Privacy Preserving Data Aggregation on Secure Cloud

Saket Komawar
Dept. of Computer Engineering & IT
College of Engineering Pune
Pune, India - 411005
Email: komawarss14.it@coep.ac.in

Mayur Batwal
Dept. of Computer Engineering & IT
College of Engineering Pune
Pune, India - 411005
Email: batwalms14.it@coep.ac.in

Shubham Shah
Dept. of Computer Engineering & IT
College of Engineering Pune
Pune, India - 411005
Email: shahsd14.it@coep.ac.in

Snehkumar Shahani
Dept. of Technology
Savitribai Phule Pune University
Pune, India - 411005
Email: snehshahani@unipune.ac.in

Jibi Abraham
Dept. of Computer Engineering & IT
College of Engineering Pune
Pune, India - 411005
Email: ja.comp@coep.ac.in

***Abstract*—Secure Cloud is a cloud based data storage and computation service that works on Encrypted data without inferring any knowledge out of it. However on such secure cloud in order to perform Privacy Preserving analysis it is necessary to decrypt that data and then perform Privacy Preserving transformation. The proposed solution provides a way to perform Privacy Preserving transform on secure cloud without decrypting it. This is done by applying appropriate Differential Privacy techniques along with Homomorphic transform. The experimentation results done over the encrypted data are consistent with the utility-privacy trade-off in Differential Privacy approach on plaintext which states that the utility of data decreases as the privacy increases.**

***Index Terms*—Differential Privacy, Homomorphic Encryption, Cloud Computing.**

I. Introduction

Cloud computing has changed the way we process and utilize data. It has provided the infrastructure, platform and the power to store, manipulate and share data efficiently and reliably. Systematic and statistical analysis of such data has paved way for various new technological innovations such as medical diagnosis, fraud detection, weather forecast and many more [13].

With such analysis, there is also need to secure the data from adversaries and untrusted entities. Exposing ones private data may lead to serious implications due to large dependencies on cloud data now-a-days. For example, if a user is found to be a smoker while doing some data analysis then this piece of auxiliary information might result in increase of premium rates for his/her health insurance[9]. Thus various measures are to be taken to provide privacy to individual clients.

Also the need of providing confidentiality to the entire dataset used in the analysis is an important aspect as it can reveal the content. Hence number of encryption techniques have been implemented which securely store and retrieve data on cloud. However, such encryption techniques restrict the computations that can be performed over encrypted data, limiting the power of cloud. A malleable scheme known as Homomorphic Encryption enables operations on cipher data producing results in encrypted format. Such a scheme can be adopted to provide security to users who share data for analysis, as the private keys need not be shared with the analyst who can perform the analysis on cipher data without actually accessing the plaintext.

To provide further privacy against statistical analysis on the encrypted data, the proposed solution uses Differential Privacy coined by Cynthia Dwork[9] to perform computations in privacy preserving manner. In this mechanism, privacy is provided by perturbing or altering the data by adding a noise, derived from a continuous probability distribution such as Laplace distribution[1]. The results obtained through experimentation using proposed model over perturbed and non-perturbed data are close to one another with some error which is also called as noise. This small error in the result helps in providing privacy to an individual against statistical analysis as the analyst cannot determine an individual entity accurately through series of statistical queries.

The outline of the rest of the paper is as follows : Section 2 describes the base concepts that are core to the presented solution. Section 3 describes current state of the art techniques. Proposed technique is introduced and explained in Section 4. In Section 5, we discuss the results of experiments that have been performed to validate our proposed mechanism. The discussion ends with a conclusion and directions for future work in Section 6.

II. Concepts

A. Homomorphic Encryption

Homomorphic encryption is an encryption technique which allows to perform computation on encrypted data, producing an encrypted output which when decrypted gives the same result if we would have performed the same computation

978-1-5386-5257-2/18/$31.00

on plain data[7]. Such a scheme does not require sharing of private keys for computation and neither does the result reveal any information about the individual data items.

