Applications of k-randomization

To clarify the use of k-randomization for Mozilla specific tasks we consider examples that contrast applications of known techniques and k-randomization to collecting sensitive user data. Before presenting the examples, I would like to emphasize the general applicability of k-randomization.

k-randomization is a technique that protects privacy of collections of multivariate user records encoded as bit vectors. Search queries, home pages, user interests, user demographics, user behavioral characteristics all fall into that type of records. Regardless of how such data is collected, storing data sets of multivariate user records in clear text is a privacy risk. Should such dataset be subpoenaed or compromised, and should specific users be linkable through other auxiliary data sources - Mozilla will be exposed. Therefore, user records must contain noise to hide the presence of a specific user record in the collection. k-randomization answers the question of how to choose randomization noise to satisfy both a privacy guarantee and business requirements for accuracy.

For more details on k-randomization, see:

- Executive summary on k-randomization

Contents

Study setup

Evaluation of k-randomization

For each of the applications presented below, we compare the performance of the k-randomization technique to that of a benchmark technique. Randomization is preferable to alternative methods of protecting privacy of individual client records for various reasons, discussed in relation to each application. As such, we choose RAPPOR as a benchmark since it is an established technique for applying randomization to individual client records, subject to a differential privacy guarantee - in that sense, k-randomization is an extension of RAPPOR.

Our comparisons for each application proceed as follows. We consider collecting records from populations of 10M users. The differential privacy guarantee is set to 2. We compute the required randomization noise level for every technique to satisfy the privacy guarantee. Once the noise settings are determined, the techniques are compared in terms of the smallest fraction of the population that can be measured to within 5% error by each technique.

When the application permits, we also consider disaggregated hashes and homomorphic encryption as alternative methods of privacy protection. Mathematical details and formulas are relegated to the Appendix.

Summary of methods

Method	**Infrastructure required**	**Noise/ precision**	**Differentially private**	**Applicability**
RAPPOR	none	relatively high, hinders estimation	yes	multidimensional data (although increasing dimensions incurs significant privacy cost)
k-randomization	anonymization/ mixnet	relatively low, can be calibrated against precision	yes	multidimensional data
Disaggregated hashes	anonymization/ mixnet	none - not randomized	no	unidimensional data
Homomorphic encryption	encrypted aggregation, decryption	none - not randomized	no	unidimensional data, low cardinality (increasing cardinality incurs significant infrastructure cost)

Note on differential privacy

By definition, differential privacy means hiding users whose values are rare, ie. different from most of the population. Randomization techniques do this by adding enough noise to thwart accurate estimation for values only shared by small proportions of the population. This is a feature, not a shortcoming. The value in using differentially private techniques is that they provide a quantifiable, comparable degree of privacy, which is widely understood and accepted in the community.

Home page collection

Collection of user home pages has been on Metrics radar for quite a while. In fact, RAPPOR implementation was submitted to Mozilla via bug **1138022**. We will take a deep look into various ways how home page data can be collected and will compare the main candidate methods (homomorphic encryption, disaggregated hashes, RAPPOR and k-randomization) to gain a clear understanding of their powers and shortcomings. We work under the setup described above: differential privacy budget is 2, sample size is 10M, and error limit is 5%. We gauge method effectiveness using the smallest fraction of the population that can be measured within 5% error provided privacy still satisfies the guarantee.

Original RAPPOR algorithm randomizes hashes of home-page URLs, but that procedure requires sendings full-length bit-vectors which precludes comparison of non-RRT methods. Hence, we change homepage collection procedure in a way to make homomorphic encryption and disaggregated hashes applicable. We send a list of 50,000 most popular hosts (ETLD+1) to each browser. If the client's homepage is in the list, it reports back a data structure specific to the method being used.

