This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

Authenticated Outlier Mining for Outsourced Databases

Boxiang Dong, Hui (Wendy) Wang, Anna Monreale, Dino Pedreschi,
Fosca Giannotti, Wenge Guo

Abstract—The Data-Mining-as-a-Service (DMaS) paradigm is becoming the focus of research, as it allows the data owner (client) who lacks expertise and/or computational resources to outsource their data and mining needs to a third-party service provider (server). Outsourcing, however, raises some issues about *result integrity*: how could the client verify the mining results returned by the server are both sound and complete? In this paper, we focus on outlier mining, an important mining task. Previous verification techniques use an authenticated data structure (ADS) for correctness authentication, which may incur much space and communication cost. In this paper, we propose a novel solution that returns a probabilistic result integrity guarantee with much cheaper verification cost. The key idea is to insert a set of artificial records (ARs) into the dataset, from which it constructs a set of artificial outliers (AOs) and artificial non-outliers (ANOs). The AOs and ANOs are used by the client to detect any incomplete and/or incorrect mining results with a probabilistic guarantee. The main challenge that we address is how to construct ARs so that they do not change the (non-)outlierness of original records, while guaranteeing that the client can identify ANOs and AOs without executing mining. Furthermore, we build a strategic game and show that a Nash equilibrium exists only when the server returns correct outliers. Our implementation and experiments demonstrate that our verification solution is efficient and lightweight.

Index Terms—authentication, outsourcing, outlier mining, probabilistic guarantees, game theory.

1 Introduction

The ability to generate and collect massive amounts of data has grown exponentially in the past years. This rises new challenges in data analytics to extract valuable insights. Fortunately, the advent in networking technologies make it possible to transmit large volume of data through the Internet. It allows data owners, especially those who have large volume of data but limited budget for data analysis, to outsource their data and data mining needs to a third-party service provider. This is referred as the *Data-Mining-as-a-Service* ($DMaS$) model [42], [49], which provides a cost-effective computing infrastructure for those users who have limited resources for sophisticate data analytics. The model allows the data owner to leverage hardware and software solutions provided by $DMaS$ providers, without developing their own.

In this paper, we focus on *outlier mining*, which is to find data objects that do not comply with the general patterns of the majority. Outlier detection plays a critical role in many real-world applications such as computer system intrusion detection, credit card fraud detection and industrial damage detection. The problem of outlier detection has been widely studied in the data mining community [2], [6], [21], [43]. Previous work [2], [21] show that outlier detection is of high computational complexity; it can be prohibitive if the data has high dimensionality. Although the researchers have proposed several optimization approaches [4], [43] to improve the efficiency of outlier detection, it is difficult for the data owner who lacks the expertise to exploit these techniques. Outsourcing outlier mining to a *DMaS* service provider becomes a natural solution.

Although the $DMaS$ paradigm is beneficial for the data owner (client) with limited capabilities to perform sophisticated analysis on their large volume of data, it brings several security challenges. One major concern is the *integrity* of the mining results returned by the service provider (server). Given the fact that the server is potentially untrusted, it is necessary for the client to authenticate if the outliers returned by the server are correct. There are many reasons for the server to return wrong results. For example, the server may return wrong mining results accidentally due to software bugs; it may keep part of the mining results to itself intentionally so that it can sell the retained results to the competitors of the client for profit. There also exists a strong financial incentive for the server to reduce the computational cost. For example, the server may execute the outlier mining on a portion of the outsourced dataset, and charge the client for mining of the whole dataset. The primary challenge in authenticating the outsourced outlier mining arises from the weak computational resources at the client side.

Following [3], we consider three types of the servers that return incorrect mining results: (1) the *Class I* server that is

- *B. Dong is with the Department of Computer Science, Montclair State University, Montclair, NJ, USA. Email: dongb@montclair.edu*
- *H.W. Wang is with the Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ, USA. Email: Hui.Wang@stevens.edu*
- *A. Monreale and D. Pedreschi are with the Computer Science Department, University of Pisa, Pisa PI, Italy . Email: annam@di.unipi.it, pedre@di.unipi.it*
- *F. Giannotti is with ISTI-CNR, Pisa, Italy. Email: fosca.giannotti@isti.cnr.it*
- *W. Guo is with the Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, NJ, USA. Email: wenge.guo@njit.edu*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

not aware of the authentication process by the client; (2) the *Class II* server that is aware of the fact that the client may perform authentication on mining result, but does not have details of the authentication process; and (3) the *Class III* server that knows the details of the authentication process and tries to avoid being caught by verification. The goal of our verification method is to check if the outliers returned by any of these three types of servers are both *sound* and *complete*. By *soundness*, we mean that each returned tuple is a true outlier. By *completeness*, we mean that all true outliers must be returned by the server.

In consistence with the other work [11], [50], we do not require a *trusted* third-party that performs the authentication for the client. A seemly straightforward solution is that for each returned record, the client checks if it is indeed an outlier against the original dataset. Though simple, this approach requires computations of $O(nk)$ complexity, where n is the size of the dataset and k is the number of outliers returned by the server. Due to the large volume of data, this naive approach may not be feasible, especially if the client uses a resource-constrained device (e.g., a smart phone) that has weak computational power for authentication. Moreover, this solution cannot verify the *completeness* of the outliers.

In theory, the techniques that verify general-purpose computation [5], [19] can be applied to authenticate any outsourced computation task. These verification techniques require a preprocessing phase in which the client generates auxiliary information of the outsourced computation. Then the server constructs interactive proofs or probabilistically checkable proofs (PCPs) to demonstrate the correctness of the returned result. Unfortunately, this body of theory is considered to be impractical, due to the complexity of the preprocessing step and the expensive cost of using general-purpose cryptographic proofs for data mining problems. A large body of the existing authentication techniques use an authenticated data structure (ADS) (e.g., [24], [52]). The key idea is that the client constructs an ADS from her dataset, and sends both the dataset and ADS to the server. During query processing, the server traverses the ADS and constructs a verification object (VO) that proves the correctness of the results. The VO is sent back to the client together with the query results for verification. Though effective, ADS-based authentication may incur significant space and communication cost.

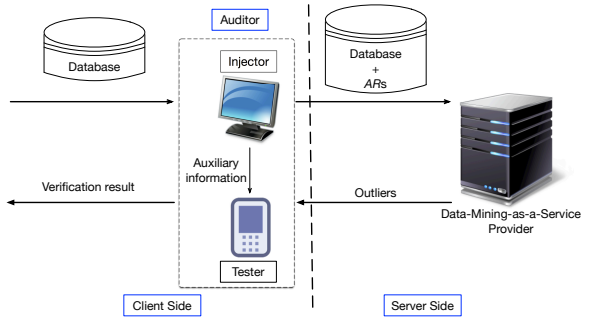


Fig. 1: The authentication architecture

We propose an efficient *probabilistic* authentication framework for outsourced outlier mining. The architecture of our approach is illustrated in Figure 1. There are two entities, the data owner (client) and the service provider (server). To verify the correctness of the outliers returned by the server, the client is equipped with an *auditor* component which can be viewed as a "black box" software. In particular, the auditor component consists of two modules, namely the *injector* module and the *tester* module. The injector module, which can be executed on a computer desktop, generates a set of artificial records (ARs) offline. The ARs are used to construct the *artificial outliers* (AOs) and *artificial non-outliers* (ANOs). The ARs are inserted into the original database and sent to the server. Meanwhile, the injector module produces a small piece of *auxiliary information* (including a hash function, three constants, and the number of AOs and ANOs). The auxiliary information is maintained at the *tester* side for verification purpose. After outsourcing the dataset and the outlier detection task, the client receives and verifies the outliers returned by the server by executing the *tester* module. In particular, the tester module analyzes the returned outliers against the AOs and ANOs, and quantifies the probabilistic guarantee of the result correctness. We show that incorrect answers from the server can be caught with high confidence by utilizing a small number of AOs and ANOs, even for large datasets. This makes it feasible to run verification on the resource-constrained devices such as smart phones. We make the following contributions:

(1) We design an efficient authentication approach that generates AOs and ANOs without any need to do outlier mining of the original dataset. One major challenge is that inserting any artificial record into the dataset may change the (non-)outlierness of the original records. We design a novel AOs/ANOs construction algorithm that does not eliminate any true outlier. We also discuss how to remove the false positive outliers (i.e., the original non-outliers that become outliers in the dataset with AOs/ANOs) introduced by AOs/ANOs and recover the true outliers efficiently. Formal analysis shows that a small number of AOs/ANOs can verify the result returned by the *Class I* server with high probabilistic guarantee.

(2) To incentivize the *Class II* server to behave honestly, we design a game theoretic approach to decide the appropriate parameter values for the authentication setting, so that a rationale server should always return correct mining results to obtain the most payoff.

(3) We discuss the *authentication-aware* cheating behaviors by the *Class III* server. As a countermeasure, we propose two approaches to catch the *Class III* server.

(4) We run an extensive set of experiments on two datasets to demonstrate the efficiency and the effectiveness of our authentication approach. The experimental results show that the AO and ANO construction takes at most 1 second on the client side. Furthermore, the mining overhead at the server side is no more than 1.2%.

An earlier and short version of this paper was published in the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) [26]. The short version only considers the *Class I* server. In this paper, we extend the short version significantly with: (1) a game theoretic approach that analyzes the *Class II* server's behaviors and incentivizes the server to return correct mining results; and (2) the security analysis of our verification approach against the *Class III* server.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

The paper is organized as following. Section 2 introduces the preliminaries. Section 3 presents our AR-based approach to construct AOs and ANOs to catch the *Class I* server. Section 4 discusses the strategic game to incentivize the *Class II* server to do mining and return correct results honestly. Section 5 analyzes the security against the cheating behaviors by the *Class III* server trying to escape from the AR-based authentication, assuming that it possesses the details of the authentication mechanism. Section 6 presents our experimental results. Section 7 discusses related work. Section 8 concludes the paper.

