

Enforcing k -anonymity in Web Mail Auditing

Dotan Di Castro
Yahoo Labs
MATAM, Haifa 31905, Israel
dot@yahoo-inc.com

Liane Lewin-Eytan
Yahoo Labs
MATAM, Haifa 31905, Israel
liane@yahoo-inc.com

Yoelle Maarek
Yahoo Labs
MATAM, Haifa 31905, Israel
yoelle@gmail.com

Ran Wolff
Yahoo Labs
MATAM, Haifa 31905, Israel
ranw@yahoo-inc.com

Eyal Zohar
Yahoo Labs
MATAM, Haifa 31905, Israel
eyalzo@gmail.com

ABSTRACT

We study the problem of k -anonymization of mail messages in the realistic scenario of auditing mail traffic in a major commercial Web mail service. Mail auditing is necessary in various Web mail debugging and quality assurance activities, such as anti-spam or the qualitative evaluation of novel mail features. It is conducted by trained professionals, often referred to as “auditors”, who are shown messages that could expose personally identifiable information. We address here the challenge of k -anonymizing such messages, focusing on machine generated mail messages that represent more than 90% of today’s mail traffic. We introduce a novel message signature *Mail-Hash*, specifically tailored to identifying structurally-similar messages, which allows us to put such messages in a same equivalence class. We then define a process that generates, for each class, masked mail samples that can be shown to auditors, while guaranteeing the k -anonymity of users. The productivity of auditors is measured by the amount of non-hidden mail content they can see every day, while considering normal working conditions, which set a limit to the number of mail samples they can review. In addition, we consider k -anonymity over time since, by definition of k -anonymity, every new release places additional constraints on the assignment of samples. We describe in details the results we obtained over actual Yahoo mail traffic, and thus demonstrate that our methods are feasible at Web mail scale. Given the constantly growing concern of users over their email being scanned by others, we argue that it is critical to devise such algorithms that guarantee k -anonymity, and implement associated processes in order to restore the trust of mail users.

1. INTRODUCTION

In the last decade, Web mail traffic has evolved, very much like regular snail mail, into being dominated by machine-generated messages. Some recent studies have verified that

more than 90% of non-spam Web email is indeed generated by automated scripts, [1, 10]. These messages vary in importance: some represent highly valuable information (and often include sensitive personal details), such as shipment notifications, flight itineraries, social events, monthly bank statements, etc., while others are almost junk mail such as promotions or newsletters, from which users have forgotten to unsubscribe. Two common characteristics of these machine-generated messages is that most of them are highly structured documents, with rich HTML formatting, and they are repeated over and over, modulo minor variations, in the global mail corpus. These characteristics clearly facilitate the application of automated data extraction and learning methods at a very large scale, in order to offer users new mail features and applications including “customized search results, tailored advertising, and spam and malware detection” [9].

As much as these methods are, by design, automated, they still require from time to time some human intervention for quality assurance or evaluation purposes, or during the research phase of their development. Major Web mail services allow only a restricted group of trained employees, that we refer to as *auditors*¹, to access personal information under strict contractual confidentiality obligations, [16, 26, 9]. We argue here that there is a need to add automated mechanisms for preserving user privacy during the auditing process. Consider for example a supervised learning application that requires a training test. The auditors, should be shown only “anonymized” samples of mail messages, before they can be reviewed, annotated, classified, etc.

As we focus on machine-generated messages, each such sample is in fact the representative of a much larger set of messages, typically originating from the same sender, and having a similar HTML structure. Determining whether a message is human or machine-generated is out of the scope of this paper. We refer the reader to previous work, as proposed by some of the authors of this paper in [10], for a possible solution to this task. More generally, it is nowadays quite feasible to devise simple methods for capturing (non-malicious) machine-generated messages by leveraging signals such as the “burstiness” (i.e., large number of messages sent at once) of the traffic originating from a sender, its volume, the structure of the messages etc. Note that a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WSDM’16, February 22–25, 2016, San Francisco, CA, USA.

© 2016 ACM. ISBN 978-1-4503-3716-8/16/02...\$15.00

DOI: <http://dx.doi.org/10.1145/2835776.2835803>

¹Note that these professionals are typically not developers as they purely focus on the qualitative aspect of the data. Some also refer to them as QA professionals or “editors”.

perfect distinction between human and machine-generated messages is not a pre-requisite of this work, as we will see that our methods naturally eliminate rare messages. Note also that we exclusively consider the privacy concerns of *human recipients* and ignore those of mass-senders (typically machines), for whom personal information is not exposed in any case. More specifically, we consider machine-generated messages that might expose personal information of a given recipient, for instance the recipient’s home address or telephone, or movies that she rented. We investigate how to preserve the recipient’s privacy when auditors review one of her messages.

We propose a mail-specific anonymization mechanism, based on a novel function operating on a single mail message, which we coin “*Mail-Hash*” by analogy to Broder et al. *Min-Hash* technique [4]. This Mail-Hash function leverages the rich DOM structure most machine-generated messages have and computes a signature, in the hope of identifying messages that seem to have been generated by the same script. Given this signature, we then generate equivalence classes of messages. Grouping messages into equivalence classes should then be performed daily, while being able to efficiently cover billions of messages in any Web scale mail service.

Intuitively, message anonymization can be achieved by simply masking the variable content of these messages. Consider as a simplifying example, two order confirmation messages sent by the same online seller to two different recipients. These messages will likely have the same structure as well as a significant amount of common data. One message body consists of “*Hello Jessie, thank you for shopping with us. We have received your order of Green Mountain Coffee, and are preparing it for shipment, etc.*”, while the other reads “*Hello Sergio, thank you for shopping with us. We have received your order of Bleu de Chanel, and are preparing it for shipment, etc.*”. It is quite straightforward to hide the delta between these two messages, using for instance the notion of mail template suggested in [1]. Both messages would be mapped into the same mail template consisting of a tuple formed by the sender of the mail and the common body strings namely “*Hello *, thank you for shopping with us. We have received your order of *, and are preparing it for shipment, etc.*”. Personal information, such as names (Jessie and Sergio in our example), addresses, or purchased products (Green Mountain coffee and Bleu de Chanel perfume in our example), which are necessarily variable across multiple recipients, would be automatically hidden. In reality, messages are more complex and include a great deal of HTML-formatting that we will discuss in details, but the intuition remains the same. We will also explain how to group messages in the same equivalence classes covering more than k recipients, so that k -anonymity can formally be verified. Once k -anonymity is achieved, since the masking process transforms all messages within a class into the same template, one can easily generate from the latter a nicely formatted mail sample that masks all the parts of the message that might break k -anonymity. We note that the privacy regulations and k -anonymity firm restrictions pose strict requirements on the process of equivalence classes and template generation, as will be presented in this work. These are inherently different from the application-driven requirements which typify template definition and computation as described in [1, 10].

