A Differentially Private Selective Aggregation Scheme for Online User Behavior Analysis

Jianwei Qian*, Fudong Qiu*, Fan Wu*, Na Ruan*, Guihai Chen*, Shaojie Tang†
*Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
†Department of Information Systems, University of Texas at Dallas, USA
qianjavier@gmail.com, fdqiu@sjtu.edu.cn, {fwu, naruan, gchen}@cs.sjtu.edu.cn, tangshaojie@gmail.com

***Abstract*—Online user behavior analysis is becoming increasingly important, and offers valuable information to analysts for developing better e-commerce strategies. However, it also raises significant privacy concerns. Recently, growing efforts have been devoted to protecting the privacy of individuals while data aggregation is performed, which is a critical operation in behavior analysis. Unfortunately, existing methods allow very limited aggregation over user data, such as allowing only summation, which hardly satisfies the need of behavior analysis. In this paper, we propose a scheme PPSA, which encrypts users' sensitive data to prevent privacy leakage from both analysts and the aggregation service provider, and fully supports selective aggregate functions for differentially private data analysis. We have implemented our design and evaluated its performance using a trace-driven evaluation based on an online behavior dataset. Evaluation results show that our scheme effectively supports various selective aggregate queries with acceptable computation and communication overheads.**

I. Introduction

Online user behavior analysis studies how and why users of e-commerce platforms and web applications behave. It has been widely applied in practice, especially in commercial environments, political campaigns, and web application developments [1]. Data aggregation is one of the most critical operations in behavior analysis. Nowadays, the aggregation tasks for user data are outsourced to third-party data aggregators including Google Analytics, comScore, Quantcast, and StatCounter. While this tracking scheme brings great benefits to analysts and aggregators, it also raises serious concerns on disclosure of users' privacy [2]. Aggregators hold detailed data of users' online behaviors, from which demographics can be easily inferred [3]. To protect users' privacy, government and industry regulations were established, *e.g.*, W3C Do-Not-Track [4], which significantly restricts the analysis of users' online behaviors [2].

To address the conflict between the utility of analysis results and users' privacy, much effort has been devoted to designing protocols that allow operations on user data while still protecting users' privacy (*e.g.*, [2], [5]–[11]). Unfortunately, existing schemes guarantee strong privacy at the expense of limitations on analysis. Most of them can only compute summation and mean of data over all users without filter or selection, *i.e.*, *overall aggregation*. Some previous methods allow more complex computations, such as polynomial evaluation [10], [11], yet still do not support selection. However, *selective aggregation* is one of the most important operations for queries on databases. It can be used to tell the difference among different user groups in a certain aspect. For instance, "select avg(income) from database group by gender".

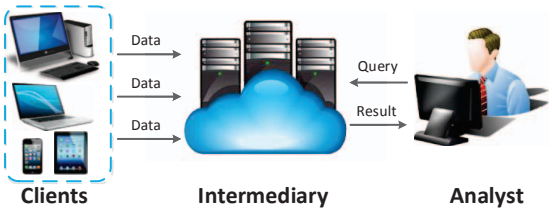


Fig. 1: System overview

As shown in Fig. 1, a typical privacy-preserving data aggregation system is composed of three parts: clients, intermediary (*i.e.*, aggregation service provider) and analyst. The intermediary collects data from clients (users' devices), does some calculations and evaluates aggregate queries issued by the analyst. A common assumption made in many existing systems is that the intermediary is not trusted.

The main goal of this paper is to design a practical protocol that is able to compute selective aggregation of user data, while still preserving users' privacy. There are mainly three challenges. *First*, the untrusted intermediary need to evaluate selective aggregation obliviously. It cannot access user data for privacy concerns, but we hope it do computations to achieve selection and aggregation on user data. We exploit homomorphic cryptosystem to address this challenge, but so far it does not directly support data selection. *Second*, PPSA needs to achieve differential privacy in homomorphic cryptosystem. To protect individuals' privacy, we need to obliviously add noise to aggregate results in addition to encrypting user data. Existing differential privacy mechanism generates noise from real numbers, but homomorphic cryptosystem requires plaintexts to be integers. Simply scaling real numbers to integers would cause inaccuracy and inconvenience. Thus, we need to resolve this conflict. *Third*, PPSA should resist *client churn*, the situation where clients switch between online and offline frequently. When an analyst issues a query, there could be few users connected, which means few data can be collected to evaluate the query. But the analyst wants the intermediary to respond to her as soon as possible. Thus, our protocol needs to tolerate client churn, and evaluate the query both timely and accurately.

