Privacy-Preserving Survey by Crowdsourcing with Smartphones

Sin G. Teo*, Narayanan Amudha†, and Jianneng Cao‡
Institute for Infocomm Research, Singapore
Email: *teosg@i2r.a-star.edu.sg, †naraa@i2r.a-star.edu.sg, ‡caojn@i2r.a-star.edu.sg

***Abstract*—In this paper we propose a solution for privacy-preserving survey. We assume a crowdsourcing platform with a big number of registered smartphones. The platform works as a coordinator between data owners and service requestors (i.e., survey users). It shifts most workload to the smartphones of data owners, and thus is lightweighted and scalable. For privacy protection, we apply cryptography techniques to ensure that the service requestors will get aggregated survey results, but will not learn personal information of any individual data owner. At the same time, the crowdsourcing platform will learn neither the information of individuals nor the aggregated survey results. In addition, different from existing work, by which data owners pass their data to survey agency and lose control on their data, our solution stores data at their owners' smartphones and allow the owners to control how their data will be used.**

I. Introduction

Survey is a very important method for companies to conduct marketing. It is also very useful for the government to learn the opinions of citizens. Traditionally, survey is done face-to-face via questionnaire. Such a method is costly and inefficient. Thus, it usually covers a very small carefully selected sample. Furthermore, when questioned face-to-face by agent, especially for sensitive questions like whether with diabetes or not, a respondent could be under pressure. Nowadays, with the prevailing of Internet and smartphones, most surveys are done online or via mobile applications. The data collection is far much more efficient, diverse (e.g., by SMS, email, and web), and less intrusive.

Data collection in survey needs to be carefully handled. We believe that the following issues need to be well addressed:

- *Data reuse*. Some data, e.g., demographic values, financial attitudes, and social behavior, are stable over a period (e.g., 1 year). For such data, it is better to reuse them, instead of asking respondents to input them every time.
- *Data Privacy*. Some data contains sensitive information (e.g., Salary, number of owned properties, and disease). There is a strong need to protect them. With the privacy protection, data owners are more likely to contribute their data in the survey.
- *Access control*. Data owners should have the control on how their data will be used.

In the above issues, data reuse is not a challenge with the survey being done via mobile applications and/or web. Data privacy and access control are challenging, and need to be carefully addressed. One possible solution is to apply existing secure data outsourcing [1], [2] techniques, which are closely related with searchable encryption. It works as follows. Data owners upload their encrypted data (together with some secure indexing for search) to a semi-trusted outsourcing server. Given queries from authorized requestors, the server processes them on the encrypted data, and returns results. Requestors may need a post-processing of the results to obtain the final results. In such a solution, the bottleneck is at the outsourcing server, since query processing on encrypted data is very expensive. For example, the solution [3] needs on average 38 seconds to run the TPC-H Query 1[1] on the encrypted data.

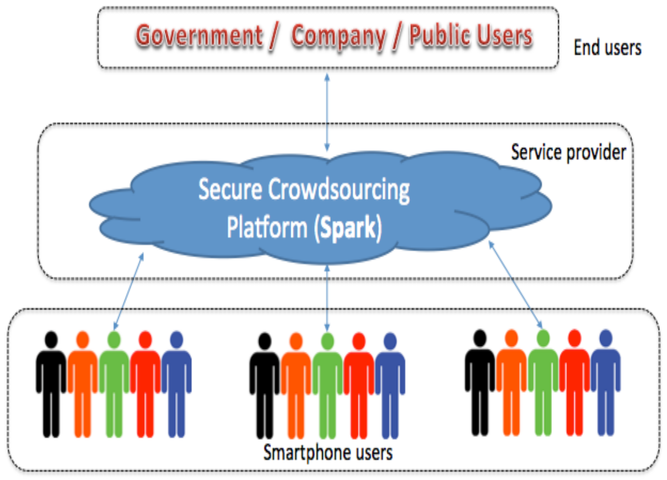
Government / Company / Public Users
End users
Service provider
Secure Crowdsourcing Platform (Spark)
Smartphone users

Fig. 1: Our Secure CrowdSourcing Platform

An alternative solution is secure crowdsourcing. The key idea is to store personal data at the smartphones of their owners and shift most workload from the server to the massive network of smartphones. Figure 1 gives the architecture. The `Service Provider` is a crowdsourcing platform, which is in the middle between `smartphone users` and `service requestors` (e.g., companies, government, and public users). Generally, the service provider accepts service requests from the requestors, and decompose them into small tasks that can be crowdsourced. It then pushes these crowdsourced tasks to the network of smartphones, and coordinates them to complete the assigned tasks. Finally, the server integrates

[1]TPC-H Query 1 can be retrieved from https://examples.citusdata.com/tpch_queries.html and TPC-H specification is on http://www.tpc.org/tpch/

the partial results from the smartphones, and send them to the service requestors.