In order to verify the benefits of our approach, we consider the *productivity of auditors* for every sample they are shown. Clearly, the more content an auditor can see, the more effective their reviewing or annotating task can be. Consider our previous example. While being short, auditors will easily be able to infer from it that the template refers to a purchase confirmation. A longer template that would indicate for instance a product category would even be more informative. Thus, we use the simple measure of *content coverage*, representing the amount of non-hidden mail content that the auditors can see. Our goal is thus to maximize content coverage in order for auditors to be highly effective, while enforcing k -anonymity. Note that there is a trade-off between these two: while an equivalence class should be large enough so as to cover at least k recipients (to enforce k -anonymization), it should be small enough to include only messages that are quite similar so as not to hide too much content, which would reduce content coverage.

Another challenge inherent to professional auditing is that auditors continue to be exposed to more and more mail samples over their tenure. Consequently k -anonymity needs to be considered not in isolation, but over time. Fortunately, professional auditing exhibits an additional property, which does not exist in usual anonymized release settings, it is tightly controlled: auditors are employed by the mail service and, as such, can be strictly governed in terms of their access not only to the current but also to future and auxiliary datasets. Taking advantage of this property, we consider the objective of releasing samples to be shown to an auditor on a daily basis, while retaining the k -anonymity of all releases over time. In addition, we take into account the capacity of auditors, and ensure the number of samples that they can be daily assigned does not go beyond some threshold representing their daily capacity. Our problem thus converges to an online mail auditing k -anonymization problem, whose objective is to release a bounded amount of samples to an auditor on a daily basis, while retaining the k -anonymity of all releases over time, and maximizing visible content. We propose an online mechanism for this problem, proving its scalability and evaluating it via experiments conducted over actual Yahoo mail traffic.

The contributions of our paper are threefold:

1. We provide, to the best of our knowledge, the first definition of k -anonymity for human auditing of mail messages.
2. We introduce a surprisingly simple yet efficient Mail-Hash function specifically tailored to identifying similarly structured mail messages.
3. We define a mail auditing process that is guaranteed to preserve the k -anonymity of mail users over time, while maximizing the productivity of mail auditors. We also demonstrate its feasibility at Web mail scale by providing detailed results of a full system we implemented over Yahoo mail.

The paper is organized as follows. Section 2 gives the problem definitions for the k -anonymization of mail messages in the auditing scenario. Section 3 presents our Mail-Hash function. Section 4 describes the complete k -anonymization process, from computation of equivalence classes based on Mail-Hash, through masking, and up to the online daily assignment of anonymized samples to the auditor. Section 5

reports on the large scale experiments we conducted. Finally, we review related work in Section 6 and present our conclusions in Section 7.

2. MAIL K -ANONYMIZATION

We first provide some general definitions for k -anonymity for mail releases in general. Then, we discuss the transformation actions, such as filtering and masking, that need to be applied in order to generate mail data releases that can make users indistinguishable and thus enforce k -anonymity. Finally, we present the realistic scenario of mail auditing, considering releases over time.

2.1 Definitions

Given a data table T in which each row contains the information of a single and unique individual, the table is k -anonymous if and only if each row repeats k or more times. An algorithm which accepts every table T , k -anonymous or not, and outputs a table T' which is guaranteed to be k -anonymous is called a k -anonymization algorithm. A table in which every row may include details of several individuals, which are denoted senders and recipients, is k -recipient anonymous if it is k -anonymous with respect to message recipients.

We represent a mail dataset D_t as a set of incoming mail messages $m_t[1], m_t[2], \dots, m_t[n]$ received by the entire set of users during a time interval t . For the sake of simplicity, we will consider as our default time interval a single day, which fits a common process of daily data release. Another simplifying assumption is to abstract any incoming mail message into a list of entities, where an entity can be a sender, or any character string in the subject or body of the message. A release R_t is then defined as a transformation of D_t into a set of uniquely identifiable users (mail recipients) and associated entities. We then refer to the tuple of all entities associated with a same message $m_t[i]$, as a *template*, reusing the terminology introduced by Ailon et al. in [1]. Like in their work, a template is defined as a tuple of a sender and a character string, which can be altered for abstraction as discussed later. One difference with their notion of templates however is that this string can be derived not only from the subject but most frequently from the body of the message.

A release R_t^u consists of the list of templates associated with user u in R_t (a template is associated with u if u received a message corresponding to this template). For any user u , we define the set of users $[u]_t$ as the set of all users v such that $R_t^v = R_t^u$. A single release R_t fulfills the requirement of k -anonymity, if for any user u in R_t , it holds that $|[u]_t| \geq k$. In other words, user u is indistinguishable from at least $k - 1$ other users in the release as it maps to exactly the same list of templates. Figure 1 depicts a release, with two user groups examples. Each group is of size at least k , and includes users with exactly the same list of templates.

In the same spirit, a sequence of releases R_1 through R_t is k -anonymous if, for every user u , the intersection of all $[u]_i$ has more than k users, $|\bigcap_{i=1}^t [u]_i| \geq k$. In other words, user u is indistinguishable from at least $k - 1$ other users over all releases, as they are associated with exactly the same list of templates over R_1, \dots, R_t .