To address these challenges, we design a scheme PPSA (Privacy-Preserving Selective Aggregation). In general, our contributions can be summarized as follows:

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

- We present the first scheme PPSA that allows privacy-preserving selective aggregation on user data, which plays a critical role in online user behavior analysis.
- We combine homomorphic encryption and differential privacy mechanism to protect users' sensitive information from both analysts and aggregation service providers, and protect individuals' privacy from being inferred.
- We extend PPSA to support aggregation selected by multiple boolean attributes, which makes it more useful in online user behavior analysis.
- We implement PPSA and do a trace-driven evaluation based on an online behavior dataset. Evaluation results show that our scheme effectively supports various selective aggregate queries with high accuracy and acceptable computation and communication overheads.

The rest of this paper is organized as follows. Section II presents an overview of PPSA and related background knowledge. Section III gives details on our protocol. Then Section IV presents an extension of PPSA. Section V analyzes simulation results, followed by related work in Section VI. Finally, Section VII concludes this paper.

II. Preliminaries

We first present PPSA model, then we introduce differential privacy and a useful homomorphic cryptosystem.

A. System Model

In behavior analytics, overall aggregation and selective aggregation are two basic types used to query over data from a group of users.

Overall aggregation means computing the sum or mean of a certain value of all users, *e.g.*, "the total amount of time online of all the users yesterday".

Selective aggregation literally refers to selecting the users who satisfy some conditions before aggregating their values, *e.g.*, "the average amount of time online of all the male users". Herein, "male" is a condition to pick out target users.

We suppose there is a centralized table T that contains attributes and collects users' answers to them. Attributes (denoted by att) can only be numeric, because non-numeric attributes cannot be directly aggregated.

Boolean attributes are a special type of numeric attributes, to which users' answers are boolean values (0 or 1), *e.g.*, "gender is male". A male user's answer would be 1. Most categorical attributes can be easily transformed into boolean attributes. For example, education level can be decomposed into several boolean attributes: "education level is bachelor", "education level is master", etc.

Numeric attributes. Since boolean attributes are very important in PPSA, we are referring to non-boolean numeric attributes when we use "numeric attributes". Users' answers to numeric attributes are non-negative integers, *e.g.*, "age=25".

To formulize, the relation schema of T is

$$T(id, a_1, a_2, \ldots, a_i, b_1, b_2, \ldots, b_j).$$

id is user ID, a represents numeric attribute, and b represents boolean attribute. The set of attributes is denoted by $\mathbb{A}$.

Fig. 1 gives an overview of PPSA, which is comprised of a set of n users, the intermediary and an analyst.

Clients are installed on user side. They can be bundled with users' softwares that require private analytics. Thus, it is reasonable to assume clients are trusted. A client collects a user's data, detects and removes outliers. Once the user gets online, the client sends encrypted data to the intermediary. Clients are not involved in the process of statistical aggregation. The set of users is denoted as $\mathbb{U} = \{1, 2, \ldots, n\}$, the plaintext answer of user u to attribute att is x^u_{att}, and the ciphertext of x^u_{att} is c^u_{att}, for $\forall att \in \mathbb{A}$, $\forall u \in \mathbb{U}$.

Analysts are individuals or institutions that want to query on user data. An analyst sends a query Q to the intermediary, and then receives a noisy answer from it. Analysts are assumed to be semi-honest, trying to learn individual users' privacy. An analyst may collude with other analysts or make one single query multiple times.

Intermediary, comprised of an *aggregator* and an *authority*, bridges clients and analysts. They are in charge of aggregating user data from clients and responding to queries of analyst. Encrypted data is stored in the aggregator, but the key is managed by the authority. They need to cooperate to provide both functionality and security. We assume they are both semi-honest, *i.e.*, they faithfully run the specified protocol, but may try to learn additional information. And they do not collude with each other. These assumptions are appropriate in realistic scenarios [7].