Clearly, our crowdsourcing solution supports data reuse. The data is stored at its owner's smartphone. So access control policies can be specified by the owner to decide who is authorized to access which part of the data. For the data privacy, we adopt cryptography techniques, so that the results sent to service requestors via crowdsourcing server are aggregated results. They do not disclose the private personal information of any individual smartphones user. At the same time, the crowdsourcing server neither knows the aggregated results [2] nor the individual information. Furthermore, our solution is also scalable — the crowdsourcing server is lightweighted and can process many requests in parallel and support big number of smartphones.

II. Background

Many partial homomorphic cryptosystems [4], [5], [6] have been proposed based on public key cryptography. One representative of them is Paillier Cryptosystem [5]. In the following we briefly discuss the concept of the public key cryptography and Paillier Cryptosystem.

Public key cryptography (PKC). pkc [7] is based on two keys: one key for encryption and the other for decryption. It consists of three phases – key generation, encryption, and decryption. In the key generation, a pair of keys, public key (pk) and secret key (sk), are generated. Let $E[.]$ and $D[.]$ be the encryption and decryption functions, respectively. In the encryption phase, given plaintext M and the public key pk, the ciphertext of M is computed as $C = E_{pk}[M]$. In the decryption phase, C can be decrypted using the secret key sk by $M = D_{sk}[C] = D_{sk}[E_{pk}[M]]$. The security of public key cryptography is based on NP-hard problems, e.g., factorization in RSA [8]. Hence, there is no efficient algorithm that can determine the secret key sk given that the public key pk, or recover the plaintext M given the ciphertext C and the public key pk.

Paillier Homomorphic Cryptosystem. The Paillier cryptosystem [5] is a public key cryptosystem with additive homomorphic properties. Let (pk, sk) be a public/private key pair, and x_1 and x_2 be plaintexts. The product of ciphertexts of x_1 and x_2 decrypts to x_1+x_2, i.e. $D[sk, (E[pk, x_1]\cdot E[pk, x_2])] = x_1+x_2$. The ciphertext of x_1 raised to the power of x_2 decrypts to $x_1 \cdot x_2$, i.e., $D[sk, (E[pk, x_1]^{x_2}] = x_1 \cdot x_2$. The Paillier cryptosystem is semantically secure under the intractability assumption of decisional composite residuosity. That is, for any pair of plaintexts x_1 and x_2, the distributions of their respective ciphertexts are computationally indistinguishable.

Secure Multiparty Computation (SMC). Secure Multi-party Computation (SMC) [**?**] allows multi-parties to jointly compute a function over their respective inputs, while keeping every input confidential. Let $x_1, x_2, \cdots, x_n$ be the inputs of n parties respectively, and $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ be an n-ary function. SMC hides all x_i's, but computes $f(x_1, x_2, \cdots, x_n) = \{f_i(x_1, x_2, \cdots, x_n)\}_{i\in\{1,2,\cdots,n\}}$, where $f_i(x_1, x_2, \cdots, x_n)$ is the output to the i-th party.

[2]Aggregated results at crowdsourcing server are encrypted, and only the authorized service requestors can decipher them.

III. Privacy-Preserving Survey by Crowdsourcing Platform

In this section, we propose our crowdsourcing platform for privacy-preserving survey. The platform's diagram is shown in Figure 1. It consists of 3 different entities: the service provider, the service requestors, and the mobile phone users. They act as different roles as defined in Table I. We assume that all of 3 entities are honest but curious (i.e., semi-honest) entities. They always follow the protocol specification but try to infer extra information from the output.

In the following we will first discuss the setting of the platform. Then we will introduce some secure operators that are needed for secure computation. We will provide case study of showing how the secure operators work. Finally, we will discuss how to enforce access control.

TABLE I: Entities with their roles in the platform

Entity	Role
Service provider	*The crowdsourcing platform.* It coordinates the mobile phone users to performs secure aggregated analytics. The platform can run on Hadoop Spark [**?**] to support large-scale parallel and scalable computation.
Mobile phone users	*Data provider.* They participate in the aggregated analytics as requested by the platform.
Service Requestors	*Data consumer.* They are data users, including government, companies, and public users. They use the platform to perform aggregated analytics that use data of the mobile phone users without violating their privacy.

