<table>
<thead>
<tr>
<th>Title</th>
<th>Privacy-preserving Set Operations in the Presence of Rational Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Miyaji, Atsuko; Rahman, Mohammad Shahriar</td>
</tr>
<tr>
<td>Citation</td>
<td>2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA): 869-874</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2012-03</td>
</tr>
<tr>
<td>Type</td>
<td>Conference Paper</td>
</tr>
<tr>
<td>Text version</td>
<td>author</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10119/10657">http://hdl.handle.net/10119/10657</a></td>
</tr>
<tr>
<td>Rights</td>
<td>Copyright (C) 2012 IEEE. 2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2012, 869-874. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.</td>
</tr>
</tbody>
</table>

Description:

Privacy-preserving set operations allow parties to perform operations on their sets without revealing the contents of their sets to each other. This is particularly useful in scenarios where parties are rational and may not trust each other. The paper presents new algorithms for performing set operations such as union, intersection, and difference in a privacy-preserving manner. The algorithms are designed to be efficient and scalable, making them suitable for large-scale applications. The paper also discusses the security and privacy guarantees provided by the proposed algorithms. The practicality of the algorithms is demonstrated through simulations and experiments.
Abstract—Privacy-preserving set operations are useful for many data mining algorithms as building tools. Protocols for privacy-preserving set operations have considered semi-honest and malicious adversarial models in cryptographic settings, whereby an adversary is assumed to follow or arbitrarily deviate from the protocol. Semi-honest model provides weak security requiring small amount of computation, on the other hand, malicious model provides strong security requiring expensive computations like homomorphic encryption. However, efficient computation of such set operations are desirable for practical implementations. In this paper, we build efficient and private set operations avoiding the use of expensive tools like homomorphic encryption, zero-knowledge proof, and oblivious transfer. Our protocol is constructed in game-theoretic model. In other words, instead of being semi-honest or malicious, the parties are viewed as rational and are assumed (only) to act in their self-interest. We show that our protocol satisfies computational Nash equilibrium.

KeyWords: Privacy-preserving data mining, Set-intersection, Game theory, Computational Nash equilibrium.

I. INTRODUCTION

A key utility of large databases today is scientific or economic research. Despite the potential gain, this is often not possible due to the confidentiality issues which arise, leading to concerns over privacy infringement while performing the data mining operations. The need for privacy is sometimes due to law (e.g., for medical databases) or can be motivated by business interests. To address the privacy problem, several privacy-preserving data mining protocols using cryptographic techniques have been suggested. Privacy-preserving algorithms are applicable for web mining, where a customer can be much better identified by many internet companies than in real world. With these algorithms, internet sellers and web searching companies are able to make much better forecast of the customer’s behavior, without disclosing private data. In data mining area, private set-intersection and set-union protocols allow two parties interact on their respective input sets. These protocols address several realistic privacy issues. For example, companies may want to decide whether to make a business alliance by the percentage of customers shared among them, without publishing their customer databases including the shared customers among them. This can be treated as an intersection cardinality problem. To determine which customers appear on a do-not-receive-advertisements list, a store must perform a set-intersection operation between its private customer list and the producer’s list.

Depending on the adversarial behavior assumptions, privacy-preserving data mining protocols use different models. Classically, two main categories of adversaries have been considered:

1. Semi-honest adversaries: Following Goldreich’s definition [15], protocols secure in the presence of semi-honest adversaries (or honest-but-curious) assume that parties faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., set size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about the other party’s input. This model is formalized by requiring that each party does not learn more information that it would in an ideal implementation relying on a trusted third party (TTP).

2. Malicious adversaries: Security in the presence of malicious parties allows arbitrary deviations from the protocol. In general, however, it does not prevent parties from refusing to participate in the protocol, modifying their private input sets, or prematurely aborting the protocol. Security in the malicious model is achieved if the adversary (interacting in the real protocol, without the TTP) can learn no more information than it could in the ideal scenario.