Our privacy goal is to prevent user data leakage to analysts, the aggregator or the authority.

B. Differential Privacy

Differential privacy *et al.*([12]–[14]) is a privacy mechanism that protects any individual's privacy by making it very hard to determine whether or not her record is in the queried table. Privacy is preserved by adding noise Z to the result of a given query Q. We prove Z can be generated according to the geometric distribution, which is on natural numbers.

Theorem 1. *Given a query Q, let two independent variables $Z_1, Z_2 \sim Geo(1-p)$ and $Z = Z_1 - Z_2$, adding Z to the result of Q achieves ϵ-differential privacy, where $p = exp(-\epsilon/\Delta Q)$.*

Please see details and proofs in the technical report [15]. PPSA utilizes this method to obliviously add noise to the real result. Z can be decomposed into two *half-noises*. If the aggregator generates Z_1 and the authority generates Z_2, we can obtain the noise by subtracting Z_2 from Z_1. Neither of them knows the whole noise. The blind noise addition prevents the system from determining the noise-free result when the noisy result is publicly released.

C. Boneh-Goh-Nissim Cryptosystem

The Boneh-Goh-Nissim Cryptosystem (BGN) [16] is a kind of homomorphic cryptosystem which allows the computation of an unlimited number of homomorphic additions and a single homomorphic multiplication of two ciphertexts. Below are functions in BGN cryptosystem.

$KeyGen(\tau)$: Given a security parameter $\tau \in \mathbb{Z}^+$, generate the public key $pk = (n, g, h, \mathbb{G}, \mathbb{G}_1, e)$, and the private key

$sk = p$. Here, e is a bilinear map, and $h^p = 1$. (More details are in [16].)

$Encrypt(pk, m)$: Given a plaintext $m \in \mathcal{P}$ and a public key pk, it chooses a random $r \in \mathbb{Z}_n$ and calculate the ciphertext

$$c = g^m h^r mod\ n \in \mathbb{G} \tag{1}$$

We use $E(\ )$ or c to refer to ciphertext hereinafter.

$Decrypt(sk, c)$: Given a ciphertext $c \in \mathcal{C}$ and a private key sk, it calculates

$$c' = c^p = (g^p)^m mod\ n \tag{2}$$

and uses Pollard's lambda method [17] to take the discrete logarithm of c' in base g^p, and then get the plaintext m in time $O(\sqrt{T})$. We actually can precompute a (polynomial-sized) table of powers of g^p so that decryption can occur in constant time.

Then, we show the homomorphic properties of BGN cryptosystem. Given $c_1 = E(m_1)$, $c_2 = E(m_2)$, then

$$c_1 c_2 = E(m_1 + m_2), \tag{3}$$

$$c_1 g^k = E(m_1 + k), \tag{4}$$

$$c_1^k = E(km_1), \tag{5}$$

$$c_1 c_2^{-1} = E(m_1 - m_2), \tag{6}$$

$$c_1 g^{-k} = E(m_1 - k), \tag{7}$$

$$e(c_1, c_2) = E(m_1 m_2) \in \mathbb{G}_1. \tag{8}$$

Equation 8, *i.e.*, homomorphic multiplication, can be used only once. Details can be seen in our technical report [15].

III. System Design

In this section, we first discuss about the design rationale of PPSA, and then present the protocol for selective aggregation.

A. Design Rationale

In this subsection, we introduce three important decisions on the system design.

1) Storing data on aggregator: Most of recent proposals are distributed systems, where each client stores its private data locally. Such systems provide privacy but are susceptible to client churn. To avoid this weakness, PPSA chooses to store all the user data in a server, the aggregator. Clients are only required to send their data when they are online. The aggregator can evaluate queries without their participation so the system operates well even if no client is online. In this case, each client encrypts its data before sending them to the aggregator, which can only do calculations obliviously on these encrypted data and output the results.

