Share a pie? Privacy-Preserving Knowledge Base Export through Count-min Sketches

Daniele Ucci, Leonardo Aniello and Roberto Baldoni
Research Center of Cyber Intelligence and Information Security (CIS)
Department of Computer, Control, and Management Engineering “Antonio Ruberti”
“La Sapienza” University of Rome, Italy
{ucci, aniello, baldoni}@dis.uniroma1.it

ABSTRACT

Knowledge base (KB) sharing among parties has been proven to be beneficial in several scenarios. However such sharing can arise considerable privacy concerns depending on the sensitivity of the information stored in each party's KB.

In this paper, we focus on the problem of exporting a (part of a) KB of a party towards a receiving one. We introduce a novel solution that enables parties to export data in a privacy-preserving fashion, based on a probabilistic data structure, namely the *count-min sketch*. With this data structure, KBs can be exported in the form of key-value stores and inserted into a set of count-min sketches, where keys can be sensitive and values are counters. Count-min sketches can be tuned to achieve a given key collision probability, which enables a party to deny having certain keys in its own KB, and thus to preserve its privacy. We also introduce a metric, the γ-*deniability* (novel for count-min sketches), to measure the privacy level obtainable with a count-min sketch. Furthermore, since the value associated to a key can expose to linkage attacks, noise can be added to a count-min sketch to ensure controlled error on retrieved values. Key collisions and noise alter the values contained in the exported KB, and can affect negatively the accuracy of a computation performed on the exported KB. We explore the tradeoff between privacy preservation and computation accuracy by experimental evaluations in two scenarios related to malware detection.

Keywords

Information sharing; knowledge base export; privacy metric; count-min sketches.

1. INTRODUCTION

Several well-known best practices show that setting up an information sharing environment, involving several parties, represents one of the main building blocks to face cyber attacks. This is why also policy makers are fostering the creation of organizations that share information within specific sectors or geographical regions (e.g., ISACs[1] and ISAOs[2]). A sharing environment includes organizations, policies and technical aspects. From the technical point of view, cyber information sharing basically allows a party to reason on a larger set of information to support its own defenses.

One of the main problems in handling information sharing environments is the different readiness of the parties. In general, distinct parties store different volumes of shareable added-value information. Indeed, there could be parties that, locally, collected huge amounts of data whose sharing can add value to others' defenses. On the other hand, there could be parties not owning relevant data to share. Collecting locally a quantity of added-value data, sufficiently big to support better defenses, can take very long (e.g., order of months). Having at disposal large amounts of data is the immediate advantage enabled by cyber information sharing, which is a key incentive for less-ready parties to join such an environment. On the long run, more-ready parties that initially shared their data will benefit from added-value information shared by other parties.

As an example, consider a malware detection system that needs to collect a huge amount of information from a monitored network before being able to accurately detect malicious samples (e.g., AMICO [26], Nazca [10], and [14]). During such bootstrapping period, an organization that deployed the system would be vulnerable to cyber attacks. More-ready parties using the same system could export (portions of) their KBs, providing to less-ready parties the information the malware detection system needs. However, exporting KBs may arise privacy issues when sensitive data are involved, indeed the receiving party may be malicious and take advantage from the shared information to the detriment of the exporting party.

In this paper, we address the following specific issue: exporting added-value information from one party to another one so that, while the privacy of the exporting party is preserved, the receiving one can immediately leverage this information to improve its defenses. Even though, in practice, exports can be periodically performed to exchange updated added-value information among the parties, in the paper we assume that the KB export is performed only once (see later in this section).

[1]Information Sharing and Analysis Center (ISAC), Presidential Decision Directive: https://fas.org/irp/offdocs/pdd/pdd-63.htm

[2]Information Sharing and Analysis Organizations (ISAO), Executive Order: https://www.whitehouse.gov/the-press-office/2015/02/13/executive-order-promoting-private-sector-cybersecurity-information-shari

CODASPY'17, March 22-24, 2017, Scottsdale, AZ, USA
© 2017 ACM. ISBN 978-1-4503-4523-1/17/03...$15.00
DOI: http://dx.doi.org/10.1145/3029806.3029817

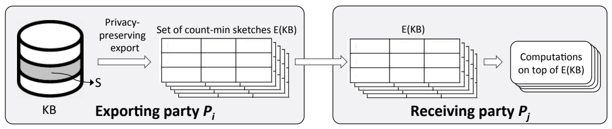
Privacy-preserving export
Set of count-min sketches E(KB)
S
KB
Exporting party P_i
E(KB)
Computations on top of E(KB)
Receiving party P_j

Figure 1: Privacy-preserving export, through a set of count-min sketches E(KB), of a KB portion (S, in gray) of a party P_i towards a possibly malicious receiving party P_j.

Aim of the paper. We introduce a novel solution for exporting KBs in a privacy-preserving fashion, based on a probabilistic data structure, namely the *count-min sketch*. This data structure allows to share KBs in the form of key-value stores, where the value associated to a key is a numeric counter. Keys can be generic strings, and their names are considered sensitive information (see Figure 1). We assume that the receiving party can be malicious and interested in breaching the privacy of the party exporting the KB. Once shared, a count-min sketch can be queried by KB's receiver to get the values associated to some keys chosen by the receiver itself. The receiver doesn't know what keys are stored in the count-min sketch, so it cannot enumerate all the key-value pairs that the owner shared. Furthermore, a count-min sketch can be tuned in such a way to have a certain key collision probability. From the owner's point of view, this means it can deny to a certain extent to have inserted a given key, which represents a statistically sound mean to preserve owner's privacy.

γ-deniability and linkage attacks. We introduce a novel metric to measure this extent, by taking inspiration from the work of Bianchi et al. [2] who applied a similar concept to bloom filters. This metric is called γ-deniability and represents the probability that any inserted key collides with some other keys (in the universe of keys known to the party exporting the KB) that have not been inserted, which in practice provides a statistical base to deny having inserted such key. We provide an analytical formula to calculate the γ-deniability of a count-min sketch configured in a certain way given the cardinalities of inserted keys and of the universe.

Likewise k-anonymity, γ-deniability is vulnerable to linkage attacks that can be used to discover what keys have been actually stored in a count-min sketch. Usually, the KB to export includes several attribute-value pairs for a given key, and a count-min sketch for each attribute is used, with all the values of this attribute for each key. For example, if we want to export a KB of physical traits (hair, eyes, height, etc...) of persons identified by their social security number (i.e., the key), then we can use a count-min sketch for each considered trait t, where for each person with social security number x and y as value of trait t, a pair $\langle x, y\rangle$ is put inside. If an attacker wanted to find out whether some key k has been really inserted in the shared KB, he would query with k all the given count-min sketches to obtain all the related attribute-value pairs. By combining obtained attribute-value pairs with some background knowledge about k, it would be possible to gain knowledge about the real key of the original KB to which these values refer to. With reference to the previous example about persons and traits, by combining the values of the traits stored in the count-min sketches for certain keys, it is possible to obtain a physical profile of the persons that are included in the original KB. By "linking" these profiles with some background knowledge about physical profiles of known persons, it would be possible to infer whether any of these known persons is contained in the original KB. This would be especially true for persons having rarer profiles. Linkage attacks can be mitigated by adding noise to the values inserted in the count-min sketches, so that it becomes more difficult for an attacker to link the knowledge extracted from the shared KB with some external information.

