# Privacy-preserving Two-party Rational Set Intersection Protocol

Atsuko Miyaji and Mohammad Shahriar Rahman
Japan Advanced Institue of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa, Japan 923-1292.
E-mail: miyaji@jaist.ac.jp, md.shahriarr@gmail.com

**Keywords:** privacy-preserving set-intersection, game theory, computational strict Nash equilibrium, stability with respect to trembles

**Received:**

*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 self-interest. In this paper, we build efficient and secure two-party set-intersection protocol in game-theoretic setting using cryptographic primitives. Our construction allow to avoid the use of expensive tools like homomorphic encryption and zero knowledge proof. We also show that our protocol satisfies computational versions of strict Nash equilibrium and stability with respect to trembles.*

*Povzetek: Predstavljen je protokol med dvema stranema na osnovi Nashevega ravnotežja.*

## 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, called Semi-honest and malicious adversaries. Following Goldreich's definition [9], 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. On the other hand, security in the presence of malicious parties allows arbitrary deviations from the protocol. It is well known that the protocols secure in the malicious model offer more security. 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 [28]. Recently, a new type of adversarial model, named covert adversary, has been proposed by Aumann et al. [3]. These adversaries are somewhere in between the semi-honest and malicious models.

Since the work of Halpern and Teague [12], protocols for some cryptographic tasks (e.g., secret sharing, multi-party computation) have begun to be re-evaluated in a game-theoretic light (see [6, 18] 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.

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 produce's 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. Secure distributed protocols have been developed for horizontally partitioned data for mining decision trees [24], k-means clustering [22], k-nn classifiers [16]. 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 [33], and k-means clusters [14, 32]. All of those previous protocols claimed to be secure only in the semi-honest model. In [7, 17], 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. [4]. Recently, Miyaji et al. presented a new adversarial model named covert adversaries [28] 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 [28]. Since homomorphic encryption and zero-knowledge proof are considered too expensive [25], 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 [15, 30] for constructing various data mining algorithms. Rational adversaries have also been considered in privacy-preserving set operations [2, 34]. These protocols consider Nash equilibrium to analyze the rational behavior of the participating entities. As discussed by Kol and Naor in [21], 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 more realistic results. There have been several works on game theory based MPC/secret sharing schemes [1, 8, 12, 13, 20, 31]. But [12, 31] 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, 20] rely on multiple invocations of protocols. Other work [13] relies on physical assumptions such as secure envelopes and ballot boxes. [8] 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[1], 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. Our construction does not use the expensive tools like homomorphic encryption and zero-knowledge proof. We have used verifiable random functions (VRF) as the underlying cryptographic primitive. We also discuss about replacing VRF with a cheaper tool like message authentication code (MAC) or trapdoor permutation (TDP). We show that our protocol satisfies computational versions of strict Nash equilibrium and stability with respect to trembles, defined by Fuchsbauer et al. [8].

**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. Section 6 includes performance comparison. We give some concluding remarks in Section 7.

[1] A preliminary version [29] of this paper appears at DBSec2011.

# 2 Background and Preliminary

## 2.1 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 [8]. 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 $\epsilon$ 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 $negl$ 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 $\vec{\sigma}$ for two parties in the computation phase, let $u_j(\vec{\sigma})$ 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, $(\sigma'_j, \vec{\sigma}_{-j})$ denotes the strategy vector $\vec{\sigma}$ 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* $u_1(\sigma'_1, \sigma_2) \leq u_1(\sigma_1, \sigma_2) + negl(n)$*, 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 $\delta$ (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 $\sigma_1$ and $\sigma_2$, 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 $\gamma_1$ 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 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* $\gamma_1$ *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* $D$*, the following holds:* $| Pr[D(1^n, view_2^{A,\gamma_1}) = 1] - Pr[D(1^n, view_2^{\Pi}) = 1] | \leq negl(n)$

**Definition 3:** $\Pi$ *induces a computational strict Nash equilibrium if: 1.* $\Pi$ *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 $\gamma_i$ is $\delta$-close to $\sigma_j$ if with probability $1 - \delta$ party $P_j$ plays $\sigma_j$, while with probability $\delta$ it follows an arbitrary PPT strategy $\sigma'_j$. In fact, a pair of strategies $(\sigma_1, \sigma_2)$ is stable with respect to trembles if $\sigma_1$ (resp., $\sigma_2$) remains the best response even if the other party plays a strategy other than $\sigma_2$ (resp., $\sigma_1$) with some small (but noticeable) probability $\delta$. 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 $\delta$, there is still no better strategy for $P_j$ than to outwardly follow the protocol.