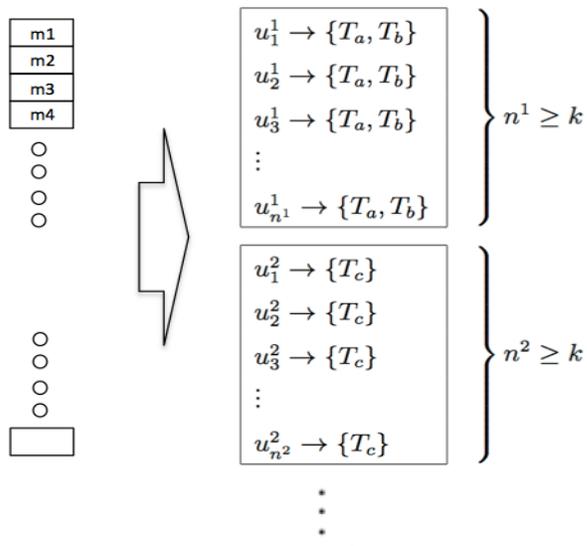


Figure 1: Two exemplary groups of users in a release. In group $\{u_1^1, \dots, u_{n_1}^1\}$ each user is associated with two templates T_a, T_b , and in group $\{u_1^2, \dots, u_{n_2}^2\}$ each user is associated with template T_c . The left side are the messages m_i included in the mail dataset.

2.2 From Dataset to Release

Intuitively, the transformation process consists of generating a list of templates for any given user u , so as to make the data informational enough to auditors and at the same time make u indistinguishable from others. Each template is an abstraction of a message sent by a given sender to u and obtained by deleting terms or sequence of terms (replacing them by a deletion marker $*$). Ailon et al. [1] showed that this abstraction of a message is particularly convenient for representing machine-generated messages generated by a same script on the sender side. In our context, we do not necessarily attempt to match a mail template to an unknown script, but still use the same type of abstraction in order to reduce variability between messages. We refer to this task as *masking*. In addition, we can remove users or messages altogether if they cannot be k -anonymized. We refer to this task as *filtering*.

In order to support masking and filtering, we need first to identify mail messages that are similar enough so that k -anonymization can be enforced while allowing a maximum portion of content to remain unmasked. We explain in Section 3 how we have devised a novel “mail-hash” function that associates highly similar messages with a same signature. We can then group messages originating from the same sender, and having the same signature, into equivalence classes. Then, for all messages in a class, variables parts are masked so as to produce a unique template for the class². In order to guarantee k -anonymization, each such class needs to cover at least k users. After the masking process, all messages in an equivalence class have in fact been reduced to the same template. Once this process is conducted, we associate with each user in the release, one or

²In the paper, we will use interchangeably the terms template and (equivalence) classes as they are associated in a 1-to-1 mapping.

several such templates, which drive the generation of *mail samples* that can be reviewed by the auditor. Each template is by definition derived from messages sent to at least k different users and as such preserves k -anonymity with respect to this data set.

One could imagine multiple ways to generate templates, as well as conducting masking and filtering. In order to select the most adequate method, we go back to the task at hand, namely assigning mail samples to auditors and maximizing their productivity, while maintaining the k -anonymity of every sample of mail they will be shown. It is obvious that the more unmasked data the auditor can see in a sample, the more effective she can be. Therefore, we propose to evaluate the usefulness of a sample via a raw quantitative measure of *content coverage*. More specifically, we define the content coverage of a sample, or rather of its underlying template and associated class, as the amount of data left in the template after masking, relative to the original source. We then derive the usefulness of a given release R_t , noted $CC[R_t]$, as the average content coverage of the templates in the release. In addition, it is clear that each auditor can review only a limited number of mail samples per work day, so we also need to ensure the number of daily samples that are assigned to an auditor is limited to some threshold Γ .

2.3 Online k-Anonymization

Optimizing R_t for content coverage, while not going over Γ samples, is however not sufficient. Indeed, as auditors get assigned new samples day after day, they are exposed to more data, and k -anonymization needs to be preserved over time. Therefore we need to generalize our problem to a new online problem, where our goal is to generate every day a new release R_t , taking into consideration both the current dataset D_t and previous releases R_1, \dots, R_{t-1} , while retaining the k -anonymity of all releases. This new release should be generated so that the overall content coverage, denoted by CC , averaged over all releases, is maximized, and the number of samples assigned daily to an auditor is at most Γ . Note that we do consider the assignment of samples to multiple auditors, as auditors are obligated not to share or discuss sample data with each other, and thus these are independent tasks. Verifying whether the assignment process can be made more efficient by considering a pool of auditors, and optimizing over a pool rather than independently for each auditor, is out of the scope of this work.

3. THE MAIL-HASH SIGNATURE

In order to increase the chances of having enough users sharing the same templates in any mail release, we need to be able to detect structurally similar messages, in an extremely fast and scalable manner as we are dealing with Web mail. We introduce here Mail-Hash, a novel mail message signature specifically designed to address this requirement. This new technique is fast and simple, yet provides superior results when dealing with the special case of anonymizing machine generated email, as demonstrated in Section 5.

3.1 Motivation

Document similarity in the context of online information has been widely explored. Note that we intentionally ignore here the traditional measures of similarity used in Information Retrieval such as cosine or Jaccard, or their associated clustering techniques, such as complete-link or k -

nearest neighbor. This is, not only for reasons of performance, (indeed computing pair-wise similarity between all messages might be prohibitive at Web mail scale) but also because of the task that requires stricter message similarity.

Therefore we elected to follow the school of work, pioneered by Manber [15], Heintze [11] and Broder³ *et al.* [2, 3, 4, 5] for fingerprinting documents. Their methods use a small fixed-size sketch, which can be computed extremely fast for each document. Documents are usually reduced to a canonical sequence of tokens, considering the text and/or the surrounding HTML tags, and forming sub-sequences of terms called *shingles*, upon which the document sketch is computed. One of the main use cases of this approach is deduplication of Web pages for search or detecting plagiarism.

The Mail-Hash method we introduce here belongs to the same school of work but is tailored both to machine-generated mail messages that contain a great deal of HTML formatting common to numerous messages, and to our specific task at hand, namely enforcing k -anonymization. Note that personal messages that contain much less formatting and are in essence private are ignored by design and will never be shown to auditors.