A. The registration

Service requestor and mobile phone user need to register with the platform before using it. They need follow the following setup steps in the registration.

i) Any new service requestor or mobile user needs to download a mobile app provided by the platform. They are only allowed to use the provided app to interact with the platform.

ii) After downloading the mobile app, they use it to create account with the platform. They need to use the credentials provided by platform for authentication purpose.

iii) Service requestor has a pair of keys – the public and private keys. The public key is registered with the platform.

B. Secure Operators

The crowdsourcing platform coordinates the massive network of mobile phone users to carry out secure aggregated analytics. When useing data of the mobile phone users, the platform ensures that personal privacy of each individual data owner is not violated privacy. The privacy guarantee is based on a set of secure operators. Most of them have already been proposed in our previous work [9]. Here, we present 2 secure operators for illustration purpose: *secure summation* and *secure max*. These secure operators are basically secure multi-party computation (SMC) protocols [10]. In the following we will describe them.

Secure Summation. Suppose that mobile phone user u_i has value z_i, where $i = 1, 2, \ldots, n$ and n is the number of users in the summation operation. The task of the summation is to compute $\sum_{i=1}^{n} z_i$. Let (pk, sk) be the public and secret key pair of the service requestor for the Paillier cryptosystem. For privacy protection, u_i encrypts z_i by pk, and sends $E_{pk}[z_i]$ to the platform. After receiving all the encryptions from all the mobile phone users, the platform computes $C = \prod_{i=1}^{n} E_{pk}[z_i]$, which is equal to $E_{pk}\left[\sum_{i=1}^{n} z_i\right]$ (see Section II for the homomorphic property of the Paillier cryptosystem). The platform sends C to the service requestor. The service requestor then decrypts C with the secret key sk, and get $D_{sk}[C] = \sum_{i=1}^{n} z_i$. At the end of the protocol, the service requestor learns only the summation output $\sum_{i=1}^{m} z_i$ and nothing else. At the same time, the platform learns nothing.

Simple analysis. The correctness of the secure summation operator is guaranteed by the homomorphic properties of Paillier cryptosystem. The computation workload of the platform is small, since all expensive modular exponentiations are done by smartphones. The platform only needs to perform multiplication operations, which can be efficiently done. The security of the summation operator is ensured by the semantic security of Paillier cryptosystem.

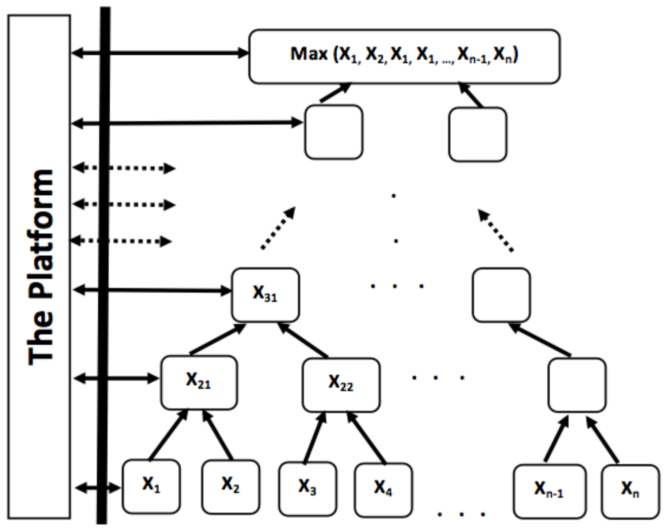

Fig. 2: Hierarchical comparison by secure max

Secure Max. Given a set of mobile phone users u_i with value x_i for $i = 1, 2, \ldots, n$, the secure max protocol computes $\max\{x_1, x_2, ..., x_n\}$, while keeping individual x_i value confidential. The protocol works in a bottom-up way as shown in Figure 2. It compares pairs of values. In each comparison, the bigger value will go one level up and join the next round of comparison. At the root, the protocol outputs the max value. Note that except for the max value (that is delivered to service requestor), all the other values are kept confidential from the service requestor.

