Privacy-Preserving Data Mining: A Game-Theoretic Approach

Atsuko Miyaji, Mohammad Rahman

Privacy-preserving Data Mining: A Game-theoretic Approach

Atsuko Miyaji and Mohammad Shahriar Rahman

School of Information Science, Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, Japan 923-1292
{miyaji,mohammad}@jaist.ac.jp

Abstract. Privacy-preserving data mining has been an active research area in recent years due to privacy concerns in many distributed data mining settings. Protocols for privacy-preserving data mining have considered semi-honest, malicious, and covert adversarial models in cryptographic settings, whereby an adversary is assumed to follow, arbitrarily deviate from the protocol, or behaving somewhere in between these two, respectively. Semi-honest model provides weak security requiring small amount of computation, on the other hand, malicious and covert models provide strong security requiring expensive computations like homomorphic encryptions. However, game theory allows us to design protocols where parties are neither honest nor malicious but are instead viewed as rational and are assumed (only) to act in their own self-interest. In this paper, we build efficient and secure set-intersection protocol in game-theoretic setting using cryptographic primitives. Our construction avoids the use of expensive tools like homomorphic encryption and oblivious transfer. We also show that our protocol satisfies computational versions of strict Nash equilibrium and stability with respect to trembles.

KeyWords: Privacy-preserving data mining, Set-intersection, Game theory, Computational strict Nash equilibrium, Stability with respect to trembles.

1 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. Depending on the adversarial behavior assumptions, those protocols use different models. Classically, two main categories of adversaries have been considered:

Semi-honest adversaries: Following Goldreich's definition [11], 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).

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.

A new type of adversarial model, named covert adversary, has been proposed recently by Aumann et al. [3].

Covert Adversaries: These adversaries are somewhere in between the semi-honest and malicious models. In many real-world settings, parties are willing to actively cheat (not semi-honest), but only if they are not caught (not arbitrarily malicious). Covert adversarial behavior accurately models many real-world situations. It explicitly models the probability of catching adversarial behavior; a probability that can be tuned to the specific circumstances of the problem. In particular, it is not assumed that adversaries are only willing to risk being caught with negligible probability, but rather allow for much higher probabilities.

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 [31]. 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 [14], protocols for some cryptographic tasks (e.g., secret sharing, multi-party computation) have begun to be re-evaluated in a game-theoretic light (see [7, 20] 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 own 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.

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. Typical application examples include:
1. Business Interest: 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. As another example, 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 producers list.
2. Aviation Security: The Department of Homeland Security (DHS) of the U.S. needs to check whether any passenger on each flight from/to the United States must be denied boarding, based on some passenger watch list. For this purpose, airlines submit their entire list of passengers to DHS, together with other sensitive information, such as credit card numbers. This poses liability issues with regard to innocent passengers' data and concerns about potential data losses. In practice, information only related to the passengers on the list should obtained by DHS without disclosing any information to the airlines.
3. Healthcare: Insurance companies often need to obtain information about their insured patients from other parties, such as other insurance carriers or hospitals. The insurance carriers cannot disclose the identity of inquired patients, whereas, the hospitals cannot provide any information on other patients.

1.1 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 [25], k-means clustering [24], k-nn classifiers [18]. 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 [35], and k-means clusters [16, 34]. All of those previous protocols claimed to be secure only in the semi-honest model. In [9, 19], 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. [5]. Recently, Miyaji et al. presented a new adversarial model named covert adversaries [31] for performing data mining algorithms. They show that protocols under covert adversarial model behave in between semi-honest and malicious models. Oblivious transfer (OT) and homomorphic encryption have been used as the building blocks in [31]. Since homomorphic encryption is considered too expensive [27] and oblivious transfer is often the most expensive part of cryptographic protocols [26], the protocols proposed in malicious and covert adversarial models are not very practical for operations on large data items. Game theory and data mining, in general, have been combined in [17, 32] for constructing various data mining algorithms. Rational adversaries have also been considered in privacy-preserving set operations [36, 2]. These protocols consider Nash equilibrium to analyze the rational behavior of the participating entities. As discussed by Kol and Naor in [23], using Nash equilibrium is not suitable in many cases, since many bad strategies are not ruled out by it. Instead, they suggest the stronger notion of strict Nash equilibrium in the information-theoretic setting, in which every player's strategy is a strict best response. Due to the restrictive nature of this notion, it is regarded as a sufficient condition and not as a necessary one. 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 strict Nash equilibrium is interesting, since it gives a more realistic results. There have been several works on game theory based MPC/secret sharing schemes [1, 14, 22, 29, 10, 33, 15]. But [14, 33] 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, 22, 29] rely on multiple invocations of protocols. Other work [15] relies on physical assumptions such as secure envelopes and ballot boxes. [10] proposed efficient protocols for rational secret sharing. But secret sharing schemes cannot be directly used for our purpose since they require the existence of TTP and their set up is different.

