Design a system that provides Web Analytics without tracking

Features:

- Give users differential privacy guarantees
- Provide better quality analytics
- No new organizational players required
- Practical to deploy

Background Information

Websites use third party web analytics for:

- Information about users
  - Demographics
  - Visits to other sites
  - Behavior

Web Analytics services TRACK individual user browsing behavior across the web

- Violation of user privacy

To overcome the privacy issue:

- Use tools that block tracking
- Use anti-tracking legislation and standards such as Do-Not-Track

Privacy Trade – off:

Improve user privacy vs. Quality of Web Analytics

Website publishers use web analytics information to:

- Analyze their traffic
- Optimize their site’s content

Web Analytics software programs:

- awstats.sourceforge.net
- openwebanalytics.com
- piwik.org
  - Video
Statistics about users on their site:

- Page views
- Clickstreams
- Browsers
- Operating Systems
- Plugins
- Frequency of returning visitors

*Do not provide users demographics*

- Third party data aggregators
  - comScore
  - Google
  - Quantcast
  - StatCounter

Data aggregator

Collects data from users visiting a publisher’s web site. Presents the data in aggregate form to the publisher. Data is collected across many publisher websites.

Publishers

Need to install a small piece of code provided by the data aggregator. Can learn statistical information they could not obtain from other server logs.

Privacy Concerns:

Tracking enables a data aggregator to compile detailed behavior of individual users and infer their demographics.

Data aggregators are thus highly criticized.

Industry Self Regulation to provide

- Opt-out mechanisms
  - BlueKai Consumers
  - BrightTag ONE-Click Privacy
  - QuantCast Opt-Out
- Do-Not-Track initiative in the W3C
- Client-side Tools to implement DNT
  - Google
  - Safari
  - Mozilla Firefox
- Client-side Tools to prevent tracking outright
  - Abine
  - EasyPrivacy
  - Ghostery
  - Internet Explorer 9 Tracking Protection Lists
**Consequences of Privacy:**

Increased protection degrades the ability of data aggregators to provide extended analytics to the publishers.

Even with tracking, inferring accurate user demographics is a difficult task → inconsistent results.

**Model Proposed**

Design and implementation of a practical, private and non-tracking web analytics system

Allow publishers to directly measure extended web analytics

Provide users with differential privacy guarantees

**How the System Works**

User information is stored in a database on the user device.

Exploit direct communication between the publisher and the users

The publisher distributes database queries to clients

Publisher acts as an anonymizing proxy for the answers from clients to data aggregator

Aggregator aggregates anonymous answers and provides results to the publisher.

Differentially – private noise is added to the data by the publisher and aggregator.

**Advantage:**

Publisher as a proxy → Avoid introducing a new organizational player in the architecture

**Disadvantage:**

New technical challenges due to malicious publishers
Goals and Assumptions

1. Publishers should get more accurate and more types of web analytics information than they do today.
2. Data aggregators should obtain web analytics information for all of their partner publishers like they do today, as an incentive for performing aggregation.
3. The system should scale adequately, and ideally be as or more efficient than today’s systems.
4. The system should not allow clients or publishers to manipulate results beyond what is possible today.

User Privacy Goals

User data should be protected within the formal guarantees of Differential Privacy.

- Each client should know its own privacy loss (defined by DP).
- DP is conservative.
- Assumes that the attacker may have arbitrary auxiliary information.
- DP’s privacy loss measure assumes a static database.
- In the current setting, the “database” is dynamic as the user population for a publisher changes constantly.
- DP is considered valuable for this model.
- Provides a worst-case measure of privacy loss.
- Noise added to answers raises the bar for the attacker while providing accuracy for aggregate results.

Differential Privacy: Aims to provide means to maximize the accuracy of queries from statistical databases while minimizing the chances of identifying its records.


Non-Tracking Web Analytics

Non-Goals

Publishers can legitimately obtain information from users

- Personally identifiable information
- Shopping activity
- Friends

This paper takes into account information that users DO NOT provide directly to publishers.

Assume that the data aggregator requires only aggregate data.
**Trust Assumptions**

- Users trust the client software in terms of locally stored data and its operation.
- Malicious client towards the publisher and aggregator.
- Distort aggregate results through click fraud.
- Violate privacy of other users.

- Honest but curious: Obeys the prescribed operation, but may try to exploit any information learned in the process.
- Does not collude with the publisher.
- Non-collusion characteristics stated in a privacy statement.
- Browser could refuse a data aggregator without a privacy statement.

**Data Aggregator**

- Assumed to be selfishly malicious towards users and aggregators.
- May try to violate the privacy of users.
- Randomly query clients.
- Drop selected client answers.
- Perform repeated queries to an isolated client to overcome the DP noise.
- Falsify results it gives to the data aggregator.