If collection is done via RRT method (RAPPOR or k-randomization), a browser may send back a 50K bit vector (6KB per record). If the list of sites for which randomization reports 1 is short, a browser can simply send back the list of chosen sites. For homomorphic encryption, a browser should, theoretically, encrypt every bit of every site, and send back 50K of encrypted bits. This is clearly not feasible, hence homomorphic encryption has to choose a subset of sites which bits are encrypted and sent. The choice and the number of sites to report back under encryption does raise serious privacy issues which we reflect upon in the appropriate section.

One-time RAPPOR

Given that a browser has only one homepage, the bit-vectors subjected to randomization are mutually-exclusive: they may only have a single bit set. With respect to differential privacy, mutually exclusive bit-vectors have much looser requirements for minimal noise to meet set privacy guarantee. Under our privacy budget of 2, we find the required noise level (probability of lying) to be 0.25.

Further, we find that the number of users with a specific homepage set can be estimated within 5% error if the homepage is set in at least 0.018 of the population of 10M, ie. we can measure homepages that only 1.8% of user base has chosen. In concrete terms, homepages with at least 180,000 users will be reliably identified. The overall size of collected data is about 60G, which seems acceptable, and no infrastructure or extra processing cost is incurred. If 1.8% resolution is sufficient for the task, RAPPOR could be a winner. However, if this data collection is to be repeated, precision will suffer due to the necessity to perform 2-step RAPPOR to maintain privacy.

K-randomization

We now show that if an anonymization infrastructure is available, k-randomization provides an order of magnitude improvement over RAPPOR precision and size. The required noise level for a single report k-randomization is found to be 0.00015. With that noise level, the measurable fraction of the population is 0.00023.

We can measure homepage audiences down 0.02% of the population, meaning that home pages that have only 2000 users can be reliably identified. K-randomization also enables a radical data-size decrease. On average, about 8 sites will be chosen for report, hence a browser only needs to send back 8 identifiers (instead of 50K bit-vector), which is 16 bytes. The total collection size is 0.16G - 375x improvement.

One thing to note here is that the k-randomization noise level of 0.00015 seems incredibly low from the point of view of the user. From their reported bit vector, only 0.015% of the bits will get flipped by the randomization, and the others are reported as is. However, the privacy protection of the k-randomization scheme is derived from each record's membership in a broader collection of anonymized user records. Suppose google.com occupies 10% of all home pages. If a user record does contain google.com, then with probability 0.998 that record’s homepage is google.com. But this is not a privacy concern, since there are ~1M other anonymized records containing google.com. On the other hand, homepages with low counts are indistinguishable from homepages with no counts. Even if no user subscribes to a homepage, the homepage will still exist in ~1500 records by chance, making it virtually impossible to tell holders of unique homepages. In fact, it is just as impossible to tell the presence of unique homepages - the estimations for rare pages will range anywhere between -120 and 120, which again makes pages that have less 120 users indistinguishable from homepages with zero counts.

However, the value of the randomization noise level q determined by the k-randomization method is a technical lower bound required to satisfy the set privacy guarantee. Using any larger q will still satisfy privacy, although incurring a penalty in precision. This is relevant in the case where we wish to impose additional policy-level constraints on our randomization methodology. For example, it is conceivable that from the point of view of the user, a randomization level of 0.00015 is too low, and our legal/policy team requires a noise level of at least 0.1. We can still proceed with k-randomization, subject to the same privacy guarantee (improving on it in fact). In this case we will be be able to identify homepages set by 0.7% of the population (70000 users), which is still an improvement over RAPPOR.

Another important point here is that we are using the simplest possibly randomization scheme: randomize each bit from each user independently according to the same distribution. Using a more sophisticated randomization scheme which incorporates domain knowledge relevant to application can be used to optimize the tradeoff between privacy and precision. For example, we know that the distribution of users' homepage preferences is highly skewed, and we can obtain a good guess of the distribution by referring to public site rankings (eg. Alexa). Weighting the randomization according to this distribution enables the privacy bias related to common homepages (ie make it less likely that you actually have google.com set if you report it).