If portions of the same KB are published in different releases, an attacker can potentially correlate them without needing any additional background knowledge. There exist attacks, namely *complementary release* and *temporal attacks*, which take advantage of possible linkages between different releases published in diverse times. Solutions to these types of attacks have already been proposed in [9]. In our case, complementary release attacks can be avoided by exporting, every time, keys that have no correlation among them. Temporal attacks can be prevented by always exporting releases containing keys stored in previous exports. This is why, in this paper, we don't specifically address this kind of attacks that leverage different exports over time of (part of) a same KB, rather we assume that the export is done only once.

The stricter are the privacy requirements of the owner, the higher should be the needed γ-deniability and the amount of noise in the exported KB. From the receiver's point of view, this affects the reliability of received KB, which in turn may impact its utility depending on the specific scenario and on what type of computation it has to execute on that KB. Deriving general results on the impact of γ-deniability and noise on computation soundness and accuracy is problematic because of the difficulty to generalize computations. We instead carried out experimental evaluations on two distinct scenarios to explore in practice this relationship.

Experimental scenarios. We present two scenarios, both related to malware detection. The first scenario concerns the malware detection algorithm implemented by AMICO system [26], which classifies samples using features based on sample download patterns. AMICO needs a large historical dataset of downloads for its bootstrap, which usually requires from one to three months to be collected. Sensitive keys in this case are information like visited websites and downloaded files. We implemented and evaluated an export of the AMICO KB based on count-min sketches, such that the receiving party can not infer which websites have ever visited the exporting party. The evaluation consisted in investigating the tradeoff between the γ-deniability of count-min sketches and the malware detection accuracy at the receiving side.

In the second scenario, a KB containing behavioral features of samples is exported, which enables an organization with such KB to execute local malware detection/analysis algorithms without having to actually execute the samples. The computation executed on the exported KB is the classification in malware/non-malware based on an already trained classifier. An attacker may extract further information on whether a certain malware m has been actually put in the shared KB by leveraging background knowledge regarding m's behavior. To cope with this kind of linkage attacks based on background knowledge, the KB's owner can add noise to the count-min sketches. The goal in this case was tuning count-min sketches to have a fixed γ-deniability and a variable amount of noise, and the evaluation consisted in showing in practice the tradeoff between the error in retrieved values and the malware detection accuracy.

Obtained results for the first scenario show that introducing deniability in the count-min sketches can impact the utility of exported KB, as higher deniability allows to export less data. When count-min sketches are queried with keys that have not been inserted, false positives can occur and mess the computation up. In this evaluation, false positive rate turned out to be independent from the required deniability. In the end, depending on the application scenario and in particular on what the KB has to be used for, the employment of count-min sketches, configured to ensure deniability guarantees, can be a viable option. The outcomes of the evaluations for the second scenario confirms the expected tradeoff between utility and privacy, as higher errors in retrieved values negatively affect the computation while complicating attacks based on background knowledge.

Paper Contributions and Organization. The main contributions of this paper are: (i) the definition of a deniability metric to measure the privacy level provided by a count-min sketch, (ii) a methodology to configure count-min sketches to obtain desired levels of deniability on keys and of noise on values, and (iii) an experimental evaluation in two scenarios showing how deniability and noise affect computation accuracy.
The rest of the paper is structured in this way. Section 2 introduces count-min sketches, how they work, and some basic properties that will be used later on. The definitions of deniability and noise for count-min sketches, together with the algorithms for configuring count-min sketches to obtain them, are presented in Section 3. Section 4 describes the two scenarios chosen for the evaluations. After a discussion on related work (Section 5) and on current limitations and open points (Section 6), conclusions and possible future work are presented in Section 7.

2. BACKGROUND

2.1 Count-min sketches

Count-min sketches are probabilistic data structures initially conceived for summarizing data streams [3].

A count-min sketch M contains key-value pairs of the universe set $\mathcal{U}$, where keys belong to the set $K_{\mathcal{U}}$ and values are in $\mathbb{R}_+ \cup \{0\}$. M is represented as a matrix of non-negative values, whose width w and depth d are defined by parameters ε and δ as follows:

$$w = \left\lceil \frac{e^3}{\varepsilon} \right\rceil \qquad d = \left\lceil \ln \frac{1}{\delta} \right\rceil \tag{1}$$

Parameters ε and δ regulate the guarantees on the accuracy of the values retrieved from M, which will be detailed in Equation 4 right after having defined the semantics of count-min sketch operations. With $M[i, j]$ we refer to the value of M stored in the cell at row i and column j, where $i = 1, \ldots, w$ and $j = 1, \ldots, d$. At the beginning, each cell has value 0.
Additionally, d hash functions $h_1, \ldots, h_d : K_{\mathcal{U}} \to \{1, \ldots, w\}$ are chosen uniformly at random from a pairwise-independent family. Pairwise-independence, also known as strong universality [13], implies low collision probabilities among distinct hash functions, which means this property holds [27]:

$$\Pr\left[h_i(x) = h_i(y)\right] = \frac{1}{w} \qquad \forall i \in \{1, \ldots, d\} \wedge \forall x, y \in K_{\mathcal{U}} \tag{2}$$

These hash functions are used to map any element of $K_{\mathcal{U}}$ to exactly d distinct cells of M, one for each row.

In this work we are interested in using the operations *point query* and *update* of count-min sketches. An *update* operation $U_M(k, v)$ updates the value associated to k in M by adding v to the values currently stored in all the cells where k is mapped to. It is defined as

$$M[i, h_i(k)] \leftarrow M[i, h_i(k)] + v \qquad \forall i \in \{1, \ldots, d\} \tag{3}$$

Since collisions are possible, an update can alter cells where also other keys are mapped to. As these values are non-negative, the effect of collisions is to make some of stored values larger than they should, so the *point query* operation $Q_M(k)$ use the following approximation to retrieve the value v associated to k: $\widehat{v} = \min_{i \in \{1,\ldots,d\}} M[i, h_i(k)]$

While count-min sketches have been originally conceived to keep and update counters, in this work we want to use them to export a KB in the form of key-value stores, so we are not actually interested in updating the values of already stored keys, rather we only need to store values once for each key we need to insert. Let S be the set of key-value pairs $\langle k, v \rangle$ to insert into a count-min sketch M. Once all the $n = |S|$ pairs have been inserted in M through as many update operations, we refer to M as $M(S)$. The value $\widehat{v}$ returned by a point query $Q_{M(S)}(k)$ provides the following guarantees:

$$\begin{cases} v \leq \widehat{v} \\ \widehat{v} \leq v + \epsilon \cdot \|S\|_1 \text{ with probability } (1 - \delta) \end{cases} \tag{4}$$

where $\|S\|_1 = \sum_{\langle k,v \rangle \in S} v$[4]. Thus, count-min sketches never underestimate real values and such estimation is bounded with probability $(1 - \delta)$ by $v + \epsilon \cdot \|S\|_1$.

By construction, count-min sketches have an implicit representation, meaning that, conversely to cleartext data, information contained in the sketch is readable only by issuing queries.

2.2 Hiding set of a count-min sketch

Let $K_S = \{k | \langle k, v \rangle \in S\}$ and $K_{\mathcal{U}} = \{k | \langle k, v \rangle \in \mathcal{U}\}$ be the sets of keys of the elements in S and $\mathcal{U}$, respectively. The hiding set for a count-min sketch $M(S)$ contains all the keys that have not been inserted in $M(S)$, but would result as inserted if $M(S)$ was queried on them.

Definition 2.1. *A set $\mathcal{V}$ is called* hiding set *for a count-min sketch $M(S)$ if $\mathcal{V}$ contains all the elements $k_i \in K_{\mathcal{U}} \setminus K_S$ s.t. $Q_{M(S)}(k_i) > 0$ (i.e., k_i is a false positive).*

Given a count-min sketch $M(S)$ with width w, depth d, and n inserted keys out of u universe keys, the probability $\psi(w, d, n)$ that a query $Q_{M(S)}(k)$ would return a false positive can be computed by observing that a false positive occurs when all the cells where k is mapped to have a value greater than zero. The event that, in a given row, k hits an

[3] e is the Euler's number.

