AnKLe: Detecting Attacks in Large Scale Systems via Information Divergence

Emmanuelle Anceaume, Yann Busnel, Sébastien Gambs

AnKLe: Detecting Attacks in Large Scale Systems via Information Divergence

Emmanuelle Anceaume
IRISA / CNRS
Rennes, France
emmanuelle.anceaume@irisa.fr

Yann Busnel
LINA / Université de Nantes
Nantes, France
Yann.Busnel@univ-nantes.fr

Sébastien Gambs
IRISA / Université de Rennes 1
INRIA – Rennes Bretagne Atlantique
Rennes, France
sebastien.gambs@irisa.fr

***Abstract*—In this paper, we consider the setting of large scale distributed systems, in which each node needs to quickly process a huge amount of data received in the form of a stream that may have been tampered with by an adversary. In this situation, a fundamental problem is how to detect and quantify the amount of work performed by the adversary. To address this issue, we propose AnKLe (for Attack-tolerant eNhanced Kullback-Leibler divergence Estimator), a novel algorithm for estimating the KL divergence of an observed stream compared to the expected one. AnKLe combines sampling techniques and information-theoretic methods. It is very efficient, both in terms of space and time complexities, and requires only a single pass over the data stream. Experimental results show that the estimation provided by AnKLe remains accurate even for different adversarial settings for which the quality of other methods dramatically decreases.**

***Index Terms*—Data Stream; Kullback-Leibler Divergence; Sampling; Byzantine Adversary; Scalability; Performance Analysis.**

I. Introduction

The main objective of this paper is to propose an algorithm for estimating the similarity between an observed data stream and the expected (*i.e.* idealized) one in the context of massive data streams. More precisely, we consider the setting of large scale distributed systems, in which each node needs to quickly process a huge amount of data. Typically, this data corresponds to IP network traffic, sensors readings, nodes identifiers or any other data issued from distributed applications. For instance, in IP network management, the analysis of the stream may be used to detect the presence of outliers or intrusions when changes in the communication patterns occur [1], to estimate the heaviest users or the more popular sites [2], or to dynamically dimension routers. In sensors networks, probabilistic laws modeling data streams are used in tracking applications for estimating the position of target sensors [3], or for correlating geographical or environmental informations [4], [5]. Finally, in large scale and dynamic systems, uniform sampling is one of the fundamental primitive [6] that allows for instance, by analyzing the information gathered across the network, to estimate the size of the system, its topological organization, or its available resources so that efficient dissemination, load-balancing or data-caching algorithms can be designed and implemented [7], [8].

In the context of massive data streams, nodes need to quickly process on the fly the flow of data. Moreover, nodes can only locally store very limited data and perform few operations on this data. Additionally, it is often the case that if some data has not been locally stored for further processing, once it has been read, it cannot be read anymore (this refers to the one-pass data streaming model). The problem of detecting changes or outliers in a data stream is similar to the problem of identifying patterns that do not conform to the expected behavior, which has been an active area of research for many decades. For instance, depending on the specificities of the domain considered and the type of outliers considered, different methods have been designed, namely classification-based, clustering-based, nearest neighbor based, statistical, spectral, and information theory. A comprehensive survey of these techniques, their advantages and their drawbacks is given in [9]. A common feature of these techniques is their space complexity and their computational cost, as they rely on full space algorithms for analyzing their data.

Given our constraint settings — one-pass analysis of a huge amount of data with limited resources, both in space and time— we propose an algorithm to detect changes in the observed stream with respect to an expected behavior by relying on sampling techniques and information-theoretic methods. More precisely, by adequately sampling the observed data stream, we estimate with high accuracy the distance between the expected stream and the observed one, and this even

if the stream has been tampered with by an adversary. The metric, we use in our context is the Kullback-Leibler (KL) divergence, which can be viewed as an extension of the Shannon entropy and is often referred to as the relative entropy [10]. Citing Chakrabarti *et al.*, [11], "[...] rationale of estimating entropy-based distances is that there are intimate connections between the randomness of traffic sequences (formalized as the entropy) and the propagation of malicious events. Indeed, detecting sudden changes in a stream may be a good indicator of attacks".

Our main contribution is the proposition of AnKLe (Attack-tolerant eNhanced Kullback-Leibler divergence Estimator), an algorithm that estimates the relative entropy between the observed stream and the expected ones in the context of massive data streams. As introduced above, AnKLe combines information-theoretic and sampling techniques to estimate accurately the relative entropy, while using only a memory of small size to cope with the very strict space constraint. Extensive simulations indicate that while AnKLe rely on sampling techniques, the accuracy of the estimation is very high. AnKLe, as a data streaming algorithm, benefits from their desired properties such as low computational and storage costs and one-pass processing of the stream. Therefore, AnKLe is perfectly adapted to the setting in which data must be read and process quickly. Finally, AnKLe is versatile enough to cope with any type of input distribution, including distribution that have been generated by an adversary. To the best of our knowledge, an algorithm combining all these strengths for the estimation of relative entropy has never been published before in the literature.

The paper is organized as follows. First, Section II reviews the related work on the estimation of the relative entropy of data streams while Section III-A describes the data stream model as well as the adversary model considered. Section III-B briefly introduces the concepts of information theory that we intensively use in this work. Section IV-A presents the different buildings blocks of our algorithm and finally Section IV-B describes AnKLe, our data streaming algorithm for estimating the relative entropy of a stream. In Section V, we empirically evaluate the accuracy of the estimation provided by AnKLe by comparing it to the exact value of the KL divergence on different data streams and also to adapted versions of state-of-the-art estimator-based algorithms, namely, Alon *et al.* [12] and Chakrabrti *et al.* [13]. Finally, we conclude in Section VI.

II. Related Work

In this paper, we consider the Kullback-Leibler (*i.e.*, the relative entropy) estimation problem. In information theory, the concept of entropy corresponds to the uncertainty of a random variable, and as a special case, the entropy of a stream quantifies the randomness of a data stream. On the other hand, relative entropy measures the difference between two distributions, and therefore the data stream relative entropy quantifies the amount of information separating one specific observed stream from expected ones.