**Incentives**

- For Publisher and Aggregator:
  - Richer and more accurate analytics
  - No incentives for the users
  - Privacy could be an incentive
  - The publisher, aggregator and browser should be motivated to provide better privacy to users.
  - Focus on stating feasible incentives
System **Overview**

System comprises three entities:
- **Client**
- **Publisher**
- **Data aggregator**

**Publisher:**
- Distributes queries to clients
- Proxies client-aggregator communication
- New software installation required
  - Many publishers already run their own analytics software
  - Hosting companies already offer servers with web analytics software pre-installed

**Client:**
- Gathers and stores information in a local database
- Answers publishers' queries using the database
- Data is obtained from web pages the user visits
  - Demographics information
  - Browsing behavior
  - Non-user related information

Queries distributed to clients are posted in well-known URLs in the publishers’ websites.

When clients visit a website, they read the queries.

Queries may be formulated by the publisher and the data aggregator.

Answers to queries are limited to 'yes' and 'no'.

Answering mechanism achieved through **Buckets** that correspond to a potential answer value.

Query results are mapped to these buckets.

The aggregator generates a per-bucket histogram of user counts.

Usage of buckets limits the distortion that malicious clients can impose on aggregate results.

Each generated answer is separately encrypted with the public key of the data aggregator.

'No' answers are omitted at the client to reduce cryptographic operations → clients generate a specified number of answers ‘yes’ or ‘null’
Publisher

Collects encrypted answers from clients
Generates DP noise separately for each bucket (as additional answers) and mixes the real and noise answers.
Forwards all answers to the data aggregator

Aggregator

Decrypts the answers
Computes the histogram of bucket counts
Adds DP noise to each count
After signing the result, it transmits the counts to the publisher

Publisher

Subtracts the noise it originally added to obtain final counts

Finally, the publisher and data aggregator, both obtain aggregate results for each query.

The results obtained are not exact because of the added noise.

This precaution prevents the publisher and aggregator from computing noise-free results

Clients can audit publishers to detect if one of them is dropping client answers

Auditing process:
The client generates and encrypts a nonce
Transmits the nonce to the publisher instead of the answer the client would have sent.
The client sends the nonce to a randomly selected publisher.
The random publisher forwards the nonce report to the data aggregator
If the data aggregator receives a nonce report without the corresponding nonce answer, the publisher is suspicious of dropping client answers.
A computation \( C \) provides \((\varepsilon - \delta)\)-differential privacy if it satisfies:

\[
\Pr[C(D_1) \in S] \leq \exp(\varepsilon) \times \Pr[C(D_2) \in S] + \delta
\]

The probability of a computation \( C \) producing a given output is almost independent of the existence of any individual record in the dataset.

Differential privacy is achieved by adding noise to the output of the computation.

The noise is independently generated for each dataset component.

---

Parameters of Differential Privacy: \( \varepsilon \) and \( \delta \)

\( \varepsilon \) controls the trade-off between the accuracy of a computation and the strength of its privacy guarantee.

\( \delta \) relaxes the strict relative shift of probability.

In this system, the \((\varepsilon, \delta)\) – differential privacy is achieved by:

- Adding Laplace distribution noise with a standard deviation of \( \sqrt{2\Delta C/\varepsilon} \) where \( \Delta C \) is the sensitivity of the computation.
- Adding a complimentary resampling mechanism.

---

Publishers need to list all their queries at a well-known URL on their website.

Data aggregator signs the query list and periodically checks it.

Detect malicious publishers isolating clients by controlling the distribution of queries to clients.

SQL may produce zero or more numerical values or strings.

Each bucket is defined as a numerical range or string expression.

Label 'yes' if a row in the SQL falls within the numerical range or matches the string expression.

‘N/A’ response when the query predicates fail.
Two purposes:
1. Detect when a publisher is dropping client answers
2. Detect when a publisher is adding substantial number of fake answers

Client generates a nonce, encrypts the nonce, the Qid and A-1 ‘null’ answers with the aggregator’s public key.
The response is transmitted to the publisher as a real response:

\[
\text{Response} = \begin{cases} 
\text{Enc}_{DA,pub}(Qid, nonce) & \text{once} \\
\text{Enc}_{DA,pub}(Qid, null) & (A-1 \text{ times}) 
\end{cases}
\]

Client obtains blind signature from the aggregator
Client sends a separate, encrypted copy of the nonce and blind signature to a randomly selected publisher (costumer of the aggregator)

\[
NR = (\text{Enc}_{DA}(Qid, nonce), \text{blind.sig})
\]

Non-Tracking Web Analytics
Audit Response

If the aggregator consistently receives nonce reports via different publishers without the corresponding nonce message from the audited publisher, the aggregator suspects the audited publisher of dropping messages, possibly to isolate a client.
The aggregator can validate the suspicion by masquerading as real clients from browsers it controls, and sending audits from this clients.
The aggregator performs this check to avoid influence from malicious clients which could have sent nonce reports via different publishers, without sending it to the audited publisher.

Blind signatures are used to limit the rate a client can generate audits
Reduces the effect of malicious clients
Prevents a malicious publisher from generating fake audits
Blind signatures are time stamped to prevent an attacker from storing them for future use.

Noise at the Publisher:
DP noise rounded to the nearest integer, for all buckets with the data aggregator’s \( \epsilon \) value.
Create additional answers
Noise can be positive or negative, relative to an offset value.
Noise answers are encrypted with the aggregator’s public key.