Disaggregated Hashes

The disaggregated hashes technique simply sends a home page site through the anonymization infrastructure. On the surface, this makes a lot of sense: why bother with randomization if a client can simply send its home page host through a mix-net. The server would collect hosts from 10M clients anonymously and directly compute number of users per each homepage. If the measurement precision is the gating factor, why add noise?

This is actually a policy rather than an implementation issue. The very fact that homepage counts are exact causes the collected dataset to lose differential privacy. It contains (potentially many) pages with count 1 - Mozilla knows that there are a number of unique users in the sample that have unique homepages. Should such dataset be subpoenaed or compromised, and should these unique users be linkable through other auxiliary data sources - Mozilla's users are exposed. Anonymization alone does not protect user data privacy in a measurable way. The added noise provides a level of permanent protection to the user records once they have left the client.

Within the differential privacy paradigm the measurement precision is the gating factor only when privacy guarantee is met. We can make “disaggregated hashes” technique differentially private by having a client to lie about the real host and send some fake ones, but then “disaggregated hashes” become a randomization algorithm.

Homomorphic encryption

Homomorphic encryption appears to have the advantage of providing pan-privacy - no actor ever sees a naked bit. It is not exactly true - some home pages will be unique and will appear so to both Mozilla and third-party decryptor. For the unique homepages homomorphic encryption provides no measurable privacy protection. Aside from that issue, homomorphic encryption for homepage collection raises numerous other privacy concerns.

Privacy considerations

Since it is infeasible to submit 50K of encrypted bits from each client, a smaller number of sites has to be chosen. Suppose, the client choose 20 and encrypts corresponding bits. The real homepage host will have to be among those 20, otherwise bit sums will be inexact. The client has to attach hosts to the encrypted bits, otherwise the adder will not know which host to add a bit to. This immediately reveals a privacy leak: when server receives a message with 20 encrypted bits, he knows that a user homepage is one of those 20, and NOT the remaining 49980 sites. It may sound idiosyncratic, but the RRT methods do not have such problem - a user always has a deniability claim.

Another serious issue is a distribution skew. Suppose 1M users (10% of the population) has google.com as home page. Assuming the random choice of 20 sites, there will be about 3600 users that do not have google.com as home page but still report it in the site list. If the server sees a record containing google.com, the probability of google.com being a homepage is:

$$\frac{1,000,000}{1,003,600} = 0.996$$

Not very private. Perhaps we can do better by choosing the 20 sites to add according to a distribution similar to that of actual homepage preference. But we do not know which sites are popular homepages until we collect data….

An even more serious issue is a privacy attack that Mozilla can mount against users with unique home pages. Mozilla simply finds a site that has only been reported by a single user (aggregate count of 1). Mozilla then finds the record that reported this site, and has thus identified the user who has this homepage. Even if the rare site occurs few times in encrypted records collection, Mozilla still knows that one of a few users has the site as his/her home page, and this is a definite privacy leak.

From a purely theoretical angle, the collection of partial site lists (even when bits are encrypted) are not private. If there are two users with different homepages, one may generate a list that does not include a home page of the other. In which case, the second user will not be able to generate same list as the the first one. We have a case when a report from one user is not possible from another. This makes the differential privacy guarantee unattainable.

In general, homomorphic encryption generates two datasets - one of site lists with encrypted bits and the other with the exact counts. Neither of these data sets is differentially private, and their combination opens a path to severe privacy attacks. An even worse consequence of the homomorphic topology is that the third-party decryptor can construct a copy of the final data set.