1.2 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 own

self-interest. Our construction avoids the use of expensive tools like homomorphic encryption and oblivious transfer. We have used verifiable random functions as the underlying cryptographic primitive which is simple and efficient. It is also possible to use our protocol for computing set-union operations. We also show that our protocol satisfies computational versions of strict Nash equilibrium and stability with respect to trembles, defined by Fuchsbauer et al. [10].

Organization of the paper: The remainder of the paper is organized as follows: Section 2 presents the background and preliminaries. Section 3 describes the protocol model. Section 4 includes protocol construction. In Section 5, we analyze the protocol formally. We give some concluding remarks in Section 6.

2 Background and Preliminary

2.1 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 [20], 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 Δ 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) [20]. 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.

2.2 Definitions

In this section, we will state the definitions of computational strict Nash equilibrium and computational strict Nash equilibrium w.r.t. trembles introduced in [10]. A protocol is in Nash equilibrium if no deviations are advantageous; it is in strict Nash equilibrium if all deviations are disadvantageous. In other words, there is no incentive to deviate in the case of a Nash equilibrium whereas there is an incentive not to deviate for a strict Nash equilibrium. Another advantage of strict Nash is that protocols satisfying this notion inhibit subliminal communication. A party who tries to use protocol messages as a covert channel has the risks to lose utility if there is any reasonable probability that the other player is following the protocol, since any detectable deviation by a party from the protocol results in lower utility while the other party follows the protocol. The computational version of strict Nash equilibrium is intuitively close to strict Nash considering the computational limitations. Moreover, our protocol satisfies a strong condition that each party can send a unique legal message that at every point in the protocol. Our protocol thus rules out subliminal communication in a strong sense. We denote the security parameter by n. A function ϵ 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 $negl$ denote a generic negligible function. We say ϵ 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 $\boldsymbol{\sigma}$ for two parties in the computation phase, let $u_j(\boldsymbol{\sigma})$ 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, $(\sigma'_j, \boldsymbol{\sigma}_{-j})$ denotes the strategy vector $\boldsymbol{\sigma}$ with P_j's strategy changed to σ'_j.

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

The computational notion of stability with respect to trembles models players' uncertainty about other parties' behavior, and guarantees that even if a party P_i believes that other parties might play some arbitrary strategy with small probability δ (but follow the protocol with probability $1 - \delta$), there is still no better strategy for P_i than to follow the protocol. 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 σ_1 and σ_2, respectively. Let 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 $view_2^{\Pi}$ includes the information given by the trusted party to P_2, the random coins of P_2, and the (partial) transcript mes. We fix a strategy γ_1 and an algorithm A. Now, let P_1 and P_2 interact, following γ_1 and σ_2, respectively. Given the entire view of P_1, algorithm A outputs an arbitrary part mes' of mes. Then $view_2^{A,\gamma_1}$ includes the information given by the trusted party to P_2, the random coins of P_2, and the (partial) transcript mes'.

Definition 2. *Strategy γ_1 yields equivalent play with respect to Π, denoted $\gamma_1 \approx \Pi$, if there exists a PPT algorithm A such that for all PPT distinguishers D*

$| Pr[D(1^n, view_2^{A,\gamma_1}) = 1] - Pr[D(1^n, view_2^{\Pi}) = 1] | \leq negl(n)$

Definition 3. *Π induces a computational strict Nash equilibrium if*

1. Π induces a computational Nash equilibrium;

2. For any PPT strategy $\sigma_1' \not\approx \Pi$, there is a $c > 0$ such that $u_1(\sigma_1, \sigma_2) \leq u_1(\sigma_1', \sigma_2) + 1/n^c$ for infinitely many values of n .