The secure max protocol is based on a basic protocol – given inputs a and b of Alice and Bob, respectively, it outputs c_1 to Alice and c_2 to Bob, such that $c_1 + c_2 = \max\{a, b\}$. The implementation of such a basic protocol can be simple by applying CMP [11] between Alice and Bob to learn whether $a > b$ or not. If $a > b$, then Alice generates a random value c_1 for Bob, and computes $c_2 = a - c_1$. Otherwise, Bob generates random value c_1 for Alice, and computes $c_2 = b - c_1$. This simple protocol leaks the information about whether $a > b$. For a more secure solution, which outputs c_1 and c_2, and hides whether $a > b$ from Alice and Bob, please refer to our technical report {**TODO:** *cite our TR*}.

On the basis of the basic max function, we propose our full protocol to securely compute the max of $\{x_1, x_2, ..., x_n\}$. For the simplicity of discussion, let us first consider the 4 values x_1, x_2, x_3, x_4 at the lower left corner in Figure 2. In the protocol, u_1 and u_2 first apply the basic max protocol to compute $c_1^1 + c_2^1 = \max\{x_1, x_2\}$, where u_1 holds c_1^1 and u_2 holds c_2^1. Similarly, u_3 and u_4 apply the basic max protocol to compute $c_1^2 + c_2^2 = \max\{x_3, x_4\}$, where u_3 holds c_1^2 and u_4 holds c_2^2. Then, the comparison goes 1 level up to compute $\max\{\max\{x_1, x_2\}, \max\{x_3, x_4\}\}$, or equivalently, whether the following inequality holds:

$$c_1^1 + c_2^1 > c_1^2 + c_2^2. \tag{1}$$

For this, u_2 sends c_2^1 to u_3, and u_4 sends c_2^2 to u_1. The inequality is then transformed to

$$c_1^1 - c_2^2 > c_1^2 - c_2^1. \tag{2}$$

u_1 and u_3 then apply CMP [11]. If Inequality 2 holds (i.e., $\max\{x_1, x_2\} > \max\{x_3, x_4\}$), u_1 sets $c_1^3 = c_1^1$ and u_2 sets $c_2^3 = c_2^1$. Otherwise, $\max\{x_1, x_2\} \leq \max\{x_3, x_4\}$, and u_1 sets $c_1^3 = c_2^2$ and u_2 sets $c_2^3 = c_1^2$.

Clearly, the above comparison can be recursively applied. At the root in Figure 2, c_1^k and c_2^k will be generated, such that $c_1^k + c_2^k = max\{x_1, x_2, ..., x_n\}$, where k is the tree depth. Let (pk, sk) be the public and secret key pair of the service requestor. The 2 users in the last round of comparison sends $E_{pk}[c_1^k]$ and $E_{pk}[c_2^k]$ to the platform. The platform then computes $E_{pk}[c_1^k + c_2^k] = E_{pk}[c_1^k] \times E_{pk}[c_2^k]$, and sends the encryption to the service requestor. The service requestor decrypts the ciphertext to get $c_1^k + c_2^k$.

Simple analysis. Again, the correctness and security are guaranteed by the Paillier cryptosystem and CMP [11]. Clearly, the max protocol is efficient. All the mobile phone users can run the protocol in parallel. The protocol complexity is linear to the depth of tree. Given 30 millions of mobile phone users in a city, the depth of the tree is 26.

User	Age/Gender	Stay Region	Annual Salary	Owned Car
User A	30/M	Central Region	50,000	No
User B	45/M	East Region	12,000,000	Yes
User C	25/F	North Region	36,000	No
User D	54/F	Central Region	120,000	Yes
User E	38/M	West Region	80,000	Yes

TABLE II: Example data of mobile phone users

C. Two examples

We use 2 examples to illustrate how our crowdsourcing platform works with the service requestors and the mobile phone users for the secure aggregated analytics. We assume a consulting company, which plans to get reliable, relevant and timely statistics via the survey on Singapore citizens. For illustration purpose, we assume 5 Singapore citizens (just for illustration purpose) are in the survey. Table II shows some private data of these 5 mobile phone users. The consulting company submits 2 questions to the platform: 1) "*What is the average yearly salary in Singapore?*", and 2) "*What is the highest annual salary in Singapore?*".