Ignoring the infeasibility issue and letting the client send the full list of sites, there is another attack that Mozilla can mount. Instead of adding bits under encryption, Mozilla simply submit individual records as they arrive to the third-party decryptor. The decryptor sends back accumulated counts, Mozilla waits for one of the sites counts to switch from 0 to 1, at which point a record causing the switch is identified. It is possible to make the decryptor smarter by flagging records with just a single site count, etc., but this is not a viable solution since Mozilla can always beat this by submitting altered records to the decryptor. For example, Mozilla can add known numbers to sites of a single record and then subtract them from accumulated counts - the decryptor would not be able to tell a legitimate record from a tampered one.

Indeed, most of the privacy issues are caused by infeasibility of encrypting all 50K bits and the need to choose a reasonably small subset to report. But if homomorphic encryption works only for small cardinality datasets, and is only capable of adding single bits, and incurs very high infrastructure cost (due to 0-1 proof cost), then its general applicability to Mozilla's privacy protecting data collection needs is doubtful.

Given the existence of better privacy preserving and less expensive alternatives that cover much wider spectrum of applications, homomorphic encryption appears to be the least preferable method. If the exact counts are absolutely required, disaggregated hashes technique will perform more cheaply and with fewer risks.

Having described the various issues, we must admit that our understanding of homomorphic encryption is rudimentary. It’s entirely possible that the security team have solutions for the above problems which we are unaware of. We would love to learn what they are, and will be happy to assist with privacy math to make homomorphic encryption viable.

Search default switches

An important question for BusDev is to understand why certain Firefox users choose to switch their search default away from Yahoo. Characterizing this group in terms of certain demographics and behavioral characteristics, in addition to the general usage measurements available in FHR, will help us understand the audience most affected by the change of search partner, with the ultimate goal of optimizing their Firefox experience within the context of our business directions. Although there are clearly many user features that are potentially relevant, for the purposes of this discussion we propose an initial study recording the following features:

- **Switched default search engine away from Yahoo**: Yes/No
- **Gender**: Male/Female
- **Age group**: Young/Mid-Aged/Elder
- **Technically advanced user**: Yes/No
- **Site diversity**: Low/Mid/High
- **Visitation frequency**: Low/Mid/High
- **Search frequency**: Low/Mid/High
- **Preferred search access point**: Search Bar/Url Bar/New Tab/about:home

It’s important that full-length user record is reported because we need to perform correlational analysis on user features. Specifically, we want to explain the differences between the group who switched and those who didn't in terms of their joint collections of features.

The need to keep user bits together precludes the use of anonymization techniques that split bits apart (e.g. homomorphic encryption), suggesting randomization techniques as a viable alternative. We focus on k-randomization and RAPPOR, using the setup described above.

One-time RAPPOR

RAPPOR assumes users to be tracked and requires record-level protection. We consider the case of RAPPOR with one-time RRT randomization, which provides optimal estimation precision. It’s a general property of RRT that features with mutually exclusive user values occupy only 2 bits for the purpose of meeting privacy guarantees. Hence the vector length reduces to 13 bits (instead of 19). The required randomization noise (probability of lying) is 0.46.

For a 5% error limit and the required privacy guarantee of 2 on a population of 10M, we can only detect events that happen no less than 2M times (ie. in 21% of the population). This is not very informative for something like homepages where we expect to see only a few pages achieving that level of popularity, and the vast majority of pages only set by a handful of users. Even though RAPPOR does not require anonymization infrastructure, the noise needed to protect individual records is so high that it limits RAPPOR usefulness for this use case.

K-randomization

We now contrast RAPPOR performance with k-randomization. k-randomization does need anonymization infrastructure, which is understandably expensive. However, should Mozilla choose to build one, it can drastically improve “statistical power” of randomization by exploiting the size of its user base.

Under k-randomization, the required randomization noise is 0.2. We more than halved randomization noise, which gives us a huge boost in accuracy.

For ***q=0.2*** the corresponding fraction of total population that can be estimated within 5% is 0.021. This is a 10 fold improvement in the “statistical power” under the same privacy guarantee. We now can estimate events that happen 210,000 times. However, this limit can be further improved if we are allowed to repeat randomizations a number of times.