[4] Since all inserted values are equal to or greater than 0, modulus can be omitted from the L_1 norm definition.

already covered cell is independent from and equiprobable to the event that this happens in any other row. In a single row, the probability that k gets mapped to one of the cells where any of the previously n keys have been mapped is $1 - \left(1 - \frac{1}{w}\right)^n$. The false positive probability can be computed as follows:

$$\psi(w, d, n) = \left(1 - \left(1 - \frac{1}{w}\right)^n\right)^d \quad (5)$$

The cardinality v of the hiding set $\mathcal{V}$ is random variable N_v with binomial probability distribution

$$\Pr[N_v = v] = \binom{u-n}{v} \psi(w, d, n)^v \left(1 - \psi(w, d, n)\right)^{u-n-v} \quad (6)$$

and mean value

$$E[N_u] = (u - n)\psi(w, d, n) \quad (7)$$

3. PRIVACY PRESERVATION WITH COUNT-MIN SKETCHES

The basic idea of using count-min sketches to provide privacy guarantees lies in leveraging key collisions for (i) denying to have inserted a key by blaming a false positive (collisions with non-inserted keys), and for (ii) adding a controlled amount of error to retrieved values (collisions with inserted keys).

The first point shares similarities with the approaches based on *k-anonymity* [24]: a key is considered k-anonymous if it cannot be distinguished from at least other $k - 1$ keys because of collisions. Assuming that such property holds, an attacker who executes a point query $Q_{M(S)}(k)$, and obtains a value $v > 0$, cannot assert for sure that the key k has been inserted in $M(S)$. Indeed, who created the count-min sketch can attest that the cells of $M(S)$ where the key k is mapped are also mapped by some other keys in the universe set, so the value $v > 0$ returned by the point query for k cannot be used by someone else as a proof that k has been inserted in $M(S)$. Such uncertainty enables who creates count-min sketches to deny having inserted specific keys, and by consequence to deny having any information at all related to specific keys.

Likewise k-anonymity, this approach is weak against attacks exploiting background knowledge [24]: a value retrieved from a count-min sketch for a given key k (or more values retrieved from distinct count-min sketches for a same given key k) can be correlated with external information about k to infer whether k is known to who created the count-min sketch(es). This weakness can be addressed by resorting to *differential privacy* [5], which adds controlled amount of noise to stored values according to some distribution, usually Laplacian. Such noise has to be bound to limit the impact on the utility of the values retrieved. Differential privacy has been shown to be vulnerable to tracker style attacks [6, 22], indeed when an unlimited number of queries can be performed, which is exactly our case, the knowledge that the attacker can gain increases exponentially with the number of queries. We address this limitation by exploiting collisions among inserted keys to add controlled amount of noise. According to Equation 4, who creates the count-min sketch can control how much noise to add by properly tuning its size. Actual collisions depend on what keys are inserted, which doesn't follow any known distribution and thus cannot be exploited by an attacker.

Setting count-min sketch dimensions is the mean to configure privacy preserving guarantees. In general, the wider and deeper the sketch, the less collisions are likely to occur, with both inserted and non-inserted keys. In this section we first define more formally system model and attacker model (Section 3.1), then we delve into how to properly dimension count-min sketches to have certain guarantees on the noise added to values (Section 3.2), and finally we introduce deniability metrics to measure privacy levels on exported keys (Section 3.3). We describe noise addition beforehand to introduce the iterative approach to tune count-min sketch width, which will be then used to better explain the simulation results obtained for the deniability metric.

3.1 System and Attacker Models

The owner O of a KB $\mathcal{U} = \{\langle k_i, v_i\rangle | i = 1, \ldots, u \wedge k_i = k_j \Rightarrow i = j\}$ creates a count-min sketch M with parameters ε and δ, having width w and depth d computed according to Equation 1. M is populated with a subset S of the universe set $\mathcal{U}$. Let $K_S = \{k_i | \langle k_i, v_i\rangle \in S\}$ be the set of the keys inserted into $M(S)$, with $|K_S| = n$. Let $V_S = \sum_{i=1}^{n} v_i$ be the sum of the values inserted into $M(S)$. Let $K_{\mathcal{U}} = \{k_i | \langle k_i, v_i\rangle \in \mathcal{U}\}$ be the set of keys in the universe set, with $|K_{\mathcal{U}}| = u$. We assume that each key of $\mathcal{S}$ and $\mathcal{U}$ is associated with exactly one value. Hence, their cardinalities coincide with the ones of their correspondent key sets (i.e. $|\mathcal{S}| = |K_{\mathcal{S}}| = n$ and $\mathcal{U} = |K_{\mathcal{U}}| = u$).

O shares $M(S)$ with a potential attacker A, who aims at obtaining information about the real content of $\mathcal{U}$ by executing on $M(S)$ as many queries she wants. In particular, A can read the values of any cell of $M(S)$, and knows all the hash functions $h_1, \ldots, h_d$, so she can derive what are the cells a given key is mapped to. A knows neither the elements in S nor its cardinality. O doesn't export over time different releases of M with A, rather O creates $M(S)$ once and sends it to A.

3.2 Noise Addition

Depending on the particular scenario, there can be different requirements on the amount of noise to add. The error is guaranteed with probability $1-\delta$ to be within a fraction ε of V_S, according to Equation 4 and to the definition of V_S reported in Section 3.1. As the value of V_S may be very large, ε may need to be set very small to limit the error, which would imply to make the count-min sketch quite large (see Equation 1). Since calculating the L1 norm of V_S requires to iterate over all the n elements, computing the width in this way has $\mathcal{O}(n)$ time complexity.

If requirements specify to have guarantees on the error that are certain and not probabilistic, an alternative way can be taken to configure count-min sketch size, which however is much less time-efficient. Trivially, the real error on the values retrieved from a count-min sketch $M(S)$ can be computed by querying $M(S)$ with all the keys in K_S, and then comparing obtained results with the real values associated to these keys in S. Intuitively, by fixing the depth, the larger is the width the less collisions are likely to occur, and by consequence errors on retrieved values are lower (see Figure 2). Obviously, shrinking the width makes errors more probable. By proceeding iteratively, different widths

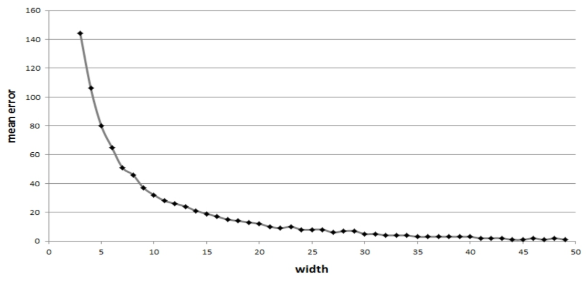

Figure 2: Mean error between real values and those retrieved from a count-min sketch, as its width varies. For each distinct value of the width, 100 simulations have been carried out by generating synthetic S with $n = 1000$. The depth was fixed to 5.

can be tested to find the optimal one, which is the minimum width guaranteeing with certainty that any retrieved value has an error within a specific bound. For each test, the count-min sketch has to be populated with all the elements in S and then queried with all the keys in K_S, which costs $\mathcal{O}(n)$[5]. Contrary to what is shown in Figure 2, where each marker represents an average error over a large number of experiments with different sets S, for a specific set S there are no guarantees that the mean error decreases monotonically as the width increases, but it depends on the collisions among the keys. This is why it is necessary to start with a small width and then keep increasing it by one until the constraint on error upper bound is met. Using an asymptotic worst-case cost model, the number of different widths to test is $\mathcal{O}(n)$, which makes the overall approach have time complexity $\mathcal{O}(n^2)$.