Processing on question 1: what is the average yearly salary in Singapore? The consulting company has a pair of keys (i.e., secret key and public key) for the Paillier cryptosystem. The public key is registered at the platform. The company submits the question 1 to the platform, which then forwards it to all the mobile phone users. For the question, `secure summation operator` is applied as follows. Each mobile phone user encrypts his/her salary with the public key of the consulting company and then sends it to the platform. Consider Table II. Mobile phone user A encrypts the salary, $E[50,000]$, mobile phone user B encrypts the salary $E[12,000,000]$, and so on. The platform the computes the encryption of the summation: $S = E[50,000] \times E[12,000,000] \times E[36,000] \times E[120,000] \times E[80,000]$. Then, S and the number of the mobile phone users (i.e., 5) are sent to the consulting company. The company first decrypts S with its private key to get the salary summation $D[S] = 12,286,000$. Then, the company computes the average salary that is equal to $\frac{12,286,000}{5} = 2,457,200$.

Processing on question 2. What is the highest annual salary in Singapore? The consulting company submits the query to the platform to find the highest salary in Table II. The platform applies the secure max protocol. Users A and B apply basic max protocol to compute $\max\{50,000, 12,000,000\}$. Suppose that $c_1^1 = 10,000,000$ and $c_2^1 = 2,000,000$ [3], such that $c_1^1 + c_2^1 = \max\{50,000, 12,000,000\}$. Users C and D similarly compute c_1^2 (say, 60,000) and c_2^2 (say 60,000), such that $c_1^2 + c_2^2 = \max\{36,000, 120,000\}$. Since $c_1^1 + c_2^1 > c_1^2 + c_2^2$, $c_1^1 + c_2^1$ will be compared with $80,000$ [4], and $E_{pk}[c_1^1]$ and $E_{pk}[c_2^1]$ will be sent to the platform, where pk is the public key of the consulting company. The platform multiplies the 2 ciphertexts and sends $E_{pk}[12,000,000]$ to the company.

[3]In real implementation by cryptography c_1^1 and c_2^1 are much big, e.g., big integers of 1024 bits. Here, the small numbers are only for illustration.

[4]For the uniformity, 80,000 can be written into 2 numbers: 80,000 and 0.

D. Access Control

Our platform allows mobile phone users to control how their data will be used. Such a property can be achieved, since the data is stored at the mobile phones of users. The access control can be specified in a data access control (DAC) list [12], which can be defined by the platform with the help of access control experts together with domain knowledge experts. The DAC list can be downloaded to the mobile phones, when mobile phone users register themselves with the platform.

The policies in DAC list allow the mobile phone users to configure the access rights to their data. The phone users can select that some policies are to be enforce to authorize access, e.g., on demographic information. They can also set some policies to deny access, e.g., no data access when the user is making a phone call. The data access may also be set with economic incentive. For example, *consulting company needs to pay 1 dollar, if it accesses the information about financial status*. Furthermore, the access control is configurable dynamically. Mobile phone users can dynamically activate or inactivate policies in DAC to customize the access control for their needs. Therefore, smart phone users can have the control over their data.

IV. Conclusions

In this paper we proposed a crowdsourcing platform for privacy-preserving survey. The platform is secure – service requestor learns only the aggregated results, but not the data of any individual mobile phone user; the platform itself learns neither the data of individuals nor the aggregated results. The platform is lightweight, since all expensive operations are shifted to smartphones. Thus, it is scalable. In addition, the smart phone users are given the access control authority to control how their data is used.

References

[1] J. Loftus and N. P. Smart, "Secure outsourced computation," in *AFRICACRYPT*, 2011, pp. 1–20.

[2] H. Hacigümüs, B. R. Iyer, and S. Mehrotra, "Secure computation on outsourced data: A 10-year retrospective," in *DASFAA*, 2014, pp. 16–27.

[3] S. Hildenbrand, D. Kossmann, T. Sanamrad, C. Binnig, F. Faerber, and J. Woehler, "Query processing on encrypted data in the cloud by," *Technical Report*, 2011.

[4] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *EUROCRYPT*, 1998, pp. 308–318.

[5] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999, pp. 223–238.

[6] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *TCC*, 2005, pp. 325–341.

[7] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[8] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, pp. 120–126, 1978.

[9] S. G. Teo, J. Cao, and V. C. S. Lee, "DAG: A model for privacy preserving computation," in *ICWS*, 2015, pp. 289–296.

[10] A. C. Yao, "How to generate and exchange secrets (extended abstract)," in *FOCS*, 1986, pp. 162–167.

[11] V. Kolesnikov, A. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *CANS*, 2009, pp. 1–20.

[12] V. C. Hu, T. Grance, D. F. Ferraiolo, and D. R. Kuhn, "An access control scheme for big data processing," in *CollaborateCom*, 2014, pp. 1–7.