**Definition 4:** $\Pi$ *induces a computational strict Nash equilibrium that is stable with respect to trembles if: 1.* $\Pi$ *induces a computational Nash equilibrium; 2. There is a noticeable function* $\delta$ *such that for any PPT strategy* $\gamma_2$ *that is* $\delta$*-close to* $\sigma_2$*, and any PPT strategy* $\gamma_1$*, 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. [27], and various efficient constructions in the standard model are known [5, 26]. It has been shown in [26] that efficient VRFs can be constructed without relying on zero-knowledge proofs[2]. A 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 properties like correctness, verifiability and pseudorandomness hold.

[2]The VRF gives us computational security. However, it is also possible to design our protocol with information-theoretic security using information-theoretically secure MACs.

# 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 [12], 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[3] (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. 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, 10, 20] 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, delaying the signal indicating whether a given iteration is real or fake until the following iteration has been used. In this case, until being sure of the occurence 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.

[3] 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.

# 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 [1, 10, 12, 20, 21]. Our intersection computation protocol proceeds in a sequence of 'fake' iterations followed by a single 'real' iteration. As in [8, 11, 19], 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. More specifically, in the pre-processing phase the trusted third party chooses $i^* \in \{1, \ldots, r\}$ uniformly at random and defines $\{a_i\} = \{a_1, \ldots, a_r\}$ and $\{b_i\} = \{b_1, \ldots, b_r\}$ as follows: First, it choose $a_1, \ldots, a_{i^*-1} \in \{0, 1\}$ and $b_1, \ldots, b_{i^*-1} \in \{0, 1\}$ independently and uniformly at random. Then, it chooses $c \in \{0, 1\}$ uniformly at random and lets $a_{i^*} = \cdots = a_r = b_{i^*} = \cdots = b_r = c$. The trusted third party creates secret shares of the values $\{a_1, \ldots, a_r\}$ and $\{b_1, \ldots, b_r\}$ using a secure 2-out- of-2 secret sharing scheme, and these shares are given to the parties. For concreteness, we use the specific secret-sharing scheme that splits a bit $x$ into $(x^{(1)}; x^{(2)})$ by choosing $x^{(1)} in \{0, 1\}$ uniformly at random and letting $x^{(2)} = x \oplus x^{(1)}$. In every round $i \in \{1, \ldots, r\}$ the parties exchange their shares for the current round, which enables $P_1$ to reconstruct $a_i$, and $P_2$ to reconstruct $b_i$ as discussed in the Intersection Computation Phase below. Clearly, when both parties are honest, the parties produce the same output bit which is uniformly distributed.

Now, we talk about how to remove the trusted party. We eliminate the need for the trusted third party by relying on a potentially unfair sub-protocol that securely computes with abort the functionality $ShareGen_r$, formally described in Figure 1. Such a protocol with a constant number of rounds can be constructed assuming the existence of oblivious transfer as in [23]. 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 $\alpha$ 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 distribu-

tion 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 $a_1^{(1)}, b_1^{(1)}, \ldots, a_r^{(1)}, b_r^{(1)}$, and $P_2$ being given $a_1^{(2)}, b_1^{(2)}, \ldots, a_r^{(2)}, b_r^{(2)}$ alongside the VRFs [4] ($ShareGen_r$ 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 check 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 [11, 19] 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 $\Pi$ consists of two stages. Let $p$ be an arbitrary polynomial, and set $r = p \cdot \mid Y \mid$. We implement the first stage of $\Pi$ using a sub-protocol $\pi$ 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 $\Pi$, 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.

[4] 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.

# 5 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.) 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 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 $\alpha$ 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 strict computational Nash equilibrium which is stable with respect to trembles. The proofs of the propositions below mostly follow those in [8].