In stability with respect to trembles, we say that γ_i is δ-close to σ_j if with probability $1-\delta$ party P_j plays σ_j, while with probability δ it follows an arbitrary PPT strategy σ_j'. In fact, a pair of strategies (σ_1, σ_2) is stable with respect to trembles if σ_1 (resp., σ_2) remains the best response even if the other party plays a strategy other than σ_2 (resp., σ_1) with some small (but noticeable) probability δ. The fact that the prescribed strategies are in Nash equilibrium ensures that any (polynomial-time) local computation performed by either party is of no benefit as long as the other party follows the protocol. Stated differently, even if a party P_j believes that the other party might play a different strategy with some small probability δ, there is still no better strategy for P_j than to outwardly follow the protocol.

Definition 4. *Π induces a computational strict Nash equilibrium that is stable with respect to trembles if*

1. Π induces a computational Nash equilibrium;

2. There is a noticeable function δ such that for any PPT strategy γ_2 that is δ-close to σ_2, and any PPT strategy γ_1, there exists a PPT strategy $\sigma_1' \approx \Pi$ such that $u_1(\gamma_1, \gamma_2) \leq u_1(\sigma_1', \gamma_2) + negl(n)$

Verifiable Random Functions (VRFs): A VRF is a keyed function whose output is random-looking but can still be verified as correct, given an associated proof. The notion was introduced by Micali et al. [30], and various efficient constructions in the standard model are known [6, 8, 28]. It has been shown in [28] that efficient VRFs can be constructed without relying on zero-knowledge proofs[1]. A verifiable random function (VRF) with range $R = \{R_n\}$ is a tuple of PPT algorithms $(Gen, Eval, Prove, Verify)$ such that: $G(1^n)$ generates the key pair (pk, sk). $Eval_{sk}(x)$ computes the value $y = F_{pk}(x)$; $Prove_{sk}(x)$ computes the proof z that $y = F_{pk}(x)$; and $Verify_{pk}(x, y, z)$ verifies that $y = F_{pk}(x)$ using the proof z. For such a VRF, the following hold:
Correctness: For all n, the algorithm $Eval_{sk}$ maps n-bit input to a set R_n. Furthermore, for any $x \in \{0,1\}^n$ we have $Verify_{pk}(x, Eval_{sk}(x), Prove_{sk}(x)) = 1$.
Verifiability: For all (pk, sk) output by $Gen(1^n)$, there does not exist a tuple (x, y, y', z, z') with $y \neq y'$ and $Verify_{pk}(x, y, z) = 1 = Verify_{pk}(x, y', z')$.

[1] The VRF gives us computational security. However, it is also possible to design our protocol with information-theoretic security using information-theoretically secure MACs. The details will appear in the full version.

Unique proofs: For all (pk, sk) output by $Gen(1^n)$, there does not exist a tuple (x, y, z, z') with $z \neq z'$ and $Verify_{pk}(x, y, z) = 1 = Verify_{pk}(x, y, z')$.
Pseudorandomness: Let $\mathcal{A}$ be a PPT adversary in the following game:

1. Generate $(pk, sk) \leftarrow Gen(1^n)$ and give pk to $\mathcal{A}$. $\mathcal{A}$ queries a sequence of strings $x_1, \ldots x_l \in \{0,1\}^n$ and is given $y_i = Eval_{sk}(x_i)$ and $z_i = Prove_{sk}(x_i)$ in response.

2. $\mathcal{A}$ outputs a string $x \in \{0,1\}^n$ s.t. $x \notin \{x_1, \ldots x_l\} \in \{0,1\}^n$.

3. $\mathcal{A}$ chooses a random $b \in \{0,1\}$. If $b = 0$ then $\mathcal{A}$ is given $y = Eval_{sk}(x)$; if $b = 1$ then $\mathcal{A}$ is given a random $y \in R_n$.

4. $\mathcal{A}$ makes queries as in step 2, as long as none of these queries is equal to x.

5. $\mathcal{A}$ outputs b' and succeeds if $b' = b$ at the end of the experiment.

We require that the success probability of any PPT adversary $\mathcal{A}$ is $1/2 + negl(n)$.

3 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 $\mathcal{F} = \{f : X \times Y \rightarrow Z\}$ is a functionality where $\mid X \mid = \mid Y \mid$ and their domain is polynomial in size ($poly(n)$). Let $\mathcal{D}$ 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. Clearly for calculating set-intersection, we need to calculate $x_e \wedge y_e$ for each e where $x_e \in X$ and $y_e \in Y$. Similarly, for set-union, we need to calculate $x_e \vee y_e$ for all e. This can be rewritten as $\neg(\neg x_e \wedge \neg y_e)$. Computing the set-union is thus straight forward.