For the sake of example, assume we can report 4 randomizations. If ***k=4***, the corresponding RRT noise level is ***q=0.25***. The required noise level did increased to cover up for repeated randomizations. However, we gain overall precision increase, because estimation deviation reduced by sqrt(k). The corresponding alpha for this case is: 0.014. We are now capable of measuring 1.5% of population with required precision and still under same privacy guarantee. And these numbers are computed under a very limiting assumption that all 10M users are identical and there’s one outlier. For realistic data sets we can do substantially better, which may effectively eliminate the need of multiple randomizations all together.

Tile impressions and click data collection

We briefly touch on the targeted tile optimization problem that Content Services are currently facing. A user is served a suggested tile if his history contains one of the sites being targeted by the tile. Some targeted sites generate better click-through-rates than others. Content Services need to optimize targeting campaigns by estimating impressions and clicks per targeted site, and removing sites with low click-through-rate from the campaign.

Collecting Impressions

Since every user with a targeted site in the history is likely to see a corresponding suggested tile, the number of impressions per site will be proportional to number of users that have a targeted site in browser history. If the number of users per each targeted site is known, we can estimate how tile impressions will be distributed among targeted sites with reasonable accuracy. Therefore, we employ a procedure similar to homepage estimation to count numbers of users for 1650 sites currently being used for targeting. We ask each client to find visited sites from the targeted list, and report at most 10 of them.

RAPPOR

Using RAPPOR, we require a randomization noise level of 0.47. This allows us to estimate counts to within 5% error for sites common to at least 18% of clients.

The majority of sites in the list will fall way below 18% user count. The noise level of RAPPOR is pretty high, which causes unacceptably low precision.

K-randomization

We observe how the parameters change as we increase the number of replicated reports.

k=1	k=6	k=10	k=100
$q = 0.27$ $\alpha = 1.8\%$	$q = 0.309$ $\alpha = 0.9\%$	$q = 0.315$ $\alpha = 0.7\%$	$q = 0.34$ $\alpha = 0.2\%$

It’s instructive to see k-randomization performance if the sample size is allowed to grow to full population. There are 64M users in US, and reporting from all of them gives us yet another major boost in precision:

k=1	k=6	k=10	k=100
$q = 0.25$ $\alpha = 0.6\%$	$q = 0.29$ $\alpha = 0.3\%$	$q = 0.297$ $\alpha = 0.2\%$	$q = 0.32$ $\alpha = 0.1\%$

K-randomization performs reasonably well, and allows for an increase in precision at the expense of extra reports and/or the sample size. Since user bits are kept together, this dataset has an additional (and valuable) property of enabling correlational analysis between targeted sites and targeted user audiences. It’s very common to target intersections of audiences (like sports and travel), which are directly computable from the collected data.

It’s also worth mentioning that these noise levels are chosen to support a theoretical upper bound of privacy. K-randomization can take full advantage of the underlying distribution of the data being collected to reduce the noise even further.

Size-wise, an individual record would contain 200 bytes, and even if the full population reports 100 times, the collection size is 1.2T which does not seem unreasonable.

Disaggregated Hashes

This technique will definitely work, but it requires the anonymization infrastructure, which brings the question why settle for individual bit counting. Since the anonymization infrastructure has to exists to support disintegrated hashes, we can as well collect full records with required precision, provide our users real privacy guarantees, and perform all sorts of interesting correlation analytics and audience sizing using k-randomization data set.

Homomorphic encryption

Homomorphic encryption will suffer from same issues described in homepage collection use case. 1600 encrypted bits are still too many to be feasible, hence the same list of privacy/cost issues.

Collecting clicks

A site's click through rate depends entirely on the appeal and the message of a particular tile. Since click distributions are very campaign-specific, we will have to collect clicks for every tile, which presents a non-trivial privacy risk: same user may click on multiple tiles that target same group of sites. RAPPOR solves this problem using 2-step randomization, which may skew our estimations because changes in a user history will never be reported. K-randomization solves this problem by adjusting RRT noise to account for potential multiple clicks from the same user.