**Proposition 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 pseudorandomness of VRFs.*

*Proof:* We first show that $\Pi$ is a valid set-intersection protocol. Computational secrecy follows from the proof, below, that the intersection computation is a computational Nash equilibrium. Because if secrecy did not hold then computing the output locally and not participating in the intersection computation phase at all would be a profitable deviation. We next focus on correctness. Assuming both parties run the protocol honestly, the correct output is computed unless:
- $i^* \geq 2^n - 1$

Input: Let the inputs to $ShareGen_r$ be $x \in X_r$ and $y \in Y_r$. (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 $\alpha$
  - For $i < i^*$ do,
    - Choose $\hat{y} \leftarrow Y_r$ and set $a_i = f_r(x, \hat{y})$
    - Choose $\hat{x} \leftarrow X_r$ and set $b_i = f_r(\hat{x}, y)$
  - For $i = i^*$, set $a_i = b_i = q = f_r(x, y)$.
  - 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 $(sk_2, sk_2', pk_1, pk_1', b_1^{(1)}, \ldots, b_r^{(1)}, (a_1^{(1)}, share2_1), \ldots, (a_r^{(1)}, share2_r), signal2)$.

Figure 1: Functionality $ShareGen_r$

- For some $i < i^* + 1$, either $signal1 = Eval'_{sk_2'}(i)$ or $signal2 = Eval'_{sk_1'}(i)$

The first event occurs with negligible probability. Pseudorandomness of the VRF, along with the fact that $i^* \leq n$ with all but negligible probability, easily imply that the latter two events happen with only negligible probability as well. We next show that $\Pi$ induces a computational Nash equilibrium. Assume $P_2$ follows the strategy $\sigma_2$ prescribed by the protocol, and let $\sigma_1'$ denote any PPT strategy followed by $P_1$. (The other case, where $P_1$ follows the protocol and we look at deviations by $P_2$, follows similarly with an even simpler approach.) In a given execution of the reconstruction phase, let $i$ denote the iteration in which $P_1$ aborts (where an incorrect message is viewed as an abort); if $P_1$ never aborts then set $i = 1$. Let $early$ be the event that $i < i^*$; let $exact$ be the event that $i = i^*$; and let $late$ be the event that $i > i^*$. Let $correct$ be the event that $P_1$ outputs the correct output. We will consider the probabilities of these events in two experiments: the experiment defined by running the actual intersection computation scheme, and a second experiment where $P_1$ is given $share1$, $signal1$ chosen uniformly at random from the appropriate ranges. We denote the probabilities in the first experiment by $Pr_{real}[\cdot]$, and the probabilities in the second experiment by $Pr_{ideal}[\cdot]$. We have the following equation using the fact (as discussed above) that whenever late occurs $P_2$ outputs the correct result. Since when both parties follow the protocol $P_1$ gets utility $U$, we need to show that there exists a negligible function $\epsilon$ such that $u_1(\sigma_1', \sigma_2) \leq U + \epsilon(n)$:
$u_1(\sigma_1', \sigma_2) \leq U^* \times Pr_{real}[exact] + U^* \times Pr_{real}[correct \wedge early] + U' \times Pr_{real}[\overline{correct} \wedge early] + U \times Pr_{real}[late]$
Now we have the following claim that follows from the pseudorandomness of the VRFs:
*Claim 1*: There exists a negligible function $\epsilon$ such that

$$
\begin{aligned}
| \, Pr_{real}[exact] &- Pr_{ideal}[exact] \, | \leq \epsilon(n) \\
| \, Pr_{real}[late] &- Pr_{ideal}[late] \, | \leq \epsilon(n) \\
| \, Pr_{real}[correct \wedge early] &- Pr_{ideal}[correct \wedge early] \, | \leq \epsilon(n) \\
| \, Pr_{real}[\overline{correct} \wedge early] &- Pr_{ideal}[\overline{correct} \wedge early] \, | \leq \epsilon(n)
\end{aligned}
$$

Now, we have $U_{ideal} = U^* \cdot Pr_{ideal}[exact] + U^* \cdot Pr_{ideal}[correct \wedge early] + U' \cdot Pr_{ideal}[\overline{correct} \wedge early] + U \cdot Pr_{ideal}[late]$

From Claim 1 we get that $| \, u_1(\sigma_1', \sigma_2) - U_{ideal} \, | \leq \epsilon(n)$ for some negligible $\epsilon$. We bound $U_{ideal}$ as follows: Let $abort = exact \vee early$, so that $abort$ is the event that $P_1$ aborts before iteration $i^* + 1$. We have $Pr_{ideal}[exact \mid abort] = \alpha$ and $Pr_{ideal}[correct \mid early] = 1/\mathcal{D}$. It is

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

Computation:

– Preliminary phase:
1. $P_1$ chooses $\hat{y} \in Y_r$ uniformly at random, and sets $a_0 = f_r(x, \hat{y})$. Similarly, $P_2$ chooses $\hat{x} \in X_r$ uniformly at random, and sets $b_0 = f_r(\hat{x}, y)$.
2. Parties $P_1$ and $P_2$ run a protocol $\pi$ 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 $\pi$ 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 $\pi$ 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 $b_r$.