2) Introducing authority: Due to using encryption in PPSA, there must be a component to manage keys. Firstly, the aggregator cannot take this responsibility, because it holds all the private data. Secondly, if clients manage keys, they have to participate in the process of evaluating queries to decrypt the results. At last, analysts cannot manage keys either because any analyst can be an adversary. As a result, we have to introduce the authority into the system to generate keys and keep the private key. The aggregator and the authority constitute the intermediary of PPSA model.

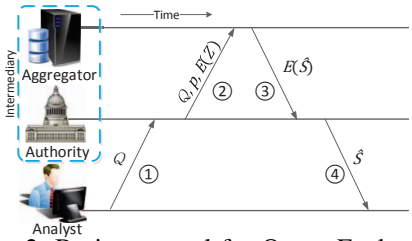


Fig. 2: Basic protocol for Query Evaluation

3) Using homomorphic cryptosystem and differential privacy mechanism: First, the aggregator needs to do calculations obliviously on encrypted data. Homomorphic cryptosystems support such operations. Second, when an analyst obtains aggregate results from the system, she can infer individual privacy with computational power and auxiliary information. To prevent such privacy disclosure, we exploit differential privacy mechanism. It guarantees stronger security of PPSA.

B. Basic Protocol

The selective aggregation protocol comprises 3 phases.

Phase 1: Setup

The authority first decides a set of attribute:

$$\mathbb{A} = \{a_1, a_2, \ldots, a_i, b_1, b_2, \ldots, b_j\}.$$

Then it runs $KeyGen(\tau)$ and gets sk and pk. Also, the authority decides the parameter ϵ in differential privacy. And it sets a minimum sample size min for too small a sample makes a query meaningless and prone to privacy leak. Finally, it publishes the tuple: $\langle \mathbb{A}, pk, min \rangle$.

Based on $\mathbb{A}$, the aggregator creates a table

$$T(id, a_1, a_2, \ldots, a_i, b_1, b_2, \ldots, b_j),$$

with values of all cells set to null.

Phase 2: Data Collection

After setup, the system begins to collect user data. Every client refers to $\mathbb{A}$, and collects user's answer to each of the given attributes. (How the client gets user data is outside the scope of this paper.) Once client u obtains an answer x_{att}^u to an attribute att, it encrypts the answer with pk published by the authority and get the ciphertext c_{att}^u. Then it sends the tuple to the aggregator: $\langle u, att, c_{att}^u \rangle, \forall att \in \mathbb{A}$.

After receiving the tuple from client u, the aggregator sets the corresponding entry in table T: $T(u, att) = c_{att}^u$.

The client still fulfills data collection when disconnected and would send obtained data to the aggregator shortly after getting online. We do not need any other interactions between clients and servers. Thus, PPSA is resistant to client churn.

Phase 3: Query Evaluation

Query evaluation can be executed before data collection is finished, because evaluating a query does not involve all the data in table T. The key to achieve selective aggregation is counting in data items of target users by multiplying them by 1 and skipping the rest by multiplying them by 0. These calculations are done in ciphertext, and thus no privacy disclosure would occur. It consists of four steps (Fig. 2).

Step 1. An analyst sends the authority a query $Q = \langle s, att, b \rangle$, where s is sample size the analyst wants, att is any attribute in $\mathbb{A}$, b is any boolean attribute for selection.

Step 2. The authority does sanity check once it receives Q. If $s < min$, $att \notin \mathbb{A}$, or b is not a boolean attribute, it returns an error to the analyst. Otherwise, it goes on as follows. It computes the parameter p as described in Section II-B: Then it generates two $Geo(1-p)$ random variable Z_1, Z_2 as half-noises and encrypts them with pk: At last, the authority sends the tuple to the aggregator:

$$\langle Q, p, E(Z_1), E(Z_2)\rangle.$$

Step 3. The aggregator runs Algorithm 1 (PPSAA). It first counts the tuples that have valid values for the requested attribute. If the number m is less than requested sample size, it returns an error which indicates data deficiency. Otherwise it goes on to randomly choose a sample $\mathbb{S}$ from valid tuples. PPSAA computes two sums:

$$E(S_b) = \prod_{u\in\mathbb{S}} c_b^u = E(\sum_{u\in\mathbb{S}} x_b^u), \tag{9}$$