Previous works have proposed efficient algorithms (in sublinear space, and sometimes even polylogarithmic space, in the size of the stream) to accurately estimate the entropy of a data stream. Most of these works rely on the seminal algorithm designed by Alon, Matias and Szegedy [12]. In their work, the authors estimate the k-th frequency moment F_k of a data stream, a statistic directly related to the input stream (*cf.*, Section III-B). For instance, the frequency moment F_0 corresponds to the number of distinct items in a stream while F_1 represents the size of the stream. Subsequently to this work, Guha *et al.* [14] have considered the entropy estimation problem in the random stream model, in which items are randomly distributed in the stream. Chakrabarti *et al.* [11] have studied the same problem but assuming the adversarial stream model, in which the items are ordered according to an adversarial strategy. Furthermore, Chakrabarti *et al.* [11], [13] and Lall *et al.* [15] have considered the challenging issue of estimating the entropy accurately when the entropy is strictly less than one. Such streams have a few items with a high occurrence frequency while all the other items appear approximately with the same low frequency. In order to guarantee a small relative estimation error in this setting, one needs to decompose the analysis of the stream into two parts, one part keeping the highly frequent items and the other part comprising the items with the same low frequency. More details will be given in Section IV-B.

Estimating the relative entropy of data streams has also been shown to be an interesting tool in the security and dependability community. For instance, Cachin [16] defines the security of a steganographic system in terms of the Kullback-Leibler entropy between the distributions of the covertext and the stegotext. Specifically, if the relative entropy is less than or equal to a given parameter ε then the stegosystem is considered ε-secure, while if the relative entropy is equal

to zero (*i.e.*, $\varepsilon = 0$), then the stegosystem is perfectly secure. Anceaume *et al.* [17] have proposed a characterization of the adversarial power to bias uniform and ergodic sampling in large scale system. This characterization is done in terms of the relative entropy between a stream composed of node identifiers and a uniform stream. More precisely, the authors have derived lower bounds on the work that an adversary has to exert to bias this input stream so that uniform and ergodic sampling does not hold.

A fundamental issue is to derive efficient algorithms both in space and time to estimate the relative entropy in presence of huge amount of data.

III. System Model and Background

A. System Model

We consider a system in which a node P receives a large data stream $\sigma = a_1, a_2, \ldots, a_m$, where the i-th element a_i of the stream is called an item. This node P might be a router that watches TCP/IP packets [2], a stegosystem [16] or a peer sampling component [17]. In the following, we describe a single instance of P, but clearly multiple instances of P may co-exist in a system. The value u of an item is assumed to be drawn from a large universe N and the length of the stream m is very high (*e.g.*, 2^{32}). Moreover, items can be repeated multiple times in the stream. The number of distinct items in the stream is denoted by n, and thus, we have $n \leq m$. We suppose that items arrive regularly and quickly, and due to memory constraints, need to be processed sequentially and in an online manner. Therefore, node P can locally store only a small fraction of the items and perform simple operations on them. The algorithms we consider in this work are characterized by the fact that they can approximate some function on σ with a very limited amount of memory (typically sublinear or polylogarithmic in the size of the data stream m). We refer the reader to [18] for a detailed description of data streaming models and algorithms.

a) Adversary Model: We suppose that the adversary is omnipotent in the sense that it may actively tamper with the data stream of any node by observing, inserting, dropping or re-ordering items of their input stream. The activity of the adversary can be detected by an honest node provided that it can accurately estimate the divergence between the observed stream and the ideal one. The presence of such a divergence is important as it may be a good indicator of attacks. For instance, in large scale systems, it might be used as an alarm to prevent the adversary from poisoning routing tables (also called eclipse attacks [19]) by freezing routing tables updates as long as the relative entropy is too high. We suppose that the algorithm used by a node to estimate the divergence is public knowledge (*i.e.*, to avoid some kind of security by obscurity), however the adversary has not access to the local random coins used in the algorithm (if any).

B. Preliminaries

Prior to describing our algorithm for estimating the KL divergence of a stream in a single pass using sublinear space, we first present notations and background on data streams analysis that make this paper self-contained.

Entropy. Intuitively, the entropy is a measure of the randomness of a data stream σ. The entropy H_σ is minimum (*i.e.*, equal to zero) when all the items in the stream are the same, and it reaches its maximum (*i.e.*, equal to $\log m$)[1] when all the items in the stream are distinct. Specifically, we have

$$H_\sigma = -\sum_{u \in N} p_u \log p_u,$$

where $p_u = m_u/m$, for each $u \in N$, with $m_u = |\{j : a_j = u\}|$ representing the number of times the value u appears in the stream σ (by convention, $0 \log 0 = 0$). It is commonly called the frequency of the item u. The norm of the entropy is defined as $F_H = \sum_{u \in N} m_u \log m_u$.

Kullback-Leibler divergence. The Kullback-Leibler (KL) divergence [20], also called the relative entropy, is a robust metric for measuring the statistical difference between two data streams. The KL divergence is a member of a larger class of distances known as the Ali-Silvey distances [21]. Given two probability distributions on events $p = \{p_1, \ldots, p_n\}$ and $q = \{q_1, \ldots, q_n\}$, the Kullback-Leibler divergence between p_u relative to q_u is defined as the expected value of the likelihood ratio with respect to q_u:

$$\mathcal{D}(p||q) = \sum_{u \in N} p_u \log \frac{p_u}{q_u} = H(p,q) - H(p),$$

where $H(p) = -\sum p_u \log p_u$ is the (empirical) entropy of p and $H(p,q) = -\sum p_u \log q_u$ is the cross entropy of p and q. As we use a logarithm in base 2, the divergence is measured in bits. When $p_n = q_n$, the KL divergence is minimal and is equal to zero. Let $p^{(\mathcal{U})}$ be the uniform distribution corresponding to a uniform stream

[1] Thereafter, we will denote by log the logarithm in base 2.

(*i.e.*, $\forall u \in \sigma, p_u^{(\mathcal{U})} = \frac{1}{n}$), and q be the probability distribution corresponding to the input stream. In the rest of this paper and according to the classical use of the KL-divergence, we consider $\mathcal{D}(q||p^{(\mathcal{U})})$ as a measure of the divergence of the current stream from the ideal one. While all the distance measures in the Ali-Silvey distances are applicable to quantifying statistical differences between data streams, the KL divergence is particularly suited to our context since it gives rise to a small number of false positives when the two data streams are not significantly different.