In the above models, a secure protocol emulates (in its real execution) the ideal execution that includes a TTP. This notion is formulated by requiring the existence of adversaries in the ideal execution model that can simulate adversarial behavior in the real execution model. In other words, the implicit assumption in the original formulation of the problem is that each party is either honest or corrupt, and honest parties are all willing to cooperate when reconstruction of the secret is desired. However, the assumption of semi-honest behavior may be unrealistic in some settings. In such cases, participating parties may prefer to use a protocol that is secure against malicious behavior. It is clear that the protocols secure in the malicious model offer more security. Regarding malicious adversaries, it has been shown that, under suitable cryptographic assumptions, any multi-party probabilistic polynomial time functionality (PPT) can be securely computed for any number of malicious corrupted parties. However, these are not efficient enough to be used in practice. Most of these constructions...
use general zero-knowledge proofs for fully malicious multi-party computation (MPC) protocols. These zero-knowledge compilers lead to rather inefficient constructions [32]. In typical cryptographic MPC protocols, parties are allowed to abort when they can find some malicious behavior from other parties. This means that the parties have to start the protocol from the scratch which is undesirable for operations on huge data sets.

Since the work of Halpern and Teague [17], protocols for some cryptographic tasks (e.g., secret sharing, multi-party computation) have begun to be re-evaluated in a game-theoretic light (see [9], [23] for an overview of work in this direction). In this setting, parties are neither honest nor corrupt but are instead viewed as rational and are assumed (only) to act in their self-interest. This feature is particularly interesting for data mining operations where huge collection of data is used, since parties will not deviate (i.e., abort) as there is no incentive to do so. In many real-world settings, parties are willing to actively deviate/cheat, but only if they are not caught. This is the case in many business, financial, political and diplomatic settings, where honest behavior cannot be assumed, but where the companies, institutions and individuals involved cannot afford the embarrassment, loss of reputation, and negative press associated with being caught cheating, hence having smaller incentive.

A. Related Work

Cryptographic techniques have been used to design many different distributed privacy-preserving data mining algorithms. In general, there are two types of assumptions on data distribution: vertical and horizontal partitioning. In the case of horizontally partitioned data, different sites collect the same set of information about different entities. For example, different credit card companies may collect credit card transactions of different individuals. Secure distributed protocols have been developed for horizontally partitioned data for mining decision trees [27], k-means clustering [26], k-nn classifiers [21]. In the case of vertically partitioned data, it is assumed that different sites collect information about the same set of entities but they collect different feature sets. For example, both a university and a hospital may collect information about a student. Again, secure protocols for the vertically partitioned case have been developed for mining association rules [37], and k-means clusters [19], [36]. All of those previous protocols claimed to be secure only in the semi-honest model. In [12], [22], authors present two-party secure protocols in the malicious model for data mining. They follow the generic malicious model definitions from the cryptographic literature, and also focus on the security issues in the malicious model, and provide the malicious versions of the subprotocols commonly used in previous privacy-preserving data mining algorithms. Assuming that at least one party behaves in semi-honest model, they use threshold homomorphic encryption for malicious adversaries presented by Cramer et al. [6]. Since homomorphic encryption is considered too expensive [28] and zero-knowledge proof is often one of the most expensive parts of cryptographic protocols, the protocols proposed in malicious adversarial model are not very practical for operations on large data items. Set operations using commutative encryption have been proposed in [2], where the adversaries have been considered as semi-honest parties. Recently, [13] proposed set-intersection protocol that allows the parties to use hardware tokens. The proposed protocol is quite efficient and privacy-preserving under the assumption that the token issuer can communicate with the token up to a certain time limit. However, dependency on extra hardware or trusted third party is what we want to avoid in this work. Game theory and data mining, in general, have been combined in [20], [34] for constructing various data mining algorithms. Rational adversaries have also been considered in privacy-preserving set operations [38], [3]. These protocols consider Nash equilibrium to analyze the rational behavior of the participating entities. As in all of cryptography, computational relaxations are meaningful and should be considered; doing so allows us to get around the limitations of the information-theoretic setting. So, analyzing set operations from the viewpoint of computational Nash equilibrium is interesting, since it gives a more realistic results. In order to build an efficient set-intersection protocol, [33] proposed a construction in a secret sharing manner based on iterations. It used verifiable random functions to make sure that there is one unique message in each iteration. But the protocol either has to depend on a trusted dealer for the pre-processing stage or has to use oblivious transfer. There have been several works on game theory based MPC/secret sharing schemes [1], [17], [25], [30], [14], [35], [18]. But [17], [35] require the continual involvement of the dealer even after the initial shares have been distributed or assume that sufficiently many parties behave honestly during the computation phase. Some schemes [1], [25], [30] rely on multiple invocations of protocols. Other work [18] relies on physical assumptions such as secure envelopes and ballot boxes. [14] proposed efficient protocols for rational secret sharing. But secret sharing schemes cannot be directly used for our purpose since they require much heavier computation, the existence of TTP, and their set up is different.