3.2 The Algorithm

The goal of the Mail-Hash algorithm is to generate a single hash value, referred to as Mail-Hash, for each mail message. The idea is to prune the DOM tree [24] so as to leave only the XPath expressions [23] (or XPaths for short), that lead to “textual leaves”, where textual leaves are non-empty alphanumeric strings. We then sign only the pruned DOM tree⁴ of the mail, while ignoring the textual leaves and being resilient to some minor changes in the HTML formatting.

In order to sign the pruned DOM tree, we use the widely used MD5 [19] over the ordered list of all XPath expressions. We chose MD5 as it is considered a collision-free (although not collision-safe) hash algorithm that takes into account every character of the input, and is sensitive to position of characters, which in our case translates to the order of XPath entries. Although MD5 has 128-bits, we use only the lower 64-bits due to the fact that it is largely sufficient for a collision-free operation when the number of inputs is in the order of billions.

Algorithm 1 Mail-Hash

- 1: **Input:** mail message
 - 2: **Output:** 64-bit signature
 - 3: Parse the mail message to DOM tree
 - 4: **For** each non-empty text element e_i in the DOM tree, generate a simple XPath t_i that leads from root to the text node, with all the elements along the path and necessary proximity positions
 - 5: Concatenate all the XPath items t_i into a single string T , by their DFS order in the DOM tree
 - 6: The Mail-Hash is the lower 64-bits of the MD5 of the string T
-

³See [12] for a large scale evaluation of Broder *et al.*'s algorithm).

⁴For simplicity, we will refer to the pruned DOM tree as the DOM tree from now on.

Messages that have the same DOM structure but differ by their textual leaves, will get the same Mail-Hash score. Note that an unwanted result might occur if messages have exactly the same DOM structure but their textual leaves differ. Another problem may arise if many messages have a unique DOM structure, resulting in a wide range of Mail-Hash values. We claim that both phenomena are rare in reality, making Mail-Hash an excellent choice in practice, as shown in Section 5. The details of the Mail-Hash generation algorithm are given in Algorithm 1. Figure 2 presents two examples of HTML mail messages and the detailed calculation of their Mail-Hash signatures. Figure 2a displays a short message with two non-empty textual leaves e_{a1}, e_{a2} . Therefore it has two XPath expressions t_{a1}, t_{a2} that form the concatenated string T_a .

Figure 2b presents a similar message sent to another receiver, with exactly the same HTML structure but slightly different textual leaves due to the receiver’s first name. Although $e_{b1} \neq e_{a1}$, the identical structure results in $\forall_i t_{ai} = t_{bi}$ and therefore the signatures of the two messages are identical.

<code><html><body><p>Dear Ava,</p><p>Thank you for contacting us.</p></body></html></code>
$t_{a1} = /html/body/p[1], t_{a2} = /html/body/p[2]/b$
$T_a = /html/body/p[1] /html/body/p[2]/b$
$\text{Mail-Hash}(a) = MD5_{64-bit}(T_a)$

(a) Mail message “a”

<code><html><body><p>Dear Bill,</p><p>Thank you for contacting us.</p></body></html></code>
$t_{b1} = t_{a1}, t_{b2} = t_{a2}$
$T_b = T_a$
$\text{Mail-Hash}(b) = \text{Mail-Hash}(a)$

(b) Mail message “b”

Figure 2: Identical Mail-Hash values for two similar messages

Figure 3 also presents a similar message, but this time the signature is different because the structure is not completely identical. To be more precise, message c ’s structure contains the structure of message a , which infers they have a high resemblance. However, for being able to apply k -anonymization in the most efficient way, we must put message c in a different class, where it may or may not be joined with other similar messages. This last example shows the biggest difference between Mail-Hash and existing fingerprinting techniques that measure resemblance. The latter techniques tend to capture both resemblance and containment, while Mail-Hash captures all structural differences and does not group containers along with the contained documents (containment is not a desirable attribute between messages in the same equivalence class, but rather precise structural similarity).

3.3 Grouping by Mail-Hash Value

The Mail-Hash value of each message can be computed in the most efficient manner in a single pass over the entire mail dataset, resulting in $O(n)$ running time where n is the

<code><html><body><p>Dear Cathy,</p><p>Thank you for contacting us.</p><p>Please call us to renew your membership.</p></body></html></code>
$t_{c1} = t_{a1}, t_{c2} = t_{a2}, t_{c3} = /html/body/p[3]$
$T_c = /html/body/p[1] /html/body/p[2]/b /html/body/p[3]$
$\text{Mail-Hash}(c) = MD5_{64-bit}(T_c)$

Figure 3: Mail-Hash value for mail message “c”

number of mail messages in the dataset. This is the lower bound on the running time of any algorithm that needs to explore the entire corpus. We consider that all messages that share the same Mail-Hash value are *equivalent* and assign them to the same equivalence class. This process generates a large number of classes of various sizes, which can also cover the long-tail of mass senders, which send hundreds as opposed to thousands of daily messages. We detail in the next section how these classes are processed to generate masked samples that preserve k -anonymity and can be safely shown to auditors.

4. THE k -ANONYMITY PROCESS

4.1 Outline

We describe the overall process of enforcing k -anonymity in the mail auditing process. Our process consists of three stages:

1. Grouping similar messages in equivalence classes, based on the Mail-Hash function.
2. Masking entities in messages belonging to a same class so as to enforce k -anonymity. The output of this process is a list of templates, where each template represents an equivalence class.
3. The assignment of templates to an auditor, note that this is an online process where templates to be shown to a given auditor are selected on a daily basis, while preserving the k -anonymity of the entire process.

Using the Mail-Hash algorithm described in Section 3, a Mail-Hash signature is computed for each message, and equivalence classes are formed. Each class contains messages that were sent from the same sender and have the same Mail-Hash signature. In order to ensure k -anonymization, classes with less than k users are filtered. Note that a user is counted only once per class, even if she received several messages belonging to the same class.