Frequency moments. Frequency moments are important statistical tools that have been introduced by Alon *et al.* [12]. Computing frequency moments F_k allows to quantify the amount of skew in a data stream. Among the remarkable moments, F_0 represents the number of distinct elements in a stream while F_1 corresponds to the size m of the stream. For each $k \geq 0$, the k-th frequency moment F_k of σ is defined as

$$F_k = \sum_{u \in N} m_u^k,$$

where m_u is defined as above.

2-universal Hash Functions. In the following, we intensively use hash functions randomly picked from a 2-universal hash family. A collection H of hash functions $h : \{1, \ldots, M\} \rightarrow \{0, \ldots, M'\}$ is said to be *2-universal* if for every two different items $x, y \in [M]$,

$$\mathbb{P}_{h \in H}\{h(x) = h(y)\} \leq \frac{1}{M'}.$$

Randomized (ε, δ)-approximation Algorithm. A randomized algorithm $\mathcal{A}$ is said to be an (ε, δ)-approximation of a function ϕ on σ if for any sequence of items in the input stream σ, $\mathcal{A}$ outputs $\hat{\phi}$ such that $\mathbb{P}\{\mid \hat{\phi} - \phi \mid > \varepsilon\phi\} < \delta$, where $\varepsilon, \delta > 0$ are given as parameters of the algorithm.

IV. Detecting Adversarial Behaviors via KL Divergence Estimation

A. Building Blocks

In this section, we describe three algorithms that form the building blocks of the AnKLe algorithm. The first one, due to Alon *et al.* [12] estimates the k-th frequency moment of a stream. Although we do not need such a quantity, we adopt the structure of their algorithm to estimate the relative entropy of a stream. The second algorithm due to Bar-Yossef *et al.* [22] estimates the number of distinct items in a stream. In our context this amounts to estimating n. Finally the third algorithm, proposed by Misra and Gries [23], estimates the k most frequent items of a stream. All these algorithms have been designed in the stream data model (*cf.* Section III-A). For self-containment reasons, we briefly review these building blocks and describe their theoretical guarantees.

1) Estimating the k^{th} Moment of a Stream: The AnKLe algorithm is inspired from the method of Alon, Matias and Szegedy [12] to approximate the KL divergence of a stream. In the following, we refer to this algorithm as the AMS algorithm. Briefly, the core of the AMS algorithm is a basic estimator, which takes the form of a random variable X whose mean value is exactly equal to the k^{th} frequency moment of a stream and whose variance is very small. Several basic estimators are computed on the stream (specifically $s_1 \times s_2$ independent basic estimators X_{ij}, for $1 \leq i \leq s_1$ and $1 \leq j \leq s_2$), and the final estimator Y is set to be

$$Y = \mathrm{median}_{1 \leq \mathrm{j} \leq \mathrm{s}_2} \left(\frac{1}{\mathrm{s}_1} \sum_{\mathrm{i}=1}^{\mathrm{s}_1} \mathrm{X}_{\mathrm{ij}} \right).$$

Alon *et al.* [12] have shown that for any $\varepsilon, \delta \in (0, 1)$, if $s_1 \geq Var[X]/(\varepsilon^2 E[X]^2)$ and $s_2 = 4\log(1/\delta)$, then Y is a (ε, δ)-approximation of $E[X]$ (*i.e.*, $\mathbb{P}\{\mid E[X] - Y \mid > \varepsilon E[X]\} < \delta$).

2) Estimating the Number of Items in the Stream: The problem of estimating the number of distinct elements has received a lot of attention in the data stream model. First, the seminal work of Flajolet and Martin [24] has shown that it is possible to compute such an estimate using only logarithmic space in n by relying on properties of hash functions. Afterwards, follow-up enhancements have improved the accuracy of the estimation [22]. (A comprehensive survey describing the literature on distinct elements in the data stream model is presented by Gibbons in [25].) Thereafter, we briefly sketch the BJKST algorithm proposed by Bar-Yossef *et al.* [22], which is so far the most efficient space and time algorithm for approximating the number of distinct elements in a stream in a single pass (and this even if the stream is adversarially ordered).

The BJKST algorithm is based on the coordinating sampling algorithm of Gibbons and Tirthapura [26]. Let $\sigma = a_1, \cdots, a_m$ be a a stream of items such that $a_i = v \in [2^r]$ and $h_1, \cdots, h_k$ be a set of k pairwise independent universal hash functions that map symbols v_i from $[2^r]$ onto $[2^r]$. Moreover, $S_1, \cdots, S_k$ is a set of k buffers of size t. The algorithm consists in running k instances of the same procedure, such that procedure j uses

Algorithm 1: BJKST algorithm

Input: An input stream σ; k and t settings;
Output: The estimate $\hat{F}_0$ of the number of distinct elements in the stream

Choose k 2-universal hash functions $h : [n] \to [n]$;
Choose k 2-universal hash functions $g : [n] \to [\mathcal{O}(\log n/\varepsilon^2)]$;
Initialization of k buffers S_j of size t;
for $j \in [1..k]$ **do** $\ell_j = 0; S_j = \emptyset$;
for $a_i \in \sigma$ **do**
    $v = a_i$;
    for $j = 1$ **to** k **do**
        b=the largest $r \geq 0$ such that the r rightmost bits in $h_j(v)$ are all 0;
        if $b \geq \ell_j$ and $(g(v, b)) \notin S_j$ **then**
            $S_j = S_j \cup \{(g(v, b))\}$;
            while $\mid S_j \mid > t$ **do**
                $S_j = S_j \setminus \{g(v', b')\}$ with $b' = \ell_j$;
                $\ell_j = \ell_j + 1$;
return $\hat{F}_0 = \mathrm{median}_{1 \leq \mathrm{j} \leq \mathrm{k}} 2^{\ell_\mathrm{j}} \mid \mathrm{S_j} \mid$;

hash function h_j. The hash function h_j determines the "level" of items from the stream such that half of the items have a level equal to 1, a quarter of them have a level equal to 2, ..., until finally $\frac{1}{2^i}$ of them have a level equal to i.