2 Preliminaries

2.1 Distance-based Outlier Mining

A variety of definitions of outliers, including distance-based outliers [21] and density-based outliers [8], have been proposed recently. In this paper, we focus on distance-based outliers, due to the properties that it works well for datasets of any dimension, and it does not assume that the data follows a standard distribution [21]. Specifically, an object t in the dataset D is a (p, d)-*outlier* if it is at least of distance d away from at least p proportion of all objects in D. In the rest of this paper, we use *outlier* and (p, d)-*outlier* interchangeably. If an object does not satisfy the condition, we call it a *non-outlier*. We measure the distance between two objects based the *Euclidean distance*. In specific, given two records $t(a_1, \ldots, a_k)$ and $t'(a'_1, \ldots, a'_k)$, $dist(t, t') = \sqrt{\sum_{i=1}^{k}(a_i - a'_i)^2}$.

2.2 Outsourcing Setting

We consider the outsourcing setting that consists of two parties, the *data owner (client)* who possesses a dataset D and aims to find outliers in D, and the *service provider (server)* that provides distance-based outlier mining as the service. When the client outsources the dataset to the server, she communicates with the server regarding the parameter setting of p and d values for (p, d)-outlier mining. She can either choose to re-configure the p and d parameters by herself or follow the parameter settings by the server. The server executes outlier mining according to the parameter setting. We assume the server finds exact outliers instead of approximate ones [22], [37].

2.3 Adversary Model

In this paper, we adopt the taxonomy of [3] of the attackers to consider three types of servers that possess different knowledge of the authentication approach.

- The *Class I (ordinary outsider)* server that is not aware of the authentication method, but is curious to find out if there is any authentication mechanism.
- The *Class II (intelligent outsider)* server that is aware of the fact that the client may perform result correctness authentication. However, it does not have the details of the authentication approach.
- The *Class III (knowledgeable insider)* server that knows the details of the authentication approach and may try to escape the verification by returning well-designed incorrect answer intentionally.

2.4 Authentication Goal

Let O be the set of outliers in the outsourced dataset D, and O^S be the set of outliers returned by the server. We define the *precision* of O as $P = \frac{|O \cap O^S|}{|O^S|}$ and the *recall* as $R = \frac{|O \cap O^S|}{|O|}$. Intuitively, precision measures the fraction of returned outliers that are correct, while recall quantifies the fraction of correct outliers that are returned. The aim of our authentication approach is to catch any wrong answer, i.e., $P < 1$ or $R < 1$, with a high probability. Next, we formally define the authentication goal as (α, a)-*completeness* and (β, b)-*soundness*.

Definition 2.1. Given a dataset D and a verification method M, let pr_p and pr_r be the probabilities that M catches the server that returns the result of precision $P \leq b$ and recall $R \leq a$ respectively, where $a, b \in [0, 1]$ are given thresholds. We say M can verify (α, a)-*completeness* if $pr_r \geq \alpha$, and can verify (β, b)-*soundness* if $pr_p \geq \beta$, where $\alpha, \beta \in [0, 1]$ are also given threshold values.

If the client catches the server's cheating behavior, then the client is 100% sure that O^S is either unsound or incomplete. Otherwise, the client believes that O^S satisfies that: (1) $R > a$ with probability α; and (2) $P > b$ with probability β.

2.5 Game Theory

A strategic game is a model of interacting decision-makers. The game involves a set of players. Each player has a set of possible actions. For each player, the preferences/payoffs over the set of action profiles are pre-defined. Each player chooses his/her action once and for all, and the players choose their actions "simultaneously" without any player being informed. A *Nash Equilibrium* is an action profile A^* with the property that no player can get more payoff by choosing an action different from her action in A^*, given that every other player adheres to A^* [33]. Intuitively, a Nash Equilibrium corresponds to a *steady state*. If, whenever the game is played, the action profile is the same as the Nash Equilibrium A^*, then no player has a reason to choose any action different from his/her component of A^*.

3 AR-based Approach to Catch Class I Server

The key idea of our authentication framework is that the client constructs a set of artificial records (ARs) before outsourcing. Let ΔD be the artificial records. Given the original dataset D, each record in ΔD is either an *artificial* outlier (AO) or an artificial non-outlier (ANO). The client combines D with ΔD to get D^+ and sends D^+ to the server. If the server conducts the outlier mining faithfully, it should return all AOs but no ANO.

In this section, we focus on the *Class I* server that is not aware of the authentication procedure and thus cannot distinguish ARs from the original records. Therefore, the client is able to obtain a probabilistic guarantee of the soundness and completeness of the returned outliers by comparing them with the AOs and ANOs. Next, we first discuss how to construct ANOs and AOs in Section 3.1 and 3.2. After that, we prove that the AR-based approach preserves the (non)outliers in the original dataset in Section 3.3. Section 3.4 and 3.5 discuss the auxiliary data and the authentication procedures at the client side.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

3.1 Construction of Artificial Non-outliers (ANOs)

Before we introduce the ANO construction procedures, we define *close records* and *distant records* first.

Definition 3.1. Given a dataset D and a record t, the close records of t with regard to d are $T_L(t,d) = \{t'|t' \in D,\ dist(t,t') < d\}$, while the distant records are $T_U(t,d) = \{t'|t' \in D,\ dist(t,t') > d\}$.

Intuitively, $T_L(t,d)$ is the set of records in D whose distance to t is smaller than d, while $T_U(t,d)$ stores the set of records that are at least d distance away from t. Next, we define the *farthest close neighbor* and *closest distant neighbor* respective.

Definition 3.2. Given a dataset D, a record t and a distance threshold d, a record $t' \in T_L(t,d)$ is the farthest close neighbor of record t, if the distance between t and t' is the largest among all records in $T_L(t,d)$. Similarly, a record $t' \in T_U(t,d)$ is the *closest distant neighbor* of record t, if the distance between t and t' is the smallest among all records in $T_U(t,d)$.

Definition 3.3. Given the dataset D of r dimensions, a record $t \in D$ and the distance threshold d, let $t_a \in D$ be the farthest close neighbor of t, and $t_b \in D$ be the closest distant neighbor of t. Let $d_a = dist(t,t_a)$, and $d_b = dist(t,t_b)$. Let Q be an r-sphere with t as the centroid, and $min(\frac{d-d_a}{2}, \frac{d_b-d}{2})$ as the radius, where d is the distance parameter of (p,d)-outlier mining. Then we say any record $t' \in D$ is a *close* record to t if it falls in Q.

Next, we show that the close records of t have the same distance property as t.

Lemma 3.1. Given a record t and a close record t_c of t, for any record $t' \neq t$, it must be true that: (1) if $dist(t,t') < d$, then $dist(t_c,t') < d$; and (2) if $dist(t,t') > d$, then $dist(t_c,t') > d$.

Proof: Since t_a is the farthest close neighbor of t, for any t' s.t. $dist(t,t') < d$, we have $dist(t,t') \leq d_a < d$, and $dist(t,t_c) < \frac{d-d_a}{2}$, leading to $dist(t',t_c) \leq dist(t,t') + dist(t',t_c) < d_a + \frac{d-d_a}{2} = \frac{d+d_a}{2}$. Since $d_a < d$, then it must be true that $dist(t_c,t') < d$. Similarly, we have $dist(t,t') \geq d_b > d$, and $dist(t,t_c) < \frac{d_b-d}{2}$. Thus, $dist(t_c,t') \geq dist(t,t') - dist(t,t_c) > d_b - \frac{d_b-d}{2} = \frac{d_b+d}{2}$. Since $d_b > d$, then it must be true that $dist(t_c,t') > d$. ∎

Note that Lemma 3.1 holds no matter t' is an outlier or non-outlier in the dataset D. Based on Lemma 3.1, we use Theorem 3.1 to prove that any record close to record t always has the same non-outlierness as t.

Theorem 3.1. Given a dataset D and any record $t \in D$ that is a non-(p,d)-outlier, any close record of t must be a non-(p,d)-outlier in D.

Proof: Let t_c be a close record of a true record t in D. It is straightforward from Lemma 3.1 that t_c and t have the same number of records whose distance is greater than d in D^+. If t is a (p',d)-outlier in D^+, t_c must be a (p',d)-outlier in D^+. On the other hand, if t is a (p',d)-non-outlier in D^+, t_c must be a (p',d)-non-outlier in D^+. ∎

Theorem 3.1 provides us the guidance to construct ANOs. In particular, we first pick a *seed* record t_{seed} that is a non-outlier record in the original dataset D. Then we construct a set of artificial records as the close records of t_{seed}, and take these artificial records as ANOs. Fig. 2 (a) illustrates the construction procedure of ANO in a 2-dimensional dataset.

To pick t_{seed}, we repeatedly pick a record from D randomly, and check its non-outlierness, until a non-outlier record is selected. The probability that t_{seed} will be picked at the x-th trial is

$$g(x) = (1 - \frac{f_{to}}{n})(\frac{f_{to}}{n})^{x-1},$$

where f_{to} is is the number of outliers, and n is the number of records in D respectively. It is straightforward that $1 \leq x \leq n - f_{to}$. We define $\phi = \frac{f_{to}}{n}$. Then $g(x) = (1-\phi)\phi^{x-1}$, where $1 \leq x \leq n - n\phi$. The expected value of x equals to

$$E(x) = \frac{(n-\phi n)\phi^{n-\phi n+1} - (n-\phi n+1)\phi^{n-\phi n} + 1}{(\phi-1)^2}.$$

As usually the outliers are only a small portion of the whole dataset, ϕ is a small number (e.g., $\phi = 0.05\%$ [35]). Therefore, $E(x) \approx 1$. In other words, it is highly likely that t_{seed} can be picked by the first random trial.