4.2 Masking

We consider each message as a set of textual entities, where each entity is a character string that consists of all the characters appearing between two subsequent HTML tags. An individual entity is retained if it contains at least one alphanumeric character. For each equivalence class, entities are deleted from the messages in the class, up to the point where all messages are identical, and thus k -anonymization is preserved at the class level. The deleted entities represent these variable term sequences that are not common to *all*

messages in the class. As our objective is to maximize the content that is shown to the auditor, the masking operation is stopped when all messages are mapped into a same identical template, as defined in Section 2. The template consists of a sequence of terms, delimited by a * deletion marker, which are common to all messages in the class. The template then allows to generate a masked mail sample (preserving mail formatting) that can be shown to auditors. We note that in some cases HTML tags might contain personal attribute values. Our solution is then applied by considering these values as textual entities.

Figure 4 shows a mail sample generated from a template representing a large set of equivalent mail messages sent by an ecommerce⁵ site to a large number of users as order confirmation. All sensitive personal information has automatically disappeared, such as the name of recipient following the “Dear” greetings or his address in the “Ship to” field, without necessitating any deep parsing or semantic analysis. Yet some common qualifiers, such as the prefix TS0T in the order number, have been preserved.

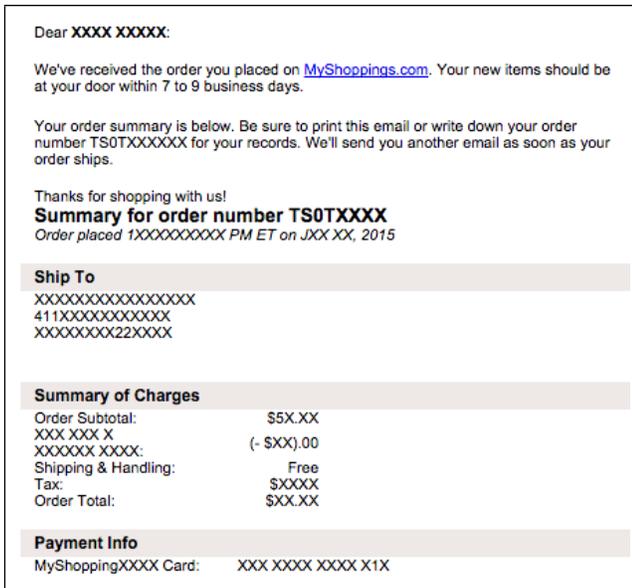


Figure 4: A masked email sample of a shopping receipt.

Another example is shown in Figure 5 with an itinerary confirmation for an airline (renamed here **FlighAirWays** for the same reason as before). Interestingly, more data is revealed here like the destination “ATLANTA”, which is frequent enough to become part of the template (precisely, appears as destination for at least k users associated with the template), while the departing city remains masked. Note that fixing the time frame of a given release in our anonymization settings, bears a great deal of influence on the eventual amount of information that will be shown. It could very well be the case that with a shorter time frame, the departing city or the flight number might be exposed. For instance an application that would analyze the users’ inboxes, in order to remind them of their departure details, might require auditors to verify that the automated script

⁵Note that we have renamed here this site [MyShoppings.com](#) in order to respect Yahoo mail terms of conditions of not sharing mail data externally.

properly extracts the departure city. On the other hand, if the departing city is not a frequent hub and is not listed in at least k itineraries on a single day release, one option might be to adjust the process so as to generate multi-day releases for less frequent entities. We reserve to future work this type of improvements, which will be needed for specific tasks that demand more details.

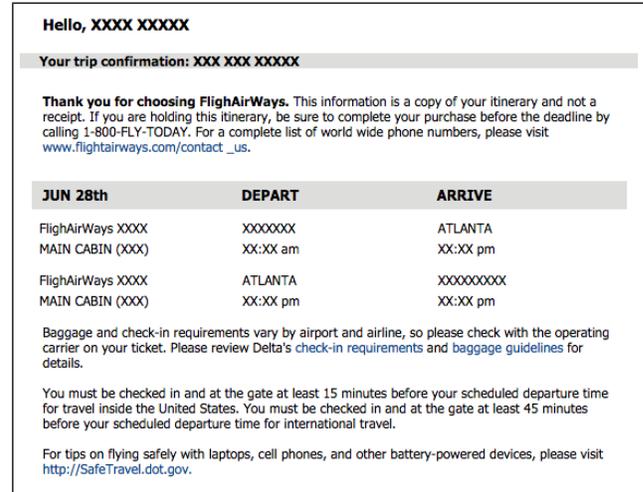


Figure 5: A masked email sample of a travel itinerary.

4.3 Templates Assignment

After the generation of the list of daily templates (where k -anonymity is guaranteed for a user with respect to a single template), the goal is now to generate a daily release globally preserving k -anonymity. The release consists of the templates that will be assigned to an auditor.

Considering a single day, our goal is thus to show at most Γ templates to the auditor while preserving k -anonymity. For each message template m obtained from an equivalence class c , we denote by U_m the set of the recipients of the messages in c . We want to avoid the worst case scenario in which presenting two different templates associated with the same user might violate k -anonymization due to the connection between their content. Note that this is clearly a worst case scenario, yet it is beyond the scope of this paper to verify whether or not the combination of different contents breaks k -anonymity. Consequently, to remain on the safe side, once a user is associated with a template shown to the auditor, the same user cannot be associated with any future template that will be shown to the same auditor.

The daily assignment process of templates to the auditor is described by Algorithm 2. The algorithm is a simple greedy algorithm, which chooses the templates at random. For each chosen template m , it chooses k random users from U_m . These will be the users associated with template m . Then, these users are deleted from the sets of users $U_{m'}$ of all other templates. This step ensures that during the assignment, each user will be associated with at most one template assigned to the auditor.

Algorithm 2 can be naturally extended to operate online over a sequence of days. For each day t , we remove users who were previously chosen (i.e., users chosen during days $1, \dots, t-1$) from the user sets of all templates created during

t. During this process, templates with a remaining set of less than k users are filtered out.

While the number of templates that can be shown to an auditor could be increased by using a different heuristic, we will show in Section 5 that this is not needed. Indeed, in our context of extremely large scale traffic covering millions of users, Algorithm 2 allows the daily assignment of templates to an auditor over a lifetime, while adhering to the constraint of assigning at most Γ templates per day.