Initially, the current level of a particular ℓ_j is set to be 0. Afterwards, items are read from the stream, and by hashing them, one can deduce their level in the following way: item a_i has level i if the i rightmost bits of $h_j(a_i)$ are all set to 0. If the level of the read item is greater than or equal to the current level ℓ_j then this item is stored (once) in S_j together with its level. The current level ℓ_j is incremented when more than t items have a level greater than or equal to ℓ_j. Afterwards, all the items with a level equal to ℓ_j are removed from buffer S_j. When this procedure stops, at level ℓ_j each item is in buffer S_j with probability $1/2^{\ell_j}$. To ensure that the estimate $\hat{F}_0 = 2^{\ell_j} \mid S_j \mid$ is a (ε, δ)-approximation of F_0, $k = 1/\delta$ instances of the procedure are executed, and $\hat{F}_0$ is set to be $\mathrm{median}_{1 \leq \mathrm{j} \leq \mathrm{k}} 2^{\ell_\mathrm{j}} \mid \mathrm{S_j} \mid$. The BJKST algorithm improves upon the original coordinating algorithm from Gibbons and Tirthapura mainly by decreasing the space bound. This is achieved by using k additional universal hash function g to store the hash of the items in buffers S_j instead of the items themselves [22]. The pseudo-code of the algorithm is shown in Algorithm 1.

Bar-Yossef *et al.* [22] have shown that for any ε, their algorithm outputs $\hat{F}_0$ such that

$$\mathbb{P}\{|\hat{F}_0 - F_0| \leq \varepsilon\} \geq 1 - \delta,$$

where $\delta = 1/3$. The worst-case running time for each input symbol is $\mathcal{O}(r + 1/\varepsilon^2(\log(1/\varepsilon) + \log r))$, and the total space required by the algorithm is $\mathcal{O}(r + 1/\varepsilon^2(\log(1/\varepsilon) + \log r))$ bits, where $\mathcal{O}(r)$ represents the space needed for implementing each hash function.

3) Determining the Most Frequent Identifiers of a Stream: As for counting the number of distinct items in a stream, the problem of determining the k most frequent items in a stream has also been studied extensively in the data stream literature. Thereafter, we describe a deterministic algorithm that outputs the k most frequent items in a stream as well as an estimate $\hat{m_u}$ for the frequency m_u of each item, if $m_u > m/k$. This algorithm due to Misra and Gries [23] maintains k counters such that for each counter, its key is the item read from the stream and its value is related to the frequency of items. Initially, all the counters are set to $(-, 0)$. Afterwards, when an item is read from the stream, if that item has already a counter associated to it, then this counter is incremented. If this is not the case and if there are still free counters available, then one of these free counters is allocated to this new item and its value is set to 1. Otherwise, all the allocated counters are decremented by one, and if after this operation some of them are equal to 0 then their keys are erased and the counters are released. The pseudo-code of the Misra Gries algorithm is presented in Algorithm 2.

The Misra Gries [23] algorithm with parameter k provides, for each item u in the stream, an estimate $\hat{m_u}$ satisfying

$$m_u - \frac{m}{k} \leq \hat{m_u} \leq m_u.$$

The algorithm uses a space of $\mathcal{O}(k(\log n + \log m))$ bits.

B. The AnKLe algorithm

This section presents AnKLe, the algorithm we propose for computing the KL divergence of a stream. Our starting point is the re-writing of the KL divergence as follows. From Definition 1, we

Algorithm 2: Misra-Gries algorithm

Input: An input stream σ; a precision parameter k;
Output: The set of the k most frequent items in a stream as well as an estimate of their frequency

```
1  for j ∈ [0..k] do A[j] ← (⊥, ⊥);
2  for a_i ∈ σ do
3      v = a_i;
4      if ∃u such that the item of A[u] is s then
       increment the count value of A[u];
5      else
6          if ∃u' such that A[u'] = (⊥, ⊥) then
7              A[u'] = (v, 1)
8          else for i = 1 to k do
9              Decrement the count of A[i];
10             if the count value of A[i] = 0 then
11                 A[i] = (⊥, ⊥)
12 return A;
```

have

$$\begin{aligned}
&\mathcal{D}(q_\sigma || p^{(\mathcal{U})}) \\
&= \sum_{i=1}^{n} q_i \log\left(q_i\right) - \sum_{i=1}^{n} q_i \log\left(p_i^{(\mathcal{U})}\right) \\
&= \frac{1}{m}\left(\sum_{i=1}^{n} m_i \log\left(\frac{m_i}{m}\right) - \sum_{i=1}^{n} m_i \log\left(\frac{1}{n}\right)\right) \\
&= \log(n) - \log(m) + \frac{1}{m}\sum_{i=1}^{n} m_i \log\left(m_i\right). \qquad (1)
\end{aligned}$$

Thus estimating the KL-divergence amounts in *(1)* estimating the number of distinct items in the stream (*i.e.*, F_0) in order to obtain a good approximation of $\log(n)$, *(2)* determining the k most frequent items in the stream, and *(3)* estimating the $\sum_{i=1}^{n} m_i \log\left(m_i\right)$, which corresponds to the norm of the entropy F_H.

AnKLe algorithm we propose for estimating the KL divergence is presented in Algorithm 3. It consists of two phases, the first one (lines 3–17) is executed upon reception of the items of the stream, while the second one (lines 18–26) is run when m items have been read from the stream. The first phase is composed of three tasks (T_1, T_2 and T_3), which are executed in parallel. Task T_1 (see line 5) estimates the number of distinct items present in the stream, task T_2 (see line 8) identifies the k most frequent items in the stream, and T_3 samples random items in the stream in order to compute their exact frequency. Specifically, Task T_3 (lines 11–17) consists in running a sampling estimator X on the stream. The basic estimator $X = X_{i,j}$ is designed so that its mean value is equal to the norm of the entropy F_H and its variance is small. More precisely, we have

$$X = m(r\log r - (r-1)\log(r-1)) \qquad (2)$$

where r is the random variable representing the number of occurrence of an item ℓ in the stream. This item ℓ is such that its position j in the stream is a random number in $[m]$. The random variable r counts the number of times ℓ appears in the stream from position j onwards. Formally, r is defined as

$$r = \mid \{j : j \geq \ell, a_j = a_\ell\} \mid .$$

We can show as in [12], [15], that the basic estimator X is unbiased (*i.e.*, the expectation of X is equal to F_H). Specifically,

$$\begin{aligned}
E[X] &= \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m_i} m(j\log j - (j-1)\log(j-1)) \\
&= \frac{m}{m}\sum_{i=1}^{n} m_i \log(m_i) \\
&= F_H. \qquad (3)
\end{aligned}$$

To improve the accuracy of the estimation, $s_1 \times s_2$ such basic estimators X_{ij} (for $1 \leq i \leq s_1$ and $1 \leq j \leq s_2$) are used, each one sampling a random position in the stream. From the implementation point of view, tracking these estimators consists in storing $s_1 \times s_2$ counters, each one counting the number of occurrences of an item whose position has been randomly chosen in the stream. When item u is read from the input stream, if u has already one or more counters assigned to it then all these counters are incremented. In addition, if the position at which u has been read in the stream is one of the chosen locations, then a counter is assigned to u, and its value is set to 1. Thus for each of these "tracked" items, an exact count of their frequency is continuously maintained starting from a random position in the stream.