Figure 2: Protocol for computing the functionality for set-intersection

easy to find that $U_{ideal} = U + (\alpha \cdot U^* + (1-\alpha) \cdot U_{rand} - U) \cdot Pr_{ideal}[abort] \leq U$ given that $\alpha \cdot U^* + (1-\alpha) \cdot U_{rand} - U < 0$. This shows that $\Pi$ induces a computational Nash equilibrium.

**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 $\Pi$ induces a computational strict Nash equilibrium.*

*Proof:* The analysis of Proposition 1 and the fact that there is always a unique valid message each party can send show us that $\Pi$ induces a computational strict Nash equilibrium. In other words, say $P_1$ plays a strategy $\sigma'_1$ with $\sigma'_1 \not\approx \Pi$. This implies that $Pr_{real}[abort] \geq 1/poly(n)$ for infinitely many values of $n$. Claim 1 then shows that $Pr_{ideal}[abort] \geq 1/poly(n)$ for infinitely many values of $n$, and so $U - U_{ideal} \geq 1/poly(n)$. Since $\mid u_1(\sigma'_1, \sigma_2) - U_{ideal} \mid$ is negligible, we conclude that $U - u_1(\sigma'_1, \sigma_2) \geq 1/poly(n)$ for infinitely many values of $n$.

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

*Proof:* Let $\delta$ be a parameter. Let $\rho_2$ be any PPT strategy that is $\delta$-close to $\sigma_2$, and let $\rho_1$ be an arbitrary PPT strategy for $P_1$. There exists a PPT strategy $\sigma'_1$ satisfying Definition 3. Let strategy $\sigma'_1$ be defined as follows:

1. Run $\rho_1$ on the output of $ShareGen_r$. Set $aborted = 0$.
2. In each iteration i:
   – Receive the iteration-$i$ message $m_i$ from $P_2$. If $P_2$ aborts, then set $aborted = 1$.

– Give $m_i$ to $\rho_1$ and get message $m_i'$ as response.

– If $aborted = 1$ then forward $m_i'$ to $P_2$; otherwise, compute the response (e.g., the protocol transcripts) as prescribed by $\Pi$ and send that to $P_2$ instead.

3. If $aborted = 0$ determine the output according to $\Pi$; otherwise, output whatever $\rho_1$ outputs.

When $\sigma_1'$ interacts with $\sigma_2$, then $aborted$ is never set to 1; thus, $\sigma_1'$ yields equivalent play w.r.t $\Pi$, and $u_1(\sigma_1', \sigma_2) = u_1(\rho_1, \rho) = U$. It remains to show $u_1(\rho_1, \rho_2) \leq u_1(\sigma_1', \rho_2) + negl(n)$. Let $\hat{\rho}_2$ is run only with probability $\delta$ by $\rho_2$. During a session where $P_1$ follows strategy $\rho_1$, let $abort$ denote the event that $\rho_1$ aborts before $P_2$ aborts, and let $pr_{abort}(a)$ be the probability of $abort$ when $P_2$ follows strategy $a$. We now state two claims. The first one says that the only advantage to $P_1$ of playing $\rho_1$ rather than $\sigma_1'$ because of $\sigma_1$ aborting first.

*Claim 2:* $u_1(\rho_1, \hat{\rho}_2) - u_1(\sigma_1', \hat{\rho}_2) \leq pr_{abort}(\hat{\rho}_2) \cdot (U^* - U')$

The following claim shows that $abort$ occurs at least as often when $\rho_1$ interacts with $\sigma_2$ as when $\rho_1$ interacts with $\hat{\rho}_2$.

*Claim 3:* $pr_{abort}(\sigma_2) \geq pr_{abort}(\hat{\rho}_2)$

We omit the proofs of the above since they are analogous to those in [8].

Now, let $\tilde{U} = \alpha \times U^* + (1 - \alpha) \times U_{rand}$, and we have $\tilde{U} < U$ by assumption. Using $U_{ideal} \leq U$, *Claim 1*, *Claim 2*,and *Claim 3* we get that $u_1(\rho_1, \rho_2) - u_1(\sigma_1', \rho_2) \leq (1 - \delta) \times (\tilde{U} - U) \times pr_{abort}(\hat{\rho}_2) + \delta \times (U^* - U') \times pr_{abort}(\hat{\rho}_2) + negl(n)$. Since $\tilde{U} - U$ is strictly negative, there exists $\delta > 0$ for which the above expression is negligible for $n$ large enough. This completes the proof sketch.

