Privacy-Preserving Selective Aggregation of Online User Behavior Data

Jianwei Qian, Fudong Qiu, *Student Member, IEEE*, Fan Wu, *Member, IEEE*, Na Ruan, *Member, IEEE*, Guihai Chen, *Member, IEEE*, and Shaojie Tang, *Member, IEEE*

Abstract—Tons of online user behavior data are being generated every day on the booming and ubiquitous Internet. Growing efforts have been devoted to mining the abundant behavior data to extract valuable information for research purposes or business interests. However, online users' privacy is thus under the risk of being exposed to third-parties. The last decade has witnessed a body of research works trying to perform data aggregation in a privacy-preserving way. Most of existing methods guarantee strong privacy protection yet at the cost of very limited aggregation operations, 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 disclosure from both outside analysts and the aggregation service provider, and fully supports selective aggregate functions for online user behavior analysis while guaranteeing differential privacy. We have implemented our method and evaluated its performance using a trace-driven evaluation based on a real online behavior dataset. Experiment results show that our scheme effectively supports both overall aggregate queries and various selective aggregate queries with acceptable computation and communication overheads.

Index Terms—Selective aggregation, differential privacy, homomorphic encryption

1 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 development [1], [2], [3]. 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 about disclosure of users' privacy [4]. Aggregators hold detailed data of users' online behaviors, from which demographics can be easily inferred [5]. To protect users' privacy, government and industry regulations were established, e.g., the EU Cookie Law [6] and W3C Do-Not-Track [7], which significantly restricts the analysis of users' online behaviors [4]. 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., [4], [8], [9], [10], [11], [12], [13], [14]). 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 [13], [14], [15]. For instance, Jung et al. [13] proposed a system that can perform multivariate polynomial evaluation. Unfortunately, they 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".

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.

Adding noise to the aggregate result is a common method to achieve stronger privacy preservation, *differential privacy* [16], without which individuals' privacy can be easily inferred from aggregate results by adversaries who have computational power and auxiliary information. There are two ways to add noise: either each client adds noise to its own data (e.g., [8], [12], [17]), or the intermediary obliviously adds noise to the aggregate result (e.g., [4], [10], [11]). Privacy-Preserving Selective Aggregation (PPSA) adopts the latter way to achieve differential privacy, in which noise needs to be added obliviously so as to prevent the intermediary from determining the noise-free result when the noisy result is publicly released.

- *J. Qian was with Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China and with the Department of Computer Science, Illinois Institute of Technology, Chicago IL 60616. E-mail: jqian15@hawk.iit.edu.*
- *F. Qiu, F. Wu, N. Ruan, and G. Chen are with Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: fdqiu@sjtu.edu.cn, {fwu, naruan, gchen}@cs.sjtu.edu.cn.*
- *S. Tang is with the Department of Information Systems, Naveen Jindal School of Management, University of Texas at Dallas, Richardson, TX 75080. E-mail: tangshaojie@gmail.com.*

Manuscript received 3 Sept. 2015; revised 22 May 2016; accepted 24 May 2016. Date of publication 27 July 2016; date of current version 20 Jan. 2017. Recommended for acceptance by P. R. Schaumont.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2016.2595562

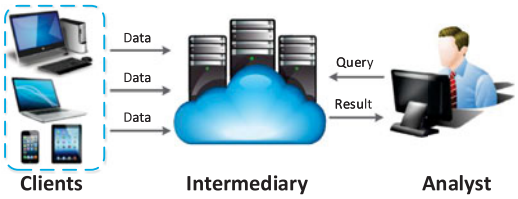


Fig. 1. *System overview:* Clients are installed on user side. The intermediary collects data from clients, computes aggregate statistics, and answers queries issued by the analyst. The intermediary should also ensure users' privacy is not leaked.

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 needs to evaluate selective aggregation obliviously. It cannot access user data for privacy concerns, but we hope it does 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*, our scheme PPSA needs to achieve differential privacy in a 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 cryptosystems require 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 be resistant to *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. In general, our contributions can be summarized as follows:

- 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 prove that differential privacy can be achieved by adding two Geometric variables, which is computed via homomorphic encryption. Furthermore, we present a privacy analysis of PPSA.
- We extend PPSA to two more scenarios to fully support more complex selective aggregation of user data. We utilize a calculation to evaluate aggregation selected by multiple boolean attributes. We design a way of oblivious comparison between two integers, and utilize it to evaluate aggregation selected by a numeric attribute.
- 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 2 presents an overview of PPSA and related background knowledge. Section 3 gives details on design rationale and protocols. Then, Section 4 presents two extensions of PPSA. Section 5 analyzes simulation results, followed by related work in Section 6. Finally, Section 7 concludes this paper.

2 Preliminaries

In this section, we first present PPSA model. Then, we introduce differential privacy and review a useful homomorphic cryptosystem.

2.1 System Model

First of all, we describe terms used in our study. 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 and mean of a certain value of every user, 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" later on. Users' answers to numeric attributes are non-negative integers, e.g., "age = 25".

To formalize, 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

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

Important symbols of this paper are listed in Table 1.

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 the user side. They can be bundled with users' software that requires 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. We suppose the set of users is denoted as $\mathbb{U} = \{1, 2, \ldots, n\}$. The plaintext answer of user u to

TABLE 1
Symbol Table

T	The centralized data table
att	An attribute in T
a	A numeric attribute in T
b	A boolean attribute in T
c	A numeric attribute for selection in T
$\mathbb{A}$	Attribute set
$\mathbb{U}$	User set
u	A user in $\mathbb{U}$
x^u_{att}	The answer of user u to attribute att
c^u_{att}	The ciphertext of x^u_{att}
Q	A query from the analyst
ε	Privacy parameter in differential privacy
p	A parameter indicating how much noise should be added
$DL(p)$	Discrete Laplace distribution on integers
$Geo(1-p)$	Geometric distribution
Z_1, Z_2	Two IID geometric variables as half-noises
pk	Public key of BGN cryptosystem
sk	Private key of BGN cryptosystem
min	Minimum sample size set by the authority
s	Sample size for the analyst requests
$T(u, att)$	The entry in T corresponding to user u and attribute att
$E(pt)$	The ciphertext of any plaintext pt
$\mathbb{S}$	A randomly chosen set of s users who has valid data
S_{att}	The sum over attribute att
$\hat{S}$	Noisy result by adding noise to S
S^b_{att}	The sum over att selected by b
$flag^u$	A parameter indicating if u satisfies the selection conditions

attribute att is denoted as x^u_{att}, and the ciphertext of x^u_{att} is denoted as c^u_{att}, for $\forall att \in \mathbb{A}, \forall u \in \mathbb{U}$.

Analysts are individuals or institutions that want to query about 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.

The *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 analysts. They provide both functionality and security. Table T is stored here. Details of them will be presented in Section 3. We assume they are both semi-honest (a.k.a. honest but curious), i.e., they faithfully run the specified protocol, but may try to learn additional information on their own. Furthermore, they do not collude with each other. This assumption is common in the literature [4], [10], [11], [17], and appropriate in realistic scenarios. For example, we assume that it is explicitly stated in the privacy policies of the aggregator and the authority, and enforced by the law. Aggregators who do not provide such privacy statements would be blocked by the client software, e.g., browsers. Likewise, analysts would turn to only honest authorities who obey such policies. Though there is a possibility that the aggregator colludes with the authority, the possibility is very small, because the party who initiates collusion takes a high risk of being reported and exposed by the other party who is honest, and then suffer from legal punishment and loss of customers and profits. As a result, we believe our assumption is reasonable in reality. In addition, this assumption can be relaxed by using trusted hardware [11].