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 $\nu_j(o)$ be the utility of player P_j for the outcome o. Following [14, 10], we make the following assumptions about the utility functions of the players:
- If $o_j > o'_j$, then $\nu(o_j) > \nu(o'_j)$
- If $o_j = o'_j$ and $\sum_j o_j < \sum_j o'_j$, then $\nu(o_j) > \nu(o'_j)$

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 $\nu_j(o) = U^*$.
- If P_j learns the output and the other player does also, then $\nu_j(o) = U$.
- If P_j does not learn the output, then $\nu_j(o) = U'$.

So, we have the expected utility of a party who outputs a random guess for the

output[2] (assuming other party aborts without any output, or with the wrong output) as follows: $U_{rand} = \frac{1}{|\mathcal{D}|} \cdot U^* + (1 - \frac{1}{|\mathcal{D}|}) \cdot U'$.
Also, we assume that $U > U_{rand}$; else players have almost no incentive to run the computation phase at all. As in [10], 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.

To complete the protocol, we need to provide a way for parties to identify the real iteration. Some work [1, 12, 22, 29] allows parties to identify the real iteration as soon as it occurs. This approach could be used in our protocol if we assume simultaneous channels. But, this approach is vulnerable to an obvious rushing strategy when simultaneous channels are not available. To avoid this, we follow the approach shown in [10]: delay the signal indicating whether a given iteration is real or fake until the following iteration. In this case, until being sure of the occurance of real iteration, a party cannot risk aborting. Moreover, once a party learns that the real iteration occurred, the real iteration is over and all parties can compute the real output. Simultaneous channels are thus not needed in this process at the price of adding only a single round.

4 Rational Set-Intersection Protocol

4.1 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. We follow the same high-level approach as in [14, 12, 29, 1, 22, 23]. Our intersection computation protocol proceeds in a sequence of 'fake' iterations followed by a single 'real' iteration. As in [13, 21, 10], our protocol is composed of two stages, where the first stage can be viewed as a pre-processing stage and the second stage that computes the intersection takes place in a sequence of $r = r(n)$ iterations. Briefly speaking, the stages have the following form:

Pre-processing stage:

- A value $i^* \in \{1, \ldots, r\}$ is chosen according to some geometric distribution $0 < \alpha < 1$ where α depends on the players' utilities (discussed later in Section 5). This represents the iteration, in which parties will learn the 'true output'.
- For $i < i^*$, $\{a_i\} = \{a_1, \ldots, a_r\}$ (resp.,$\{b_i\} = \{b_1, \ldots, b_r\}$) are chosen according to some distribution that is independent of y (resp., x). For $i \geq i^*$, $a_i = b_i = f(x, y)$.
- Each a_i is randomly divided into shares $a_i^{(1)}$, $a_i^{(2)}$ with $a_i^{(1)} \oplus a_i^{(2)} = a_i$ (and similarly for each b_i). The stage concludes with P_1 being given $a_1^{(1)}, b_1^{(1)}, \ldots, a_r^{(1)}, b_r^{(1)}$

[2] We do not consider U''- the utility when neither party learns the output, since 'not learning the output' is not the target of a rational adversary in practice.

, and P_2 being given $a_1^{(2)}, b_1^{(2)}, \ldots, a_r^{(2)}, b_r^{(2)}$ alongside the VRFs [3] ($ShareGen_r$ provides the parties with VRFs so that if a malicious party modifies the share it sends to the other party, then the other party will almost certainly detect this due to the property of VRFs. It will be treated as an abort if such manipulation is detected.).

After this stage, each party has a set of random shares that reveal nothing about the other party's input.

Intersection Computation Phase:

In each iteration i, for $i = 1, \ldots, r$, the parties do the following: First, P_2 sends $a_i^{(2)}$ to P_1 who reconstructs a_i; then P_1 sends $b_i^{(1)}$ to P_2 who reconstructs b_i. (Parties also checks the VRF but we omit this here.) If a party aborts in some iteration i, then the other party outputs the value reconstructed in the previous iteration. Otherwise, after reaching iteration r the parties output a_r and b_r, respectively. To compute the correct intersection, parties run a sequence of iterations until the real iteration is identified, and both parties output the result at that point. If some party fails to follow the protocol, the other party aborts. In fact, it is rational for P_j to follow the protocol as long as the expected gain of deviating is positive only if P_j aborts exactly in iteration i^*; and is outweighed by the expected loss if P_j 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. Since we want to avoid simultaneous communication, we simply require P_2 to communicate first in each iteration.