We show through simulations how the two approaches differ for what concerns the width they compute. Figure 3 shows that the required width decreases for both as the constraint on the upper bound becomes weaker. Indeed, as ε increases, tolerated error is higher and less wide count-min sketches are needed. Moreover, it is clear that the width derived iteratively is always lower than the width computed with Equation 1, and the difference between the two grows as the constraint on error upper bound becomes stricter. As will be shown in Section 3.3, a lower width may be preferable because it allows higher deniability.

For what concerns the role of the depth in the iterative approach, it can be observed that a larger depth implies a lower collision probability among inserted elements, which allows to obtain the required certain error upper bound with a lower width. Figure 4 gives evidence of this aspect by showing the width required to have error 0 with certainty as the number of inserted elements varies with respect to the universe set cardinality, for different values of depth. In conclusion, adding controlled amount of noise to the values of a count-min sketch can be done in two ways: (i) *probabilistically*, by computing w and d according to Equation 1, which is faster ($\mathcal{O}(n)$) but likely to generate very wide count-min sketches, and (ii) *iteratively*, by explicitly finding the minimum width that satisfies the constraint on the error, which is time-consuming ($\mathcal{O}(n^2)$) but ensures lower width.

[5]Since usually $n \gg d$, to keep notation as simple as possible we assume that both point query and update operations cost $\mathcal{O}(1)$, even though their actual time complexity is $\mathcal{O}(d)$

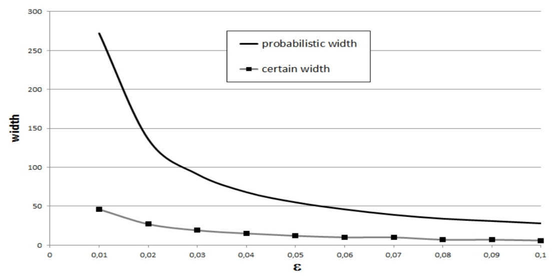

Figure 3: Comparison between the width computed using Equation 1 and the minimum width that provides the required guarantees on error upper bound, which depends on ε. For each distinct value of ε, 100 simulations have been carried out by generating synthetic S with $n = 1000$. The depth was fixed to 5.

3.3 Deniability

The basic idea behind the concept of *deniability* is providing probabilistic coverage for the cells where inserted elements are mapped, with elements that have not been inserted. If O created a count-min sketch where each element in K_S is mapped to cells that can be covered with elements in $\mathcal{K_U} \setminus K_S$, then O would have solid arguments to deny having inserted any key $k \in K_S$. To define the concept of deniability and prove some probabilistic results, we take an approach very similar to the one used in [2], where the deniability is applied to bloom filters.

Definition 3.1. *A key $k \in K_S$ inserted in a count-min sketch $M(S)$ is* deniable *if $\forall i = 1, \ldots, d$, there exist at least one key $z \in \mathcal{V}$ s.t. $h_i(k) = h_i(z)$.*

Definition 3.2. *A count-min sketch $M(S)$ is γ-*deniable *if a randomly chosen key $k \in K_S$ is deniable with probability γ. In this case, $M(S)$ is said to have γ-deniability γ.*

It is important to note that this γ-deniability definition imposes that deniable keys have to be covered by keys in the hiding set, and not by other keys in S, otherwise the count-min sketch as a whole could not be deniable.

We provide an expression to compute the γ-deniability of a count-min sketch.

Theorem 3.1. *The γ-deniability of a count-min sketch $M(S)$ having width w, depth d, $|S| = n$, and $|\mathcal{U}| = u$ can be computed exactly as*

$$\gamma\big(M(S)\big) = \left(\sum_{b=1}^{w} U_w(n;b) \sum_{v=0}^{u-n} \binom{u-n}{v} \left(\frac{b}{w}\right)^v \left(1 - \frac{b}{w}\right)^{u-n-v} \sum_{r=0}^{b} U_b(v;r) \frac{r}{b} \right)^d$$

and can be approximated in closed form as

$$\gamma\big(M(S)\big) \approx \left(1 - \left(1 - \frac{1}{wp}\right)^{(u-n)p}\right)^d \qquad (8)$$

where $p = 1 - \left(1 - \frac{1}{w}\right)^n$

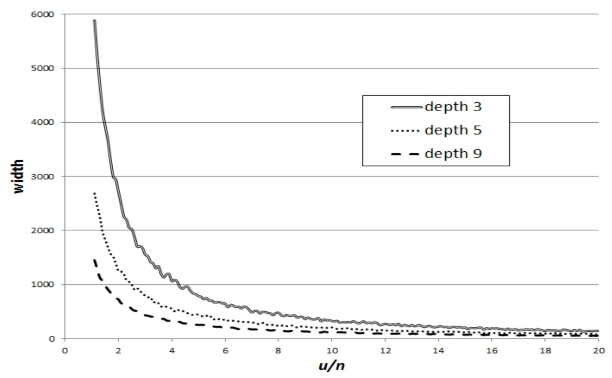

Figure 4: Comparison on how the width (computed with the iterative approach) changes by varying the ratio between the cardinality u of the universe set $\mathcal{U}$ (fixed to 1000 elements) and the cardinality n of the set S of inserted elements. The width has been computed iteratively to obtain 0 as certain error upper bound. Given a pair $\langle u/n, depth \rangle$, the value of the width reported in the graph has been computed as the average width over 100 experiments.

Proof. See Appendix A.

We carried out simulations to validate our model, where we compared the deniability obtained using Equation 8 with the real deniability of count-min sketches populated with synthetically generated elements. We fixed the cardinality u of the universe set to 1000 and varied the number n of inserted elements. We tuned the width w of count-min sketches so as to achieve 0 error using the iterative approach, the one which guarantees certainty on the error upper bound (see Section 3.2). For the depth d, we used values 3, 5, and 9. For each tuple $\langle u, n, w, d \rangle$ we ran 1000 experiments and took the average deniability. For each experiment we derived the actual deniability by counting how many keys out of the n inserted were deniable, according to Definition 3.1.

The results are reported in Figure 5. The average error between theoretic and real deniability is very small: 0.92% for depth 3, 0.57% for depth 5, and 0.36% for depth 9. In general, the deniability increases as the number of inserted elements is reduced, but the trend is not strictly monotonically increasing. Intuitively, the lower is n the greater is the cardinality of the candidate hiding set $K_{\mathcal{U}} \setminus K_S$, and thus there are more elements to choose from to deny any inserted element. The fact that the deniability grows with the depth actually depends on the width: a larger depth implies a lower collision probability among inserted elements, which allows to obtain certain 0 error upper bound with a lower width. In turn, a lower width guarantees more potential collisions with elements in $K_{\mathcal{U}} \setminus K_S$, which makes the deniability increase.