Algorithm 2 Daily Release

- 1: **Input:** Set of templates M , parameters $k \geq 2, \Gamma \leq |M|$
 - 2: **Output:** A set $Q \leq \Gamma$ of templates to display
 - 3: **while** M is non-empty *and* $|Q| < \Gamma$ **do**
 - 4: Choose at random a template $m \in M$
 - 5: Remove m from M
 - 6: **if** $|m| \geq k$ **then**
 - 7: Add m to Q
 - 8: Choose at random a set $u_m(k)$ of k users from U_m
 - 9: Remove the k chosen users from the user set of every template $m' \in M$, update $U_{m'} = U_{m'} \setminus u_m(k)$
 - 10: **end if**
 - 11: **end while**
-

Finally, we are ready to establish the following claim.

CLAIM 1. *k-anonymity holds for the online mail audition k-anonymization problem by applying our three-phase process.*

PROOF. By construction, k -anonymity holds for each user at the template level. This follows the fact that a template represents identical masked samples associated with at least k users. Furthermore, k -anonymity holds for each daily release following the fact that a user is associated with only a single template assigned to the auditor. Finally, k -anonymity holds for the entire online process of continuous daily assignments due to the fact that this constraint is kept for the entire lifetime period: once a user has been associated with an assigned template, she/he will never be associated with any future template. \square

5. EXPERIMENTS

5.1 Dataset

We use as dataset the actual Yahoo mail traffic, consisting of billions of incoming messages received in several subsequent days, after filtering out several large domains such as facebook or yahoogroups. Indeed, these domains lack interest in our context, as although being machine generated, they mostly include personal unstructured information. For the deep analysis of the hashing and grouping techniques, we focus on a random subset of over a billion messages received in a single day.

5.2 Mail-Hash vs. Min-Hash

In order to verify the benefits of Mail-Hash, we conducted an experiment comparing our proposed technique with the well-known Min-Hash function as our baseline. We use Min-Hash with a single hash function over the mail message, even if it can use a set of hash functions, as this is not needed when using our approach. We applied both functions to the entire message, excluding only URLs, which tend to contain

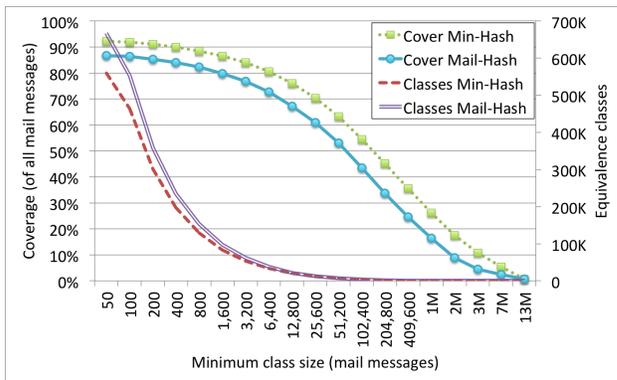


Figure 6: Mail-Hash vs. Min-Hash: compare coverage and number of equivalence classes.

long random strings, and obtained a single signature for each mail message. We then grouped all messages according to their signature, and measured the content coverage of each equivalence class, as defined in Section 2, derived from either Mail-Hash or Min-Hash values.

Figure 6 presents the coverage and number of classes obtained, as a function of the minimal number of messages in a class. It shows that Min-Hash’s coverage is slightly higher while the number of classes it generates is slightly lower. In other words, Min-Hash classes tend to be somewhat larger than Mail-Hash classes. The reason is that Min-Hash often treats containment⁶ and resemblance in a similar way, putting the contained and container in the same group, while Mail-Hash usually generates two distinct signatures, one for each, thus putting them in two different equivalence classes.

5.3 Template/Class Content Coverage

We measured the content coverage of templates (and their associated classes) induced by Min-Hash and Mail-Hash. As per the definition given in Section 2, we considered the amount of data that can be revealed to auditors, relative to the overall original amount of data, if each template was the only template ever exposed.

Figure 7 depicts the content coverage of an equivalence class induced by a template, as a function of the class size. It separates the results according to class size so as not to bias the results towards the larger ones that might just be a kind of “garbage can”. Naturally, larger classes are less useful in practice as they will expose fewer amount of shared data. The largest templates are a result of collisions that group together messages that were not necessarily generated by the same script (e.g., very short messages that are likely to have the same DOM tree but totally different textual values) and/or vary in a way that requires more masking due to the high variability of the data. The chart shows that Mail-Hash performs clearly better than Min-Hash, as it reveals a larger portion of common information.

In the experiment presented by Figure 7, we chose for our k anonymization parameter a value of $k = 25$. The same experiment was also run with values of $\{50, 75, 100\}$ leading to similar results. The reason is that this parameter influences only the minimal size of an equivalence class, but not its content coverage. This can also be seen from looking at the

⁶Containment refers to the case where a significant part of one document is contained in another, typically larger one.

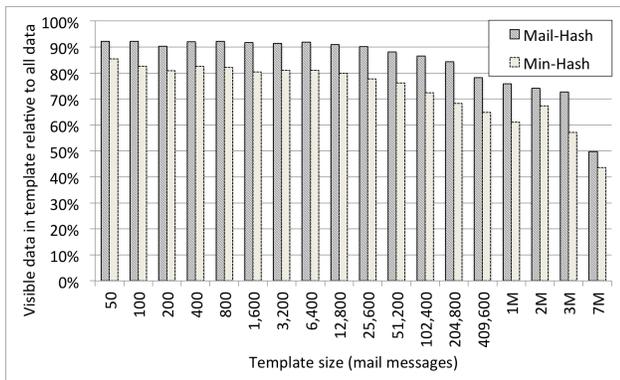


Figure 7: Mail-Hash vs. Min-Hash: the k -anonymity experiment with $k = 25$, comparing the coverage of templates as a function of the template size.

different classes in the left side of the graph that corresponds to the smaller sizes, all leading to a similar coverage of above 90%. In fact, a content coverage close to 90% is achieved for almost all classes (except the largest ones, which as explained before are valueless classes of high variability that do not contribute to our process). Consequently, the content coverage achieved for the daily release is close to 90% as well.