Our privacy goal is to prevent user data leakage to analysts and the intermediary.

2.2 Differential Privacy

Differential privacy proposed by Dwork et al. [16], [18], [19] 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.

Definition 1 ((ε, δ)-differential privacy [20]). *A computation C achieves (ε, δ)-differential privacy if, for any two tables T_1 and T_2 that differ in at most one record, and all $O \subseteq range(C)$*

$$Pr(C(T_1) \in O) \leq exp(\varepsilon) \times Pr(C(T_2) \in O) + \delta, \quad (1)$$

where Pr is a probability distribution over the randomness of the computation.

That means the probability that a computation generates a given output is almost independent of the presence of any individual record in the dataset. Differential privacy is independent of the adversary's computational power and auxiliary information so it is a very strong guarantee. Specifically, there are two privacy parameters, ε and δ, in the expression. The former parameter ε mainly controls the tradeoff between the accuracy of a computation and the strength of its privacy guarantee. Higher ε represents higher accuracy but weaker privacy guarantee, and vice versa. The latter parameter δ relaxes the strict relative shift of probability in some cases, where Inequation (1) cannot be satisfied without a non-zero δ [20].

A standard technique is adding random noise to the answers. The noise distribution is carefully calibrated to the sensitivity of the query.

Definition 2 (Sensitivity [16]). *For a single snapshot query $Q : T \rightarrow R^k$, the sensitivity of Q is*

$$\Delta Q = \max \|Q(T_1) - Q(T_2)\|_1, \quad (2)$$

for all T_1, T_2 differing in at most one element.

Informally, the sensitivity ΔQ is the maximum amount the result $Q(T)$ can change, given any change to a single user's data in table T. It is a property of the query Q alone and is independent of the table. The sensitivity decides how much noise should be added to the answer of a query.

To achieve differential privacy, we prove that noise Z can be generated according to the discrete Laplace distribution DL [21], which is on integers. The probability mass function of $DL(p)$ is

$$Pr(Z = k) = \frac{1-p}{1+p} p^{|k|}, k \in \mathbb{Z}. \quad (3)$$

Theorem 1. *Given a query Q and a random integer $Z \sim DL(p)$, adding Z to the result of Q achieves ε-differential privacy, where $p = exp(-\varepsilon/\Delta Q)$.*

Proof. For any $r \in range(C)$

$$\begin{aligned} Pr(C(T_1) = r) &= Pr(Q(T_1) + Z = r) \\ &= Pr(Z = r - Q(T_1)) \\ &= \frac{1-p}{1+p} p^{|r-Q(T_1)|}. \end{aligned} \tag{4}$$

Similarly,

$$Pr(C(T_2) = r) = \frac{1-p}{1+p}\ p^{|r-Q(T_2)|}. \tag{5}$$

Then, we have

$$\begin{aligned} \frac{Pr(C(T_1) = r)}{Pr(C(T_2) = r)} &= p^{|r-Q(T_1)|-|r-Q(T_2)|} \\ &= exp\left(\frac{\varepsilon(|r - Q(T_2)| - |r - Q(T_1)|)}{\Delta Q}\right) \\ &\leq exp\left(\frac{\varepsilon|Q(T_1) - Q(T_2)|}{\Delta Q}\right) \\ &\leq exp(\varepsilon). \end{aligned} \tag{6}$$

It is straightforward that Inequation (1) can be achieved. □

Inusah et al. [21] proved that the difference of two independent and identically distributed geometric variables (denoted as $Geo(\cdot)$) has the same distribution as a discrete Laplace distribution variable.

Proposition 1. *Let two independent variables $Z_1, Z_2 \sim Geo(1-p)$ and $Z = Z_1 - Z_2$, we have $Z \sim DL(p)$.*

PPSA utilizes this method to obliviously add noise to the true result. Because the noise can be decomposed into two part, we call each of them a *half-noise*. If the aggregator generates a Z_1 and the authority generates Z_2, we can obtain a discrete Laplacian noise by subtracting Z_2 from Z_1. Neither of the two components knows the whole noise, so neither can remove it. Blind noise addition prevents the system from determining the noise-free result when the noisy result is publicly released.

Unfortunately, differential privacy has an intrinsic drawback: the privacy parameter ε degrades with the growing number of queries answered, as shown by [22], [23], [24]. For example, when the attacker issues the same query and gets the results for unlimited times, she can estimate the real result by averaging those results and thus breach differential privacy.

Theorem 2 (Composition Theorem [22], [24]). *For any $\varepsilon > 0, \delta, \delta' > 0$, and $k \in \mathbb{N}$, a k-fold adaptive composition of (ε, δ)-differentially private mechanisms is $(\varepsilon', k\delta + \delta')$-differentially private, where*

$$\varepsilon' = \sqrt{2k\ln(1/\delta')} \cdot \varepsilon + k\varepsilon(e^{\varepsilon} - 1). \tag{7}$$

Later on, more strict quantification of the privacy degradation level was given by Oh et al. [23]. As researchers keep working on tightening the bound, we denote $\varepsilon' = f(k, \varepsilon)$ for simplicity and in case of inaccuracy. Given the number of query k and objective privacy level ε', we can solve the equation and calculate ε for answering every single query in turn.

2.3 Boneh-Goh-Nissim Cryptosystem

The Boneh-Goh-Nissim (BGN) Cryptosystem [25] is a kind of homomorphic cryptosystem (please refer to [26] for a detailed survey) which utilizes a bilinear pairing [27] to allow the computation of an unlimited number of homomorphic additions and a single homomorphic multiplication of two ciphertexts. Bilinear map and bilinear group are the bases of the BGN cryptosystem.

Definition 3 (Bilinear map [25]). *Let $\mathbb{G}$ and $\mathbb{G}_1$ be two cyclic groups of order n with g as a generator of $\mathbb{G}$. A map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is said to be bilinear if $e(g, g)$ is a generator of $\mathbb{G}_1$, and*

$$e(u^a, v^b) = e(u, v)^{ab}, \tag{8}$$

for all $u, v \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$.

Definition 4 (Bilinear group [25]). *$\mathbb{G}$ is a bilinear group if there exists a group $\mathbb{G}_1$ and a bilinear map e such that,*

1) *$\mathbb{G}$ and $\mathbb{G}_1$ are two multiplicative cyclic groups of finite order n.*
2) *g is a generator of $\mathbb{G}$.*
3) *$e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map and $g_1 = e(g, g)$ is a generator of $\mathbb{G}_1$.*

Boneh et al. [25] presented an approach to construct a bilinear group of order n. We do not detail it here.

Next, we introduce the BGN cryptosystem as follows. Let $n = pq$ for distinct odd primes p and q, and let $\mathbb{G}, \mathbb{G}_1$ be two multiplicative groups of order n with a bilinear pairing e: $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, let g be a random generator of $\mathbb{G}$, and let h be a random generator of the subgroup of $\mathbb{G}$ of order p. Let $T < q$, then $\mathcal{P} = \mathbb{Z}_T, \mathcal{C} = \mathbb{G}, \mathcal{R} = \mathbb{Z}_n$.

KeyGen(τ): Given a security parameter $\tau \in \mathbb{Z}^+$, we generate two random τ-bit primes p, q, set $n = pq$, and select a positive integer $T < q$. Then, we choose two multiplicative groups $\mathbb{G}, \mathbb{G}_1$ of order n, which support a bilinear pairing $e : (\mathbb{G} \times \mathbb{G} \to \mathbb{G}_1)$, as well as random generators $g, u \in G$, and sets $h = u^q$ such that h is a generator of the subgroup of order p. The public key is $pk = (n, g, h, \mathbb{G}, \mathbb{G}_1, e)$, and the private key is $sk = p$.