3.4 Count-min sketch tuning

We envision a scenario where a subset of a KB has to be exported, and count-min sketches have to be configured to provide guarantees on the maximum error of retrieved values and on the minimum deniability, and at the same time with the goal of maximizing the number of exported elements. We propose a heuristic to maximize n and compute w for each count-min sketch, such that

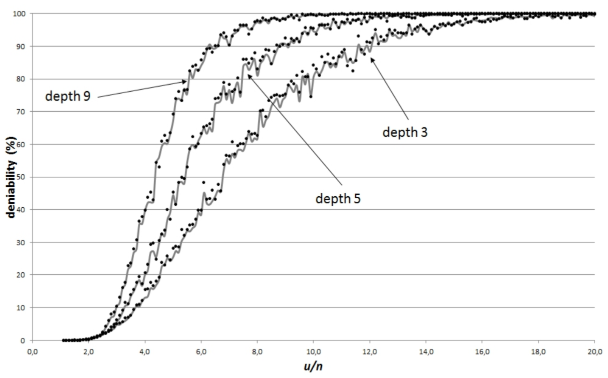

Figure 5: Comparison between the deniability computed with Equation 8 (gray lines) and the real deniability obtained through simulation (black markers), by varying the ratio between the cardinality u of the universe set $\mathcal{U}$ (fixed to 1000 elements) and the cardinality n of the set S of inserted elements.

$$\begin{cases} \gamma\big(M(S)\big) \geq \gamma_{min} \\ \forall \langle k, v \rangle \in S, \quad |v - Q_M(k)| \leq err_{max} \end{cases} \tag{9}$$

The basic idea is to leverage the fact that the deniability almost grows as n is decreased, so an approximation of the optimum n can be obtained by performing a sort of binary search over n in the range $[0, u]$ to find the largest n satisfying the constraint on the γ-deniability. For each value of n to test, we compute the minimum w required to meet the requirement of the maximum allowed error on retrieved value. As the search goes on, we keep track of the best result got so far, which is the maximum n and correspondent w that are compliant with Equation 9. Once the search is over, the best result we found becomes the output of such heuristic. The time complexity is $\mathcal{O}(n^2 \log n)$.

We don't provide exact heuristics to choose the depth, rather we describe guidelines. A larger depth makes it easier to ensure the compliance with the conditions reported in Equation 9. On the other hand, the size of count-min sketches grows linearly with the depth. We leave as future work a more thorough investigation on the tuning of count-min sketch depth.

4. EXPERIMENTAL EVALUATION

This section describes experimental evaluations carried out in application scenarios related to malware detection, where machine learning techniques are used to classify samples in benign or malicious. Both scenarios need some form of KB to construct the feature vectors to be given as input to the classifiers, and obtaining such KB is time-consuming, so export turns out to be advantageous for who would like to begin using these malware detection techniques. For each scenario, we describe the content of such KB and why it takes time and effort to procure, how to export it through properly configured count-min sketches, and to what extent the γ-deniability and error upper bound properties affect the utility of the exported KB itself, measured by using the malware detection accuracy as main comparison metric.

4.1 Malware Detection based on Download Patterns

AMICO [26] is a system for malware detection which classifies samples using features on how they have been downloaded. Downloads are captured at the edge of the monitored network, and a random forest algorithm is used for the classification, trained on a set of labeled samples.

The features of interest include several information on the downloads occurred in the monitored network: past file downloads (i.e., how many clients downloaded a specific file), domain features (e.g., how many malware have been downloaded from a given domain), server IP features (e.g., how many benign files have come so far from some IP), URL features (number of files sharing a same URL and URL structure). This download history KB is required to properly create the feature vectors to be fed to the classifier, and a bootstrap period in the order of months is needed to collect enough statistics to start classifying samples [26]. Sharing such KB, along with the already trained classifier, can enable others to start using the malware detector without waiting months. While the classifier can be shared without incurring in privacy concerns, the KB can be shared using count-min sketches, by preparing a count-min sketch for each feature based on past download events. Each feature can be seen as key-value store. As an example, the feature *domain_malware_downloads* can be seen as a set of pairs $\langle d, m \rangle$, each representing the fact that m malware have been downloaded from domain d. Since distinct count-min sketches represent different aspects of download history, they will have distinct key sets with distinct cardinalities, which in general are different from the number of downloads taken in account to populate the KB.

4.1.1 Count-min sketch configuration

Some count-min sketches have keys that can be considered sensitive. For example, the owner of the KB may have concerns in disclosing what domains have been visited by its network. In this scenario, we export different subsets of the whole KB according to different γ-deniability constraints (25%, 50%, 75%, 100%), and show how the classification is affected. A γ-deniability constraint means that all the count-min sketches have to have a deniability $\geq \gamma$. To show the impact of the deniability only, we configured the count-min sketches to have error upper bound equal to 0. Furthermore, we imposed the additional constraint that what we export has to be consistent with respect to the downloads: we populated all the count-min sketches with data coming from the same downloaded samples, to avoid that any two distinct count-min sketches would contain key-value pairs extracted from two different sets of downloads.

We fixed the depth to 11, according to empirical tests aimed at minimizing the dimension (width $\cdot$ depth) of count-min sketches. Figure 6 shows how the width varies for different γ-deniability constraints. Even though the width is reported for a limited number of count-min sketches only, the general trend is that the width decreases as the required deniability grows. A similar trend characterizes the number of exportable downloads: the higher the deniability the lower they are, as Table 1 highlights.

4.1.2 Detection accuracy

AMICO classifies a sample as benign or malicious on the base of a *malware score* output by the shared classifier, and using a given threshold such that if the score is above that threshold then the sample is classified as a malware. We first show the impact of the deniability on the detection accuracy, in terms of false positive and false negatives considering as ground truth the classification obtained by using the original KB. Figure 7 shows how increasing the size of the exported KB, that is loosing deniability constraint, does not necessarily imply a significant reduction of the false positive and false negative rates. Indeed the accuracy seems to remain almost the same as deniability is varied, in fact counterintuitively it seems that the highest deniability leads to the best accuracy.

To understand why this happens, we have to analyze the impact of the deniability on the malware score. Indeed, the classification accuracy is affected only when the malware score is perturbed in such a way that it crosses the given threshold. As can be observed in Figure 8, the average difference between the malware score obtained by using the exported KB, the score computed on the original KB, and the shared classifier is highest when the deniability is 100%, which is the expected result. Such a mismatch between the impact on the malware score (which corresponds to what expected) and the impact on the detection accuracy (which does not) seems to depend on the shared classifier employed in the evaluation. Indeed, it was trained with features computed over more than 15000 download events, while in our evaluations the features used to test the classifier derive from a download history including at most 1582 events (see Table 1). Such a difference in size is the likely reason why misclassifications occur, and seems to depend on the shared classifier and on the size of shared KB employed in the evaluation. It is to note that such issue regards sharing AMICO

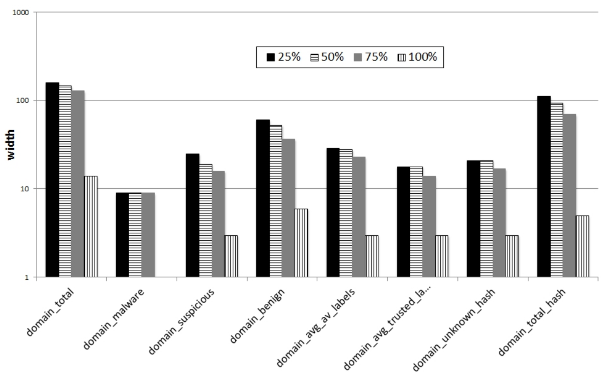

Figure 6: Comparison among widths of count-min sketches as the γ-deniability is varied. For space constraints, only a subset of the count-min sketches is reported here.

γ-deniability	exportable downloads
25%	1582
50%	1178
75%	961
100%	281

Table 1: Maximum number of exportable downloads for different constraints on γ-deniability.