$$E(S_{att}^b) = \prod_{u\in\mathbb{S}} e(c_{att}^u, c_b^u) = E(\sum_{u\in\mathbb{S}} x_{att}^u x_b^u), \tag{10}$$

where S_b represents sum over attribute b, S_{att}^b represents sum over att on users who are selected by b. If user u meets the predicate of b, *i.e.*, $x_b^u = 1$, then his answer x_{att}^u is added to the sum. Otherwise, it is skipped since it is multiplied by a 0. The aggregator also generates two half-noises, and obliviously adds them together with $E(Z_1), E(Z_2)$ to the sums:

$$E(\hat{S}_b) = E(S_b) \times E(Z_1) \times E(Z_3)^{-1} = E(S_b + Z_1 - Z_3),$$

$$E(\hat{S}_{att}^b) = E(S_{att}^b) \times E(Z_2) \times E(Z_4)^{-1} = E(S_{att}^b + Z_2 - Z_4).$$

At last, the algorithm outputs two encrypted noisy aggregate results $E(\hat{S}_b), E(\hat{S}_{att}^b)$. Here hat ($\hat{}$) denotes noisy data. Then it sends the results to the authority.

Algorithm 1 Privacy-Preserving Selective Aggregation Algorithm (PPSAA)

Require: Table T, attribute set $\mathbb{A}$, query Q, noise parameter p, encrypted half-noises $E(Z_1)$, $E(Z_2)$, public key pk.
Ensure: Tuple $\langle E(\hat{S}_b), E(\hat{S}_{att}^b)\rangle$, or $Error$.
1: Count and mark users who have valid values in att and b. Let the count be m.
2: **if** $m < s$ **then**
3: **return** $Error$.
4: **else**
5: Let $\mathbb{S}$ be a set of s random users that are marked.
6: $E(S_b) \leftarrow \prod_{u\in\mathbb{S}} c_b^u$.
7: $E(S_{att}^b) \leftarrow \prod_{u\in\mathbb{S}} e(c_{att}^u, c_b^u)$.
8: Generate two $Geo(1-p)$ random variables Z_3, Z_4.
9: $E(Z_3) \leftarrow Encrypt(pk, Z_3)$.
10: $E(Z_4) \leftarrow Encrypt(pk, Z_4)$.
11: $E(\hat{S}_b) \leftarrow E(S_b) \times E(Z_1) \times E(Z_3)^{-1}$.
12: $E(\hat{S}_{att}^b) \leftarrow E(S_{att}^b) \times E(Z_2) \times E(Z_4)^{-1}$.
13: **return** $\langle E(\hat{S}_b), E(\hat{S}_{att}^b)\rangle$.
14: **end if**

Step 4. If the output is $Error$, the authority sends an error message to inform the analyst of data deficiency. Otherwise, it decrypts the encrypted noisy results with its private key sk, and sends the noisy results $\hat{S}_b, \hat{S}_{att}^b$ to the analyst:

$$\hat{S}_b = S_b + Z_1 - Z_3, \hat{S}_{att}^b = S_{att}^b + Z_2 - Z_4. \tag{11}$$

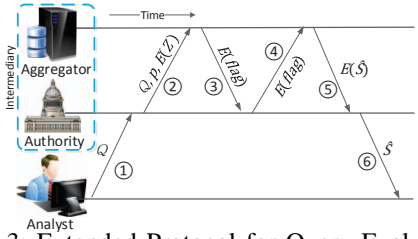


Fig. 3: Extended Protocol for Query Evaluation

The analyst can calculate the noisy mean ξ_{att}^b:

$$\xi_{att}^b = \hat{S}_{att}^b / \hat{S}_b. \tag{12}$$

Note it also makes sense if att is a boolean attribute. For instance, att is "education level is master" and b is "gender is female". We can get females' ratio of holding a master degree.

IV. Extension

In this section, we present steps to evaluate aggregation selected by multiple different boolean attributes. For example, the average number of searches made by females who have master degrees, which involves two boolean attributes: "gender is female" and "education level is master". To evaluate such queries, we present the Query Evaluation phase as described in Fig. 3. For simplicity, we assume there are only two boolean attributes restricting the aggregation, from which we can easily extend to scenarios with more boolean attributes.