B. Our Contribution

In this work, we build two-party secure set-intersection protocol in game-theoretic setting using cryptographic primitives. It is assumed that parties are neither honest nor corrupt but are instead rational and are assumed to act only in their self-interest. Our construction avoids the use of expensive tools like homomorphic encryption, zero knowledge proof, and oblivious transfer. We have used commutative encryption as the underlying cryptographic primitive which is simple and efficient. The parties run the protocol in a sequence of \( r \) rounds and learn the complete result at the end of the \( r \)-th round. Also, our construction does not rely on the existence of any trusted third party. It is also possible to use our protocol for computing set-union operations. We also show that our protocol satisfies computational version of strict Nash equilibrium. In short, our protocol achieves the following:
Either of the parties may cheat with incorrect input. But cheating does not help any party to win the game.

At any round earlier than \( r \), aborting the protocol does not give any higher pay off to the aborting party than following the protocol.

**Organization of the paper:** The remainder of the paper is organized as follows: Section II presents the background and preliminaries. Section III describes the protocol model. Section IV includes protocol construction. In Section V, we analyze the protocol formally. We give some concluding remarks in Section VI.

II. BACKGROUND AND PRELIMINARY

A. Cryptographic Considerations in Game Theory

Achieving a secure protocol is the objective in the cryptographic setting. Eliminating the trusted party is one of the main tasks while maintaining the privacy. On the other hand, in game theory, some particular equilibrium is defined to achieve stability. The existence of the trusted party/mediator is a parameter setting resulting in a more desirable, but harder to implement equilibrium concept for rational behaviors. Thus, privacy is a goal in the cryptographic setting while in the game theory setting it is a means to an end.

Games are treated in a modified way with a differently defined equilibrium notions in a cryptographic setting with. Katz, in [23], gives some examples of how this might be done for the specific case of parties running a protocol in the cryptographic setting. A security parameter \( n \) is introduced which is provided to all parties at the beginning of the game. The action of a player \( P_j \) now corresponds to running an interactive Turing Machine (TM) \( T_j \). The \( T_j \) takes the current state and messages received from the other party as the input, and outputs message of player \( P_j \) along with updated state. The message \( m_j \) is sent to the other party. In a computational sense, it is required that \( T_j \) runs in PPT meaning that the function is computed in time polynomial in \( n \). \( T_j \) is thus allowed to run for an unbounded number of rounds and, it can be added that the expected number of rounds is also polynomial for which \( T_j \) runs. The security parameter \( n \) is given as input to the utility functions. Utility functions map transcripts of a protocol execution to the reals that can be computed in time polynomial in \( n \). Let \( A \) be a computational game in which the actions of each player correspond to the PPT TMs. Also, the utilities of each player are computed in time polynomial in \( n \). Thus, mixed strategies are no longer needed to be considered, since a polynomial time mixed strategy corresponds to a pure strategy (since pure strategies correspond to randomized TMs) [23]. The parties are not assumed to be curious in negligible changes in their utilities, and this is an important difference between the cryptographic setting and the setting that has been considered here.