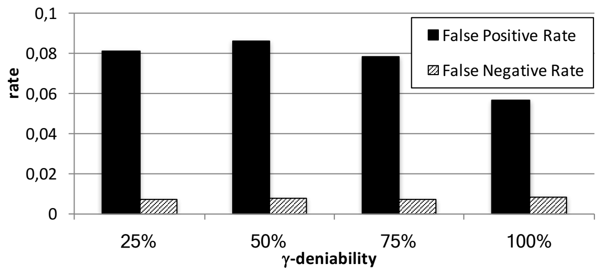

Figure 7: False positive/negative rates of AMICO classification using the exported KB to create feature vectors, for different γ-deniability constraints.

classifiers, and not sharing download events through count-min sketches, as proved by the results shown in Figure 8.

Another interesting aspect regards what happens when count-min sketches are queried with keys that had not been inserted: if the query returned a value witnessing the presence of such keys (i.e., a false positive for the single count-min sketch, not to be confused with the false positives of AMICO classification), then that value would be potentially harmful for whole classification. Figure 9 presents the average false positive rate of count-min sketches for each γ-deniability constraint. Although the reported values seem almost constant, there is a little (from 39% to 44%) increase when the deniability constraint is weakened. Such increase in the false positive rate is probably due to the fact that, as the amount of shared keys grows, the count-min sketches are more filled, and thus more likely to have a non-empty cell for each row for several keys that have not been actually inserted. This is in line with Equation 5, indeed false positive rate grows with n, and Section 6 reports some final considerations on this aspect.

4.2 Malware detection based on Behavior

In this scenario, we use the dataset provided by the 2nd Cybersecurity Data Mining Competition[6]. This dataset contains the logs of API/system calls invoked by malicious and benign samples, where each executable is uniquely identified by its MD5. The number of invocations for each API/system call can be used by a trained classifier to determine whether a sample is malicious or benign.

[6]Cybersecurity Data Mining Competition 2010: h
d

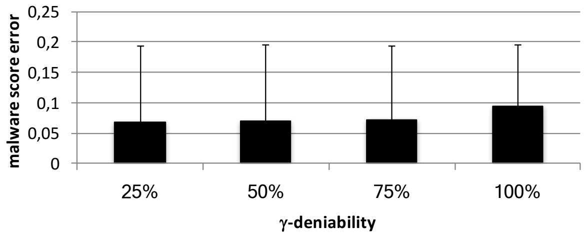

Figure 8: Mean difference (and bars representing standard deviation) between the malware score obtained by using the original KB and the one computed from sketches.

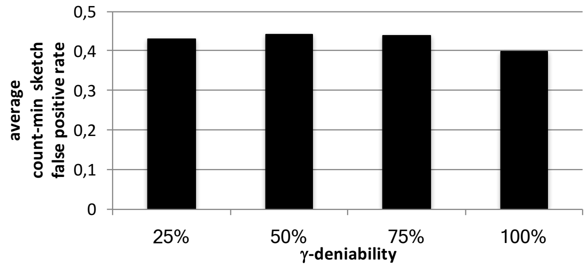

Figure 9: Average false positive rate of count-min sketches for different γ-deniability constraints. Sketches have been queried on keys associated to 100 distinct test download events.

The KB contains indeed all the information required to answer to questions like "how many time the API/system call c has been invoked by the sample having MD5 x?" for a possibly large number of distinct c and x. Collecting such a KB requires a lot of time to execute the dynamic analysis of samples for generating the counters of invocations. In turn, performing dynamic malware analysis necessitates of proper resources to setup the virtualization/emulation environments where samples can be executed, monitored, and traced. Sharing this kind of KB would enable others to test and evaluate their classifiers, based on API/sytem call invocation counters, without having to pay the effort in time and resources to run any dynamic analysis. Analogously to the previous scenario, both KB and trained classifier are exchanged between parties.

The KB can be exported through count-min sketches by using a sketch for each API/system call invoked by any sample included in the KB itself. For a count-min sketch related to API/system call c, a pair $\langle k, v\rangle$ represents the fact that the sample with MD5 k invoked c v times. Differently from the previous case, here all the count-min sketches have the same universe set for the keys, that is the set of MD5 signatures of analyzed samples.

4.2.1 *Count-min sketch configuration*

The owner of the KB may have concerns in sharing it because the fact that she has seen and analyzed a specific sample (identified univocally by its MD5) can be considered a sensitive information. For example, a party which has seen a sample known to be related to an APT, targeted at specific organizations or countries, may make suspicions arise on whether such party could be either one the target or involved in its development. While in the previous scenario we investigated the effect of the deniability on exported KB utility, here we are interested in deepening the role of noise. In this scenario, an attacker aiming at discovering whether the KB owner has seen a specific sample with MD5 x, could query the shared count-min sketches with key x to get API/system call counters, then compare them with the counters obtained by executing a dynamic analysis of the sample itself. The more the counters match, the more the attacker can be confident that the KB owner has x in its KB, regardless of the deniability guarantees provided by the way shared count-min sketches have been configured.

We configured the count-min sketches with 100%-deniability, inserting 378 MD5, and varied the err_{max} with values 0%, 5%, 10% and 20%. Universe cardinality has been computed taking into account the number of samples that a party could have analyzed. In order to have a lower bound on the universe cardinality, we have considered the number of samples received daily by VirusTotal[7], a free on-line analysis service of URLs and files. Also in this case, we fixed the depth to 11. Figure 10 presents the widths of a subset of the count-min sketches, for different constraints on error upper bound. As expected, in general, higher error upper bounds reduce the width, since indeed lower width leads to more collisions and thus more errors.

4.2.2 *Detection accuracy*

Similarly to what has been shown for the other scenario, here we present experimental results regarding how the amount of noise added to the values stored in the count-min sketches affects the detection accuracy of the shared classifier. As ground truth, we again use the classification provided with the classifier by creating the input feature vectors with the original values. Figure 11 reports the false negative rates obtained by varying err_{max}. As expected, the accuracy worsens as err_{max} grows, as the classification uses

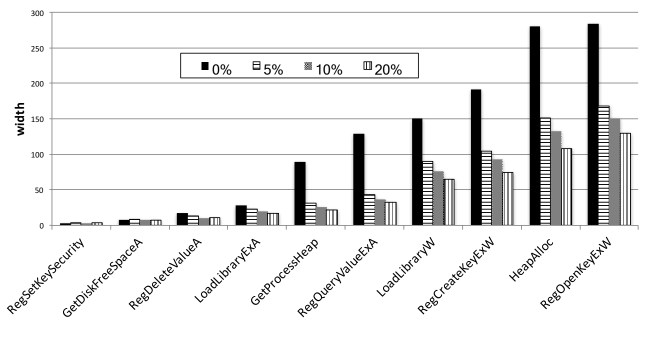

Figure 10: Comparison among the widths of count-min sketches as error upper bound varies. Due to space constraints, only a subset of APIs is reported.

5. RELATED WORK

Information sharing has already been proven to be promising in different fields [17, 23, 8, 25, 7, 11, 15]. Indeed, it allows to have larger KBs and, hence, perform more accurate statistical analyses and data mining tasks.

In [17], the Semantic Room abstraction is presented. It allows parties to share their data into a collaborative environment constituted by event-based platforms. The goal of a Semantic Room is to correlate coordinated Internet-based security threats and frauds. The privacy and the protection of possible sensitive data is demanded to the subscription of contracts, ruling requirements and service level specifications. Thus, the proposed abstraction assumes that its members are all trustworthy and prevent information leakage from the output diffused to Semantic Room members.