After t_{seed} is identified, we can construct ANOs by constructing the r-sphere Q (defined in Definition 3.3) of t_{seed}, and picking any record in Q as a ANO. The computation of t_{seed} needs to traverse D once. It requires a scan of the dataset D to identify t_{seed}, and takes another round to determine the radius of the r-sphere Q (defined in Definition 3.3). Therefore, we can construct ANOs by two passes of D.

3.2 Construction of Artificial Outliers (AO)

A seemly straightforward way to construct AOs is similar to the construction of ANOs: we first find a seed outlier from the dataset D and then construct artificial records that are *close* to the seed as AOs. However, this approach may be prohibitively expensive, since the outliers are rare and finding an outlier may require mining of the original dataset. Our goal is to construct AOs without mining the original dataset to find any outlier. In the following, we discuss the details of how to construct AOs efficiently.

Our construction procedure relies on the definition of *distant* records.

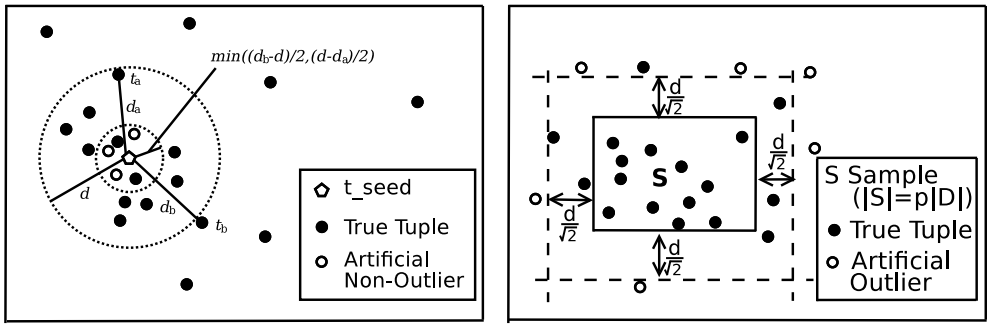
Definition 3.4. Given a r-dimension dataset D and a set of records $S \subseteq D$, we say a record $t \notin S$ is a *distant* record of S if for each record $t' \in S$, $dist(t,t') > d$, where d is the parameter setting of (p,d)-outlier mining.

We have the following lemma to show a important property of distant records.

Lemma 3.2. Given a r-dimension dataset D and a set of records $S \subseteq D$, let min_i and max_i be the minimum and maximum value of the i-th ($1 \leq i \leq r$) attribute of the records in S. Then any record $t \notin S$ is a *distant* record of S if there are k ($1 \leq k \leq r$) attributes such that on each attribute $A_i (1 \leq i \leq k)$, $t[A_i] < (min_i - \frac{d}{\sqrt{k}})$ or $t[A_i] > (max_i + \frac{d}{\sqrt{k}})$, where $t[A_i]$ is the i-th attribute value of t.

The correctness of Lemma 3.2 is straightforward. Next, we use Theorem 3.2 to show that (p,d)-outliers can be generated from the distant records.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing



(a) ANO construction (b) AO construction

Fig. 2: Construction of ANOs and AOs

Theorem 3.2. Given the dataset D and a set of records $S \subseteq D$, any distant record t of S must be a (p, d)-outlier if $|S| \geq p|D|$.

The correctness of Theorem 3.2 is straightforward. For any distant record t, there must exist a set of records $S \subseteq D$, whose size is greater than p fraction of the size of D, such that the distance between t and any record $t' \in S$ must be greater than d. Therefore, t must be a (p, d)-outlier. Based on Theorem 3.2, we design the AO construction procedure as follows.

First, the client picks a sample S of size $\lceil p|D| \rceil$ records randomly from D. Second, the client treats S as an r-dimension hypercube $\mathcal{R}$. The range $[min_i, max_i]$ represents the edge at the i-th dimension of $\mathcal{R}$. Then the client randomly picks $k \leq r$ dimensions (possibly $k = 1$) of $\mathcal{R}$. Last, the client expands the picked k dimensions of $\mathcal{R}$ by $\frac{d}{\sqrt{k}}$ (i.e., change the minimum and maximum value of the i-th attribute to be $min_i - \frac{d}{\sqrt{k}}$ and $max_i + \frac{d}{\sqrt{k}}$). Let the expanded hypercube be $\mathcal{R}'$. Then any record t_{ao} that is created outside of $\mathcal{R}'$ must be a (p, d)-outlier of D. Fig. 2 (b) illustrates the construction procedure in a 2-dimensional dataset. How to decide f_{ao} will be discussed in Section 3.5.1. The complexity of AO construction is $O(n)$, where n is the size of D.

3.3 Preservation of (non)outlierness

One issue of inserting ARs into D is that (p, d)-outliers in the original dataset D may not be (p, d)-outliers in $D^+ = D \cup \Delta D$ anymore, as inserting records into D may change the (non)outlierness of some original records in D. This may ruin the authentication method as even the server executes the outlier mining and returns the result faithfully, it will be wrongly caught by the AR-based verification approach. Therefore, the client must make sure that all (p, d)-outliers in D are also outliers in D^+. In this section, we discuss the followings:

- How to ensure that the true (p, d)-outliers in D are still recognized as outliers in D^+; and
- How to eliminate the *false positive* outliers (i.e., the records that are not (p, d)-outliers in D but become outliers in D^+).

Preservation of true AOs. First, we show how to make sure that the (p, d)-outliers in D are still outliers in D^+ by only using a different p value in (p, d)-outlier mining.

Theorem 3.3. Given a dataset D and a set of AOs and ANOs constructed from the set of artificial records ARs, any (p, d)-outlier in D must be a (p_1, d)-outlier in $D^+ = D \cup \Delta D$, where

$$p_1 = \frac{p|D|}{|D^+|}. \quad (1)$$

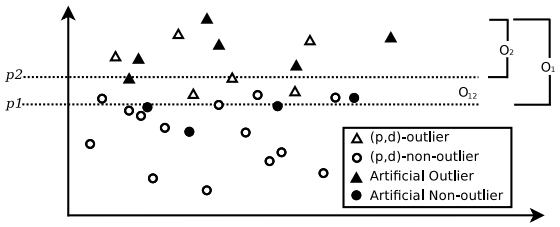


Fig. 3: (p_1, d)-outliers and (p_2, d)-outliers in D^+ VS. (p, d)-outlier in D; O_1: (p_1, d)-outliers in D^+, O_2: (p_2, d)-outliers in D^+, O_{12}: $O_1 - O_2$.

Proof: For a true record $t \in D$, let x and y be the number of original and artificial records (including both AOs and ANOs) whose distance to t is greater than d in D^+. Now we prove that it must be true that $\frac{x+y}{|D^+|} \geq p_1 = \frac{p|D|}{|D^+|}$. This can be proven by the following. Since record t is a (p, d)-outlier in D, it must be true that $x \geq p|D|$. This naturally leads to that $\frac{x+y}{|D^+|} \geq \frac{p|D|}{|D^+|}$, with $x \geq 0$. ∎

Following Theorem 3.3, instead of requesting for (p, d)-outliers in the outsourced dataset D^+, the client asks for (p_1, d)-outliers in D^+, where p_1 is defined in Formula (1). All (p, d)-outliers in D must appear in the answer if the server is honest. Note that all AOs must be appear in (p_1, d)-outliers of D^+ too.

Elimination of false positive outliers. It is not necessary that all (p_1, d)-outliers in D^+ must be (p, d)-outliers in D. In the words, by asking for (p_1, d)-outliers on D^+, the server may return some *false positive* outliers. To eliminate those false positives, we have:

Theorem 3.4. Given a dataset D, let ΔD be the set of ARs that are used to construct AOs and ANOs. Then any (p_2, d)-outlier in $D^+ = D \cup \Delta D$ must be a (p, d)-outlier in D, where

$$p_2 = \frac{p|D| + |\Delta D|}{|D^+|}. \quad (2)$$

Proof: For a record $t \in D^+$, let x and y be the number of original and artificial records (including both AOs and ANOs) whose distance to t is greater than d in D^+. Since the x true records must exist in D, we aim to prove that $\frac{x}{|D|} \geq p$. This can be proven as follows. Since t is a (p_2, d)-outlier in D^+, it must be true that $\frac{x+y}{|D^+|} \geq p_2$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

In other words, $x + y \geq p|D| + |\Delta D|$. Since $y \leq |\Delta D|$, it must be true that $x \geq p|D|$. Therefore, the theorem holds. ∎

Following Theorem 3.4, all the constructed ANOs must be (p_2, d)-non-outliers in D^+.

Fig. 3 illustrates the relationship among (p_1, d)- and (p_2, d)- outliers (p_1 and p_2 are defined by Eqn. 1 and 2 respectively) in D^+ and (p, d)-outliers in D. For a given record $t \in D$, let ρ be the fraction of records in $D^+ = D \cup \Delta D$ whose distance to t is greater than d, where d is the parameter for (p, d)-outlier mining. Then t must fall into one of the following three categories:

- t is a (p, d)-outlier in D, if $\rho \geq p_2 = \frac{p|D|+|\Delta D|}{|D^+|}$;
- t is a (p, d)-non-outlier in D, if $\rho < p_1 = \frac{p|D|}{|D^+|}$;
- t is either a (p, d)-outlier or a (p, d)-non-outlier in D, otherwise.

Based on both Theorem 3.3 and Theorem 3.4, the outsourcing and the authentication procedures are designed as the following. The client constructs $D^+ = D \cup \Delta D$ and outsources D^+ to the server. She sends two mining requests to the server for (p_1, d)-outliers and (p_2, d)-outliers, where $p_1 = \frac{p|D|}{|D^+|}$, and $p_2 = \frac{p|D|+\Delta D}{|D^+|}$. Let O_1 and O_2 be the set of outliers received from the two requests separately. It is worth noting that $O_2 \subseteq O_1$. Therefore, the client can get both (p_1, d)-outliers and (p_2, d)-outliers by outsourcing the task only once.