B. Definitions

In this section, we will state the definitions of computational Nash equilibrium and Commutative encryption. A protocol is in Nash equilibrium if no deviations are advantageous. In other words, there is no incentive to deviate in the case of a Nash equilibrium. We assume that a party exhibits its malicious behavior by aborting early or sending non-participate message. However, a malicious party does not manipulate its own datasets to provide wrong data. Preventing malicious parties from sharing false data is difficult since the data are private and non-verifiable information. To prevent such malicious behavior, there can be auditing mechanism where a TTP can verify the integrity of data. Further investigation is needed to thwart this kind of misbehavior without a TTP. In this regard, mechanism design could be a potential tool to motivate parties to share their data. We denote the security parameter by \( n \). A function \( \epsilon \) is negligible if for all \( c > 0 \) there is a \( n_c > 0 \) such that \( \epsilon(n) < 1/n^c \) for all \( n > n_c \); let \( \text{negl} \) denote a generic negligible function. We say \( \epsilon \) is noticeable if there exist \( c,n_c \) such that \( \epsilon(n) > 1/n^c \) for all \( n > n_c \).

We consider the strategies in our work as the PPT interactive Turing machines. Given a vector of strategies \( \sigma \) for two parties in the computation phase, let \( u_j(\sigma) \) denote the expected utility of \( P_j \), where the expected utility is a function of the security parameter \( n \). This expectation is taken over the randomness of the players’ strategies. Following the standard game-theoretic notation, \( (\sigma'_j,\sigma'_{-j}) \) denotes the strategy vector \( \sigma \) with \( P_j \)'s strategy changed to \( \sigma'_j \).

**Definition 1:** \( \Pi \) induces a computational Nash equilibrium if for any PPT strategy \( \sigma'_1 \) of \( P_1 \), we have \( u_1(\sigma'_1,\sigma_2) \leq u_1(\sigma_1,\sigma_2) + \text{negl}(n) \), and similarly for \( P_2 \).

The following definition is stated for the case of a deviating \( P_1 \) (definition for a deviating \( P_2 \) is analogous). Let \( P_1 \) and \( P_2 \) interact, following \( \sigma_1 \) and \( \sigma_2 \), respectively. Let \( \text{mes} \) denote the messages sent by \( P_1 \), but not including any messages sent by \( P_1 \) after it writes to its (write-once) output tape. Then \( \text{view}_1^\Pi \) includes the information given by the trusted party to \( P_2 \), the random coins of \( P_2 \), and the (partial) transcript \( \text{mes} \). We fix a strategy \( \gamma_1 \) and an algorithm \( A \). Now, let \( P_1 \) and \( P_2 \) interact, following \( \gamma_1 \) and \( \sigma_2 \), respectively. Given the entire view of \( P_1 \), algorithm \( A \) outputs an arbitrary party \( \text{mes}' \) of \( \text{mes} \). Then \( \text{view}_2^A,\gamma_1 \) includes the information given by the trusted party to \( P_2 \), the random coins of \( P_2 \), and the (partial) transcript \( \text{mes}' \).

**Definition 2:** Strategy \( \gamma_1 \) yields equivalent play with respect to \( \Pi \), denoted \( \gamma_1 \approx \Pi \), if there exists a PPT algorithm \( A \) such that for all PPT distinguishers \( D \)

\[
\left| \Pr[D(1^n,\text{view}_2^A,\gamma_1) = 1] - \Pr[D(1^n,\text{view}_1^\Pi) = 1] \right| \leq \text{negl}(n)
\]

**Commutative Encryption:** Our definition of commutative encryption below is similar to the constructions used in [5], [7], [11] and others. Informally, a commutative encryption is a pair of encryption functions \( f \) and \( g \) such that \( f(g(v)) = g(f(v)) \). Thus by using the combination \( f(g(v)) \) to encrypt \( v \), we can ensure that a party cannot compute the encryption of a value without the help of others. In addition, even though the encryption is a combination of two functions, each party can apply their function first and still get the same result.

**Definition 3:** Let \( \omega_n \in \{0,1\} \) be a finite domain of \( n \)-bit numbers. Let \( D_1 = D_1(\omega_n) \) and \( D_2 = D_2(\omega_n) \) be distributions over \( n \). Let \( A_n(x) \) be an algorithm that, given
x ∈ ω_n, returns either true or false. We define distribution $D_1$ of random variable $x ∈ ω_n$ to be computationally indistinguishable from distribution $D_2$ if for any family of PPT algorithms $A_n(x)$, any polynomial $p(n)$, and all sufficiently large $n$

$$Pr[A_n(x)|x ∈ D_1] − Pr[A_n(x)|x ∈ D_2] < \frac{1}{p(n)}$$

where $x$ is distributed according to $D_1$ or $D_2$, and $Pr[A_n(x)]$ is the probability that $A_n(x)$ returns true.