Step 1. An analyst sends the authority a query

$$Q = \langle s, att, b_1, b_2\rangle,$$

where b_1, b_2 represents two different boolean attributes. (The aggregation makes sense if att is also a boolean attribute not equal to b_1, b_2.)

Step 2. The authority does sanity check and sends the tuple to the aggregator: $\langle Q, p, E(Z_1), E(Z_2)\rangle$.

Step 3. The aggregator does a calculation:

$$E(flag^u) = c_{b_1}^u c_{b_2}^u = E(x_{b_1} + x_{b_2}). \tag{13}$$

We use $flag$ as an indicator. The ciphertext $E(flag^u)$ is an encryption of 2 if and only if user u satisfies both b_1, b_2 conditions. Then the aggregator permutates all these $E(flag^u)$ and sends them to the authority.

Step 4. Now the authority wonders which of the encrypted flags are $E(2)$. Note decrypting them is unnecessary. She just computes $E(flag^u)^p$ (recall p is the private key) and compared them with $(g^p)^0, (g^p)^1, (g^p)^2$ (Equation 2). Then these $E(flag^u)$ are set to:

$$E(flag^u) = \begin{cases} E(1), & if\ E(flag^u)^p = (g^p)^2 \\ E(0), & otherwise. \end{cases} \tag{14}$$

Then it sends these $E(flag^u)$ back to the aggregator.

Step 5. Suppose $S_{b_1 b_2}$ represents the number of users who satisfy both b_1, b_2, $S_{att}^{b_1 b_2}$ represents the sum over att selected by both b_1, b_2. The aggregator computes:

$$E(S_{b_1 b_2}) = \prod_{u\in\mathbb{S}} E(flag^u) = E(\sum_{u\in\mathbb{S}} flag^u). \tag{15}$$

$$E(S_{att}^{b_1 b_2}) = \prod_{u \in \mathbb{S}} e(c_{att}^u, E(flag^u)) = E(\sum_{u \in \mathbb{S}} x_{att}^u flag^u). \tag{16}$$

A user is counted in only if she meets both conditions of b_1, b_2. The aggregator then adds noise and sends them to the authority.

Step 6. The analyst decrypts $E(\hat{S}_{b_1 b_2}), E(\hat{S}_{att}^{b_1 b_2})$, and sends the noisy results to the analyst.

V. Evaluation

In this section, we implement PPSA and evaluate its performance.

A. Methodology

We first implemented the BGN cryptosystem using the Pairing-based Cryptography Library (PBC) [18], a library built on the GMP library to perform mathematical operations in pairing-based cryptosystems. The security parameter τ was set to 80. Then we implemented PPSA based on the cryptosystem.

Then we did a trace-driven simulation on a data set of 1000 nationwide users' demographics and online behaviors in four weeks. It is extracted from a data set from China Internet Network Information Center (CNNIC). The behaviors include webpages browsed, time spent on each webpage, browsers used. The demographic profiles include gender, age, education level, income, occupation, etc.

B. Utility & Accuracy & Client Churn Resistance

We set differential privacy parameter $\epsilon = 1$ in our tests. To evaluate utility, we consider the following 5 queries. They all aggregate on a sample of 1000 users in the whole four weeks.

- Q_1: Average number of times of using Internet Explorer.
- Q_2: Ratio of male users.
- Q_3: Average number of webpages browsed by users, who are grouped by gender.
- Q_4: Average number of shopping webpages browsed by users, who are grouped by gender.
- Q_5: Average number of times of using Internet Explore by users who are female and have a bachelor degree.

After running simulations, we get all the noisy results. The results of Q_1, Q_2, Q_5 are 1765, 0.773, 1994, respectively. The results of Q_3, Q_4 are shown in Fig. 4(a). Analysts can use them to compare the online intersts of males and females. Relative errors of the five results are $0.2\%, 0.4\%, 2.5\%, 4.3\%, 6.3\%$, respectively. Thus, it is shown PPSA supports multiple types of queries with acceptable accuracy. To prove PPSA is resistant to client churn, we compare it with a most recent system SplitX [7] in accuracy. We focus on Q_2 because SplitX can only evaluate such queries. We vary online user ratio and sample size s, and use average relative error as a metric. As presented in Fig. 4(b), PPSA has higher accuracy than SplitX, especially when there are less than 10 percent users online. When few users are online, SplitX cannot operate at all.