3.4 Auxiliary Data for Authentication

Auxiliary data sent to the server. Before the client sends out her dataset, she generates two digests, namely dig_a and dig_b, for each record $t \in D^+$ with a cryptographic hash function. Let s be the secret key of the client, and H denote the signing function of a message authentication code (MAC), e.g., the Hash-based message authentication code ($HMAC$). Given a record $t(a_1, \ldots, a_r)$, we have $dig_a = H(s, t)$, and

$$dig_b = \begin{cases} H(c_1, dig_a), & \text{If } t \text{ is a true record in } D; \\ H(c_2, dig_a), & \text{If } t \text{ is an } AO; \\ H(c_3, dig_a), & \text{If } t \text{ is an } ANO, \end{cases}$$

where c_1, c_2 and c_3 are three unique constant values kept secret at the client side.

Intuitively, dig_a helps the client to check if the server has modified any record in D^+, while dig_b can be used to distinguish the original records from the AOs and ANOs.

After completing the computation of digests, the client sends D^+ to the server. Each record is associated with its two digests dig_a and dig_b. When the server returns any outlier to the client, it is required to return the two digests too.

Auxiliary data maintained at the client side. The client keeps the private key s locally, and maintains the number of AOs and ANOs, and the three constants c_1, c_2, and c_3 that are used to construct the digests. It is straightforward that the space overhead of these auxiliary information is negligible. For each outlier record t returned by the server, the client computes its digest dig_a, and the three signatures $dig_b^1 = H(dig_a * c_1)$, $dig_b^2 = H(dig_a * c_2)$, and $dig_b^3 = H(dig_a * c_3)$, by using the three constants c_1, c_2, and c_3 that it has stored locally. Then by comparing the digests dig_b^1, dig_b^2, dig_b^3 against the dig_b that is attached to t, the client can distinguish whether t is a original record in D, an AO or an ANO.

3.5 Authentication and Post-processing at Client Side

After receiving O_1 and O_2 from the server, the client checks the integrity and post-processes the result as the following.

3.5.1 Authentication at Client Side

The client runs the 2-phase verification procedure to check the soundness and completeness of the result.

Phase-1. For each record $t \in O_1$ or $t \in O_2$, the client re-computes the digest dig_a by applying the stored hash function H (Section 3.4) on t. If the computed digest does match the one that is associated with t, the client concludes that the server has returned records that do not exist in the outsourced database, and thus fails to pass the verification.

Phase-2. If the server passes the phase-1 verification, the client further computes three digests: $dig_b^1 = H(dig_a * c_1)$, $dig_b^2 = H(dig_a * c_2)$, and $dig_b^3 = H(dig_a * c_3)$, by using the three constants c_1, c_2 and c_3 that the client stores locally. By comparing these digests with the digest dig_b associated with t, the client can identify whether the returned tuple is a true record, an AO, or an ANO. Then the client verifies the completeness and soundness respectively.

Completeness authentication. To verify whether the server has returned all true outliers, the client checks whether every AO is included in O_1. If not, the client catches the incomplete outlier answer with 100%; otherwise, the client trusts that the the recall R of the result is at least α with probability $pr_r = 1 - \alpha^{f_{ao}}$, where f_{ao} is the number of artificial outliers. This is because for each AO in D^+, the probability that the server includes it in O_1 is α. The probability that the server avoids being caught by the client, i.e., the probability that the server returns all AOs in O_1, is $\alpha^{f_{ao}}$. Therefore, to satisfy (α, a)-completeness (i.e., $pr_r \geq a$), it is required that:

$$f_{ao} = \lceil log_\alpha(1 - a) \rceil. \tag{3}$$

Soundness authentication. For the soundness authentication, the client checks whether if any ANO appears in O_2. If it does, the client catches the incorrect answer with 100%; otherwise, the client believes that the precision P of the result returned by the server is at least β with probability $pr_p = 1 - \beta^{f_{ano}}$, where f_{ano} is the number of artificial non-outliers. Similar to the reasoning in the completeness authentication, the probability that the server does not return any ANO in O_2 is $\beta^{f_{ano}}$. Thus to meet the (β, b)-soundness (i.e., $pr_p \geq b$) requirement, f_{ano} must satisfy that

$$f_{ano} = \lceil log_\beta(1 - b) \rceil. \tag{4}$$

Equation 3 and 4 show that f_{ao} and f_{ano} (the number of AOs and ANOs) are independent of the size of D. Therefore, our authentication framework would be especially efficient for large datasets. Furthermore, it does not need large number of AOs and ANOs to provide high soundness and completeness guarantee. For instance, when $\alpha = 0.99$ (i.e., the server misses 1% of the outliers) and $a = 0.99$ (i.e., the probability to catch such answer is at least 0.99), it only needs 459 AOs to verify $(0.99, 0.99)$-completeness.

Overhead Analysis. The complexity of soundness and completeness authentication is $O(f_{ao} + f_{ano})$. Our empirical

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

study shows that f_{ao} and f_{ano} are relatively small even for large datasets (Section 6). This enables the client to accomplish authentication on resource-constrained devices (e.g., smart phones).

3.5.2 Recovery of True (p, d)-Outliers at Client Side

Since the returned (p_1, d)-outliers O_1 and (p_2, d)-outliers O_2 may contain some false positive records that are not (p, d)-outliers in D, the client needs to recover the original (p, d)-outliers in D from O_1 and O_2. To accomplish this, first, the client excludes all AOs (if there is any) from O_2 (how to distinguish original records from AOs, and ANOs is discussed in Section 3.5.1). Let the remaining (p_2, d)-outliers be O_2'. Second, the client checks each record in $O_1 \setminus O_2$ against D, and keeps those that are (p, d)-outliers in D. Let these records be O_{12}. Then $O_{12} \cup O_2'$ are the set of original (p, d)-outliers in D. As will shown in the Experiment section (Section 6), the records in $O_1 \setminus O_2$ take a negligible portion of D (less than 0.2%). Therefore, the complexity of outlier recovery at the client side is $O(|D|)$.

4 A Game Theoretic Approach To Incentivize Class II Server

While our AR-based authentication approach enables the client to catch the *Class I* server with high probability, a *Class II* server that is aware of the probabilistic authentication strategy may still make the most outcome by performing cheaper outlier mining computations and returning incorrect results. In this section, we aim at incentivizing the *Class II* server not to cheat on the outlier mining computations. A possible solution is to build a strategic game [48] between the client and server, in which a Nash equilibrium [36] exists only when the server's action is to return correct mining results. Based on the theory of rational choice, due to the Nash Equilibrium, a rational server will always play the game honestly, if he can not gain more payoff by switching to cheating. We emphasize that even though we focus on the outsourced outlier mining, our incentive game model can be applied to other types of outsourced computations.

There are two players, i.e., the server and the client, in this game. Each player has the following actions.

- **Client**
  - **Verify:** The client uses the AR-based approach to authenticate the returned outlier mining results.
  - **Trust-without-verification:** The client simply accepts the results returned by the server. With a certain level of belief, the client trusts the result to be correct. The belief can be obtained based on the server's reputation.

  If the client accepts the server's mining results (either after verification or without verification), the client pays the server for its mining efforts. The payment is considered as the server's payoff.
- **Server**
  - **Cheat:** The server returns wrong outliers to the client. If the cheating behavior is caught by the client, the server receives no payment from the client. Instead, it pays the penalty for the cheating.
  - **Honest play:** The server executes mining faithfully and returns the correct outliers to the client.

Notation	Meaning
c_v	Client's cost for AR-based authentication
c_m	Server's cost for faithful outlier mining
p_c	Server's penalty if the cheating is caught
p_m	Server's payment for mining
u_m	Client's benefit gained from correct outlier mining result
$\epsilon \in [0, 1]$	Probability of catching server by AR-based authentication
$\eta \in [0, 1]$	Result correctness belief for trust-without-verification
$\lambda \in [0, 1]$	Result correctness belief for pass of verification
$\sigma \in [0, 1]$	Fraction of correct outliers that are returned by the server

TABLE 1: Notations

For the following discussions, we use the notations defined in Table 1.

Next, we present the payoff matrix (Table 2) for the client and server with various actions. We assume c_v, c_m, and p_c are in the monetary format. We also assume that the client's benefit gained from correct outlier mining result (i.e., u_m) can be measured in the same format as c_v, c_m, and p_c. Regarding the results that were considered as correct with probability η (λ, resp.), the client's benefit gained from the result is measured as ηu_m (λu_m, resp.). We use $(,)$ to denote the payoffs, where the left (right, resp.) value is the client's (server's resp.) total payoff by the specific actions. We explain the details of various payoffs as below.

- **(Verify, Not Cheat)**: When the server does not cheat, there is no need that the client catches the server (with N/A in the corresponding *Catch* row). Thus the client trusts the returned outlier results with λ confidence and pays the full price for the server's mining efforts. The client's payoff is $\lambda u_m - c_v - p_m$, and the server's payoff is $p_m - c_m$.
- **(Verify, Cheat)**: Due to the probabilistic property of the AR-based authentication approach, the client may not catch the server's cheating behaviors. Assume that the server returns σ ($0 < \sigma < 1$) fraction of the outliers, the client can still obtain σu_m utility, while the server's mining cost is σc_m. If the client catches the server (i.e., for the *Catch* case), the server needs to pay the penalty to the client. So the client's payoff is $\sigma u_m + p_c - c_v$, and the server's payoff is $-\sigma c_m - p_c$. If the client does not detect the cheating behavior (i.e., for the *Not Catch* case), the client still needs to pay the server for the mining. So the client's payoff is $\sigma u_m - c_v - p_m$, and the server's payoff is $p_m - \sigma c_m$.
- **(Not Verify, Not Cheat)**: The client trusts the result correctness with probability η. Thus the client's payoff is $\eta u_m - p_m$, and the server's is $p_m - c_m$.
- **(Not Verify, Cheat)**: As the client does not verify the result, the server receives the payment from the client, but the client does not benefit from the mining result. Thus, the client's payoff is $\sigma u_m - p_m$, and the server's is $p_m - \sigma c_m$.