Shokri and Shmatikov [23] implement a system that allows multiple parties to jointly learn an accurate neural net-

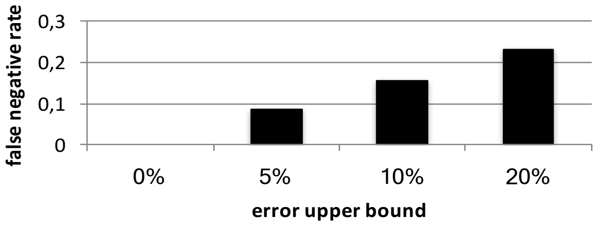

Figure 11: False negative rate for different error upper bounds. No false positive occurred.

work model without sharing their knowledge bases, formed by input datasets. Each party runs, in parallel and asynchronously, the stochastic gradient descent optimization method, training the model on its own dataset and, then, selectively shares small subsets of the trained models' key parameters. This enables parties to benefit from other participants' KBs, while preserving their own KB privacy.

Privacy-preservation becomes more and more important when parties own and need to share KBs related to Electronic Medical Record or Electronic Health Record (EMRs and EHRs, respectively). In the last decades, EMR/EHR systems have been largely deployed, disseminating data and preventing to store sensitive records in a unique KB [8]. From a statistical perspective, having a large centralized statistical database would allow to perform more accurate analyses and data mining tasks. On the other hand, EMRs/EHRs are highly distributed and, when these KBs are shared, patients' privacy needs to be preserved.

5.1 Privacy-preserving info sharing

Many models have been proposed to preserve the privacy of KBs. The most popular are k-anonymity and differential privacy. The first is vulnerable to many attacks [9], and solutions have been proposed for the majority of them, such as l-diversity [18] and t-closeness [16]. However, there not exists a real solution against the background knowledge attacks, which allow to re-identify records within an anonymized KB. A more recent attack, based on background knowledge, has threatened the anonymity of Netflix Prize[8] users [20]: taking advantage of very few information crawled from IMDb[9], Narayanan and Shmatikov have successfully de-anonymized the records of two users belonging to the Netflix Prize dataset. This proof of concept demonstrates how easy it is for an attacker to de-anonymize users by leveraging background knowledge, even when privacy-preserving mechanisms are put in place: the IDs of Netflix customers had been removed and noise was added to a very small portion of the total ratings. Whereas in the case of EMR/EHR systems the shared KB is a statical database, the aim of the Netflix was sharing a KB storing a large sparse dataset in order to let parties test their proposed collaborative filtering algorithms. This is very similar to the scenario where a company decides to share its own KB with other parties, allowing them to use their algorithms on its data. This is very common in privacy-preserving machine learning and data mining applications.

Differential privacy is another model for preserving privacy. Proposed by Dwork, it gives some guarantees on the

[7]VirusTotal: https://www.virustotal.com/en/statistics/

[8]Netflix Prize: http://www.netflixprize.com/

[9]Internet Movie Database: http://www.imdb.com

responses returned by arbitrary queries issued to a statistical database [5]. A database is differentially private if the attacker is not able to distinguish if a specific record is stored in it. As mentioned in Section 3, since Laplacian noise addition has been commonly used to implement differential privacy, this latter is vulnerable to tracker style attacks.

Other approaches rely on cryptographic protocols to share information in privacy-preserving fashion. Both [4] and [28] design and leverage secure protocols for sharing sensitive data. In [4], the authors focus on client-server interactions, in which the server shares the required minimum sensitive information. On the other hand, [28] addresses privacy concerns of sharing information in a collaborative environment. In both cases, the privacy leakage is non-zero.

5.2 Count-min sketches as privacy-preserving data structures

The idea of applying sketches to privacy-preserving computation is very recent. Similarly to our work, Balu and Furon leverage count sketches for adding a controlled amount of noise to matrix factorization tasks and providing differential privacy guarantees [1]. In [19], the authors design a cryptographic protocol in which count-min sketches, storing statistics of three different data sources, are encrypted. It is worth mentioning that the security of the proposed scheme depends on the protocol and sketches are only employed to reduce the network communication overhead. Conversely, Cormode [12] cites the work of Roughan and Zhang [21] that takes advantage of count-min sketches to run computations in privacy-preserving fashion. Nevertheless, the works on count-min sketches do not provide any guarantee about the privacy achieved, no privacy metric for count-min sketches has been designed or employed. One of the contributions of this article is also to propose a privacy metric for count-min sketches. The majority of the privacy metrics presented in Section 5.1 are not suitable for count-min sketches. Indeed, while the common differential privacy implementation is a noise addition technique, whose drawbacks have been extensively described in [22] and in the previous section, k-anonymity definition can be mapped to collisions within count-min sketches.

6. DISCUSSION

This section discusses some limitations and possible improvements with respect to the current state of this work.

Tuning the width. A relevant limitation is the $\mathcal{O}(n^2)$ time complexity of the procedure to configure the width of a count-min sketch to achieve a given guarantee on error upper bound (see Section 3.2), given the universe set cardinality u, the depth d of the count-min sketch, and the number n of elements to insert. The other procedure also is too time consuming, indeed it takes $\mathcal{O}(n)$ and, moreover, provides weaker guarantees on error upper bound, which depends on δ parameter. A direction to investigate could be devising a statistical model to provide probabilistic guarantees on error upper bound without having to iterate through all the n elements to insert. Rather, a mathematical expression based on u, d, w and n should allow to estimate the error on retrieved values. In this way it could be possible to directly derive the proper value for the width or, in the worst case, use some iterative method of numerical analysis to obtain a good approximation with $\mathcal{O}(\log n)$ time complexity.

Tuning the depth. How to configure the depth d of a count-min sketch is still an open point. Event though some practical guidelines are provided in Section 3.4, a more formal and neater way should be defined. A promising approach would consist in tuning together the width w and the depth d to achieve certain guarantees on error upper bound, and at the same time to minimize the product $w \cdot d$ to save on the memory required to store the count-min sketch. Some preliminary simulations show that this could be viable: by iterating incrementally through possible depths, and for each depth then increasing the width to find the one ensuring 0 error upper bound, it turned out that wd decreases up to some value $\hat{d}$, and then begins to increase.

False positive keys. The property of deniability enables the KB owner to deny having inserted a given key in a count-min sketch. An aspect that has not been investigated regards what guarantees could be possible to ensure with respect to the values returned by queries on keys that have not been inserted. In case of false positives, the utility of the exported KB could be affected relevantly. A desiderata could be including the count-min sketch false positive rate (defined by Equation 5) in the tuning process to achieve specific guarantees on what should happen when a query with a non inserted key occurs.

Stronger privacy guarantees. While the deniability requires that each cell of an inserted element should be covered by *at least one element* of the hiding set, stronger privacy properties could require each cell to be covered by *at least K elements* of the hiding set. This concept is defined as γ-K*-anonymity* and introduced by [2], where we mainly took inspiration from for the basic deniability property.An interesting addition to this work could be to translate also the concept of γ-K-anonymity to count-min sketches and extend the configuration procedure accordingly.

7. CONCLUSION

In this work we proposed to employ count-min sketches to export KBs in scenarios where privacy-preserving guarantees are required. We provided an analysis on the properties we can ensure on count-min sketches concerning the deniability of inserted keys (considered as sensitive information) and the error on retrieved values (considered as an information that could be leveraged for linkage attacks). We further presented a metric to measure the level of privacy that can be supplied, the γ-deniability, an adaptation of a metric used by [2] for bloom filters, and devised a probabilistic model proposing a procedure to tune count-min sketches, given specific goals on deniability and error upper bound. Such procedures have been applied to two scenarios related to malware detection, where count-min sketches can be used to export KBs in a privacy-preserving fashion: results mainly show the privacy/utility tradeoff that usually holds when data need to be altered to meet privacy requirements.