Definition 4: A commutative encryption $F$ is a computable (in polynomial time) function $f : KeyF × DomF → DomF$, defined on finite computable domains, that satisfies all properties listed below. We denote $fe(x) ≡ f(e, x)$.

1) Commutativity: For all $e, e′ ∈ KeyF$ we have $fe ◦ fe′ = f e′ ◦ fe$.
2) Each $fe : DomF → DomF$ is a bijection.
3) The inverse $fe^{-1}$ is also computable in polynomial time given $e$.
4) The distribution of $(x, fe(x), y, fe(y))$ is indistinguishable from the distribution of $(x, fe(x), y, z)$, where $x, y, z ∈ DomF$ and $e ∈ KeyF$.

Informally, Property 1 says that when we computably encrypt with two different keys, the result is the same irrespective of the order of encryption. Property 2 says that two different values will never have the same encrypted value. Property 3 says that given an encrypted value $fe(x)$ and the encryption key $e$, we can find $x$ in polynomial time. Property 4 says that given a value $x$ and its encryption $fe(x)$ (but not the key $e$), for a new value $y$, we cannot distinguish between $fe(y)$ and a random value $z$ in polynomial time. Thus we can neither decrypt $y$ nor encrypt $fe(y)$ in polynomial time. Note that this property holds only if $x$ is a random value from $DomF$, i.e., the adversary does not control the choice of $x$.

Remark: One-way functions exist under the discrete log-type hardness assumption; namely, exponentiation modulo a prime $p$. To be precise, given that $DomF$ is all quadratic residues modulo $p$ where $p$ is a safe prime and $q = (p − 1)/2$ such that $p$ and $q$ are primes, and $KeyF$ is $\{1, 2, ..., q − 1\}$, the exponentiation function $fe(x) = x^e \mod p$ has the properties of commutative encryption. That is, the powers commute, each of the powers $fe$ is a bijection with its inverse, and indistinguishability is satisfied under the discrete log-type hard problem.

III. MODEL

In a typical protocol, parties are viewed as either honest or semi-honest/malicious. To model rationality, we consider players’ utilities. Here we assume that $F = \{f : X × Y → Z\}$ is a functionality where $|X| = |Y| = |Z|$ and their domain is polynomial in size (poly(n)). Let $D$ be the domain of output which is polynomial in size. The function returns a vector $I$ that represents the set-intersection where $I_t$ is set to one if item $t$ is in the set-intersection. In other words, for all the data items of the parties (i.e., $X$ and $Y$), we will compute $X \cap Y$, and we get $I$ as the output of the function. Let us assume that $x_l$ is set to 1 if $P_1$ has item $i$ in its private set else it is set to 0 (similarly for $y_l$ for $P_2$). Clearly, for calculating set-intersection, we need to calculate $x_l ∩ y_l$ for each $l$ where $x_l ∈ X$ and $y_l ∈ Y$. Similarly, for set-union, we need to calculate $x_l ∪ y_l$ for all $l$. This can be rewritten as $¬(¬x_l ∩ ¬y_l)$. Computing the set-union is thus straightforward.

Given that $j$ parties are active during the computation phase, let the outcome $o$ of the computation phase be a vector of length $j$ with $o_j = 1$ iff the output of $P_j$ is equal to the exact intersection (i.e., $P_j$ learns the correct output). Let $ν_j(o)$ be the utility of player $P_j$ for the outcome $o$. Following [17], [14], we make the following assumptions about the utility functions of the players:
- If $o_j > o_j′$, then $ν_j(o_j) > ν_j(o_j′)$.
- If $o_j = o_j′$ and $o_j < o_j′$, then $ν_j(o_j) > ν_j(o_j′)$.