Encrypt(pk, m): Given a message $m \in \mathcal{P}$ and a public key pk, $Encrypt(pk, m)$ chooses a random $r \in R$ and calculates the ciphertext

$$c = g^m h^r mod\ n \in \mathbb{G}. \tag{9}$$

In this paper, we use c or $E(\cdot)$ to denote ciphertext.

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

$$c' = c^p = (g^p)^m mod\ n, \tag{10}$$

and uses Pollard's lambda method [28] 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. In the rest of this paper, "$mod\ n$" is omitted and understood to have been operated.

The security of the BGN cryptosystem is summarized in the following theorem [25].

Theorem 3. *Let $n = pq$ for distinct odd primes p and q. The BGN cryptosystem is semantically secure if deciding whether*

or not a random element of $\mathbb{G}$ has order p is hard when p and q are unknown.

Lastly, we show the homomorphic properties of BGN cryptosystem. Given $c_1 = g^{m_1}h^{r_1}$, $c_2 = g^{m_2}h^{r_2}$, then

$$c_1 \boxplus c_2 = c_1 c_2 = g^{m_1}h^{r_1}g^{m_2}h^{r_2} = g^{m_1+m_2}h^{r_1+r_2}, \tag{11}$$

is a valid encryption of $m_1 + m_2$ ($\boxplus$ means homomorphic addition)

$$c_1 g^k = g^{m_1}h^{r_1}g^k = g^{m_1+k}h^{r_1}, \tag{12}$$

is a valid encryption of $m_1 + k$, and

$$c_1^k = (g^{m_1}h^{r_1})^k = g^{km_1}h^{r_1 k}, \tag{13}$$

is a valid encryption of km_1. Subtraction of encrypted messages and constants can be accomplished by computing $c_1 c_2^{-1}$ and $c_1 g^{-k}$, respectively. Given c_1, c_2 and a random $r \in \mathcal{R}$, a ciphertext representing the product $m_1 m_2$ can be calculated

$$\begin{aligned} c_1 \boxtimes c_2 &= e(c_1, c_2)h_1^r \\ &= e(g^{m_1}h^{r_1}, g^{m_2}h^{r_2})h_1^r \\ &= e(g^{m_1}g^{\alpha r_1}, g^{m_2}g^{\alpha r_2})h_1^r \\ &= e(g^{m_1+\alpha r_1}, g^{m_2+\alpha r_2})h_1^r \\ &= e(g,g)^{(m_1+\alpha r_1)(m_2+\alpha r_2)}h_1^r \\ &= g_1^{m_1 m_2 + m_1\alpha r_2 + m_2\alpha r_1 + \alpha^2 r_1 r_2}h_1^r \\ &= g_1^{m_1 m_2}h_1^{m_1 r_2 + m_2 r_1 + \alpha r_1 r_2 + r} \\ &= g_1^{m_1 m_2}h_1^{\bar{r}} \in \mathbb{G}_1. \end{aligned} \tag{14}$$

Here, $\boxtimes$ means homomorphic multiplication, e is a bilinear map, $g_1 = e(g,g), h_1 = e(g,h)$, and $\bar{r}$ is a uniformly random element of $\mathcal{R}$. All further homomorphic operations but multiplications are allowed in $\mathbb{G}_1$.

3 System Design

In this section, design choices of PPSA are explained first. Then, the protocols for overall aggregation and selective aggregation are described respectively. Finally, we present a privacy analysis.

3.1 Design Rationale

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

3.1.1 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 on 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 our mechanism, each client encrypts its data before sending it to the aggregator, which can only do calculations obliviously on these encrypted data and output the results.

3.1.2 Introducing Authority

Due to using encryption in PPSA, there must be a component to manage keys. First, the aggregator cannot take this responsibility, because it holds all the private data. Second, if clients manage keys, they have to participate in the process of evaluating queries to decrypt the results. In that case, the system could be out of service, when clients frequently shift between online and offline, or when few clients are connected, which is opposite to our goal of resisting client churn. 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.

3.1.3 Using Homomorphic Encryption and Differential Privacy Mechanism

On one hand, the aggregator needs to do calculations obliviously on encrypted data. Homomorphic encryption supports such operations. Therefore, we introduce BGN cryptosystem to perform this task. On the other hand, when an analyst obtains aggregate results from the system, she can infer individual privacy with her computational power and auxiliary information. To prevent such privacy disclosure, we exploit differential privacy mechanism proposed by [18]. The incorporation of homomorphic encryption and differential privacy guarantee strong security of PPSA.

3.2 Protocol for Overall Aggregation

The aggregation mechanism comprises three 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 $sk = p$ and $pk = (n, g, h, \mathbb{G}, \mathbb{G}_1, e)$. The authority also decides the objective overall privacy level by setting the parameter ε'. It also limits the number of queries to be answered to k and maintains a query counter cnt. Accordingly, the privacy budget ε for every single query can be calculated as mentioned in Section 2.2. Besides, the authority sets a minimum and a maximum value min, max of acceptable sample size for the analysts. Then, it publishes the following tuple

$$\langle \mathbb{A}, pk, min, max \rangle.$$

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 items 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

$$c_{att}^u = Encrypt(pk, x_{att}^u) = g^{x_{att}^u}h^{r_{att}^u}. \tag{15}$$

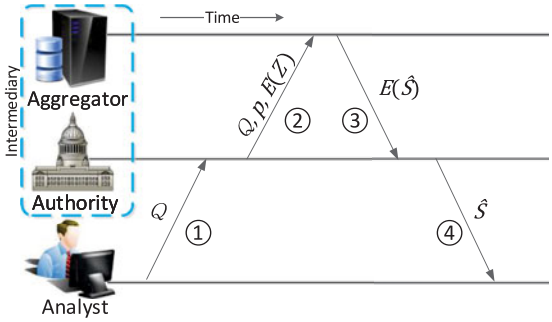


Fig. 2. *Basic protocol for Query Evaluation:* The intermediary consists of two non-colluding entities, aggregator and authority. They cooperate to calculate the noisy aggregate result following the given protocol.

Then, client u sends the following tuple to the aggregator, if they are connected

$$\langle u, att, c^u_{att}\rangle, \forall att \in \mathbb{A}.$$

After receiving the tuple from client u, the aggregator updates the value of corresponding item in table T:

$$T(u, att) = c^u_{att}.$$

$T(u, att)$ refers to the item in the row corresponding to user u and column corresponding to attribute att. So, the content of T is dynamic.

Instead of sending all the data at a time, the client sends every single answer shortly after it is obtained and encrypted. It still fulfills data collection even when disconnected with the aggregator. In this case, the client would send obtained data to the aggregator once 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 achieving selective aggregation is counting 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

$$Q = \langle \text{ “}OA\text{ ”}, s, att\rangle,$$

where “OA” is short for overall aggregation, s is sample size the analyst wants, and att is any attribute in $\mathbb{A}$.