Ideally, we want the action profile (*, Not Cheat) to be the steady state (Nash Equilibrium), where $*$ stands for any action for the client. This is because any Nash Equilibrium (*, Not Cheat) incentivizes the server to discover the outliers honestly. However, it is impossible to reach an equilibrium for the (Not Verify, Not Cheat) case, as the server can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

		Not Cheat	Cheat
Verify	Catch	N/A	$(\sigma u_m + p_c - c_v, -\sigma c_m - p_c)$
	Not Catch	$(\lambda u_m - c_v - p_m, p_m - c_m)$	$(\sigma u_m - c_v - p_m, p_m - \sigma c_m)$
Not Verify		$(\eta u_m - p_m, p_m - c_m)$	$(\sigma u_m - p_m, p_m - \sigma c_m)$

TABLE 2: Payoff Matrix for the Client and Server

easily get more payoff by switching to Cheat. Thus, we can only get the desired equilibrium from (Verify, Not Cheat). According to the definition of Nash Equilibrium, an action profile is an equilibrium only if any player cannot gain more payoff by changing the action. Thus we require that:

$$P^{Server}_{(Verify,NotCheat)} \geq P^{Server}_{(Verify,Cheat)}, \tag{5}$$

and

$$P^{Client}_{(Verify,NotCheat)} \geq P^{Client}_{(NotVerify,NotCheat)}. \tag{6}$$

Based on the payoffs in Table 2, by doing the integral over σ, we have

$$\begin{aligned} P^{Server}_{(Verify,Cheat)} &= \int_0^{\lambda} \epsilon(-\sigma c_m - p_c)d\sigma \\ &+ \int_0^{\lambda} (1-\epsilon)(p_m - \sigma c_m)d\sigma \\ &+ \int_{\lambda}^{1} (p_m - \sigma c_m)d_{\sigma} \\ &= -\epsilon\lambda p_c - \epsilon\lambda p_m + p_m - \frac{1}{2}c_m, \end{aligned} \tag{7}$$

and

$$P^{Client}_{(NotVerify,NotCheat)} = \eta u_m - p_m. \tag{8}$$

The first part in Equation 7 measures the server's expected payoff when $\sigma < \lambda$ and its cheating is not detected by the client (the catching probability is ϵ). The second part evaluates the server's payoff when getting detected in cheating (probability is $1-\epsilon$). Whereas the last part considers that the returned result contains more than λ fraction of correct result and is not detected by the AR-based appraoch.

While

$$P^{Server}_{(Verify,NotCheat)} = p_m - c_m, \tag{9}$$

and

$$P^{Client}_{(Verify,NotCheat)} = \lambda p_m - c_v - c_m. \tag{10}$$

According to the requirement, the action profile $(Verify, NotCheat)$ is a Nash Equilibrium if:

$$\epsilon\lambda \geq \frac{c_m}{2(p_c + p_m)}, \tag{11}$$

and

$$\epsilon \geq \eta + \frac{c_v}{u_m}. \tag{12}$$

By combining Equation 11 and 12, we can get the required value of λ as:

$$\lambda \geq \frac{c_m u_m}{2(p_m + p_c)(\eta u_m + c_v)}. \tag{13}$$

According to our AR-based verification approach, ϵ corresponds to α and β in the (α, a)-completeness and (β, b)-correctness model (Def. 2.1), while λ confirms to a and b in the same model. Therefore, in order to incentivize the *Class II* server to behave honestly, when constructing the AOs and ANOs, the client should make sure that:

$$\alpha, \beta \geq \eta + \frac{c_v}{u_m}, \tag{14}$$

and

$$a, b \geq \frac{c_m u_m}{2(p_m + p_c)(\eta u_m + c_v)}. \tag{15}$$

5 Security Analysis against Class III Server

In this section, we focus on the *Class III* server that possesses the details of our AR-based authentication approach. We first discuss the security weakness of the AR-based approach. Then we present two robust verification approaches to catch the *Class III* server.

5.1 Weakness of AR-based Authentication

We consider the **authentication-aware cheating** behaviors by the server: When the server is aware of the details of the authentication mechanism, it can identify (at least the candidates of) AOs and ANOs. Even though the server cheats in the outlier mining, it can avoid getting detected by returning all AOs and excluding all ANOs from the returned result.

Unfortunately, our AR-based approach cannot catch the authentication-aware cheating. When the server knows the details about how ARs are constructed, it is able to identify them from D^+. In particular, it can find all AOs by two passes of D^+. Because the p value of the (p, d)-outlierness is always very close to 1 in practice, the sample S used to construct AOs (Section 3) includes almost all records in D. Therefore, the constructed AOs will be the skyline points of D. Based on this, the server takes the records that have the maximum/minimum values of any attribute as AOs. On the other hand, due to the fact that all ANOs must be the *close* records (Definition 3.3) to a non-outlier record, the server can identify them by searching for non-outlier records that are *close* to at least one non-outlier. This requires the server to find all non-outliers, whose complexity is at least as high as the cost of mining all outliers. Though expensive, after finding the AOs and ANOs, the server is able to escape from the soundness and completeness authentication.

5.2 Catching *Class III* Server

Sampling-based probabilistic approach. To catch the authentication-aware cheating, the client can randomly pick records from the original dataset as a *sample*, and verify the outlierness of the sample. To check the soundness and completeness of the result returned by the server, the client checks the outlierness of the samples against the result. Intuitively, a large sample is necessary in order to obtain a high soundness/completeness guarantee. This may not be affordable by the client with limited computational power. We conjecture that the complexity of catching the authentication-aware cheating is as expensive as outlier

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

mining itself, especially if the client requires a high result correctness probability.

Replication-based approach. The client can assign the outlier mining task to two servers, and verify the correctness by crosschecking the results from the two servers. To forbid collusion where the two servers return the same incorrect results, the client can create distrust between the servers by incentivizing them to betray their partner in the collusion coalition [16]. In specific, the client first takes deposit from both servers as the security for the delivery of correct answer. If any server is detected to return incorrect results, it loses the security deposit. Moreover, if one server is honest while the other is cheating, the cheating party's deposit will be taken by the honest one. Theoretical game theory analysis [16] shows that by setting appropriate deposit amounts, both rational servers will choose not to cheat to receive the maximum payoff. In this way, it is guaranteed that the *Class III* server returns correct outlier mining results.

6 Experimental Evaluation

We conducted an extensive set of experiments to evaluate both the robustness and performance of our AR-based authentication approach. In particular, we measured: (1) the soundness and completeness guarantee; (2) the verification overhead at the client side, which includes a) the construction time of AOs and ANOs, b) the verification time to based on AOs and ANOs, and c) the time of examining the outlierness of records to eliminate false positives; (3) the mining overhead at the server side.

Setup. In our experiment, we use two datasets, the $Letter$ dataset[1] from UCI MLC++ Library that contains $20k$ records, and the $KDDCUP$ dataset[2] that contains $100k$ records. The $Letter$ dataset has 16 numerical (integer) attributes. The $KDDCUP$ dataset contains 41 (numerical or categorical) attributes. In our experiments, we use five numerical attributes, including `duration`, `dst_bytes`, `number_access_files`, `count`, and `serror_rate` attributes, of the $KDDCUP$ dataset. For $Letter$ dataset, we used $p = 0.8$ and $d = 15$ that return 1% of the records as outliers, while for $KDDCUP$ dataset, we used $p = 0.99$ and $d = 5000$ that return 2% of the records as outliers. All of our experiments are evaluated on a PC with a 2.4GHz Intel Core 2 Duo CPU and $4GB$ RAM running Windows 7. We implemented the algorithm in Java.

ADS-based deterministic authentication approach. We implement the deterministic authentication approach that can verify the result correctness with 100% certainty. We design the deterministic approach by using the *Merkle B-tree* [24] as the *authenticated data structure (ADS)*. By using the Merkle B-tree, the server constructs a verification object (VO) and sends the VO to the client. From the VO, the client checks if the server has tampered with the outsourced dataset. After that, the client verifies the soundness and completeness by computing the distance between every pair of tuples.

6.1 Robustness of AR-based Approach

We simulate the incomplete result by removing 5%, 10%, 15%, 20%, and 25% outliers randomly (i.e., $a = 95\%$, 90%, 85%, 80%, and 75%), and generate the incorrect result as inserting 5%, 10%, 15%, 20%, and 25% non-outliers into the result (i.e., $b = 95\%$, 90%, 85%, 80%, and 75%). Then, we measure the probability of catching these incomplete and unsound results of our approach as the following: we repeat 500 trials on the $Letter$ dataset and 100 trials on the $KDDCUP$ dataset. In each trial, we construct a certain number of AOs and ANOs and compare the result against them.

For completeness verification, we check if all the AOs are included in the server's result. If at least one AO is missing, we take the trial as *a successful detection*. Similarly, for soundness verification, we check if the result contains any ANO, and take the trial as a detection if it does. After we finish all trials, we calculate the detection probability as the $pr_d = \frac{n_{det}}{n_{tr}}$, where n_{det} is the number of successful trials and n_{tr} is the total number of trials.