When X and Y (the domains of f) are polynomial size, we follow [13, 21] and set $a_i = f(x, \hat{y})$ for $\hat{y}$ chosen uniformly from Y , and set $b_i = f(\hat{x}, y)$ for $\hat{x}$ chosen uniformly (and independently) from X. Note that a_i (resp., b_i) is independent of y (resp., x), as desired.

4.2 Protocol Construction

As described above, our protocol Π consists of two stages. Let p be an arbitrary polynomial, and set $r = p \cdot \mid Y \mid$. We implement the first stage of Π using a subprotocol π for computing a randomized functionality $ShareGen_r$ (parameterized by a polynomial r) defined in Figure 1. This functionality returns shares to each party, alongside r-time VRF $(Gen, Eval, Prove, Verify)$. In the second stage of Π, the parties exchange these shares in a sequence of r iterations as described in Figure 2. The protocol returns I at the end of the operations on all the data items.

[3] It is the parties' own interest that they input the correct values for $ShareGen_r$. Otherwise, they will receive incorrect shares that will give them no chance to compute the correct intersection result, which will only enable them of having smaller incentives.

Input: Let the inputs to $ShareGen_r$ be $x \in X_n$ and $y \in Y_n$. (If one of the received inputs is not in the correct domain, a default input is substituted.)

Computation:

- Define values $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ in the following way:
  - Choose i^* according to some geometric distribution α
  - For $i < i^*$ do,
    - Choose $\hat{y} \leftarrow Y_n$ and set $a_i = f_n(x, \hat{y})$
    - Choose $\hat{x} \leftarrow X_n$ and set $b_i = f_n(\hat{x}, y)$
  - For $i = i^*$, set $a_i = b_i = q = f_n(x, y)$.
  - For $i > i^*$, set $a_i = b_i = NULL$
- For all iteration i, choose $(a_i^{(1)}, a_i^{(2)})$ and $(b_i^{(1)}, b_i^{(2)})$ as random secret shares of a_i and b_i, respectively. (I.e., $a_i^{(1)} \oplus a_i^{(2)} = a_i$, $b_i^{(1)} \oplus b_i^{(2)} = b_i$)
- Let $\mathcal{D} = \{0,1\}^l$ be the domain of the output. Let $(Gen, Eval, Prove, Verify)$ and $(Gen', Eval', Prove', Verify')$ be VRFs with range $\{0,1\}^l$ and $\{0,1\}^n$, respectively. Compute (pk_1, sk_1), $(pk_2, sk_2) \leftarrow Gen(1^n)$ and (pk_1', sk_1'), $(pk_2', sk_2') \leftarrow Gen'(1^n)$. For all i, compute $share1_i = Eval_{sk_2}(i \| b_i^{(1)})$ and $share2_i = Eval_{sk_1}(i \| a_i^{(1)})$. Also compute $signal1 = Eval'_{sk_2'}(i^* + 1)$ and $signal2 = Eval'_{sk_1'}(i^* + 1)$

Output:

- Send to P_1 the values $(sk_1, sk_1', pk_2, pk_2', a_1^{(1)}, \ldots, a_r^{(1)}, (b_1^{(1)}, share1_1), \ldots, (b_r^{(1)}, share1_r), signal1)$.
- Send to P_2 the values $(sk_2, sk_2', pk_1, pk_1', b_1^{(1)}, \ldots, b_r^{(1)}, (a_1^{(1)}, share2_1), \ldots, (a_r^{(1)}, share2_r), signal2)$.

Fig. 1. Functionality $ShareGen_r$

5 Protocol Analysis

Here we will give some intuition as to why the reconstruction phase of Π is a computational Nash equilibrium for an appropriate choice of α. 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.) As soon as it receives $z_2^{(i)} = signal1$, P_1 can abort in iteration $i = i^* + 1$, or it can abort in some iteration $i < i^* + 1$. While aborting in $i = i^* + 1$, P_1 'knows' that it learned the correct output in the preceding iteration (iteration i^*) and can thus output the correct result; however, P_2 will output the correct result as well since it sent the $z_2^{(i)} = signal1$ value to P_1. So P_1 does not increase its utility beyond what it would achieve by following the protocol. In the second case, when P_1 aborts in some iteration $i < i^* + 1$, the best strategy P_1 can adopt is to output $a_1^{(i)}$ hoping

Input: Party P_1 has input x and party P_2 has input y.