Step 2. The authority would reject the query request if $cnt = k$ to limit the degradation of differential privacy level. Otherwise, it does sanity check once it receives Q. If the arguments are not matched, $s < min$, $s > max$, or $att \notin \mathbb{A}$, the query is also rejected. Otherwise, it goes on as follows. It computes the privacy parameter (as described in Section 2.2):

$$p = exp(-\varepsilon/\Delta Q) = exp(-\varepsilon/\max\|Q(T_1) - Q(T_2)\|)). \quad (16)$$

Then, the authority generates a $Geo(1-p)$ random variable Z_1 as a half-noise and encrypts it with pk:

$$E(Z_1) = Encrypt(pk, Z_1) = g^{Z_1} h^{r_1}. \quad (17)$$

Here $E(pt)$ refers to the ciphertext of any plaintext pt. At last, the authority sends the following tuple to the aggregator:

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

Step 3. The aggregator runs Algorithm 1 (PPOAA). 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 and sums up their values on att:

$$E(S_{att}) = \prod_{u\in\mathbb{S}} c^u_{att} = E\left(\sum_{u\in\mathbb{S}} x^u_{att}\right). \quad (18)$$

It also generates a $Geo(1-p)$ random variable Z_2 as the second half-noise. Then, the aggregator combines two half-noises into the whole Laplacian noise in encrypted form and obliviously adds it to the sum:

$$E(\hat{S}_{att}) = E(S_{att}) \times E(Z_1) \times E(Z_2)^{-1} = E(S_{att} + Z_1 - Z_2). \quad (19)$$

At last, the algorithm outputs an encrypted noisy aggregate result $E(\hat{S}_{att})$. Here S_{att} refers to sum over attribute att, and hat ($\hat{}$) denotes noisy data. Then, the aggregator sends the result to the authority.

Algorithm 1. Privacy-Preserving Overall Aggregation Algorithm (PPOAA)

Input: Table T, attribute set $\mathbb{A}$, query Q, noise parameter p, encrypted half-noise $E(Z_1)$, public key pk.
Output: Encrypted noisy sum $E(\hat{S}_{att})$, or $Error$.

```
m ← 0.
for each u ∈ 𝕌 do
  if T(u, att) is not null then
    m ← m + 1.
    Mark user u.
  end if
end for
if m < s then
  return Error.
else
  Let 𝕊 be a set of random s users that are marked.
  E(S_att) ← ∏_{u∈𝕊} c^u_att.
  Generate a Geo(1 − p) random variable Z_2.
  E(Z_2) ← g^{Z_2} h^{r_2}.
  E(Ŝ_att) ← E(S_att) × E(Z_1) × E(Z_2)^{−1}.
  return E(Ŝ_att).
end if
```

Step 4. If the result is $Error$, the authority sends an error message to inform the analyst of data deficiency. Otherwise, it decrypts the encrypted noisy result with its private key sk, gets the noisy result:

$$\hat{S}_{att} = Decrypt(sk, E(\hat{S}_{att})), \quad (20)$$

and sends it to the analyst, who can calculate the noisy mean by himself. In addition, the authority increases the query counter cnt. Here is the end of Query Evaluation.

3.3 Protocol for Selective Aggregation

We consider aggregation selected by one single boolean attribute for simplicity. More complex scenarios will be

discussed in Section 4. To support selective aggregation, we need to make some changes to the third phase, Query Evaluation.

Step 1. An analyst sends the authority a query

$$Q = \langle \text{“}SA\text{”}, s, att, b\rangle,$$

where “SA” is short for selective aggregation and b is any boolean attribute, att is the attribute to be aggregated whereas b is used for selection.

Step 2. The authority checks cnt and query validity, which is similar to the previous protocol. Then, it generates two half-noises and sends the following tuple to the aggregator:

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

Step 3. The aggregator runs Algorithm 2 (PPSAA) to evaluate the query Q. PPSAA first computes a sum on attribute b, i.e., $E(S_b)$. Then, it computes $E(S^b_{att})$ such that:

$$E(S^b_{att}) = E\left(\sum_{u\in\mathbb{S}} x^u_{att}x^u_b\right) = \prod_{u\in\mathbb{S}} e\left(c^u_{att}, c^u_b\right), \tag{21}$$

where S^b_{att} represents sum over att selected by b. If user u meets the predicate of b, i.e., $x^u_b = 1$, then his answer x^u_{att} is added to the sum. Otherwise, it is skipped since it is multiplied by a 0. At last, the aggregator obliviously adds two noises to the two sums respectively and outputs them. Then, the aggregator sends them to the authority.

Algorithm 2. Privacy-Preserving Selective Aggregation Algorithm (PPSAA)

Input: Table T, attribute set $\mathbb{A}$, query Q, noise parameter p, encrypted half-noises $E(Z_1)$, $E(Z_2)$, public key pk.
Output: Tuple $\langle E(\hat{S}_b), E(\hat{S}^b_{att})\rangle$, or $Error$.

$m \leftarrow 0$.
for each $u \in \mathbb{U}$ **do**
  if neither $T(u, att)$ nor $T(u, b)$ is null **then**
    $m \leftarrow m + 1$.
    Mark user u.
  end if
end for
if $m < s$ **then**
  return $Error$.
else
  Let $\mathbb{S}$ be a set of s random users that are marked.
  $E(S_b) \leftarrow \prod_{u\in\mathbb{S}} c^u_b$.
  $E(S^b_{att}) \leftarrow \prod_{u\in\mathbb{S}} e(c^u_{att}, c^u_b)$.
  Generate two $Geo(1-p)$ random variables Z_3, Z_4.
  $E(Z_3) \leftarrow g^{Z_3}h^{r_3}$.
  $E(Z_4) \leftarrow g^{Z_4}h^{r_4}$.
  $E(\hat{S}_b) \leftarrow E(S_b) \times E(Z_1) \times E(Z_3)^{-1}$.
  $E(\hat{S}^b_{att}) \leftarrow E(S^b_{att}) \times E(Z_2) \times E(Z_4)^{-1}$.
  return $\langle E(\hat{S}_b), E(\hat{S}^b_{att})\rangle$.
end if

Step 4. If the results are not $Error$, the authority decrypts them and sends the noisy results to the analyst, who calculates noisy mean value by dividing $\hat{S}^b_{att}$ by $\hat{S}_b$. 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.

3.4 Privacy Analysis

This section demonstrates the privacy-preserving property of PPSA. As mentioned before, we have assumed that both the aggregator and the authority are semi-honest. They run the specified protocol faithfully even if either of them colludes with others, so differential privacy is always preserved.

3.4.1 Analysts

An analyst sends queries to and receives results from the authority. She normally does not communicate with the aggregator or any user. She can only get noisy results, which are differentially private so she cannot infer individual privacy with auxiliary information. If she colludes with a user, they still cannot acquire other users’ data. If she colludes with the aggregator, they have all encrypted data but no private key to decrypt them. If she colludes with the authority, they get the private key but no user data. If analysts collude with each other, they get no more information.

3.4.2 Aggregator

The aggregator holds encrypted user data but only the authority knows the private key. It is assumed they do not collude with each other so it cannot decrypt those data. All of its computations are done obliviously. If it colludes with a user, they still cannot get other users’ data. Half of differential private noise added to a result comes from the aggregator. But it does not know the whole noise, so it cannot get a noise-free result by removing noise from the noisy one (It may get the noisy result from an analyst).

3.4.3 Authority

The authority holds the private key but has no access to user data. It can access noisy results, but like the aggregator, it cannot remove noises.

3.4.4 Clients

A client is installed on the user side and only knows its user’s data. If some users collude with each other, they only know their own data. Normally a client is not manipulated by its user. Even if a user makes her client send considerable duplicate values, the aggregator will only keep the latest value as each user takes up exactly one row in table T.