Suggested tiles campaigns rarely reach 10M users. It usually reaches about 1M users, and only 0.5% of them click. Hence, we only measure clicks from 50,000 users. We also do not need a razor sharp precision - measuring with 25% error will work fine for optimization.   Since there's no specific need to correlate, counts of targeted sites that happen in the history when the clicks occur can be submitted individually.

RAPPOR

RAPPOR doesn't have a luxury of sending separate records for each history site. The randomized version of the full targeted list must be submitted. Assuming no more than 3 targeted sites are ever reported, RAPPOR's noise and alpha (minimum detectable population proportion) are:

q = 0.417
alpha = 16%
minimal audience size = 8000

RAPPOR might work for short list of equally popular targeted sites. RAPPOR may be too noisy for longer lists of targeted sites, especially if popularity is skewed.

K-randomization

Due to presumed existence of an anonymizer, K-randomization can send randomizations for each targeted site found in the history when a click occurs. But k-randomization needs to adjust for the possibility of the same user clicking on different tiles that target the same group of sites. This adjustment is achieved by assuming that **every** user always clicks 3 times on similarly targeted tiles. In this case, the parameters are:

q = 0.0027
alpha = 0.002
minimal audience size = 140

This actually will do very well for the current optimization needs because targeted sites lists range from 10 to 70 sites.

Disaggregated hashes

Again, this technique definitely works and does produce exact counts, but still has an issue with generating unprotected collection due to its exactness. The privacy can be protected by adding noise to reported bits, but then it becomes the same algorithm as k-randomization.

Homomorphic encryption

Homomorphic encryption can actually help with this use case, because targeted lists are short. On the other hand, investing in homomorphic encryption infrastructure just for one use case seems uneconomical, especially when cheaper and more private techniques exist.

Search queries

10M users are asked to submit 10 most frequent queries they issue. A browser hashes each query into 28 bits hash, randomizes the hash and sends in 10 randomized hashes to Mozilla. Mozilla collects 100M randomized hashes to determine 100K most popular queries.

Disaggregated hashes and homomorphic encryption are inapplicable for this use case, since bit-vectors of hashes can’t be broken into separate bits.

RAPPOR claims that queries with frequency of 10% can be distinguished from noise, which makes queries issued by less than 10M users unidentifiable.   This is unlikely to provide sufficient precision for query collection, although RAPPOR can take advantage of distributional properties of query hashes.

K-randomization

K-randomization takes advantage from the fact that query hashes are distributed uniformly. Differential privacy limits are computed under assumption that at least 5% of 100M queries are distinct. This is a very restrictive assumption, since the number of distinct queries probably ranges above 90%, but we want to insure hard privacy. This yields a randomization noise parameter of 0.075.

This yields an alpha value (estimable proportion of the population) of 0.00024. This is actually a very good result, since we do not need to measure the exact counts of each query, but rather rank queries by popularity and select a small subset of most popular ones. For which task, the above precision is more than adequate. For predominant majority of queries we can tell their occurrences with +/- 400 error, which will provide for fairly precise ranking of highly used queries.

Size-wise, we are looking at 200M of 40 bit hashes, which fits 400 megabytes of storage for 10M users. Even if we ran the algorithm on full Mozilla user base, this is still under 10G data size, which is actually a fairly small dataset.