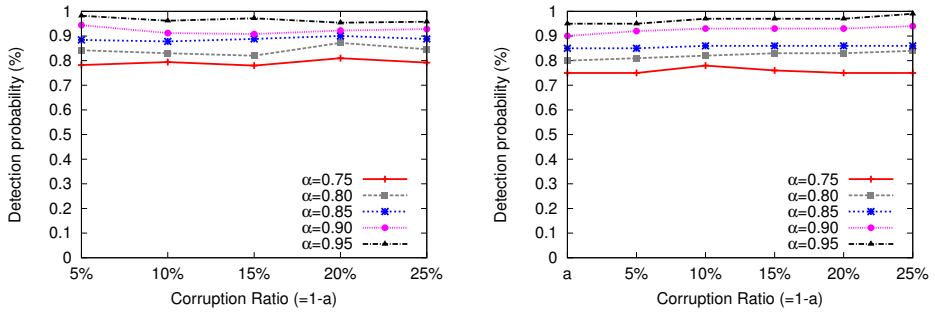
First, we measured the detection probability of incomplete answer. Figure 4 (a) shows the result on the $Letter$ dataset. We observe that the detection probability is always higher than the required α value (i.e., the probability threshold). We have the same observation on the $KDDCUP$ dataset (Figure 4 (b)). We also measured the detection probability of soundness violations and show the results in Figure 4 (c) & (d) for $Letter$ dataset and $KDDCUP$ dataset respectively. It can be seen that the detection probability of unsound result is always higher than the required β value, i.e., the soundness catching probability threshold. This proves the robustness of our AR-based approach for both completeness and soundness verification.

6.2 Cost Analysis at Client Side.

First, we measured the time performance of the AR-based approach on the *Letter* dataset. Figure 5 (a) shows the AO construction time. We observe that the AO construction is extremely fast, as it only takes 0.012 seconds even when $\alpha = 0.95$ and $a = 0.95$. Furthermore, when α and a values increase, AO construction time increases too, but only slightly. Figure 5 (b) shows the ANO construction time on $Letter$ dataset. It takes more time than AO construction since it needs to find the t_{seed}. Nevertheless, it is still very efficient; it only needs 0.022 second at most even when $\beta = 0.95$ and $b = 0.95$. Compared with the ADS-based deterministic authentication approach (around 0.18 second to construct the authenticated data structure for both completeness and soundness verification on the $Letter$ dataset), the construction time of AOs and $ANOs$ is negligible. We also measured the verification time at the client side. Figure 5 (c) shows the time of completeness verification on $Letter$ dataset. We observed that the time grows when α and a increase. This is straightforward as higher α and a require larger number of AOs. In comparison with the ADS-based deterministic approach, the AR-based authentication approach dramatically saves the verification cost at the client side. For example, the ADS-based deterministic approach demands 131 seconds to check the result completeness. Whereas, the AR-based approach only needs at most 0.008 second. Figure 5 (d) shows the time of soundness verification. Contrary to the completeness verification, the soundness verification time decreases with the growth of β and b. This is because with a larger b value, there are fewer original non-outliers inserted as unsound answer (for simulation). Even though a larger b value demands more ANOs,

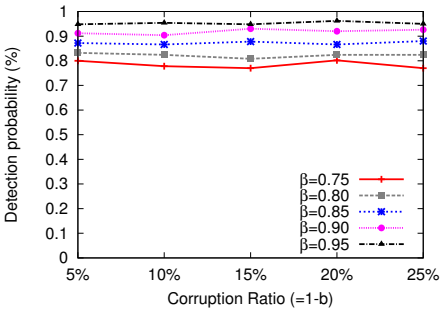
1. http://www.sgi.com/tech/mlc/db/letter.all
2. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

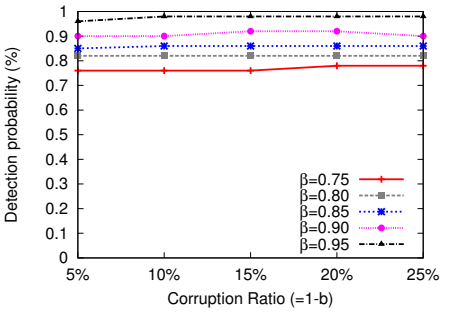


(a) Completeness: $Letter$ Dataset

(b) Completeness: $KDDCUP$ Dataset

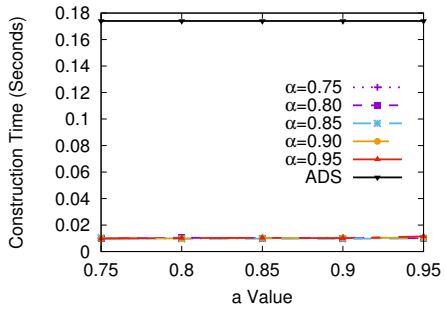


(c) Soundness: $Letter$ Dataset

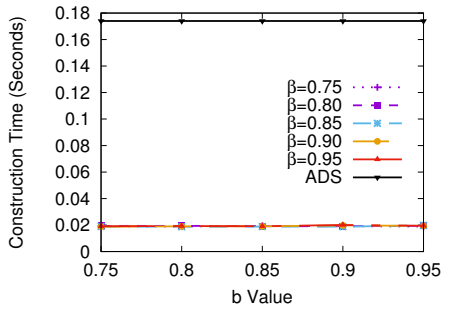


(d) Soundness: $KDDCUP$ Dataset

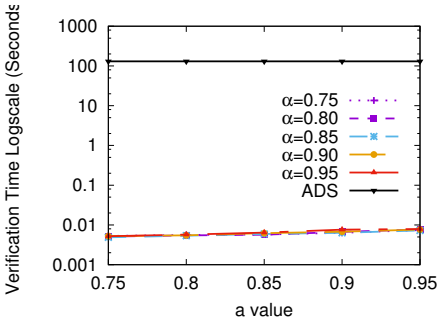
Fig. 4: Robustness of AR-based Approach



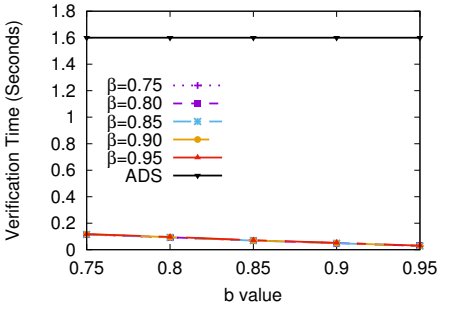
(a) AO Construction Time



(b) ANO Construction Time



(c) Verification Time: Completeness



(d) Verification Time: Soundness

Fig. 5: $AO\&ANO$ Construction Time and Verification Time, $Letter$ Dataset

(α, a)	AO Construction	Completeness Verification
(0.9, 0.75)	0.1649216111	0.0287629264
(0.9, 0.8)	0.1497754622	0.0305521828
(0.9, 0.85)	0.1478960015	0.032380496
(0.9, 0.9)	0.1486778035	0.0343804893
(0.9, 0.95)	0.1472718158	0.0411139978
Deterministic approach	15.341	N/A

(a) Completeness Verification

(β, b)	ANO Construction	Correctness Verification
(0.9, 0.75)	0.031591748	0.4989598102
(0.9, 0.8)	0.0283305431	0.4090301879
(0.9, 0.85)	0.0284852679	0.3174848479
(0.9, 0.9)	0.0286167844	0.2299931051
(0.9, 0.95)	0.0287029023	0.1376942443
Deterministic approach	15.341	42,384

(b) Correctness Verification

TABLE 3: $AO\&ANO$ Construction Time and Verification Time (Second), $KDDCUP$ Dataset

the number of inserted original non-outliers decreases faster than that of $ANOs$. This leads to the decreasing number of outliers (including real non-outliers and $ANOs$) that the client receives, and consequently less verification time. Compared with the ADS-based deterministic approach, the ANO-based soundness verification is at least 10 times more

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2754493, IEEE Transactions on Dependable and Secure Computing

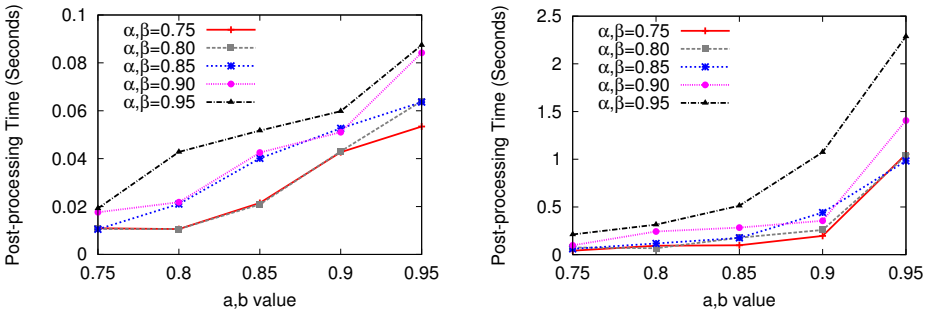
a, b	α, β 0.75	α, β 0.8	α, β 0.85	α, β 0.9	α, β 0.95
0.75	1	1	1	2	2
0.80	1	1	2	2	4
0.85	2	2	4	4	5
0.90	4	4	5	5	6
0.95	5	6	6	8	8

(a) $Letter$ Dataset ($20K$ records)

a, b	α, β 0.75	α, β 0.8	α, β 0.85	α, β 0.9	α, β 0.95
0.75	2	4	4	6	13
0.80	4	4	6	13	14
0.85	6	8	14	17	23
0.90	14	19	23	23	48
0.95	47	70	77	101	167

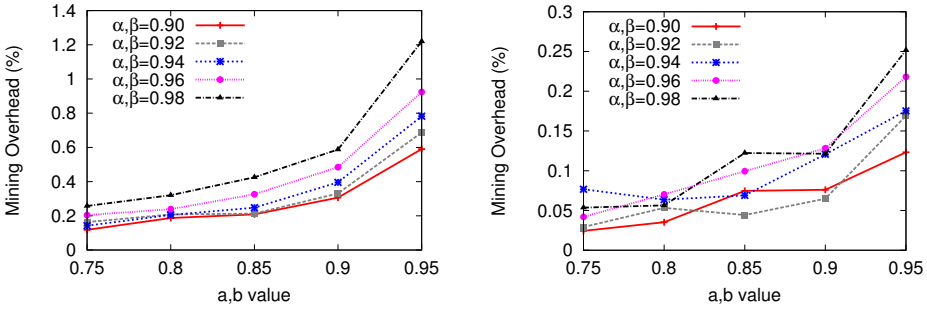
(b) $KDDCUP$ Dataset ($100K$ records)

TABLE 4: Number of records Whose Outlierness Are Examined during Post-Processing



(a) $Letter$ Dataset (b) $KDDCUP$ Dataset

Fig. 6: Post-processing Time



(a) $Letter$ Dataset (b) $KDDCUP$ Dataset

Fig. 7: Mining overhead

efficient.