3.4.5 Differential Privacy

In our scheme, Laplacian noise is added to the answer of every query. The noisy is combined from two half-noises and added by two parties, the aggregator and the authority, which are assumed not to collude with each other. Since nobody knows how much noise is added, nobody can remove the noise from the released noisy query answer and then infer a person’s data from it. As a result, differential privacy for single queries is preserved. The parameter ε balances the trade-off between privacy and accuracy; a smaller ε leads to stronger privacy protection but greater answer distortion.

However, differential privacy has its own limitation that privacy degrades with growing number of queries answered. Even though the mechanism for answering a single query is ε-differentially private, we cannot achieve it when many queries are answered (unless they are

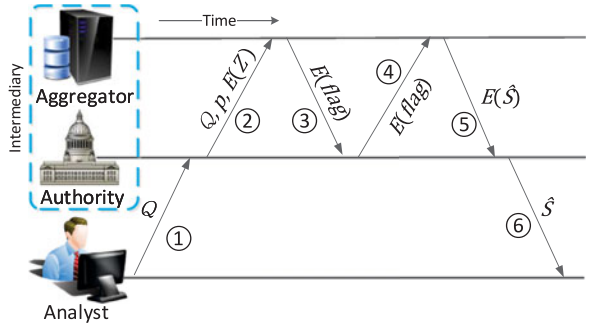


Fig. 3. *Protocol for query Evaluation in extension 1:* The analyst queries the aggregation of an attribute selected by multiple different boolean attributes. The aggregator and the authority communicate for two rounds and calculate the noisy result.

evaluated on different disjoint subsets of the data set). In our settings, queries are evaluated on a random sample of the original data and the data keeps changing, but the data subsets where queries are computed are somewhat, if not greatly, overlapped and correlated to each other. This scenario can be modeled as the composition of many different privacy mechanisms on different data releases that are correlated. According to the composition theorem [22], [24], answering k adaptive queries, of which each is ε-differentially private, can still achieve ε'-differential privacy though it is not as strong as the privacy of answering a single query. As mentioned in Section 2.2, ε' is a function of ε and the query number limit k. Given ε' and k, the authority calculates the privacy budget ε and then applies it to answering single queries. It rejects more query request when the query number limit has been reached so as to ensure the overall privacy guarantee. Therefore, it is concluded that ε'-differential privacy is achieved in our scheme.

To answer more queries, we could choose to degrade either the overall privacy level or the accuracy of answering single queries. When the query number limited is reached, the intermediary can pause the service and wait for the update of all the data to finish. By then, the data set is completely new and disjointed with the previous one so the query counter can be reset and the service can be open again, which is referred to as parallel composition [29].

4 Extensions

In this section, we first present steps to evaluate aggregation selected by multiple different boolean attributes. Then, we describe the protocol of aggregation selected by a single numeric attribute, based on a way of comparison between two encrypted integers.

4.1 Aggregation Selected by Multiple Boolean Attributes

In reality, an analyst may want to know the aggregation of an attribute selected by multiple different boolean attributes, e.g., 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

$$Q = \langle \text{“}SA\text{”}, s, att, b_1, b_2 \rangle,$$

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

Step 2. The authority does sanity check and sends the following tuple to the aggregator:

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

Step 3. The aggregator does a calculation:

$$E(flag^u) = c_{b_1}^u c_{b_2}^u = E(x_{b_1} + x_{b_2}). \tag{22}$$

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 $E(2)$. Noting that decrypting is unnecessary, she just computes $E(flag^u)^p$ (recall p is the private key) and compare them with $(g^p)^0, (g^p)^1, (g^p)^2$ (Equation (10)). Then, these $E(flag^u)$ are set to

$$E(flag^u) = \begin{cases} E(1), & if\ E(flag^u)^p = (g^p)^2 \\ E(0), & otherwise. \end{cases} \tag{23}$$

Next, 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:

$$E(S_{b_1 b_2}) = \prod_{u \in \mathbb{S}} E(flag^u) \tag{24}$$

$$E(S_{att}^{b_1 b_2}) = \prod_{u \in \mathbb{S}} e(c_{att}^u, E(flag^u)). \tag{25}$$

A user is counted 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.

4.2 Aggregation Selected by a Numeric Attribute

To be more practical, the analyst may wonder the selective aggregation of an attribute A selected by a numeric attribute, e.g., the average time online of 13 to 19 years old teenagers, where age is a numeric attribute for selection and $[13, 20)$ is an interval. Here, the attribute table is $T(id, a_1, a_2, \ldots, a_i, b_1, b_2, \ldots, b_j, c_1, c_2, \ldots, c_k)$. a and c are both numeric, but the former is used for aggregation, the latter for selection. We tell them apart because they are encrypted differently. Client u encrypts each bit of x_c^u, while encrypting x_a^u entirely.

Before processing such queries, we introduce a way of comparison between two encrypted integers, which is motivated by Katti et al. [30]. Suppose two l-bit non-negative integer x and y in binary form: $x = (x^l, x^{l-1}, \ldots, x^1)$,

$y = (y^l, y^{l-1}, \ldots, y^1)$. For each $i = 1, 2, \ldots, l$, let

$$d^i = x^i - y^i, \tag{26}$$

(Note $d^i \in \{-1, 0, 1\}$) and

$$e^i = x^i - y^i + 1 + \sum_{j=i+1}^{l} d^j 2^{l-j+1}. \tag{27}$$

Proposition 2 (See proof in [30]). *For any two l-bit non-negative integer x and y, we have $x < y$ if and only if there exists i, such that $e^i = 0$.*

If we encrypt each bit of x and y, then we get $c_1 = (c_1^l, c_1^{l-1}, \ldots, c_1^1)$, $c_2 = (c_2^l, c_2^{l-1}, \ldots, c_2^1)$. To obliviously compare x and y using c_1, c_2, the aggregator can utilize BGN cryptosystem to compute:

$$E(d^i) = c_1^i (c_2^i)^{-1}, \tag{28}$$

$$E(e^i) = c_1^i (c_2^i)^{-1} g \prod_{j=i+1}^{l} E(d^j)^{2^{l-j+1}}. \tag{29}$$

Then, it sends each $E(e^i)$ to the authority, which checks whether there is any i such that $E(e^i)$ is the encryption of 0 (by checking whether $E(e^i)^p = 1$).

We now present the process of Query Evaluation. Still, we consider one single attribute for selection.

Step 1. An analyst sends the authority a query

$$Q = \langle \text{“}numSA\text{”}, s, att, c, z, w \rangle,$$

where c is any numeric attribute for selection, and z, w are bounds of interval $[z, w)$.

Step 2. Same as before.

Step 3. For each user data x_c^u ($u \in \mathbb{S}$), the aggregator computes $E(e_{u,z}^i)$ and $E(e_{u,w}^i)$ mentioned above. Then, it permutates the two tables of $E(e)$ and sends them to the authority.

Step 4. The authority checks and learns the relationships between x_c^u and z, w. If $x_c^u < w$ and $x_c^u \geq z$, it sets $flag^u = 1$. Otherwise, $flag^u = 0$. The authority finally encrypts and sends these flags to the aggregator.

Step 5, 6. Same as before.

5 Evaluation

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

5.1 Methodology

We first implemented the BGN cryptosystem using the Pairing-based Cryptography Library (PBC) [31], which is a library built on the GMP library to perform mathematical operations in pairing-based cryptosystems. The security parameter τ was set to 80. Based on the cryptosystem, we implemented PPSA in C.

Then, we did a trace-driven simulation based on a dataset of 1,000 nationwide users' demographics and online behaviors in four weeks. It is extracted from a dataset from China Internet Network Information Center (CNNIC) [32]. The behaviors include webpages browsed, time spent on each webpage, browsers used. The demographic profiles include gender, age, education level, income, occupation, etc.