Appendix: Further details

Home page collection

For one-time RAPPOR, the privacy guarantee is ensured by inequality below:

$$\left(\frac{1-q}{q}\right)^2 \leq e^2$$

From here, we have the bound for RAPPOR noise ***q***:

$$q \geq \frac{1}{1+e} = 0.25$$

Since we only concerned with the distribution of the number of users who have a specific homepage, the corresponding sigma is:

$$\sigma = \sqrt{\frac{q(1-q)N)}{(1-2q)^2)}}$$

Here, ***N*** is the sample size and equal to 10M. For an acceptable error level of 5%, we look for the smallest fraction of ***N*** such that a 3σ -confidence interval stays within 5% deviation from the real value. Denoting this fraction as α, we express acceptable error condition in the inequality below:

$$\alpha \geqslant \frac{3\sigma}{0.05N} = \frac{3}{0.05} \cdot \sqrt{\frac{q(1-q)}{N(1-2q)^2}} = 0.018$$

The required noise level for a single report k-randomization is governed by inequality below:

$$\left(\frac{q}{1-q}\right)^2 N \geqslant \frac{9}{(e^2-1)^2}$$

For here, the required noise is given by:

$$q \geq \frac{1}{1+\sqrt{\frac{(e^2-1)^2 N}{9}}} = 0.00015$$

Search default switches

RAPPOR's privacy requirements are governed by the inequality below:

$$L \cdot ln(\frac{1-q}{q}) \leq 2$$

Here, ***q*** is RRT noise (probability of lying) and ***L*** is the length of bit-vector adjusted to account for mutually exclusivity of user values. From here, the noise parameter is computed:

$$q \geq \frac{1}{1+e^{2/13}} = 0.46$$

Reconstructing covariance matrices for two user sub-groups require estimations of 3-feature tuples counts. The deviation of such estimates is roughly:

$$\sigma = \sqrt{\frac{3q(1-q)N}{(1-2q)^2}}$$

Here, ***N*** is the sample size and equal to 10M. For an acceptable error level of 5%, we look for the smallest fraction of ***N*** such that a 3σ-confidence interval stays within 5% deviation from the real value. Denoting this fraction as α, we express acceptable error condition in the inequality below:

$$0.05 \cdot \alpha \cdot N \geqslant 3\sigma$$
$$\alpha \geqslant \frac{3\sigma}{0.05N} = \frac{3}{0.05} \cdot \sqrt{\frac{3q(1-q)}{N(1-2q)^2}}$$

Using ***q=0.46*** and ***N=10M***, we find that $\alpha \geqslant 0.21$.

For k-randomization, the privacy requirement that governs the k-randomization scheme is given below:

$$\left(\frac{q}{1-q}\right)^L kN \geqslant \frac{9}{(\sqrt[k]{\lambda}-1)^2}$$

Here, ***k*** is the number of randomization repetitions and λ is a privacy ratio. λ is exponent of the privacy guarantee - $\lambda = e^2$. For a one-time report ***k=1,*** and we arrive to the noise parameter ***q*** in the expression below:

$$q \geqslant \frac{1}{1 + \sqrt[L]{\frac{(e^2-1)^2 N}{9}}} = 0.2$$

The analogous estimation accuracy computation gives $\alpha \geqslant 0.021$ for k = 1, and $\alpha \geqslant 0.014$ for k = 4.

Search queries

It can be shown that if there are ***D*** uniformly distributed hashes, the privacy guarantee is met when:

$$(1-q)^L - e^2 q^L \leq \frac{(D-1)(e^2-1)}{2^L}$$

where ***L*** is hash length and equal to 28, and ***D*** is a number distinct queries equal to 5M (5% of 100M queries). From here, desired ***q*** is:

q=0.075

The estimation of query occurrence is done by computing a likelihood estimator based on a prior dictionary of queries. For a known query hash, one computes the sum of probabilities between a query hash and every randomized hash in the collection.

$$E(Q) = \sum_{i=0}^{N} (1-q)^l q^{L-l}$$

It can be shown that if number of query occurrences is ***T***, then

$$E(Q) = ((1-q)^2 + q^2)^L \cdot T + \beta \frac{N-T}{2^L}$$

β controls how skewed the distribution of queries is, and to be on a conservative side, we set it to 100. From here, we derive our usual precision measurement.