Computation:

- Preliminary phase:
  1. P_1 chooses $\hat{y} \in Y_n$ uniformly at random, and sets $a_0 = f_n(x, \hat{y})$. Similarly, P_2 chooses $\hat{x} \in X_n$ uniformly at random, and sets $b_0 = f_n(\hat{x}, y)$.
  2. Parties P_1 and P_2 run a protocol π to compute $ShareGen_r$, using their inputs x and y.
  3. If P_2 receives $\perp$ from the above computation, it outputs b_0 and halts. Otherwise, the parties proceed to the next step.
  4. Denote the output of P_1 from π by $(sk_1, sk'_1, pk_2, pk'_2, a_1^{(1)}, \ldots, a_r^{(1)}, (b_1^{(1)}, share1_1), \ldots, (b_r^{(1)}, share1_r), signal1)$.
  5. Denote the output of P_2 from π by $(sk_2, sk'_2, pk_1, pk'_1, b_1^{(1)}, \ldots, b_r^{(1)}, (a_1^{(1)}, share2_1), \ldots, (a_r^{(1)}, share2_r), signal2)$.
- Intersection Computation Phase
  For all i do:
  P_2 **sends message to** P_1:
  1. P_2 computes $y_2^{(i)} = Prove_{sk_2}(i \| a_i^{(2)}), z_2^{(i)} = Eval'_{sk'_2}(i), \bar{z}_2^{(i)} = Prove'_{sk'_2}(i)$. It sends $(a_i^{(2)}, share2_i, y_2^{(i)}, z_2^{(i)}, \bar{z}_2^{(i)})$ to P_1.
  2. If P_2 does not send anything to P_1, then P_1 outputs a_{i-1} and halts. P_2 sends $(a_i^{(2)}, share2_i, y_2^{(i)}, z_2^{(i)}, \bar{z}_2^{(i)})$ to P_1. If $Verify_{pk_2}(i \| a_i^{(2)}, share2_i, y_2^{(i)}) = 0$ or $Verify'_{pk'_2}(i, z_2^{(i)}, \bar{z}_2^{(i)}) = 0$, then P_1 outputs a_{i-1} and halts. If $signal1 \neq z_2^{(i)}$ then P_1 outputs a_{i-1}, sends its iteration-i message to P_2, and halts.
  3. If $Verify_{pk_2}(i \| a_i^{(2)}, share2_i, y_2^{(i)}) = 1$ and $a_i^{(1)} \oplus a_i^{(2)} \neq NULL$ (i.e., $x = x_i$), then P_1 sets $a_i = a_i^{(1)} \oplus a_i^{(2)}$, and continues running the protocol.
  P_1 **sends message to** P_2:
  1. P_1 computes $y_1^{(i)} = Prove_{sk_1}(i \| b_i^{(1)}), z_1^{(i)} = Eval'_{sk'_1}(i), \bar{z}_1^{(i)} = Prove'_{sk'_1}(i)$. It sends $(b_i^{(1)}, share1_i, y_1^{(i)}, z_1^{(i)}, \bar{z}_1^{(i)})$ to P_2.
  2. If P_1 does not send anything, then P_2 outputs b_{i-1} and halts. P_1 sends $(b_i^{(1)}, share1_i, y_1^{(i)}, z_1^{(i)}, \bar{z}_1^{(i)})$ to P_2. If $Verify_{pk_1}(i \| b_i^{(1)}, share1_i, y_1^{(i)}) = 0$ or $Verify'_{pk'_1}(i, z_1^{(i)}, \bar{z}_1^{(i)}) = 0$, then P_2 outputs b_{i-1} and halts. If $signal2 \neq z_1^{(i)}$ then P_2 outputs b_{i-1}, sends its iteration-i message to P_1, and halts.
  3. If $Verify_{pk_1}(i \| b_i^{(1)}, share1_i, y_1^{(i)}) = 1$ and $b_i^{(1)} \oplus b_i^{(2)} \neq NULL$ (i.e., $y = y_i$), then P_2 sets $b_i = b_i^{(1)} \oplus b_i^{(2)}$, and continues running the protocol.