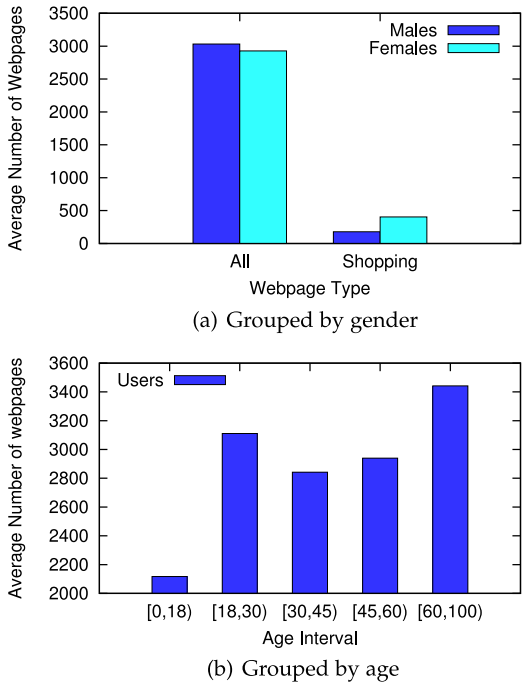


Fig. 4. *Average number of webpages browsed:* The statistics can be used to compare the online interests of males and females, and online activeness of different age groups.

We perform all simulations on a PC with an Intel Core i3-2310M 2.10 GHz CPU and an 8 GB memory under Ubuntu 12.04 OS.

5.2 Utility & Accuracy & Client Churn Resistance

We set differential privacy parameter $\varepsilon = 1$ for every single query in our tests. To evaluate utility, we consider the following six queries. They all aggregate on a sample of 1,000 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 Explorer by users who are female and have a bachelor degree.
- Q_6: Average number of webpages browsed by users, who are grouped by age intervals.

After running simulations, we get all the noisy results. The results of Q_1, Q_2, Q_5 are 1,765, 0.773, 1,994, respectively. The results of Q_3, Q_4 are shown in Fig. 4a. Analysts can use them to compare the online interests of males and females. The results of Q_6 is shown in Fig. 4b, which can be used to tell the differences among people of different ages. Thus, it is shown that PPSA supports multiple types of queries with acceptable accuracy.

Relative errors of the six results are 0.2, 0.4, 2.5, 4.3, 6.3, and 11.3 percent, respectively. Here, relative error is caused by the noise added, and depends on the privacy parameter

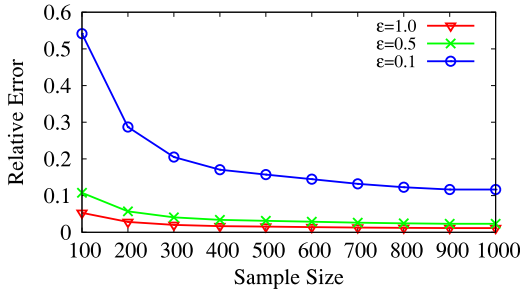


Fig. 5. *Relative error versus* s, ε*:* The relative error decreases when the sample size s or the privacy budget ϵ in differential privacy increases.

ε and the real result of the query. The smaller the real result is, the larger relative error the noisy result has, which is a common issue of differential privacy mechanism. Fig. 5 depicts the impacts of s, ε on the accuracy of evaluating Q_1. The relative error decreases when the sample size increases since the real result rises with sample size. It is also suggested that the smaller ε is, the greater the inaccuracy is.

To prove PPSA is resistant to client churn, we compare it with a most recent system called SplitX [10] in terms of 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. 6, 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.

5.3 Overhead

5.3.1 Computation Overhead

We analyze run time by every phase and step of PPSA. The setup time is constant and less than 10 ms. The data collection time is up to the 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 1.5 ms. 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. 7a presents computation overheads of two algorithms, which are run by the aggregator. PPOAA costs very little time. PPSAA has greater overhead due to pairing 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 10 s when the sample size is 10,000. (The test data is generated from the real dataset.)

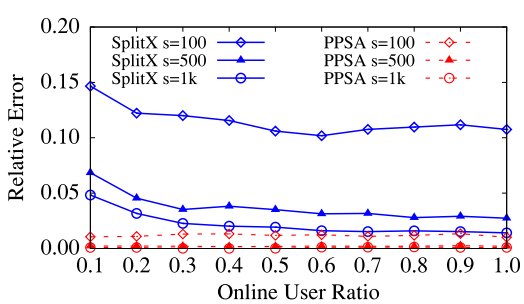


Fig. 6. *Comparison with SplitX:* PPSA has higher accuracy than SplitX, especially when there are less than 10 percent users online.

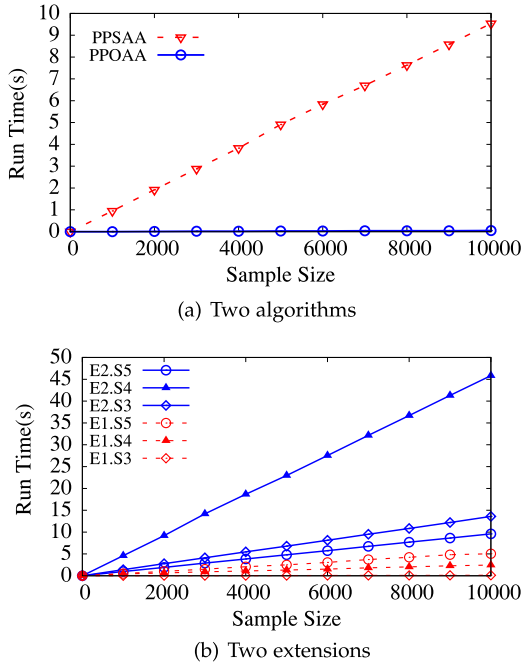


Fig. 7. *Computation overhead versus sample size:* The computation overheads are in proportion to the sample size for any of the four cases.

In Extensions 1 and 2, most of the overheads lie in Steps 3, 4, 5. We plot them in Fig. 7b. Their computation overheads are also in proportion to the sample size. Among them, Step 4 of Extension 2 (E2.S4) is the most time-consuming, whose run time is around 45 s when the sample size is 10,000. This is still acceptable. In addition, as depicted in Fig. 8, overheads of Steps 3 and 4 in Extension 2 are also linearly related to l, the number of digits of the data for attribute c (In Fig. 7b, l is set to 7). From these tests we show that PPSA has acceptable computation overhead.

5.3.2 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 30 bytes, which is negligible. Therefore, we only focus on the aggregator and the authority. Table 2 summarizes their communication overheads in bytes. (Recall s is sample size and l is the number of digits of data for attribute c.) For overall aggregation and basic selective aggregation, communication overhead is very small and constant. For

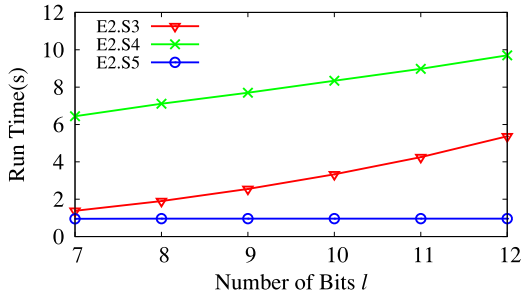


Fig. 8. *Computation overhead versus number of bits:* The computation overheads of Steps 3 and 4 in Extension 2 are linearly related to l, the bit number of the attribute of interest.