Second, we measured the time performance on $KDDCUP$ dataset and compared it with the baseline approach. We note that the ADS-based deterministic approach cannot finish the completeness verification within 100 hours. Thus we note "N/A" in the corresponding cell of Table 3. We found that both the AO & ANO construction and verification time are more than that of the $Letter$ dataset, as the size of the $KDDCUP$ dataset is four times larger than that of the $Letter$ dataset. However, the AO/ANO construction is still fast; AO construction only takes 0.165 second at most, while ANO construction only takes 0.035 seconds at most. The ADS-based deterministic approach takes 15.341 seconds to build the authenticated data structure, which is 100 times slower than the AR-based approach. The AR-based approach is also significantly more efficient than the ADS-based deterministic approach. In particular, the completeness verification takes at most 0.042 second, while the correctness verification finishes within 0.5 second. However, the ADS-based deterministic approach needs more than 10 hours to check the correctness, and cannot finish the completeness verification within 100 hours. This proves that our AR-based approach is in particular suitable for the verification under resource-constrained environment (e.g., on smart phones). With little sacrifice on the correctness guarantee, the client saves significantly for verification.

Third, we measured the performance of outlier recovery at the client side. In particular, we count the number of records whose outlierness needs to be examined against the dataset D by the client, and report the result in Table 4 (a) & (b). Both tables show that the number of records whose outlierness needs to be examined is very small compared with the size of the dataset. For example, at most 8 records in $Letter$ dataset (0.04% of the dataset) and at most 167 records in $KDDCUP$ dataset (0.16% of dataset) are examined. Another interesting observation is that even though it is possible that the records that need to be examined include both true and false positive outliers, in our experiments, all of them are false positive outliers (original non-outliers). We also measured the time of outlier recovery at the client side. Figure 6 shows the detailed result. Specifically, it is very efficient to eliminate the false positive outliers, due to the small number of records that need to be examined. For example, it needs no more than 0.1 second on the $Letter$ dataset.

6.3 Overhead at Server Side

We measured the mining overhead at the server side as $|T_{D^+} - T_D|/T_D$, where T_D and T_{D^+} are the time of mining outliers from the original database D and the dataset $D^+ = D \cup \Delta D$. Figure 7 (a) and (b) show the mining overhead of the $Letter$ dataset and $KDDCUP$ dataset respectively. We observed that the insertion of artificial records (ARs) does not introduce significant mining overhead. For example, the mining overhead is no more than 1.2% for the $Letter$

dataset, and at most 0.25% for the $KDDCUP$ dataset The overhead on $KDDCUP$ dataset is much smaller because we insert the same number of artificial records for the same α, β, a, b values into a larger dataset. This proves that our approach is more suitable for datasets of large size.

7 Related Work

Verifiable computations for general-purpose computations. Goldwasser et al. [18] use interactive proofs to verify tractable computation. The proofs enable the client to verify the result's correctness in nearly-linear time. Gennaro et al. [17] formally define verifiable computing as a technique that enables a computationally weak client to verify the result correctness of outsourced computation. They propose a non-interactive proof construction approach. The proof enables the client to verify the result with complexity polynomial to the result's size. However, it demands an expensive pre-processing and input preparation phase to generate auxiliary information for verification. Setty et al. [44] build a general-purpose verification approach to support a wide range of computation including floating-point fractions and inequality comparisons. Parno et al. [40] propose to construct a VC scheme with public delegation and public verifiability from any attribute-based encryption scheme. PEPPER was proposed in [45]. It dramatically reduces the verification cost by using an argument system in which the verifier queries the linear probabilistically checkable proofs in an inexpensive way. PEPPER also reduces the prover's overhead so that its total work is not significantly more than the cost of executing the computation. [39] is the first general-purpose verification framework to demonstrate that the verification can be cheaper than the native computation. While it is reasonable in some scenarios that the same computation is executed on many different inputs, this is not ideal for the DMaS paradigm where the client outsources the data mining computations with a single input dataset.

Authenticated outsourced computations. Quite a few studies [1], [23], [38], [53] focus on the authentication of specific computations. Zhang et al. [53] define a new type of accumulation values so that the outsourced sum operation can be authenticated with two group elements and in constant time. Papadopoulos et al. [38] combine the accumulation values with the suffix tree to facilitate the authentication of outsourced pattern matching. The proof size is proved to be optimal as it includes at most ten accumulation values. Meanwhile, the proof can be readily constructed as the components have been pre-computed in the setup phase. Li et al. [25] design a novel authenticated indexing structure based on Merkle tree to detect the misbehavior by the cloud broker in the service selection process. Abadi et al. [1] propose a protocol that supports verifiable delegated private set intersection on outsourced datasets. The verification cost is dependent on the result size. In the crowdsourcing setting, Kupcu [23] combines cryptography and game theory to incentivize the rational workers to perform the computation correctly, and guarantee the result quality even in the existence of irrational workers who intensionally submit incorrect result.

Authenticated query evaluation in Data-Mining-as-a-Service (DMaS) paradigm. Our problem falls into the category of integrity assurance of the Data-Mining-as-a-Service (DMaS) paradigm. Wong et al. [50] propose auditing techniques for outsourcing frequent itemset mining to an untrusted party. They generate a (small) artificial database such that all itemsets in the database are guaranteed to be frequent and their exact support counts are known. By hosting the artificial database with the original one and checking whether the server has returned all artificial itemsets, the data owner can verify whether the server has returned correct and complete frequent itemsets. However, the artificial itemsets can be easily identified by the server with publicly-available information about the original dataset. In order to fix the issue, Dong et al. [12] provide an approach to construct the artificial itemsets in the original dataset. These artificial itemsets are indistinguishable from the original ones, and guarantee to catch the server's dishonest result with high probability. Dong et al. [11] extend the work by proposing a verification framework based on cryptographic proofs to catch any incorrect mining result with 100% certainty. Other work [27] solves the result integrity verification problem for various computations such as Bayesian network learning and clustering. Their techniques on computations which are dramatically different from outlier mining and thus cannot be directly applied to our problem.

Authenticated query evaluation in Database-as-a-Service (DaS) paradigm. The issue of integrity assurance for database management was initially raised in the Database-as-a-Service (DaS) paradigm [20]. The focus is to assure the integrity of SQL query evaluation over the hosted relational databases. A few techniques have been proposed to provide assurance for SQL query evaluation [14], [20], [24], [46], [51]. For example, Hacigümüş et al. [20] propose a solution that uses conventional encryption techniques and additionally assign a bucket id to each attribute value to facilitate efficient evaluation. that partially executing query on the provider side, they can support range searches and joins in addition to exact match queries. The proposed solutions include Merkle hash trees [24], signatures on a chain of records [34], challenge tokens [46], and counterfeit records [51]. The concept of *verifiable database (VDB)* was first proposed in [7]. It uses the verifiable computation mechanisms to authenticate the query results on outsourced databases. Chen et al. [9], [10] design a new VDB framework to defend against the *forward automatic update* attack and support efficient incremental data updates. These work share the same result integrity concerns in the outsourcing paradigm. However, their focus is mainly SQL query evaluation, which is dramatically different from ours.

Data and pattern security of data mining. The problem of how to protect sensitive data and data mining results for the DMaS paradigm has caught much attention recently. A few work [13], [15], [30], [31], [47], [49] have been done under this theme. Wong et al. [49] consider utilizing a one-to-n item mapping together with non-deterministic addition of cipher items to protect the identification of individual items in the scenario that frequent pattern mining task is outsourced. Unfortunately, this work has potential privacy flaws; Molloy et al. [30] show how privacy can be breached in the framework of [49]. Tai et al. [47] consider the same scenario and proposed a database transformation scheme that is based on a notion of k-support anonymity. To achieve k-support anonymity, they introduced a pseudo taxonomy tree; the third party server will discover the gen-

eralized frequent itemsets instead. Dong et al. [13] design data encoding schemes that preserve high accuracy in data deduplication computation while provide provable privacy guarantee. Dong et al. [15] consider the privacy leakage where the server leverages *functional dependency* to initiate the inference attack. They proposed a heuristic algorithm to block the inference channel by encrypting a small amount of insensitive data cells. Although these works focus on other mining tasks, their encryption techniques can be applied to our work to provide further protection on data and mining results.

Integrity incentive games. Morgan [32] initiates the study by incentivizing heterogeneous individuals in a game to improve the provision of public good. A fixed-prize raffle game is proposed. The author discovers a unique equilibrium when the benefit function is quasi-quadratic. Loiseau et al. [28] extends the study by adjusting the players' actions. Vaidya et al. [48] propose an incentive compatible protocol in the outsourced collaborative filtering computation. However, [48] does not consider the fact even though the result passes the verification, the client still can not put total belief in the result. Pham et al. [41] design an optimal *contract* by setting appropriate rewards, penalty and verification rate to guarantee the result correctness. Moghaddam et al. [29] model the interaction between the service provider and the client as a dynamic game. The service provider sends either legitimate or compromised signal to the client. Under various circumstances, the desired payoff values are set in order to motivate the server to play honestly. Different from [29], the server's decision of whether to cheat on the mining result is static in our model.

8 Conclusion

In the outsourcing paradigm, it is extremely important for the client to check if the result returned by the untrusted service provider is correct. In this paper, we concentrate on the authentication of outsourced outlier mining. We consider three types of untrusted servers with different adversarial knowledge We proposed a lightweight authentication framework by constructing a set of artificial records (ARs) to catch any unsound or incomplete result with high probability guarantee. In order to incentivize the server to discover the outliers faithfully, we design a strategic game where the server always chooses to behave honestly to get the best payoff. We demonstrated the efficiency and effectiveness of our approach via an extensive set of experiments.