Output: If all r iterations have been run, party P_1 outputs a_r and party P_2 outputs b_r.

Fig. 2. Protocol for computing the functionality for set-intersection

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 a_r 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 α 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 Π induces a strict computational Nash equilibrium which is stable with respect to trembles.

The detailed proof of the following propositions will be given in the full version of the paper.

Proposition 1. *The protocol Π induces a computational Nash equilibrium given that $0 < \alpha < 1$, $U > \alpha \times U^* + (1 - \alpha) \times U_{rand}$, and the pseudorandomness of VRFs.*

Proposition 2. *If $0 < \alpha < 1$, $U > \alpha \times U^* + (1 - \alpha) \times U_{rand}$, VRFs are pseudorandom, and there is always a unique valid message each party can send, then the protocol Π induces a computational strict Nash equilibrium.*

Proposition 3. *The protocol Π is stable with respect to trembles given that $0 < \alpha < 1$ and $U > \alpha \times U^* + (1 - \alpha) \times U_{rand}$.*

According to the above propositions and their proofs, we give the theorem as follows:

Theorem 1. *If $0 < \alpha < 1$, $U > \alpha \times U^* + (1 - \alpha) \times U_{rand}$, and VRFs are pseudorandom, then Π induces a computational strict Nash equilibrium that is stable with respect to trembles.*

6 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 verifiable random functions as the underlying cryptographic primitive which is simple and efficient. It is also possible to use our protocol for computing set-union operations. We also show that our protocol satisfies computational versions of strict Nash equilibrium and stability with respect to trembles. Applying game-theoretic approach for multi-party setting where parties are allowed to collude is an interesting open problem.

References

1. Abraham, I., Dolev, D., Gonen, R., and Halpern, J.: Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multi-party Computation. In 25th ACM Symposium Annual on Principles of Distributed Computing, pp. 53-62, 2006.
2. Agrawal, R. and Terzi, E.: On Honesty in Sovereign Information Sharing. In the 10th International Conference on Extending Database Technology- EDBT'06, pp. 240-256 2006.
3. Aumann, Y. and Lindell, Y.: Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries, In Theory of Cryptography- TCC'07, pp. 137-156, 2007.
4. Bellare M. and Micali S.: Non-interactive Oblivious Transfer and Applications. In Advances in Cryptology- CRYPTO'89. pp. 547-557, 1989.
5. Cramer, R., Damgard, I., and Nielsen, J.B.: Multi-party Computation from Threshold Homomorphic Encryption. In Advances in Cryptology- EUROCRYPT'01, pp. 280-299, 2001.
6. Dodis, Y.: Efficient Construction of (distributed) Verifiable Random Functions. In 6th International Workshop on Theory and Practice in Public Key Cryptography- PKC'03, pp. 1-17, 2003.
7. Dodis, Y. and Rabin, T.: Cryptography and Game Theory. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, Algorithmic Game Theory, pp. 181-207, Cambridge University Press, 2007.
8. Dodis, Y. and Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In 8th International Workshop on Theory and Practice in Public Key Cryptography- PKC'05, pp. 416-431, 2005.
9. Emura, K., Miyaji, A., and Rahman, M.S.: Efficient Privacy-Preserving Data Mining in Malicious Model. In The 6th International Conference on Advanced Data Mining and Applications, ADMA'10. pp. 370-382, 2010.
10. Fuchsbauer, G., Katz, J., and Naccache, D.: Efficient Rational Secret Sharing in Standard Communication Networks. In Theory of Cryptography- TCC'10, pp. 419-436, 2010.
11. Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge Univ. Press, Cambridge, 2004.
12. Gordon, S.D., Katz, J.: Rational Secret Sharing, Revisited. In 5th International Conference on Security and Cryptography for Networks- SCN'06, pp. 229-241, 2006.
13. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete Fairness in Secure Two-party Computation. In 40th Annual ACM Symposium on Theory of Computing-STOC'08, pp. 413-422, 2008.
14. Halpern, J. and Teague, V.: Rational Secret Sharing and Multi-party Computation: Extended abstract. In 36th Annual ACM Symposium on Theory of Computing-STOC'04, pp. 623-632, 2004.
15. Izmalkov, S., Micali, S., and Lepinski, M.: Rational Secure Computation and Ideal Mechanism Design. In 46th Annual Symposium on Foundations of Computer Science- FOCS'05, pp. 585-595, 2005.
16. Jagannathan, G. and Wright, R.N.: Privacy-preserving Distributed k-means Clustering over Arbitrarily Partitioned Data. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining- KDD'05, pp. 593-599, 2005.