5.4 Deduplication in Equivalence Classes

This section presents an experiment we conducted to measure the average message similarity within each equivalence class, as an indicator for how “equivalent” our equivalence classes are.

We express the average message similarity in each class by measuring the deduplication level of k random messages in the class. For deduplication we use PACK chunking [28] which converts each message to a list of variable-size chunks. The deduplication level is calculated by initiating a pool of chunks using a random pivot message, and then adding the other $k - 1$ messages from the class to the pool, one by one. Whenever a message is added, we measure the number of distinct chunks belonging to the added message, which also belong to at least one message that was already in the chunk pool. We then average over the $k - 1$ add operations to get the total deduplication level.

Figure 8 shows the deduplication ratio as a function of the class size. Surprisingly, the average similarity achieved by Mail-Hash is always higher than Min-Hash, even though Mail-Hash covers only the DOM structure while Min-Hash covers the entire message. This indicates that Mail-Hash is able to infer chunk-level similarity without looking at the content of the chunks. In fact, it performs even better than a technique that looks into chunks and chooses some of them, as done by Min-Hash, which chooses a subset of the chunks for generating its signature.

This result also explains the higher content coverage achieved by Mail-Hash, as the more similar the messages within a class, the larger the amount of shared information that can be revealed.

5.5 Daily Releases

The last stage of our evaluation consists of demonstrating that it is feasible to generate daily releases, while preserv-

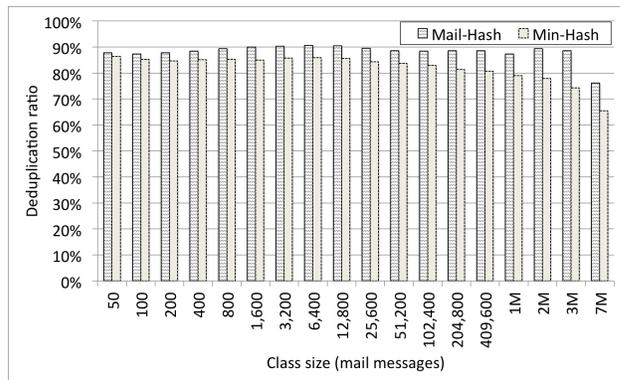


Figure 8: Mail-Hash vs. Min-Hash: compare deduplication ratio as a function of the equivalence-class size.

ing k -anonymity over time, and ensuring that auditors can safely continue review samples during their entire tenure. In addition, we demonstrate the scalability of our process by running Algorithm 2 over the actual Yahoo daily traffic during three consecutive days. Using $k = 100$ for k -anonymity, we measured the daily number of equivalence classes that were created, as well as their accumulative number, counting the same class that was created during different days only once. We used a threshold of $\Gamma = 1000$ on the number of samples that can be shown to an editor per day. Thus, due to the fact we restrict a user to be associated with a template only once (see Section 4.3), it follows that the daily number of users becoming “invalid” per day is 100,000 (that is, users that cannot be associated with a template in the future).

According to our online algorithm, such users are removed from future templates, potentially causing the filtering of templates with a remaining set of less than k users (see Algorithm 2). However, as can be observed in Table 1, the number of templates filtered per day is negligible ($\sim 250 - 630$) compared to the total number of accumulated templates ($> 720K$). Note that while the accumulated number of templates grows during the first days, it is expected to increase at a much slower pace after a while, when most of the stable equivalence classes have been identified in the system.

The ratio between the daily filtered templates and the overall number of templates (along new templates as well as new users entering the system over time) demonstrates that, in a realistic scenario over a traffic of such scale, k -anonymization can be enforced online practically “forever” over the full tenure of an auditor.

Day	day 1	day 2	day 3
# accumulated templates	466K	597K	727K
# daily templates filtered	252	352	626

Table 1: Results for experiments conducted over Yahoo traffic during three consecutive days.

6. RELATED WORK

Using k -anonymization as a method for protecting individual privacy has first been suggested by Samarati and Sweeney [20]. It was then greatly popularized and adopted

into key privacy related legislation such as HIPAA [22] and the EU Data Protection Directive [6]. k -Anonymity continues to be the single most practical tool for protecting individuals against inadvertent intrusion into their privacy. This is, in many ways, because it is simple and well understood. This same simplicity is the source of some of k -anonymity falls back.

As many have noticed (see [18]), k -anonymity breaks when there is lax control of auxiliary releases of information. In our work we claim that the exposure of internal auditors to such auxiliary releases can be limited via internal regulation. Additionally, it was noticed [14] that if data subjects are not sufficiently diverse then k -anonymity becomes ineffective. In the context of mail users this can occur with alias accounts and with inactive accounts and both those cases are evident to the Web-mail provider and can be filtered with no adverse effect on the application. Last, it was argued that the protection provided by k -anonymous releases can be de-anonymized [17] and that stronger models such as differential privacy [7] may be needed. While this is true in the general case, our concern is not protection against an adversary who invests effort in de-anonymization but rather protection against inadvertent exposure. Therefore we argue that k -anonymization provides sufficient protection for our application.

Optimal k -anonymization with respect to any non-trivial optimization criteria is NP-Hard [13]. Consequently, most k -anonymization techniques are heuristic. Our k -anonymization algorithm, like Mondarian’s [13], is a multi dimensional technique: every template represents a different dimension and different equivalence classes of messages are anonymized using different combinations of dimensions. The unique aspect of our algorithm is the choice of classes of messages that will be co-anonymized. This choice relies on a deep domain understanding (mail traffic, and mail documents characteristics), which is atypical for general purpose anonymization algorithms.

The issue of multiple sequential releases have been studied by Wang, Fung et al. [25, 8] and by Shmueli et al. [21]. Their focus, however, is on the efficiency of computing the current release given all previous releases in terms of both computation and limited disclosure. Another work is that of *multi-release publishing* [27], where the data owner needs to release the data to different recipients, each seeing a different projection of the data. The challenge in this case is to preserve the k -anonymity of previous releases when a new projection is needed.