In other words, player $P_j$ first prefers outcomes in which he learns the output; otherwise, $P_j$ prefers strategies in which the fewest number of other players learn the result (in our two-party case, the other player learns). From the point of view of $P_j$, we consider the following three cases of utilities for the outcome $o$ where $U^* > U > U′$:
- If only $P_j$ learns the output, then $ν_j(o) = U^*$.
- If $P_j$ learns the output and the other player does also, then $ν_j(o) = U$.
- If $P_j$ does not learn the output, then $ν_j(o) = U′$.

So, we have the expected utility of a party who outputs a random guess for the output1 (assuming other party aborts without any output, or with the wrong output) as follows: $U_{rand} = \frac{1}{|P|} ∑ j \ ν_j(o)$.

Also, we assume that $U > U_{rand}$; else players have almost no incentive to run the computation phase at all. As in [14], we make no distinction between outputting the wrong secret and outputting a special ‘don’t know’ symbol both are considered as a failure to output the correct output.

IV. RATIONAL SET-INTERSECTION PROTOCOL

A. An Overview of the Protocol

Let $x$ denote the input of $P_1$, let $y$ denote the input of $P_2$, and let $f$ denote the set-intersection function they are trying to compute. Our protocol is composed of two stages, where the first stage can be viewed as a key generation stage and the second stage that computes the intersection takes place in a sequence of $r = r(n)$ iterations. More specifically, in the key generation stage the parties generate their encryption keys. They also choose $i^∗ ∈ r$ according to some random distribution $α$ in which step they can learn the complete intersection result. In every round $i ∈ \{1, ..., r\}$ the parties exchange the encrypted data for the current round, which enables $P_1$ and $P_2$ to perform the Intersection Computation. Clearly, when both parties are honest, the parties produce the same output result which is uniformly distributed. Briefly speaking, the stages have the following form:

Key Generation Stage:

1 We do not consider $U^{−\infty}$, the utility when neither party learns the output, since ‘not learning the output’ is not the target of a rational adversary in practice.
• Each party randomly chooses a secret key for itself, i.e. $e_S \in KeyF$ for $P_1$ and $e_R \in KeyF$ for $P_2$, for commutative encryption.

• A value $i^* \in \{1, \ldots, r\}$ is chosen according to some random distribution $0 < \alpha < 1$ where $\alpha$ depends on the players’ utilities (discussed later). This represents the iteration, in which parties will learn the complete result.

Intersection Computation Stage:
In each iteration $i$, for $i = 1, \ldots, r$, the parties do the following: First, $P_2$ sends $e_2$ to $P_1$ and then $P_1$ sends $e_2$ to $P_2$, where $e_1$ and $e_2$ are the ciphertexts computed by party $P_1$ and $P_2$ respectively. After receiving the ciphertexts, $P_2$ and $P_1$ compute the set-intersection using commutative property of the encryption scheme. If a party aborts in some iteration $i$, then the other party outputs the value computed in the previous iteration. If some party fails to follow the protocol, the other party aborts. In fact, it is rational for $P_2$ to follow the protocol as long as the expected gain of deviating is positive only if $P_2$ aborts exactly in iteration $i^*$; and is outweighed by the expected loss if $P_2$ aborts before iteration $i^*$. The intersection computation phase proceeds in a series of iterations, where each iteration consists of one message sent by each party.

B. Protocol Construction

As described above, our protocol $\Pi$ consists of two stages. Let $p$ be an arbitrary polynomial, and set $r = p \cdot |Y|$. We implement the first stage of $\Pi$ using a key generation algorithm. This functionality returns required keys to each party. In the second stage of $\Pi$, the parties exchange their ciphertexts in a sequence of $r$ iterations. The protocol returns $I$ at the end of the operations on all the data items as follows:

Key Generation Stage:
• Each party randomly chooses a secret key $e_1 \in KeyF$ for $P_1$ and $e_2 \in KeyF$ for $P_2$ for commutative encryption.

• A value $i^* \in \{1, \ldots, r\}$ is chosen according to some random distribution $0 < \alpha < 1$ where $\alpha$ depends on the players’ utilities. This represents the iteration, in which parties will learn the complete result.

