3. Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.
4. Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified vss and fact-track multiparty computations with applications to threshold cryptography. In *PODC*, pages 101–111, 1998.
5. Susan Hohenberger and Stephen A. Weis. Honest-verifier private disjointness testing without random oracles. In *Privacy Enhancing Technologies*, pages 277–294, 2006.
6. Aggelos Kiayias and Antonina Mitrofanova. Testing disjointness of private datasets. In *Financial Cryptography*, pages 109–124, 2005.
7. Aggelos Kiayias and Antonina Mitrofanova. Syntax-driven private evaluation of quantified membership queries. In *ACNS*, pages 470–485, 2006.
8. Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In *CRYPTO*, pages 241–257, 2005.
9. Ronghua Li and Chuankun Wu. An unconditionally secure protocol for multi-party set intersection. In *ACNS*, pages 226–236, 2007.
10. Ron Ben Natan. *Implementing Database Security and Auditing.* ELSEVIER, 2005.
11. Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *Public Key Cryptography*, pages 343–360, 2007.
12. Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. Information theoretically secure multi party set intersection re-visited. Cryptology ePrint Archive, Report 2009/116, 2009. http://eprint.iacr.org/.
13. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
14. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[4] A malicious adversary takes complete control of the parties under its control and can make them behave in any arbitrary fashion during the protocol execution.

15. Jaideep Vaidya and Chris Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.
16. Qingsong Ye, Huaxiong Wang, and Josef Pieprzyk. Distributed private matching and set operations. In *ISPEC*, pages 347–360, 2008.
17. Qingsong Ye, Huaxiong Wang, Josef Pieprzyk, and Xian-Mo Zhang. Efficient disjointness tests for private datasets. In *ACISP*, pages 155–169, 2008.

APPENDIX A: Properties of Protocol 2-Party DPMP

Claim 3: *In protocol 2-Party DPMP if Alice is honest, then a passive adversary controlling at most t servers does not get any information about a even after knowing t shares of $a, \dots, a^m$.*

Proof: Without loss of generality, let the adversary passively controls the servers $S_1, \dots, S_t$. Thus the adversary will know the first t shares of $a, \dots, a^m$. We first show that knowing t shares of a and a^2, the adversary does not get any extra information than just knowing the t shares of a. So let a and a^2 be Shamir shared using degree-t polynomials $f(x)$ and $g(x)$ respectively, where $f(0) = a$ and $g(0) = a^2$. Moreover, for $i = 1, \dots, t$, we have $f(\alpha_i) = a_i$ and $g(\alpha_i) = a_i^2$. Here a_i and a_i^2 denotes i^{th} share of a and a^2 respectively. Moreover, $\alpha_1, \dots, \alpha_t$ are publicly known distinct elements from $\mathbb{F}$.

From the shares of a, the adversary can form a $t-1$ degree polynomial $f_{int}(x)$ such that $f_{int}(x) = f(x)$, for $x = \alpha_1, \dots, \alpha_t$. The polynomial $f(x)$ can thus be expressed in terms of $f_{int}(x)$ in the following way :

$$f(x) = f_{int}(x) + \gamma(x - \alpha_1) \dots (x - \alpha_t) \tag{3}$$

The adversary knows $f_{int}(0)$ and $\alpha_1, \dots, \alpha_t$. However, $f(0)$ is information theoretically secure because of the fact that γ is still unknown to the adversary. Thus the security of a lies on the inability of the adversary to gain information on γ.

Similarly, the adversary can form a $t-1$ degree polynomial $g_{int}(x)$, such that

$$g(x) = g_{int}(x) + \beta(x - \alpha_1) \dots (x - \alpha_t) \tag{4}$$

If Alice would have only Shamir shared a^2, then a^2 would be information theoretically secure because adversary would have no information about β. However, from Eqn (3) and Eqn (4), the adversary can form the following system of equations:

$$f(x) = f_{int}(x) + \gamma(x - \alpha_1) \dots (x - \alpha_t) \tag{5}$$
$$g(x) = g_{int}(x) + \beta(x - \alpha_1) \dots (x - \alpha_t) \tag{6}$$

With the above two equations for $f(x)$ and $g(x)$, the adversary can obtain the following relation :

$$a = f_{int}(0) + \gamma(-\alpha_1) \dots (-\alpha_t) \tag{7}$$
$$a^2 = g_{int}(0) + \beta(-\alpha_1) \dots (-\alpha_t) \tag{8}$$

Using the relation between a and a^2, adversary can obtain the following relation between γ and β:

$$(f_{int}(0) + (-1)^t\gamma \times \prod_{i=1}^{t} \alpha_i)^2 = (g_{int}(0) + (-1)^t\beta \times \prod_{i=1}^{t} \alpha_i) \tag{9}$$

As we see, we can only get β in terms of γ and vice versa. As long as γ is secure, the whole set of dependent information is secure and we can see that, because of the dependency, the security of a^2 is also in terms of γ. The argument can be further extended to other powers of a or to any set of values that are dependent on a showing that ultimately all their security depend on γ and hence on the security of a as is the case when we share just a (and not its powers). $\square$