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* $\Pi$ *induces a computational strict Nash equilibrium that is stable with respect to trembles.*

# 6 Performance Comparison

For a single data item, the protocol in covert model [28] requires only a constant number of rounds, single oblivious transfer to the number of input items, and requires $n|C|$ number of communication bits where $n$ is the security parameter and $|C|$ is the size of the circuit being computed. Whereas the protocol in malicious model [17] requires $d$ number of rounds ($d$ is the depth of $C$), more communication bits (dependent on the number of parties), and expensive computation like ZK proof which is linear to the number of data items. Both the covert and malicious models rely on homomorphic operations. On the other hand, in our rational model, we do not need any ZK proof or homomorphic encryption computation. As discussed earlier, use of ZK proof and homomorphic encryption leads to inefficiency in practical world and we want to avoid using the expensive tool like ZK proofs. As for the other parameters in rational model, the share size is $|t| + O(n)$, where $t$ is the size of data items and $n$ is the security parameter. The round complexity of the protocol for each item is $O(\alpha^{-1})$, where $\alpha$ is the geometric distribution used to pick up the value of $i^*$ (typically, we will need only two rounds for each items in our protocol). In our construction, we have showed the use of VRFs, which is also an expensive tool. However, it is possible to design our protocol with information-theoretic security using information-theoretically secure MACs. It is also possible to replace VRFs with TDPs, since the properties of VRF that we require for our constructions are also available with TDPs. Using TDPs would give us much more efficient protocol as compared to using VRFs. Construction using TDP is straightforward and we omit the details here. Clearly, the rational model requires much lighter computation than the protocol designed in malicious model and performs even better than the covert model in terms of computational overhead given that MAC or TDP is used instead of VRF.

# 7 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 VRFs as the underlying cryptographic primitive. We also suggest replacing VRFs with information-theoretic secure MACs or TDPs, which are simple and efficient. Our protocol satisfies strong equilibrium notions like computational versions of strict Nash equilibrium and stability with respect to trembles.

# References

[1] Abraham, I., Dolev, D., Gonen, R., and Halpern, J.: Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty 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] Cramer, R., Damgard, I., and Nielsen, J.B.: Multiparty Computation from Threshold Homomorphic Encryption. In Advances in Cryptology- EUROCRYPT'01, pp. 280-299, 2001.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge Univ. Press, Cambridge, 2004.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Katz, J.: Bridging Game Theory and Cryptography: Recent Results and Future Directions. In Theory of Cryptography- TCC'08. pp. 251-272, 2008.

[19] 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.

[20] Kol, G. and Naor, M.: Cryptography and Game Theory: Designing Protocols for Exchanging Information. In Theory of Cryptography- TCC'08, pp. 320-339, 2008.

[21] Kol, G. and Naor, M.: Games for Exchanging Information. In 40th Annual ACM Symposium on Theory of Computing- STOC'08, pp. 423-432, 2008.

[22] 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.

[23] Lindell,Y: Parallel coin-tossing and constant-round secure two-party computation. Journal of Cryptology, 16(3):143-184, 2003.

[24] Lindell, Y. and Pinkas, B.: Privacy-preserving Data Mining. In Advances in Cryptology- CRYPTO'00, pp. 36-54, 2000.

[25] Liu, J., Lu, Y.H., and Koh, C.K.: Performance Analysis of Arithmetic Operations in Homomorphic Encryption. In ECE Technical Reports, Purdue University, 2010.

[26] Lysyanskaya, A.: Unique Signatures and Verifiable Random Functions from the DH-DDH Separation. In Advances in Cryptology- CRYPTO'02, pp. 597-612, 2002.

[27] 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.

[28] 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.

[29] Miyaji, A., and Rahman, M.S.: Privacy-Preserving Data Mining: A Game-Theoretic Approach. In The 25th Annual WG 11.3 Conference on Data and Applications Security and Privacy, DBSec'11. pp. 186-200, 2011.

[30] Nix, R. and Kantarcioglu, M.: Incentive Compatible Distributed Data Mining. In IEEE International Conference on Privacy, Security, Risk and Trust, pp. 735-742, 2010.

[31] 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.

[32] 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.

[33] 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.

[34] 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.