After the query end time, the set of client answers and noise answers are mixed and sent to the aggregator along with the Qid and offset value

\[
P \rightarrow DA : Qid, R_{DA}, 0
\]

The aggregator decrypts the answers, counts them, and subtracts the offset to obtain the noisy result:

\[
\begin{align*}
R_{DA}' &= (r_1 + n_{1,b_1}, r_2 + n_{2,b_2}, ..., r_b + n_{b,b}) \\
    &= (r_1 + n_{1,b_1}, r_2 + n_{2,b_2}, ..., r_b + n_{b,b})
\end{align*}
\]

\( r_i \) is the count of client answers belonging to a bucket \( b_i \) and \( n_{b,b} \) is the publisher’s noise value for bucket \( b \).
The aggregator checks for potential malicious publisher behavior:
Estimate the number of expected answers based on the number of audits received for the query and the audit probability
Check for anomalies in the publisher’s results.
Consistent combination of low-value buckets with high-value buckets
**Analysis**

**Possible threats:**
Data Aggregator might be motivated to track clients and exploit any information it learns.
- Record identifiers associated with clients
- IP addresses
- Combination of answers from a client
- Manipulate query parameters and audit activities

**Mitigation procedure:**
By using the publisher as an anonymizing proxy, the proposed system hides IP addresses from the aggregator.
DP noise added by the publisher.
Separately encrypting answers.
In auditing, blind signature and coarse-grained timestamps prevent the aggregator to connect nonce reports back to the clients.

**Publisher**
Exploit its position in the middle to learn individual client’s information and falsify results.

**Possible threats:**
- Publisher attacking clients
  - Isolation via selectively dropping other clients’ answers
  - Isolation via dropping target client’s answer
  - Isolation via buckets or SQL
  - Isolation via query distribution
  - DoS attacks
- Publisher falsifying results

**Mitigation procedure:**
The aggregator has various means of determining the expected number of answers from the publisher through query parameters.
Establishing a maximum A value
Possible threats:
- Distort the aggregate result by lying in its responses.
- Multiple answers to a given query.
- Send fake nonce reports without corresponding nonces.

Mitigation procedure:
- Publisher keeps a record of client IP addresses.
- Distortions are limited by the A value set by the publisher.
- Control of blind signature assignment.

Client:
- Firefox add-on
- Keeps user information in a local database
- Looks for queries at a well-known database
- 1000 lines of JavaScript code and 3000 lines of RSA libraries

Publisher:
- Simple server-side script that stores encrypted responses at the publisher’s website
- Plugin for open source web analytics software

Data Aggregator software:
- Simple program that enables the publisher to upload encrypted answers.
- Aggregator decrypts and aggregates answers, adds noise and returns signed values to the publisher.

Implementation and Evaluation

To analyze the system’s overhead, the scenario is as follows:

- Each week, a publisher poses queries to 50K clients.
- First eight queries collect the same information current aggregators provide to publishers.
- Last three are additional queries the system allows the publisher to pose.
- Aggregator uses a 2048-bit key.
Client ran on Laptop with Mac OS X 10.6.8 on an Intel Core 2 Duo 2.66 GHz Smartphone running Android 2.3.5 with 1 GHz processor

JavaScript client can achieve about 380, 20 and 16 encryptions per second on Google Chrome, Firefox and on the smartphone respectively.

Publisher and aggregator software ran on Machine with 2GB of memory running Linux 2.6.38 kernel on an Intel Xeon two cores 2.4 GHz

Publisher software can encrypt around 7980 answers per second. Expected total of additional answers for all 11 queries is 4.9M (< 11 minutes per week to generate them)

Data aggregator software can decrypt and aggregate around 270 messages per second. Aggregation takes 3.6 hours/week for 8 queries and 6.6 hours/week for all 11 queries.

Most of the overhead is due to additional noise answers.

Compressed size of the biggest query: 35 KB

_buckets may not change very often and can be cached.

The client’s bandwidth overhead is in the order of a few kilobytes for sending responses. A client would consume about 8KB/week for all 11 queries

Most bandwidth consumption is related to noise answers; however, overhead is still acceptable.

Test of system’s feasibility:

Client deployed via friends and mturk.com for 15 days with 236 unique clients.

Report on their browsing activities.

118 clients daily, on average.

Queries (everyday) on:

- Number of pages browsed
- Visited sites
- Site visit frequency
- Search engines used.

3K most popular sites from Alexa were used.

εp = εDA = 0.5

Clients in the test were fairly active

Almost half of them visited at least 100 pages

Most reported sites:

Google
YouTube
Facebook

Data on the sites usage frequency was gathered:

Google more visited than Facebook or YouTube

Only three search engines covered, which implies that searches on other sites with search functionality might have been omitted.
There is no other model to compare their results with.
Lack of information about current aggregators’ infrastructure makes comparison of results difficult.
The example provided does not gather meaningful data as it is used as a means to gain experience on the model.
It is hard to determine the accuracy and effectiveness of the privacy measures taken.

Future Implementations:
Apply the model to distributed scenarios.
Perform the test with data that could provide data additional to bandwidth overhead measurements.