The post-processing phase of AnKLe algorithm estimates the KL divergence of the input stream according to Relation (1). This phase is executed when m items have been read from the input stream. In this work, we suppose that m is a parameter of the algorithm, however by using techniques proposed in Chakrabarti *et al.* [13] we can extend our solution to streams whose size is *a priori* unknown. To accurately estimate the KL divergence of the stream, one needs to cope with patterns in

Algorithm 3. AnKLe algorithm

Input: An input stream σ of length m, k (number of counters in the Misra-Gries algorithm), s_1 and s_2 (size of the AMS-based matrix)

Output: An estimation of $\mathcal{D}(q_\sigma || p^{(\mathcal{U})})$, the KL divergence between the observed stream and the uniform one

1 Choose $s_1 \times s_2$ random integers in $[1..m]$;
2 **for** $u_1 \in [0..s_1], u_2 \in [0..s_2]$ **do** $S[u_1, u_2] \leftarrow (\perp, \perp)$;
3 **for** $a_j \in \sigma$ **do**
4 $v = a_j$;
5 **begin Task** T_1**:**
6 $\hat{F}_0 \leftarrow$ BJKST Algorithm (Algorithm 1) fed with v
7 **end**
8 **begin Task** T_2**:**
9 $\hat{F} \leftarrow$ Misra-Gries Algorithm (Algorithm 2) fed with v
10 **end**
11 **begin Task** T_3**:**
12 **forall** entries (u_1, u_2) of matrix S such that $(s_{(u_1,u_2)}, r_{(u_1,u_2)}) \neq (\perp, \perp)$ **do**
13 **if** $s_{(u_1,u_2)} = s$ **then**
14 $r_{(u_1,u_2)} \leftarrow r_{(u_1,u_2)} + 1$;
15 **if** j is one the $s_1 \times s_2$ random integers **then**
16 assign $(v, 1)$ to the first unused entry of S;
17 **end**
18 **forall** entries (u_1, u_2) of matrix S **do**
19 **if** $(s_{(u_1,u_2)}, -) \in \hat{F}$ **then**
20 $X_{u_1,u_2} \leftarrow 0$ // *$s_{(u_1,u_2)}$ is one of the frequent items returned by Task T_2*;
21 **else**
22 $X_{u_1,u_2} \leftarrow m\left(r_{(u_1,u_2)} \log r_{(u_1,u_2)} - (r_{(u_1,u_2)} - 1)\log(r_{(u_1,u_2)} - 1)\right)$;
23 $Y_S \leftarrow$ average of all *non null* entries X_{u_1,u_2};
24 $Y_{\hat{F}} \leftarrow \sum_{(s_i,r_i)\in\hat{F}} r_i \log r_i$;
25 $p \leftarrow 1 - \max\left(0, \frac{\min\left(Y_S, Y_{\hat{F}}\right) - m}{10 \cdot m}\right)$;
26 **return** $D = \log \hat{F}_0 - \log m + \frac{p}{m}\left(Y_S + Y_{\hat{F}}\right)$;

which a small number of items occur with a very high frequency with respect to the other items. When such patterns occur, the basic estimator X alone is unable to compute the norm of the entropy in bounded space [13]. Indeed, by analogy of the calculation performed in [12], the variance of the estimator grows with the norm of the entropy. Thus in presence of high frequency patterns, one needs to estimate the relative entropy using a different approach. In Chakrabarti *et al.*, the authors propose to decompose the computation of the entropy as the sum of the entropy of the most frequent items and the estimation of the entropy of the remaining items of the stream. In AnKLe, we extend their method to deal with any stream distribution in order to guarantee that whatever the strategy of the adversary, the error on the estimation is kept small (as shown in Section V). Specifically, the basic estimator X is computed on unfrequent items (*cf.*, lines 18–23) as done in Relation (3), while the contribution of highly frequent items on the norm of the entropy is directly computed as $\sum_{(s_i,r_i)\in\hat{F}} r_i \log r_i$ (*cf.*, lines 24). The set $\hat{F}$ represents the set of highly frequent items dynamically computed in Task T_2. Finally, to prevent some of the items to appear in both terms, we weight the contribution of both terms by p (*cf.*, line 26).

V. Performance Analysis

In this section, we evaluate the accuracy of AnKLe by comparing its estimation with the exact value of the KL divergence computed between the observed input stream and the uniform one. We also compare AnKLe to adapted versions of the estimator-based algorithms of Alon *et al.* [12] and Chakrabarti *et al.* [13]. In the former case, the original estimator computes the k-th frequency moment of a stream, while in the latter case, the original estimator measures the entropy of a stream. In both cases, the adapted versions compute instead the norm of the entropy.

All the experiments have been conducted on synthetic traces of streams whose distributions are shown in Figure 1. (Note that we use a logarithmic scale for the y-coordinate of all the distributions). More precisely, all the generated streams have a length of $m = 200,000$ items. We have tested 750 different settings of the following parameters: n, the number of distinct items in the stream, s_1 and s_2, which are related to size of the estimator matrix in Task T_3, and k, the number of counters used in Task T_2. For each setting of parameters, we have conducted 10 trials of the same experiment and compute the average and the standard deviation.