In the future, we plan to explore the deterministic authentication approach to catch the cheating behavior of the server with deterministic guarantee. We will also examine how to design authentication techniques that support data updates.

9 Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1350324 and 1464800.

References

[1] Aydin Abadi, Sotirios Terzis, and Changyu Dong. Vd-psi: verifiable delegated private set intersection on outsourced private datasets. *Financial Cryptography and Data Security*, 2016.

[2] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM SIGMOD Record*, pages 37–46, 2001.

[3] Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In *International Workshop on Security Protocols*, 1997.

[4] Fabrizio Angiulli and Fabio Fassetti. Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets. *ACM Transactions on Knowledge Discovery from Data*, 2009.

[5] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 1998.

[6] Vic Barnett and Toby Lewis. *Outliers in statistical data*. Wiley New York, 1994.

[7] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. *Advances in Cryptology*, 2011.

[8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM SIGMOD Rrecord*, 2000.

[9] Xiaofeng Chen, Jin Li, Xinyi Huang, Jianfeng Ma, and Wenjing Lou. New publicly verifiable databases with efficient updates. *IEEE Transactions on Dependable and Secure Computing*, 2015.

[10] Xiaofeng Chen, Jin Li, Jian Weng, Jianfeng Ma, and Wenjing Lou. Verifiable computation over large database with incremental updates. *IEEE Transactions on Computers*, 2016.

[11] Boxiang Dong, Ruilin Liu, and Hui Wendy Wang. Integrity verification of outsourced frequent itemset mining with deterministic guarantee. In *International Conference on Data Mining*, 2013.

[12] Boxiang Dong, Ruilin Liu, and Hui Wendy Wang. Result integrity verification of outsourced frequent itemset mining. In *Annual Conference on Data and Applications Security and Privacy*, 2013.

[13] Boxiang Dong, Ruilin Liu, and Hui Wendy Wang. Prada: Privacy-preserving data-deduplication-as-a-service. In *International Conference on Information and Knowledge Management*, 2014.

[14] Boxiang Dong and Hui Wendy Wang. Arm: Authenticated approximate record matching for outsourced databases. In *International Conference on Information Reuse and Integration*, 2016.

[15] Boxiang Dong, Hui Wendy Wang, and Jie Yang. Secure data outsourcing with adversarial data dependency constraints. In *International Conference on Big Data Security on Cloud*, 2016.

[16] Changyu Dong, Yilei Wang, Amjad Aldweesh, Patrick McCorry, and Aad van Moorsel. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In *ACM Conference on Computer and Communications Security*, 2017.

[17] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Cryptology Conference*, 2010.

[18] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Symposium on Theory of Computing*, 2008.

[19] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 1989.

[20] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *International Conference on Management of Data*, 2002.

[21] Edwin M Knox and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *International Conference on Very Large Data Bases*, 1998.

[22] George Kollios, D Gunupulos, Nick Koudas, and Stefan Berchtold. An efficient approximation scheme for data mining tasks. In *International Conference on Data Engineering*, 2001.

[23] Alptekin Kupcu. Incentivized outsourced computation resistant to malicious contractors. *IEEE Transactions on Dependable and Secure Computing*, 2015.

[24] Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Dynamic authenticated index structures for outsourced databases. In *International Conference on Management of Data*, 2006.

[25] Jingwei Li, Anna Squicciarini, Dan Lin, Smitha Sundareswaran, and Chunfu Jia. Mmbcloud-tree: Authenticated index for verifiable cloud service selection. *IEEE Transactions on Dependable and Secure Computing*, 2015.

[26] Ruilin Liu, Hui Wendy Wang, Anna Monreale, Dino Pedreschi, Fosca Giannotti, and Wenge Guo. Audio: An integrity auditing framework of outlier-mining-as-a-service systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012.

[27] Ruilin Liu, Hui Wendy Wang, and Changhe Yuan. Result integrity verification of outsourced bayesian network structure learning. In *SIAM International Conference on Data Mining*, 2014.

[28] Patrick Loiseau, Galina Schwartz, John Musacchio, Saurabh Amin, and S Shankar Sastry. Congestion pricing using a raffle-based scheme. In *International Conference on Network Games, Control and Optimization*, 2011.

[29] Monireh Mohebbi Moghaddam, Mohammad Hossein Manshaei, and Quanyan Zhu. To trust or not: A security signaling game between service provider and client. In *International Conference on Decision and Game Theory for Security*, 2015.

[30] Ian Molloy, Ninghui Li, and Tiancheng Li. On the (in) security and (im) practicality of outsourcing precise association rule mining. In *International Conference on Data Mining*, 2009.

[31] Anna Monreale and Hui Wendy Wang. Privacy-preserved data mining in cloud. In *International Workshop on Secure Identity Management in the Cloud Environment*, 2016.

[32] John Morgan. Financing public goods by means of lotteries. *The Review of Economic Studies*, 67(4):761–784, 2000.

[33] Peter Morris. *Introduction to game theory*. Springer Science & Business Media, 2012.

[34] Maithili Narasimha and Gene Tsudik. Dsac: integrity for outsourced databases with signature aggregation and chaining. In *ACM International Conference on Information and Knowledge Management*, 2005.

[35] Hoang Vu Nguyen, Vivekanand Gopalkrishnan, and Ira Assent. An unbiased distance-based outlier detection approach for high-dimensional data. In *International Conference on Database Systems for Advanced Applications*, 2011.

[36] Martin J Osborne. *An introduction to game theory*. Oxford University Press New York, 2004.

[37] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *International Conference on Data Engineering*, 2003.

[38] Dimitrios Papadopoulos, Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Practical authenticated pattern matching with optimal proof size. *Proceedings of the VLDB Endowment*, 2015.

[39] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Symposium on Security and Privacy*, 2013.

[40] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography*. 2012.

[41] Viet Pham, MHR Khouzani, and Carlos Cid. Optimal contracts for outsourced computation. In *International Conference on Decision and Game Theory for Security*, 2014.

[42] Ling Qiu, Yingjiu Li, and Xintao Wu. Protecting business intelligence and customer privacy while outsourcing data mining tasks. *Knowledge and Information Systems*, 2008.

[43] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, 2000.

[44] Srinath Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security Symposium*, 2012.

[45] Srinath TV Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *The Network and Distributed System Security Symposium*, 2012.

[46] Radu Sion. Query execution assurance for outsourced databases. In *International Conference on Very Large Data Bases*, 2005.

[47] Chih-Hua Tai, Philip S Yu, and Ming-Syan Chen. k-support anonymity based on pseudo taxonomy for outsourcing of frequent itemset mining. In *International Conference on Knowledge Discovery and Data mining*, 2010.

[48] Jaideep Vaidya, Ibrahim Yakut, and Anirban Basu. Efficient integrity verification for outsourced collaborative filtering. In *International Conference on Data Mining*, 2014.

[49] Wai Kit Wong, David W Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. Security in outsourcing of association rule mining. In *International Conference on Very Large Data Bases*, 2007.

[50] Wai Kit Wong, David W Cheung, Edward Hung, Ben Kao, and Nikos Mamoulis. An audit environment for outsourcing of frequent itemset mining. In *International Conference on Very Large Data Bases*, 2009.

[51] Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. Integrity auditing of outsourced data. In *International Conference on Very Large Data Bases*, 2007.

[52] Yin Yang, Dimitris Papadias, Stavros Papadopoulos, and Panos Kalnis. Authenticated join processing in outsourced databases. In *International Conference on Management of Data*, 2009.

[53] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. Integridb: Verifiable sql for outsourced databases. In *Conference on Computer and Communications Security*, 2015.

Dr. Boxiang Dong is an assistant professor in the Computer Science Department, Montclair State university, New Jersey. He received his Ph.D degree in computer science from Stevens Institute of Technology, New Jersey. He is dedicated to facilitating the integration of big data analysis and cybersecurity. His research interests include verifiable computing, data mining, anomaly detection, data security and privacy.

Dr. Wendy Hui Wang is an associate professor in the Computer Science Department, Stevens Institute of Technology, New Jersey. She received her PhD degree in computer science from University of British Columbia, Vancouver, Canada. Her research interests include data management, data mining, database security, and data privacy. She is a member of the editorial boards of Journal of Information Technology and Architecture and Journal of Information Systems.

Dr. Anna Monreale is a post-doc at the Computer Science Dept. of the Pisa University and a member of the KDD-LAB, a joint research group with the Information Science and Technology Institute of the National Research Council in Pisa. She received his MS degree and PhD degree in Computer Science from University of Pisa in 2007 and 2011. Her research is in privacy-aware data mining and data publishing and in privacy-preserving outsourcing of analytical tasks.

Dr. Dino Pedreschi Dino Pedreschi is a full professor of Computer Science at the University of Pisa. His research interests are in data mining, and in privacy-preserving data mining. He is a member of the PC of the main international conferences on data mining and knowledge discovery. He has been granted a Google Research Award (2009) for his research on privacy-preserving data mining and anonymity-preserving data publishing.

Dr. Fosca Giannotti is a senior researcher at the Information Science and Technology Institute of the National Research Council at Pisa, Italy, where she leads the KDD-LAB, a joint research initiative with the University of Pisa. Her research interests include data mining query languages, knowledge discovery support environment, web-mining, spatio-temporal data mining, and privacy preserving data mining. She has been the coordinator of various projects and she served in the scientific committee of various conferences.

Dr. Wenge Guo is an associate professor in the Department of Mathematical Sciences, New Jersey Institute of Technology. He received his PhD degree in Biostatistics from University of Cincinnati, Ohio. His research interests include large-scale multiple testing, high-dimensional data analysis, Bioinformatics, and machine learning. He is a member of the editorial boards of Statistics and Probability Letter, PLOS ONE, and Calcutta Statistical Association Bulletin.