C. Overhead

1) Computation Overhead: We analyze run time by every phase and step of PPSA. The setup time is constant and less than 10ms. The data collection time is up to computational ability of clients and end-to-end time between clients and the aggregator. And the encryption time is almost negligible because one BGN encryption costs less than 1.5ms. The decryption time of the authority is also a small constant. As for analysts, they only send their queries and do a few simple further calculations, also not time-consuming. Consequently, we will not detail them. Fig. 4(c) presents computation overheads of algorithm 1 and the extension. PPSAA's overhead results mainly from homomorphic multiplication operations, and it increases linearly as the growth of sample size, *i.e.*, the number of users whose data are involved in query evaluation. But they are still very efficient in computation, as the run time is less than 10s when sample size is 10,000. (The test data is generated from the real dataset.) In the extension, most of overheads lie in Steps 3, 4, 5. Their computation overheads are also in proportion to the sample size. From these tests we show that PPSA has acceptable computation overhead.

2) Communication Overhead: In PPSA, clients encrypt their data and send the ciphertexts to the aggregator. Each ciphertext has a size of 30 bytes. So the communication overhead for them is up to the number of ciphertexts. For an analyst, she only sends a query whose size is at most 16 bytes, which is negligible. Therefore, we only focus on the aggregator and the authority. Table I summarizes their communication overheads in bytes. For basic selective aggregation, communication overhead is very small and constant. For the extension, it is linearly related to s (sample size). In the evaluation of query Q_5, the overheads of authority and aggregator are both about 30KB. We can make it clear that PPSA has acceptable communication overhead.

VI. Related Work

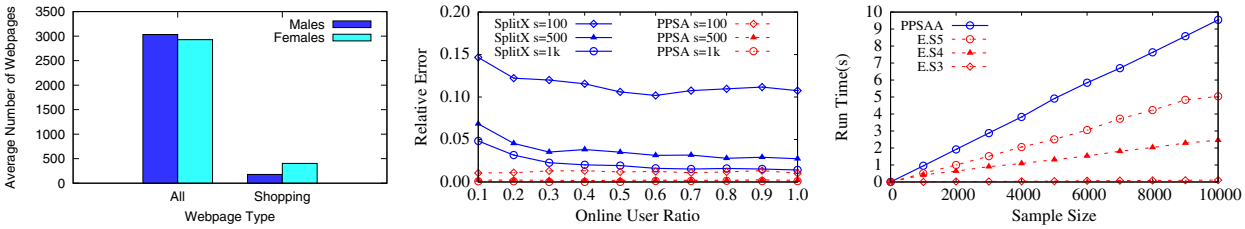
Privacy-preserving aggregation on user data has raised much attention recently. In general, there are two types of systems in previous work.

In a centralized system, all the user data are stored in the server. It is important that users encrypt or encode their data before sending them to the server. The server holds the encrypted data, but it can only compute answers to queries obliviously, *e.g.*, [19]–[21]. However, they do not guarantee differential privacy. Homomorphic encryption is a common method to allow aggregation of encrypted data without decryption, *e.g.*, [11], [22], [23]. Chen *et al.* [24] used an order-preserving hash-based function to encode both data and queries instead. But they do not have the same goal as us and cannot evaluate selective aggregation. Li *et al.* [25] proposed a system that processes range queries, which yet does not compute aggregation and assumes analysts to be trusted.

In a distributed system, clients need to proactively (*e.g.*, [5], [7], [9], [26]), or passively (*e.g.*, [2], [8]) send required data to the aggregator in a private way. But both rely on the participation of clients. Secure Multi-Party Computation [27] requires that all participants must be simultaneously online and interact with each other perodically. Shi *et al.* [9] proposed

Component	Basic PPSA	Extension
Authority	87	$30s + 107$
Aggregator	60	$30s + 60$

TABLE I: Communication overheads (bytes)



(a) #webpages browsed by different genders (b) Comparison with SplitX (c) Computation overhead

Fig. 4: Evaluation on utility, accuracy and computation overhead

a system that can tolerate some offline clients, but a trusted dealer and a trusted initial setup phase between all participants and the aggregator are still needed. Rottondi *et al.* [28] also discussed node failures but they addressed a different issue from this paper.