Except from the uniform distribution and the zipf distribution with parameter $\alpha = 1$, that model respectively an ideal stream in which each item appears exactly with the same frequency (*cf.* Figure 1(a)) and a realistic one in absence of any attacks (*cf.* Figure 1(d)), the other four distributions capture different adversarial strategies. More precisely:

- Figure 1(b) shows a distribution modeling streams in which the frequency of a large quantity of items is significantly higher than the frequency of the remaining items. This type of stream might reflect sybil attacks in which the adversary aims at over-representing a large number of node identifiers that it owns.
- Figure 1(e) depicts a distribution modeling streams in which there is a small number of highly frequent items. This type of stream might correspond to an eclipse attack in which the objective of the adversary is to poison the routing tables of honest nodes.
- Figures 1(c) and 1(f) displays distributions modeling streams in which a very small number of items have a very high frequency[2]. These distributions might illustrate streams in which very few items (typically $1, 2$ or 3) are over-pushed by the adversary.

[2]Pascal distribution is also known as Negative Binomial distribution.

Table 1 summarizes the results obtained for the AnKLe, AMS and CCM estimators, averaged over 45,000 experiments (*i.e.* 750 different settings with 10 repetitions for each setting, over 6 distributions). The results clearly show that AnKLe outperforms the estimator CCM for all the distributions, even in scenario in which CCM should excel (*i.e.*, Figure 1(f)), as this corresponds to a stream in which a very frequent item exists in the observed stream. Compared to the AMS estimator, the results obtained with AnKLe are often really better than or sometimes comparable to it for all the distributions, with the exception of the zipf distribution with $\alpha = 2$. But even for this specific distribution, the standard deviation of AnKLe is four times smaller than the one of AMS (*i.e.*, 0.09 versus 0.36), thus demonstrating that AnKLe provides a more robust and stable estimation than AMS on this distribution.

Figure 2 shows the evolution of the KL divergence estimation as a function of n, k, s_1 and s_2. In all the figures, the x-coordinate represents the number of distinct items in the stream as a ratio of its length m. For each value of $n \in \{m/100, \ldots, m/20\}$, all the other parameters k, s_1 et s_2 also vary in the experiments. More precisely, the parameter k takes a value in $\{0.1n, \ldots, n\}$, $s_1 \in \{m/100, \ldots, m/20\}$, and $s_2 \in \{m/100, \ldots, m/20\}$. The main observation that can be drawn from Figure 2(a) is that the CCM estimator behaves relatively badly in presence of a small number of distinct items with frequency uniformly distributed in the stream. However, its accuracy increases when the number of distinct items increases. The other two estimators are very close to the real value of the KL divergence, with moreover a clear advantage for AnKLe. This observation is further confirmed in Figure 2(b) that corresponds to a zoom of Figure 2(a). This figure demonstrates that the estimation provided by AnKLe is very good. In average, the AnKLe estimation overlaps with the real value of the KL divergence, contrary to AMS, and its standard deviation remains small, for any values of n, and for any variations of k, s_1, and s_2. Figure 2(c) and its zoom in (*cf.* Figure 2(d)) further validate the above results. In particular, we observe that CCM is clearly not adapted to uniform and near uniform streams, while AMS and AnKLe provide very good estimates for these distributions. For instance, the zoom in Figure 2(d) shows that these

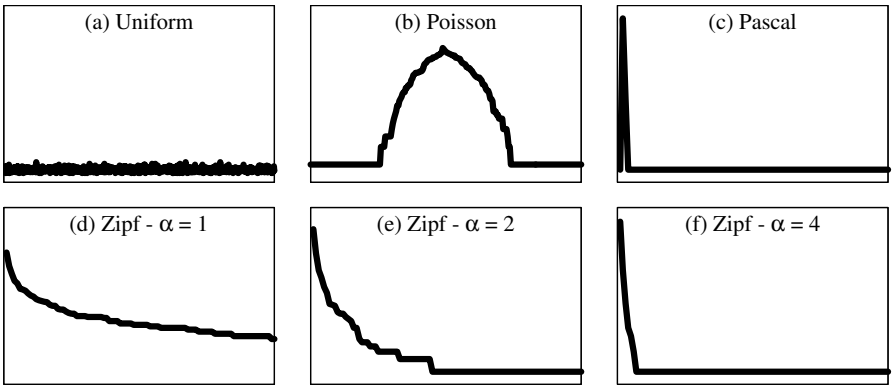


Fig. 1. Shape of distributions used for evaluating estimators. The y-ordinate is logarithmic

Distribution		Exact	AnKLe	AMS	CCM
Uniform	average	0.018240161	0.027314791	0.253219967	-1.161750384
	std. dev.	0.00271478	0.029827495	0.071137525	0.015038305
Zipf – $\alpha = 1$	average	0.825819381	0.688826548	1.055650933	-8.313990878
	std. dev.	0.027970186	0.142217553	0.293984322	0.858649847
Zipf – $\alpha = 2$	average	2.58717975	2.999794044	2.827368288	0.866992924
	std. dev.	0.031286484	0.092712953	0.369015065	0.237650647
Zipf – $\alpha = 4$	average	3.611623614	3.631385192	3.85458675	3.532833916
	std. dev.	0.018752397	0.130210517	0.333661261	0.030785665
Pascal	average	3.40688118	3.357277524	3.650869233	2.148588258
	std. dev.	0.017502656	0.075845977	0.317275996	0.205970693
Poisson	average	0.957558622	0.743131204	1.197167903	-2.089271044
	std. dev.	0.013611449	0.123500666	0.193894289	0.082006954

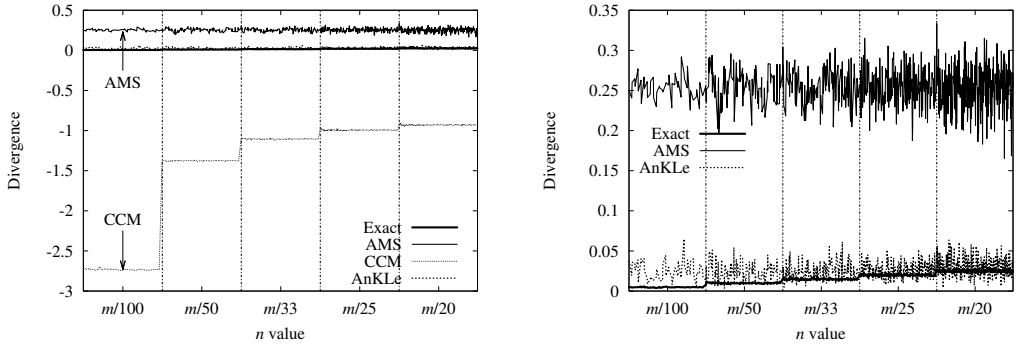
TABLE I
SUMMARY OF PERFORMANCES

two estimators are pretty close to the exact value of the divergence, but still once more AnKLe provides a better robustness to parameters variations. However, Figures 2(e) and 2(f) demonstrate that CCM is more adapted to streams in which a very small fraction of items occur more frequently than the remaining ones. This is clearly shown in Figure 2(e). The estimation of AnKLe in presence of such streams still remains good. In average, AnKLe overlaps with the real value of the KL divergence, but its standard deviation is a little higher than the one of CCM for the Poisson distribution (*cf.*, Table I).