To the best of our knowledge this work is the first to consider sequential releases in a large scale data environment. Considering billions of new daily (mail) documents, we provide a complete solution enforcing k -anonymity over time, demonstrating through experiments at Web scale that our algorithm can be applied over a lifetime period of years, while producing daily a k -anonymized release that maximizes content coverage as we defined it.

7. CONCLUSIONS AND FUTURE WORK

We have defined here the novel problem of k -anonymization for human auditing of mail messages, which we believe is timely given the growing concerns of users over the privacy of their inboxes. We focus on the specific problem of enforcing the k -anonymity of recipients of machine-generated messages, when their messages are reviewed by human au-

ditors for various quality assurance tasks. We proposed a mechanism that guarantees k -anonymization by grouping structurally-similar messages into a same equivalence class, and by generating for such classes masked mail samples that can be safely shown to auditors. To this effect, we introduced a highly efficient, yet surprisingly simple, message signature function, *Mail-Hash*, that allows to identify these structurally-similar messages, most probably generated by a same script. We also proposed a mail sample assignment process that guarantees k -anonymity of users while maximizing the auditors’ productivity, according to a generic quantitative measure of content coverage. We demonstrated over real Yahoo mail traffic that our approach is feasible and presents detailed results of our experiments.

This work represents only a first step in the direction of improving the auditing process in most large Web mail companies. We sincerely hope it will trigger more efforts and alternate solutions towards devising more automated and more sophisticated methods for protecting users’ privacy. We intend to continue exploring this venue along different directions. First, we would like to devise more qualitative measures of auditors’ productivity, as opposed to our quantitative generic measure of raw content coverage. Such measures should probably be tailored to the specific task that required auditing in the first place. We also believe that some specific tasks will demand more control of the timeframe of a release, in order to reveal specific entities in mail samples, as in our example of flight itinerary in Figure 5.

One effect of our solution is the large number of classes potentially created per domain, sometimes fragmenting the same “logical template” into multiple equivalence classes due to slight changes in their structure. While for masking and anonymization purposes, it does not represent a limitation, it could pose a problem depending on the auditorial task at hand. Aggregating such classes together within the same logical component is left for future work.

In addition, it will be interesting to verify whether productivity can be improved by considering auditors not independently but as a pool, for which sample assignment need to be globally optimized. Finally, the next big challenge will be to tackle the much harder task of considering k -anonymization of highly variable human-generated messages.

Acknowledgements

We are grateful to our colleague Doug Sharp who motivated this problem and convinced us of the urgency of proposing principled methods for safe mail auditing. We owe a lot to Zohar Karnin for his constant insights. We also cannot thank enough our friends and colleagues in the Mail research and engineering teams at Yahoo, without whom this work would not have been possible.

8. REFERENCES

- [1] Nir Ailon, Zohar S. Karnin, Edo Liberty, and Yoelle Maarek. Threading machine generated email. In *Proceedings of WSDM’2013*, pages 405–414, New York, NY, USA, 2013.
- [2] Andrei Z Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE, 1997.

- [3] Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, COM '00, pages 1–10. Springer-Verlag, 2000.
- [4] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.
- [5] Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166, 1997.
- [6] EU Directive. 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the EC*, 23(6), 1995.
- [7] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2006.
- [8] Benjamin Fung, Ke Wang, Ada Wai-Chee Fu, and Jian Pei. Anonymity for continuous data publishing. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 264–275. ACM, 2008.
- [9] Google. Privacy policy, Jun 2015. <http://www.google.com/intl/en/policies/privacy/>.
- [10] Mihajlo Grbovic, Guy Halawi, Zohar Karnin, and Yoelle Maarek. How many folders do you really need? classifying email into a handful of categories. In *Proceedings of CIKM'2014*, 2014.
- [11] Nevin Heintze. Scalable document fingerprinting. In *Proceedings of the USENIX Workhop on Electronic Commerce*, 1996.
- [12] Monika Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291. ACM, 2006.
- [13] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proceedings of the 22Nd International Conference on Data Engineering*, ICDE '06, pages 25–50. IEEE Computer Society, 2006.
- [14] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1):3, 2007.
- [15] Udi Manber. Finding similar files in a large file system. In *Proc. of the USENIX winter technical conference*, pages 1–10, 1994.
- [16] Microsoft. Services agreement, Jun 2014. <http://windows.microsoft.com/en-us/windows/microsoft-services-agreement>.
- [17] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 111–125. IEEE, 2008.
- [18] Paul Ohm. Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization. *UCLA Law Review*, 57:1701–1777, 2010.
- [19] R. Rivest. The MD5 Message-Digest Algorithm. *RFC 1321*, April 1992.
- [20] Pierangela Samarati and Latanya Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical Report SRI-CSL-98-04, 1998.
- [21] Erez Shmueli, Tamir Tassa, Raz Wasserstein, Bracha Shapira, and Lior Rokach. Limiting disclosure of sensitive data in sequential releases of databases. *Information Sciences*, 191:98–127, 2012.
- [22] US Dept. of HHS. Standards for privacy of individually identifiable health information; final rule, August 2002.
- [23] W3C. XML Path Language (XPath) Version 1.0. <http://www.w3.org/TR/xpath/>, November 1999.
- [24] W3C. Document Object Model (DOM) Level 3. <http://www.w3.org/DOM/DOMTR#dom3>, April 2004.
- [25] Ke Wang and Benjamin Fung. Anonymizing sequential releases. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 414–423. ACM, 2006.
- [26] Yahoo. Privacy center, Sept 2014. <https://policies.yahoo.com/us/en/yahoo/privacy/index.htm>.
- [27] Chao Yao, X Sean Wang, and Sushil Jajodia. Checking for k-anonymity violation by views. In *Proceedings of the 31st international conference on Very large data bases*, pages 910–921. VLDB Endowment, 2005.
- [28] Eyal Zohar, Israel Cidon, and Osnat Mokryn. The Power of Prediction: Cloud Bandwidth and Cost Reduction. In *Proc. of SIGCOMM*, 2011.