Acknowledgment

This work has been partially supported by a grant of the Italian Presidency of Ministry Council, by the Cybersecurity National Laboratory of CINI (Consorzio Interuniversitario Nazionale Informatica), and by Filierasicura project funded by Cisco Research and Leonardo SPA. We thank Roberto Perdisci for the helpful comments and references, provided during manuscript writing, and the anonymous reviewers for their careful reading and useful insights.

8. REFERENCES

[1] R. Balu and T. Furon. Differentially private matrix factorization using sketching techniques. *IMMSEC*, June 2016.

[2] G. Bianchi, L. Bracciale, and P. Loreti. Better than nothing privacy with bloom filters: To what extent? In J. Domingo-Ferrer and I. Tinnirello, editors, *Privacy in Statistical Databases*, volume 7556 of *Lecture Notes in Computer Science*, pages 348–363. Springer Berlin Heidelberg, 2012.

[3] G. Cormode and S. Muthukrishnan. Approximating data with the count-min sketch. *Software, IEEE*, 29(1):64–69, 2012.

[4] E. De Cristofaro, Y. Lu, and G. Tsudik. *Efficient Techniques for Privacy-Preserving Sharing of Sensitive Information*, pages 239–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[5] C. Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006.

[6] C. Dwork and A. Smith. Differential privacy for statistics: What we know and what we want to learn. *Journal of Privacy and Confidentiality*, 1(2):2, 2010.

[7] D. Gallego and G. Huecas. An empirical case of a context-aware mobile recommender system in a banking environment. In *MUSIC*, pages 13–20. IEEE, 2012.

[8] A. Gkoulalas-Divanis, G. Loukides, and J. Sun. Publishing data from electronic health records while preserving privacy: A survey of algorithms. *Journal of Biomedical Informatics*, 50:4–19, 2014.

[9] A. Hussien, N. Hamza, and H. Hefny. Attacks on anonymization-based privacy-preserving: A survey for data mining and data publishing. *Journal of Information Security*, 4:101–112, 2013.

[10] L. Invernizzi, S. Miskovic, R. Torres, C. Kruegel, S. Saha, G. Vigna, S.-J. Lee, and M. Mellia. Nazca: Detecting malware distribution in large-scale networks. In *NDSS*, volume 14, pages 23–26, 2014.

[11] A. J. P. Jeckmans, M. R. T. Beye, Z. Erkin, P. H. Hartel, R. L. Lagendijk, and Q. Tang. Privacy in recommender systems. In *Social Media Retrieval*, Computer Communications and Networks, pages 263–281. Springer Verlag, London, January 2013.

[12] M. Kao, editor. *Encyclopedia of Algorithms*. Springer, 2015.

[13] O. Kaser and D. Lemire. Strongly universal string hashing is fast. *CoRR*, abs/1202.4961, 2012.

[14] M. Kruczkowski and E. N. Szynkiewicz. Support vector machine for malware analysis and classification. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*, pages 415–420. IEEE Computer Society, 2014.

[15] B. Li, J. Springer, G. Bebis, and M. H. Gunes. A survey of network flow applications. *Journal of Network and Computer Applications*, 36(2):567 – 581, 2013.

[16] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[17] G. Lodi, L. Aniello, G. A. Di Luna, and R. Baldoni. An event-based platform for collaborative threats detection and monitoring. *Information Systems*, 39:175–195, 2014.

[18] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[19] L. Melis, G. Danezis, and E. D. Cristofaro. Efficient private statistics with succinct sketches. *CoRR*, abs/1508.06110, 2015.

[20] A. Narayanan and V. Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[21] M. Roughan and Y. Zhang. Secure distributed data-mining and its application to large-scale network measurements. *SIGCOMM Comput. Commun. Rev.*, 36(1):7–14, Jan. 2006.

[22] R. Sarathy and K. Muralidhar. Some additional insights on applying differential privacy for numeric data. In J. Domingo-Ferrer and E. Magkos, editors, *Privacy in Statistical Databases*, volume 6344 of *Lecture Notes in Computer Science*, pages 210–219. Springer, 2010.

[23] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321. ACM, 2015.

[24] L. Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, Oct. 2002.

[25] E. Toch, Y. Wang, and L. F. Cranor. Personalization and privacy: A survey of privacy risks and remedies in personalization-based systems. *User Modeling and User-Adapted Interaction*, 22(1-2):203–220, Apr. 2012.

[26] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis. Measuring and detecting malware downloads in live network traffic. In *Computer Security - ESORICS 2013*, pages 556–573. Springer Berlin Heidelberg, 2013.

[27] M. N. Wegman and J. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265 – 279, 1981.

[28] N. Zhang and W. Zhao. Distributed privacy preserving information sharing. In *Proceedings of the 31st international conference on Very large data bases*, pages 889–900. VLDB Endowment, 2005.

APPENDIX

A. PROOF OF THEOREM (SKETCH)

According to Definition 3.1, a key k is deniable if, for each row $i \in \{1, \ldots, d\}$, $h_i(k) = h_i(z)$ for some $z \in K_{\mathcal{U}} \setminus K_S$. This event is independent and equiprobable for each row. Let q be the probability that this event occurs for a single row, then $\gamma\big(M(S)\big) = q^d$.

In order to prove Theorem 3.1, let consider a single row in which b is the number of cells (out of w) covered by keys in K_S, and r is the number of cells (out of b) covered by keys in the hiding set $\mathcal{V}$, with $r \leq b \leq w$. We have to restrict the cells covered by the hiding set to be a subset of the cells covered by K_S because we need to impose the occurrence of a collision. Given these definition, we have that $q = r/b$.

Both b and r are random variables following the probability distribution defined by Equation 10 (see Appendix B). Indeed, b is the number of non-empty cells, in a count-min sketch row with width w, after having inserted n elements. r is the number of cells, out of these b cells, where the elements of the hiding set are mapped, which can be seen as the number of non empty cells in a row with width b, after having inserted v elements. In turn, v is a random variable following a binomial distribution, but with a probability different from that used in Equation 6: rather than using the approximation given by the false positive probability, we can be more precise by conditioning the distribution to the knowledge that there are b non-empty cells, so we can use b/w. The exact formula reported in Theorem 3.1 can be derived by simply applying the law of total probability.

Equation 8 derives from the observation that the two probability distributions employed in this formula are both tightly centered around their mean [2]. The approximation consists in replacing r and b in q as follows

- $r \leftarrow b(1 - (1 - 1/b)^v)$
- $b \leftarrow w(1 - (1 - 1/w)^n)$
- $v \leftarrow (u - n)\frac{b}{w} = (u - n)(1 - (1 - 1/w)^n)$

B. PROBABILITY OF NON EMPTY CELLS

A result that is used in Appendix A regards the probability of having, in a row of a count-min sketch, a certain number of non empty cells after having inserted n elements.

LEMMA B.1. *Consider a row of a count-min sketch having width w where n elements with different keys have been inserted. Let U be the random variable representing the resulting number of non empty cells. The U has the following probability distribution:*

$$U_w(n, b) = \frac{\left\{ {n \atop b} \right\} \binom{w}{b} b!}{w^n}, \forall b \in \{1, \ldots, w\} \tag{10}$$

and mean value

$$E[U] = w\left(1 - \left(1 - \frac{1}{w}\right)^n\right) \tag{11}$$

$\left\{ {n \atop b} \right\}$ is a Stirling number of the second kind, expressing the number of ways to partition a set of n elements into b non empty subsets, and is computed as

$$\left\{ {n \atop b} \right\} = \frac{1}{b!} \sum_{i=0}^{b-1} (-1)^i \binom{b}{i} (b - i)^n$$

Proof. See [2].