Figures 3(a) and 3(b) show the KL divergence estimation as a function of s_1 and s_2. For each value of s_1, s_2 is increased from $m/5000$ to $m/90$. Several observations can be drawn from both figures. First, the robustness of CCM estimator greatly improves with increasing values of s_1, as the cone-shaped curves converge for $s_1 > m/500$. On one hand, the value towards which the CCM converges under-estimates the KL divergence. Thus, both s_1 and s_2 have a greater impact on CCM robustness than on its accuracy. On the other hand, variations of both s_1 and s_2 have not impact on AMS robustness. This feature does not appear in AnKLe as the weight given to Task T_2 makes it preponderant with respect to Task T_3, limiting accordingly the lack of robustness of Task T_3.

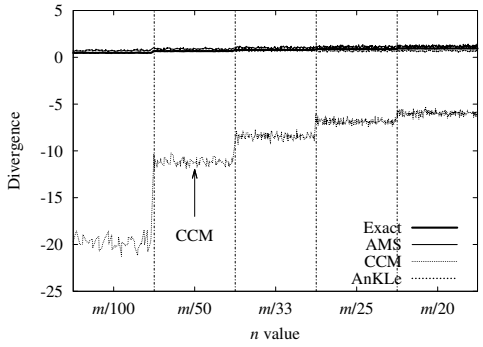
Finally, Figures 4(a) and 4(b) show the KL divergence estimation as a function of k. The main observation drawn from these figures is that AnKLe fully overlaps with the exact value of the KL divergence, which clearly demonstrates the robustness of this estimator in presence of any input streams. Regarding CCM, we can observe that when the number of counters k is less than $0.1n$, then the Misra-Gries algorithm under-estimates the k most frequent items, which degrades the estimation of CCM. This confirms the theoretical bound of $k \geq \lceil 7\varepsilon^{-1} \rceil$ shown in [13]. On the other hand, variations of parameter k has not impact on AMS as this estimator does not decompose its computation according to items frequency characteristics.

To summarize, experiments have validated the impressive accuracy and robustness of AnKLe in presence of a very large spectrum of distributions. This illustrates the importance of the weighting factor applied to both terms of the estimator.

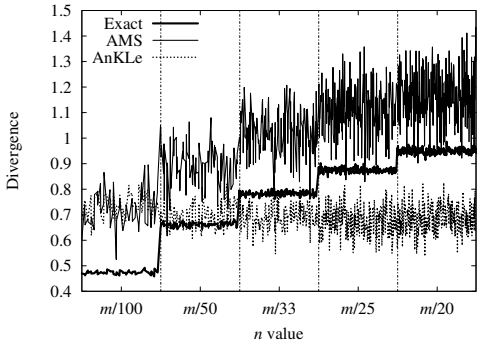


(a) Distribution: Uniform

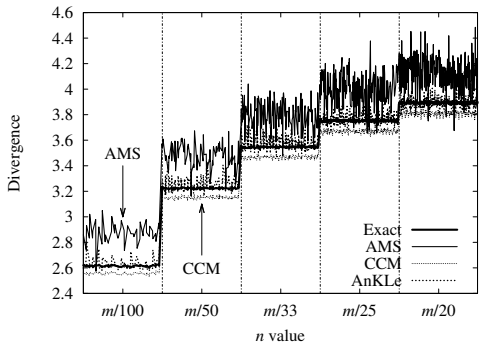
(b) Distribution: Uniform (Closer view)



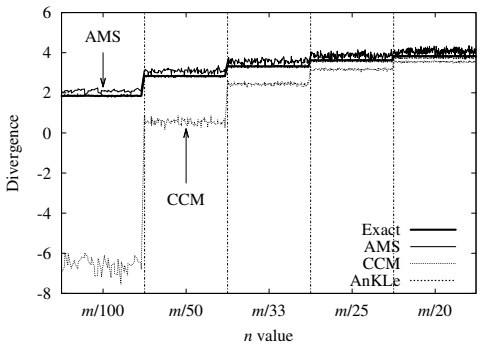
(c) Distribution: Zipf – $\alpha = 1$



(d) Distribution: Zipf – $\alpha = 1$ (Closer view)



(e) Distribution: Zipf – $\alpha = 4$



(f) Distribution: Pascal

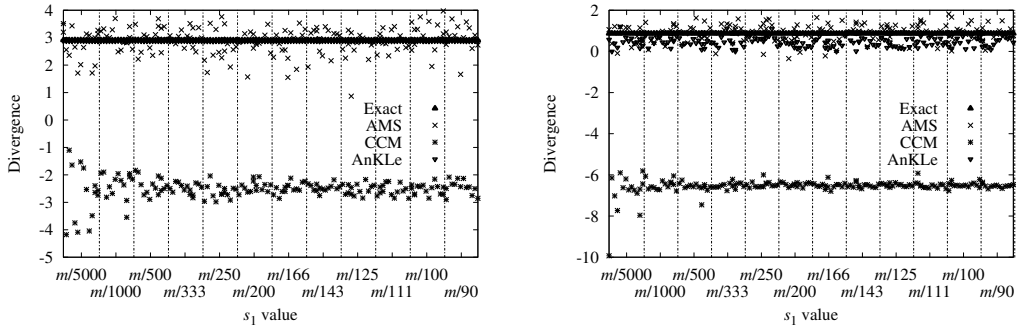
Fig. 2. KL divergence estimation as a function of n, k, s_1 and s_2

VI. Conclusion and future works

In the setting of large scale distributed systems, node receives continuously huge amount of data in the form of a stream that they need to be able to process and analyze on the fly without being able to store the whole stream due to memory constraints. A challenging issue in this setting is to able to detect if the observed stream is conform to the expected one or if it has been tampered with by an adversary. Indeed, an important divergence between the observed stream and the expected one is usually the indication that an attack is being conducted.