VII. Conclusion

In this paper, we have described the challenges of making online user data aggregation while preserving users' privacy. Based on BGN homomorphic cryptosystem, we have designed the first system that is able to securely and selectively aggregate user data, making it practical in realistic data analytics. It guarantees strong privacy preservation by utilizing differential privacy mechanism to protect individuals' privacy. We have presented PPSA to evaluate aggregation selected by one boolean attribute, and extended it to aggregation selected by multiple boolean attributes. Extensive experiments have shown that PPSA supports various selective aggregate queries with acceptable overhead and high accuracy.

Acknowledgment

This work was supported in part by the State Key Development Program for Basic Research of China (973 project 2012CB316201), in part by China NSF grant 61422208, 61472252, 61272443 and 61133006, in part by CCF-Intel Young Faculty Researcher Program and CCF-Tencent Open Fund, in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, and in part by Jiangsu Future Network Research Project No. BY2013095-1-10. The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

F. Wu is the corresponding author.

References

[1] Behavioral analytics - wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/wiki/Behavioral%20analytics

[2] I. E. Akkus, R. Chen, M. Hardt, P. Francis, and J. Gehrke, "Non-tracking web analytics," in *CCS*, 2012, pp. 687–698.

[3] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *NSDI*, 2012.

[4] Web tracking protection. [Online]. Available: http://www.w3.org/Submission/web-tracking-protection/

[5] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *SIGMOD*, 2010, pp. 735–746.

[6] B. Applebaum, H. Ringberg, M. J. Freedman, M. Caesar, and J. Rexford, "Collaborative, privacy-preserving data aggregation at scale," in *PETS*, 2010, pp. 56–74.

[7] R. Chen, I. E. Akkus, and P. Francis, "SplitX: high-performance private analytics," in *SIGCOMM*, 2013, pp. 315–326.

[8] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke, "Towards statistical queries over distributed private user data," in *NSDI*, 2012.

[9] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *NDSS*, 2011.

[10] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: multivariate polynomial evaluation," in *INFOCOM*, 2013, pp. 2634–2642.

[11] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *CCS*, 2014, pp. 844–855.

[12] C. Dwork, "Differential privacy," in *ICALP*, 2006, pp. 1–12.

[13] ——, "Differential privacy: A survey of results," in *TAMC*, 2008, pp. 1–19.

[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, 2006, pp. 265–284.

[15] A differentially private selective aggregation scheme for online user behavior analysis. [Online]. Available: http://www.cs.sjtu.edu.cn/%7Efwu/res/Paper/PPSA-TR.pdf

[16] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography*, 2005, pp. 325–341.

[17] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[18] Pairing-based cryptography library. [Online]. Available: http://crypto.stanford.edu/pbc/

[19] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *SOSP*, 2011, pp. 85–100.

[20] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *S&P*, 2007, pp. 350–364.

[21] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *S&P*, 2000, pp. 44–55.

[22] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 25, no. 5, pp. 1125–1134, 2013.

[23] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *STOC*, 2012, pp. 1219–1234.

[24] F. Chen and A. X. Liu, "Privacy and integrity preserving multi-dimensional range queries for cloud computing," in *IFIP Networking*, 2014, pp. 1–9.

[25] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," in *PVLDB*, vol. 7, no. 14, 2014.

[26] B. Mood, D. Gupta, K. Butler, and J. Feigenbaum, "Reuse it or lose it: More efficient secure computation through reuse of encrypted values," in *CCS*, 2014, pp. 582–596.

[27] O. Goldreich, *Secure multi-party computation*. [Online]. Available: http://www.wisdom.weizmann.ac.il/%7Eoded/PS/prot.ps

[28] C. Rottondi, G. Verticale, and A. Capone, "A security framework for smart metering with multiple data consumers," in *INFOCOM WKSHPS*, 2012, pp. 103–108.