TABLE 2
Communication Overheads (Bytes): For Overall Aggregation and Basic Selective Aggregation, the Communication Overheads Are Small and Constant

Component	OA	Basic SA	Extension 1	Extension 2
Authority	50	87	$30\ s + 107$	$30\ s + 114$
Aggregator	30	60	$30\ s + 60$	$60l\ s + 60$

For the two extensions, they are linearly related to the sample size s (and the bit number, for the aggregator in Extension 2).

Extensions 1 and 2, it is linearly related to s and l. In the evaluation of query Q_6, the overheads of authority and aggregator are about 30 and 420 KB respectively. We can make it clear that PPSA has acceptable communication overhead.

6 Related Work

Privacy-preserving aggregation on sensitive user data has raised much attention recently, including health care data ([33], [34]), time-series data ([8], [9]), wireless sensor network data ([35], [36]), social network data ([37], [38]) and online behavior data for analysis and advertising ([10], [17]). In general, there are two types of systems in previous work.

6.1 Centralized Systems

In a centralized system, all the user data are stored on 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., [39], [40], [41]. However, these proposals have different goals than our system and do not support selective aggregation. Moreover, they do not guarantee differential privacy. Homomorphic encryption is a common method to achieve aggregation of encrypted data without decryption, such as [14], [42], [43]. Chen et al. [44] 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. [45] proposed a system that processes range queries, which yet does not compute aggregation and assumes analysts to be trusted. On the contrary, PPSA combines differential privacy and homomorphic encryption, and is able to selectively aggregate encrypted user data.

6.2 Distributed Systems

In a distributed system, clients need to proactively (e.g., [8], [10], [12], [46]), or passively (e.g., [4], [11], [47]) send required data to the aggregator in a private way. But both rely on the participation of clients. These systems all require online users, so analysis cannot go on when most of the users are offline. Homomorphic encryption is also applicable in distributed system. For instance, PASTE [8] exploits differential privacy and homomorphic cryptography but it allows only summation of user data and the aggregator knows the private key. Castelluccia et al. [36] use symmetric homomorphic encryption so they need a trusted aggregator, and they also allow only additive aggregation. DJoin [47] aims to support distributed and differentially private query answering service, but it applies to privacy-preserving data join between two parties, which is a different scenario with ours. Secure Multi-Party Computation (SMC) [48] requires that all participants must be simultaneously online and interact with each other periodically, which is infeasible for practical scenarios. Some previous researches have noticed the problem of client churn. For instance, Shi et al. [12] proposed 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. [49] also discussed node failures but they addressed a different issue from the problem in this paper.

When participating in query evaluation, clients need to preprocess their data first to prevent individuals from being identified. There are basically two approaches.

6.2.1 Each Client Anonymizes or Adds Noise to Its Own Data

This approach has obvious weak points. Clients still might be de-anonymized with auxiliary information or the utility of user data is significantly degraded. Anonygator [50] assumes that the content of the data collected would not leak privacy and that clients can hide their identities by adopting an anonymous network. Compared with it, PPSA has weaker assumptions. Dwork et al. [20] used expensive secret-sharing protocols leading to complex computations for each client, which is opposite to our goal of achieving client churn resistance, i.e., providing timely service even when clients frequently shift between online and offline. Duchi et al. [51] proposed local differential privacy which does not assume a trusted data curator and is stronger than standard differential privacy. In our scheme, the aggregator is trusted and thus local differential privacy is achieved. Other related works [8], [12], [17] allow duplicate answers sent by a single malicious client to pollute the query result substantially.

6.2.2 Server Obliviously Adds Noise to Aggregate Results

Akkus et al. [4] presented the first system that provides web analytics without tracking with differential privacy guarantee, yet it requires a public-key operation per single bit of user data, causing high overheads. PDDP [11] also has this limitation. SplitX [10] utilizes XOR operation to achieve low-cost encryption, thus has much better scalability. All of the three designs define buckets to correspond to different intervals of numeric values, which degrades data quality because the aggregator can only get very obscure answers from each client.

7 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 and by one numeric attribute. Extensive experiments have shown

that PPSA supports various selective aggregate queries with acceptable overhead and high accuracy.

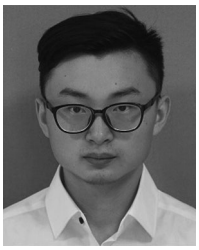
Acknowledgments

This work was supported in part by the State Key Development Program for Basic Research of China (973 project 2014CB340303), in part by China NSF grant 61422208, 61472252, 61272443 and 61133006, in part by Shanghai Science and Technology fund 15220721300, in part by CCF-Tencent Open Fund, and in part by the Scientific Research Foundation for the Returned Overseas Chinese Scholars. 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. Fan Wu is the corresponding author.

References

[1] R. E. Bucklin and C. Sismeiro, "Click here for internet insight: Advances in clickstream data analysis in marketing," *J. Interactive Marketing*, vol. 23, no. 1, pp. 35–48, 2009.
[2] R. Bose, "Advanced analytics: Opportunities and challenges," *Ind. Manage. Data Syst.*, vol. 109, no. 2, pp. 155–172, 2009.
[3] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact," *MIS Quarterly*, vol. 36, no. 4, pp. 1165–1188, 2012.
[4] I. E. Akkus, R. Chen, M. Hardt, P. Francis, and J. Gehrke, "Non-tracking web analytics," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 687–698.
[5] F. Roesner, T. Kohno, and D. Wetherall, "Detecting and defending against third-party tracking on the web," in *Proc. 9th USENIX Conf. Networked Syst. Des. Implementation*, 2012, pp. 12–12.
[6] Directive 2009/136/ec of the European parliament and of the council, (2009). [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:337:0011:0036:en:PDF
[7] Web tracking protection, (2011). [Online]. Available: http://www.w3.org/Submission/web-tracking-protection/
[8] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proc. ACM Int. Conf. Manage. Data*, 2010, pp. 735–746.
[9] B. Applebaum, H. Ringberg, M. J. Freedman, M. Caesar, and J. Rexford, "Collaborative, privacy-preserving data aggregation at scale," in *Proc. 10th Privacy Enhancing Technol. Symp.*, 2010, pp. 56–74.
[10] R. Chen, I. E. Akkus, and P. Francis, "SplitX: High-performance private analytics," in *Proc. ACM Special Interest Group Data Commun.*, 2013, pp. 315–326.
[11] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke, "Towards statistical queries over distributed private user data," in *Proc. 9th Symp. Networked Syst. Des. Implementation*, 2012, pp. 13–13.
[12] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2011, Art. no. 4.
[13] 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 *Proc. 32nd IEEE Int. Conf. Comput. Commun.*, 2013, pp. 2634–2642.
[14] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 844–855.
[15] T. Jung, X.-Y. Li, and M. Wan, "Collusion-tolerable privacy-preserving sum and product calculation without secure channel," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 45–57, Jan./Feb. 2015.
[16] C. Dwork, "Differential privacy: A survey of results," in *Proc. 5th Int. Conf. Theory Appl. Models Comput.*, 2008, pp. 1–19.
[17] M. Hardt and S. Nath, "Privacy-aware personalization for mobile advertising," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 662–673.
[18] C. Dwork, "Differential privacy," in *Proc. 33rd Int. Colloquium Automata Languages Programming*, 2006, pp. 1–12.
[19] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. 3rd Conf. Theory Cryptography*, 2006, pp. 265–284.
[20] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Proc. 25th Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2006, pp. 486–503.
[21] S. Inusah and T. J. Kozubowski, "A discrete analogue of the laplace distribution," *J. Statistical Planning Inference*, vol. 136, no. 3, pp. 1090–1102, 2006.
[22] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *Proc. 51st Annu. IEEE Symp. Found. Comput. Sci.*, 2010, pp. 51–60.
[23] P. Kairouz, S. Oh and P. Viswanath, "The composition theorem for differential privacy," in *Proc. 32nd Int. Conf. on Machine Learning*, Lille, France, 2015.
[24] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theoretical Comput. Sci.*, vol. 9, no. 3/4, pp. 211–407, 2014.
[25] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. 2nd Int. Conf. Theory Cryptography*, 2005, pp. 325–341.
[26] K. Henry, "The theory and applications of homomorphic cryptography," Master thesis, Dept. of Computer Science, Univ. Waterloo, Waterloo, Canada, 2008.
[27] V. S. Miller, "The Weil pairing, and its efficient calculation," *J. Cryptology*, vol. 17, no. 4, pp. 235–261, 2004.
[28] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.
[29] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 19–30.
[30] R. S. Katti and C. Ababei, "Secure comparison without explicit XOR," in *CoRR, abs/1204.2854*, 2012.
[31] The pairing-based cryptography library, (2006). [Online]. Available: https://crypto.stanford.edu/pbc/
[32] An online user behavior dataset, (2012). [Online]. Available: http://www.datatang.com/data/43910
[33] M. P. Armstrong, G. Rushton, and D. L. Zimmerman, "Geographically masking health data to preserve confidentiality," *Statistics Med.*, vol. 18, no. 5, pp. 497–525, 1999.
[34] HealthVault, (2016). [Online]. Available: https://www.healthvault.com
[35] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. 26th IEEE Int. Conf. Comput. Commun.*, 2007, pp. 2045–2053.
[36] C. Castelluccia, A. C. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 5, no. 3, 2009, Art. no. 20.
[37] J. Qian, X.-Y. Li, C. Zhang, and L. Chen, "De-anonymizing social networks and inferring private attributes using knowledge graphs," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.
[38] X.-Y. Li, C. Zhang, T. Jung, J. Qian, and L. Chen, "Graph-based privacy-preserving data publication," in *Proc. IEEE INFOCOM*, 2016, pp. 1–9.
[39] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. 33rd ACM Symp. Operating Syst. Principles*, 2011, pp. 85–100.
[40] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proc. 39th IEEE Symp. Secur. Privacy*, 2007, pp. 350–364.
[41] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. 32nd IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.
[42] X. Yi, M. G. Kaosar, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 1125–1134, May 2013.
[43] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proc. 44th IEEE Symp. Theory Comput.*, 2012, pp. 1219–1234.