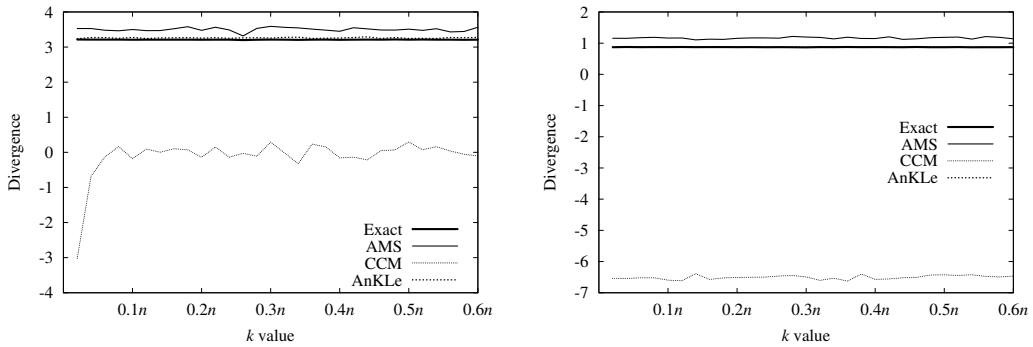
In this paper, we have proposed AnKLe, a novel algorithm for estimating the KL divergence between the observed stream and the uniform one. AnKLe is very efficient both in terms of space and time, and requires only a single pass over the data stream. Simulations also show that AnKLe performs always better, in terms of accuracy and robustness, than other state-of-the-art estimator-based algorithms such as AMS [12] and CCM [13].

We left as future work the exact theoretical analysis of the behavior of the algorithm. In particular, we want to characterize how the different



(a) Pascal Distribution. Settings: $n = m/125$ – $k = 0.1n$ (b) Poisson Distribution. Settings: $n = m/125$ – $k = 0.1n$

Fig. 3. KL divergence estimation as a function of s_1 and s_2



(a) Pascal Distribution. Settings: $n = m/100$ and $s1 = s2 = m/80$ (b) Poisson Distribution. Settings: $n = m/100$ and $s1 = s2 = m/80$

Fig. 4. KL divergence estimation as a function of k

parameters impact the precision of the estimation and the space complexity of AnKLe (and *vice-versa*). Moreover, while currently the length of the stream m is a parameter that has to be fixed in advance, we will design online version of the algorithm for which the length is not specified in advance by using standard windowing techniques. This corresponds to realistic situations in which the nodes regularly receive new data that they need to take into account to update their estimator.

REFERENCES

[1] B. K. Subhabrata, E. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Internet Measurement Conference*, 2003, pp. 234–247.

[2] E. D. Demaine, R. López-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space," in *In Proceedings of the 10th Annual European Symposium on Algorithms*. Springer-Verlag, 2002, pp. 348–360.

[3] J. Bruck, J. Gao, and A. A. Jiang, "Localization and routing in sensor networks by local angle information," *ACM Transaction on Sensor Networks*, vol. 5, pp. 7:1–7:31, February 2009. [Online]. Available: http://doi.acm.org/10.1145/1464420.1464427

[4] Y. Busnel, M. Bertier, and A.-M. Kermarrec, "SOLIST or How To Look For a Needle in a Haystack?" in *the 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2008)*, Avignon, France, October 2008.

[5] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 293–313, 2002.

[6] Y. Busnel, R. Beraldi, and R. Baldoni, "On the uniformity of peer sampling based on view shuffling," *Elsevier Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1165–1176, August 2011.

[7] M. Bertier, Y. Busnel, and A.-M. Kermarrec, "On Gossip and Populations," in *Proceedings of the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, 2009.

[8] E. Anceaume, Y. Busnel, and S. Gambs, "Uniform and Ergodic Sampling in Unstructured Peer-to-Peer Systems with Malicious Nodes," in *Proceedings of the 14th international conference on Principles of distributed systems (OPODIS)*, vol. 6490, 2010, pp. 64–78.

[9] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[10] T. Cover and J. Thomas, "Elements of information theory," *Wiley New York*, 1991.

[11] A. Chakrabarti, K. D. Ba, and S. Muthukrishnan, "Estimating entropy and entropy norm on data streams,"

in *In Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS)*. Springer, 2006.

[12] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (STOC)*, 1996, pp. 20–29.

[13] A. Chakrabarti, G. Cormode, and A. McGregor, "A near-optimal algorithm for computing the entropy of a stream," in *In ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 328–335.

[14] S. Guha, A. McGregor, and S. Venkatasubramanian, "Streaming and sublinear approximation of entropy and information distances," in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006, pp. 733–742.

[15] A. Lall, V. Sekar, M. Ogihara, J. Xu, and H. Zhang, "Data streaming algorithms for estimating entropy of network traffic," in *Proceedings of the joint international conference on Measurement and modeling of computer systems (SIGMETRICS)*. ACM, 2006.

[16] C. Cachin, "An information-theoretic model for steganography," *Information and Computation*, vol. 192, no. 1, pp. 41–56, 2004.

[17] E. Anceaume, Y. Busnel, and S. Gambs, "Characterizing the adversarial power in uniform and ergodic node sampling," in *Proceedings of the 1st International Workshop on Algorithms and Models for Distributed Event Processing (AlMoDEP)*. ACM, 2011.

[18] Muthukrishnan, *Data Streams: Algorithms and Applications*. Now Publishers Inc., 2005.

[19] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Proc. for the 1st Int'l Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[20] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951. [Online]. Available: http://dx.doi.org/10.2307/2236703

[21] S. M. Ali and S. D. Silvey, "General Class of Coefficients of Divergence of One Distribution from Another," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 28, no. 1, pp. 131–142, 1966.

[22] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques (RANDOM)*. Springer-Verlag, 2002, pp. 1–10.

[23] J. Misra and D. Gries, "Finding repeated elements," *Science of Computer Programming*, vol. 2, no. 2, pp. 143–152, 1982.

[24] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, 1985.

[25] P. Gibbons, *Data Streams Management: Processing High-Speed Data Streams*. Elsevier, 2007, ch. Distinct-Values Estimation over Data Streams.

[26] P. B. Gibbons and S. Tirthapura, "Estimating simple functions on the union of data streams," in *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2001, pp. 281–291.