17. Jiang, W., Clifton, C. and Kantarcioglu, M.: Transforming Semi-Honest Protocols to Ensure Accountability. In Data and Knowledge Engineering (DKE), 65(1), pp. 57-74, 2008.
18. Kantarcioglu, M. and Clifton, C.: Privately Computing a Distributed k-nn Classifier. In 7th European Conference on Principles and Practice of Knowledge Discovery in Databases- PKDD'04, pp. 279-290, 2004.
19. Kantarcioglu, M., and Kardes, O.: Privacy-preserving Data Mining in the Malicious model. In International Journal of Information and Computer Security, Vol. 2, No. 4, pp. 353-375, 2008.
20. Katz, J.: Bridging Game Theory and Cryptography: Recent Results and Future Directions. In Theory of Cryptography- TCC'08. pp. 251-272, 2008.
21. Katz, J.: On Achieving the Best of Both Worlds in Secure Multi-party Computation. In 39th Annual ACM Symposium on Theory of Computing- STOC'07, pp. 11-20, 2007.
22. Kol, G. and Naor, M.: Cryptography and Game Theory: Designing Protocols for Exchanging Information. In Theory of Cryptography- TCC'08, pp. 320-339, 2008.
23. Kol, G. and Naor, M.: Games for Exchanging Information. In 40th Annual ACM Symposium on Theory of Computing- STOC'08, pp. 423-432, 2008.
24. Lin, X., Clifton, C. and Zhu, M.: Privacy-preserving Clustering with Distributed EM Mixture Modeling. In Knowledge and Information Systems, July, Vol. 8, No. 1, pp. 68-81, 2005.
25. Lindell, Y. and Pinkas, B.: Privacy-preserving Data Mining. In Advances in Cryptology- CRYPTO'00, pp. 36-54, 2000.
26. Lipmaa, H.: Verifiable Homomorphic Oblivious Transfer and Private Equality Test. In Advances in Cryptology- ASIACRYPT'03, pp. 416-433, 2003.
27. Liu, J., Lu, Y.H., and Koh, C.K.: Performance Analysis of Arithmetic Operations in Homomorphic Encryption. In ECE Technical Reports, Purdue University, 2010.
28. Lysyanskaya, A.: Unique Signatures and Verifiable Random Functions from the DH-DDH Separation. In Advances in Cryptology- CRYPTO'02, pp. 597-612, 2002.
29. Lysyanskaya, A., Triandopoulos, N.: Rationality and Adversarial behavior in Multi-party computation. In Advances in Cryptology- CRYPTO'06, pp. 180-197, 2006.
30. Micali, S., Rabin, M. O., and Vadhan, S. P.: Verifiable Random Functions. In 40th Annual Symposium on Foundations of Computer Science- FOCS'99, pp. 120-130, 1999.
31. Miyaji, A., and Rahman, M.S.: Privacy-preserving Data Mining in Presence of Covert Adversaries. In The 6th International Conference on Advanced Data Mining and Applications, ADMA'10. pp. 429-440, 2010.
32. Nix, R. and Kantarcioglu, M.: Incentive Compatible Distributed Data Mining. In IEEE International Conference on Privacy, Security, Risk and Trust, pp. 735-742, 2010.
33. Ong, S. J., Parkes, D., Rosen, A., and Vadhan, S.: Fairness with an Honest Minority and a Rational Majority. In Theory of Cryptography- TCC'09, pp. 36-53, 2009.
34. Su, C., Bao, F., Zhou, J., Takagi, T., Sakurai, K.: Security and Correctness Analysis on Privacy-Preserving k-Means Clustering Schemes. In IEICE Trans. Fundamentals, Vol.E92-A, No.4, pp. 1246-1250, 2009.
35. Vaidya, J. and Clifton, C.: Privacy Preserving Association Rule Mining in Vertically Partitioned Data. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining- KDD'02, pp. 639-644, 2002.
36. Zhang, N. and Zhao, W.: Distributed Privacy-preserving Information Sharing. In the 31st International Conference on Very large data bases- VLDB'05, pp. 889-900, 2005.