[44] F. Chen and A. X. Liu, "Privacy and integrity preserving multi-dimensional range queries for cloud computing," in *Proc. IFIP Netw. Conf.*, 2014, pp. 1–9.
[45] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," *Proc. VLDB Endowment*, vol. 7, no. 14, pp. 1953–1964, 2014.
[46] B. Mood, D. Gupta, K. Butler, and J. Feigenbaum, "Reuse it or lose it: More efficient secure computation through reuse of encrypted values," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 582–596.
[47] A. Narayan and A. Haeberlen, "DJoin: Differentially private join queries over distributed databases," in *Proc. 10th USENIX Symp. Operating Syst. Des. Implementation*, 2012, pp. 149–162.
[48] O. Goldreich, *Secure Multi-Party Computation*, (2002). [Online]. Available: http://www.wisdom.weizmann.ac.il/~oded/PS/prot.ps
[49] C. Rottondi, G. Verticale, and A. Capone, "A security framework for smart metering with multiple data consumers," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2012, pp. 103–108.
[50] K. P. Puttaswamy, R. Bhagwan, and V. N. Padmanabhan, "Anonygator: Privacy and integrity preserving data aggregation," in *Proc. ACM/IFIP/USENIX 11th Int. Middleware Conf.*, 2010, pp. 85–106.
[51] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, 2013, pp. 429–438.



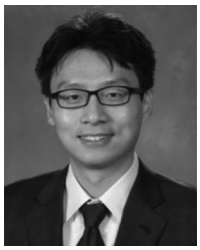
Jianwei Qian received the BE degree in computer science and technology from Shanghai Jiao Tong University, P.R. China, in 2015. He is working toward the PhD degree in the Department of Computer Science, Illinois Institute of Technology. His research interests include privacy and security issues in big data and social networking.

Fudong Qiu received the BS degree in computer science and technology from Xi'an Jiao Tong University, in 2012. He is working toward the PhD degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, P. R. China. His research interests include security and privacy preservation in wireless networks. He is a student member of the ACM, CCF, and the IEEE.

Fan Wu received the BS degree in computer science from Nanjing University, in 2004, and the PhD degree in computer science and engineering from State University of New York, Buffalo, in 2009. He is an associate professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He has visited the University of Illinois at Urbana-Champaign (UIUC) as a post doc research associate. His research interests include wireless networking and mobile computing, algorithmic game theory and its applications, and privacy preservation. He has published more than 100 peer-reviewed papers in leading technical journals and conference proceedings. He is a recipient of China National Natural Science Fund for Outstanding Young Scientists, CCF-Intel Young Faculty Researcher Program Award, CCF-Tencent "Rhinoceros bird" Open Fund, and Pujiang Scholar. He has served as the chair of CCF YOCSEF Shanghai, on the editorial board of Elsevier Computer Communications, and as the member of technical program committees of more than 40 academic conferences. He is a member of the IEEE.

Na Ruan received the BS degree in information engineering and the MS degree in communication and information system from China University of Mining and Technology, in 2007 and 2009, respectively. She received the DE degree from the Faculty of Engineering, Kyushu University, Japan, in 2012. Since 2013, she joined the Department of Computer Science and Engineering, Shanghai Jiaotong University as assistant professor. Her current research interests include wireless network security and game theory. She is a member of the Information Processing Society of Japan (IPSJ), China Computer Federation (CCF), ACM and the IEEE.

Guihai Chen received the BS degree from Nanjing University, in 1984, the ME degree from Southeast University, in 1987, and the PhD degree from the University of Hong Kong, in 1997. He is a distinguished professor at Shanghai Jiao Tong University, China. He had been invited as a visiting professor by many universities including Kyushu Institute of Technology, Japan, in 1998, University of Queensland, Australia in 2000, and Wayne State University, USA, during Sep. 2001 to Aug. 2003. He has a wide range of research interests with focus on sensor networks, peer-to-peer computing, high-performance computer architecture, and combinatorics. He has published more than 250 peer-reviewed papers, and more than 170 of them are in well-archived international journals such as the *IEEE Transactions on Parallel and Distributed Systems*, the *Journal of Parallel and Distributed Computing*, the *Wireless Networks*, the *Computer Journal*, the *International Journal of Foundations of Computer Science*, and the *Performance Evaluation*, and also in well-known conference proceedings such as HPCA, MOBIHOC, INFOCOM, ICNP, ICPP, IPDPS and ICDCS. He is a member of the IEEE.



Shaojie Tang received the PhD degree in computer science from Illinois Institute of Technology, in 2012. He is currently an assistant professor in Naveen Jindal School of Management, University of Texas, Dallas. His research interest includes social networks, mobile commerce, game theory, e-business and optimization. He received the Best Paper Awards in ACM MobiHoc 2014 and IEEE MASS 2013. He also received the ACM SIGMobile service award in 2014. He served in various positions (as chairs and TPC members) at numerous conferences, including ACM MobiHoc and IEEE ICNP. He is an editor for the *Elsevier Information Processing in the Agriculture* and the *International Journal of Distributed Sensor Networks*. He is a member of the IEEE.