Set Intersection Computation Stage:
for all $i$ do
1) $P_2$ encrypts its input dataset $Z_2 = f_{e_2}(Y)$ and sends $Z_2$ to $P_1$.
2) $P_1$ encrypts its input dataset $Z_1 = f_{e_1}(X)$ and sends $Z_1$ to $P_2$.
3) For $P_2$, if it has not received any message from $P_1$ then output the result of iteration $i - 1$ and halt. Otherwise, compute $Z'_2 = f_{e_2}(f_{e_1}(X))$ and sends the pairs $\langle Z_1, Z'_2 \rangle$ to $P_1$.
4) For $P_1$, if it has not received any message from $P_2$ then output the result of iteration $i - 1$ and halt. Otherwise, compute $Z'_1 = f_{e_1}(f_{e_2}(Y))$. Also, from pairs $\langle f_{e_1}(x), f_{e_2}(f_{e_1}(x)) \rangle$ obtained in earlier step for each $x \in X$, it creates pair $\langle x, f_{e_2}(f_{e_1}(x)) \rangle$ replacing $f_{e_1}(x)$ with corresponding $x$.
5) For $P_1$, for $x \in X$ for which $(f_{e_2}(f_{e_1}(x)) \in Z'_1$, these values form the intersection result $I = X \cap Y$.

6) $P_2$ computes and outputs $I$ similarly.

We provide an intuitive description of the computation phase. Let us assume that $P_1$ has a set of data items $\{Tokyo, London, Washington, Beijing\}$ and $P_2$ has $\{Tokyo, Paris, Toronto, Rome\}$. At first, they encrypt each of the items with their secret keys and exchange the ciphertexts with each other at each round (here, we will have 4 rounds at most to complete the whole protocol). After a party receives the ciphertext from the other party, it reencrypts the ciphertext using its own secret key. After they exchange such data items at each round, due to the commutative property of the underlying encryption scheme, they will come to know the intersection output (1 if the items match, 0 otherwise). For this example, they will come to know that $Tokyo$ is the intersected result from round 1, and all the subsequent rounds will output 0. So, the final result they will know only is the intersected value.

V. Protocol Analysis

Here we will give some intuition as to why the reconstruction phase of $\Pi$ is a computational Nash equilibrium for an appropriate choice of $\alpha$. Let us assume that $P_2$ follows the protocol, and $P_1$ deviates from the protocol. (It is easier to analyze the deviations by $P_2$ since $P_2$ starts in every iteration.) When $P_1$ aborts in some iteration $i < i^*$, the best strategy $P_1$ can adopt is to output $Z_1^i$ hoping that $i = i^*$. Thus, following this strategy, the expected utility that $P_1$ obtains can be calculated as follows:

• $P_1$ aborts exactly in iteration $i = i^*$. In this case, the utility that $P_1$ gets is at most $U^*$. 

• When $i < i^*$, $P_1$ has ‘no information’ about correct $I$ and so the best it can do is guess. In this case, the expected utility of $P_1$ is at most $U_{rand}$.

Considering the above, $P_1$’s expected utility of following this strategy is at most:

$$\alpha \times U^* + (1 - \alpha) \times U_{rand}$$

Now, it is possible to set the value of $\alpha$ such that the expected utility of this strategy is strictly less than $U$, since $U_{rand} < U$ by assumption. In such a case, $P_1$ has no incentive to deviate. Since there is always a unique valid message a party can send and anything else is treated as an abort, it follows that the protocol $\Pi$ induces a computational Nash equilibrium. Due to lack of space, we omit the proof of the following theorem. It will appear in the full version.

Theorem 1: The protocol $\Pi$ induces a computational Nash equilibrium given that $0 < \alpha < 1$, $U > \alpha \times U^* + (1 - \alpha) \times U_{rand}$, and the properties of commutative encryption.

VI. Conclusion

In this paper, we have proposed a privacy-preserving set-intersection protocol in two-party settings from the game-theoretic perspective. We have used commutative encryption as the underlying cryptographic primitives which is simple and very efficient. We do not rely on expensive tools like homomorphic encryption, zero knowledge proof, oblivious transfers,
etc. It is also possible to use our protocol for computing set-union operations. We also show that our protocol satisfies computational version of Nash equilibrium. Applying game-theoretic approach for multi-party setting where parties are allowed to collude is an interesting open problem.

REFERENCES