Definition 1: If x and y are any two data values , then function f satisfies homomorphism iff:

$$Enc(f(x,y)) = f(Enc(x), Enc(y)) \tag{1}$$

1) Paillier Encryption: Paillier Encryption is a partial homomorphic encryption technique which allows to compute addition operation on encrypted data such that the addition of encrypted data when decrypted gives same result when addition operation is performed on plain data[8]. Thus, Paillier Encryption satisfies additive homomorphic property. So, for any given plain data x and y, we have

$$Enc(x+y) = Enc(x) + Enc(y) \tag{2}$$

Along with additive homomorphic property, in case of paillier encryption, the product of two ciphertexts when decrypted will correspond to the sum of corresponding plain text.

$$D((E(m1,r1) * E(m2,r2))mod(n^2)) = (m1+m2)mod(n) \tag{3}$$

where *D* is decryption function, *E* is encryption function *m1, m2* correspond to plain texts and *r1, r2* correspond to random number chosen for encryption.

Paillier Encryption is an asymmetric algorithm for public key cryptography [8]. Public and Private keys are generated by selecting two large prime numbers and performing computations on them as prescribed by the Paillier Encryption algorithm. The plain data is encrypted by using public key and random number chosen using public key parameters. Thus, it is also called as randomized algorithm. The encrypted data can be decrypted by using private key generated. Thus, data is encrypted using public key and decrypted using private key.

B. Differential Privacy

Differential Privacy is a mechanism which guarantees privacy of personally identifiable information of an individual whether or not he/she decides to participate in the analysis[5][6].

‘

Definition 2: Mechanism M is said to provide *ϵ-differential* privacy if for all x and $y \in D_n$, which differ in only one entry and for all C $\subset$ range(M), we have

$$Pr[M(x) \in C] \quad \leq \quad e^{\epsilon} * Pr[M(y) \in C] \tag{4}$$

where *Pr* is probability function and $\epsilon > 0$.

Differential privacy ensures that an analyst cannot figure out whether an individual participated in a dataset, since the mechanism gives almost equally likely results irrespective of a particular entity's participation. The measure of privacy that can be compromised is given by the privacy parameter ϵ.

C. Sensitivity

The *sensitivity* of function is a measure of how much inclusion or exclusion of one element in a dataset can affect the output of function[9].

Definition 3: If x and y are datasets which differ by at most one element, then the *sensitivity* of function f is given by:

$$\Delta f = max|f(x) - f(y)| \tag{5}$$

Sensitivity gives the magnitude by which an individual's data can affect the function in worst case. This gives a quantifiable measure for the uncertainty we must introduce in our dataset, like data perturbation by adding appropriate noise. The sensitivity of a counting query is 1, while that of an aggregation query like summation is the maximum element in the dataset.

D. Utility

Utility is the measure of usefulness of data. It is inversely proportional to privacy. Utility gives a measure of usability of data after DP transform and thus helps in determining the effectiveness of the transform for further analysis[12]. Utility metric used in the work is given by,

$$Utility = \mu \frac{x_i - \bar{x_i}}{x_i + 1} \tag{6}$$

where μ stands for arithmetic mean, x_i is the original data value and $\bar{x_i}$ is the perturbed value.

E. Laplacian Noise

The noise which is derived by using *Laplacian Distribution* is called *Laplacian Noise*. It is generally added to output result to ensure differentially private results.

A random variable has Laplace(θ, λ) Distribution if its probability density function is :

$$f(x,\theta) = \frac{1}{2\lambda} \exp\left(-\frac{|x-\theta|}{\lambda}\right) \tag{7}$$

where variable x is a real number, θ is location parameter and λ is a real positive number [1].

III. Related Work

In 2009, Microsoft implemented a prototype of Privacy Integrated Queries (PINQ) which intends to provide differential privacy by allowing to express only those programs which are differentially private[2]. PINQ is a programming language which provides analysts with access to records through Language Integrated Queries (LINQ) which is an SQL-like declarative language which acts like an interface to original data before analysis. Before analyst executes some queries over source data, privacy cost of each query is calculated. A query fired is executed only if its privacy cost satisfies restrictions defined by the data endpoints, otherwise PINQ restricts its execution.

In 2017, Uber released Flex, which acts as an interface to SQL queries while preserving differential privacy requirements[3]. When a query is fed to Flex, it analyses the query to calculate elastic sensitivity[4].Elastic sensitivity enables to

efficiently approximate sensitivity of query without making changes in the database. Thus, sensitivity is calculated using elastic sensitivity and then noise derived is add to query's result to preserve differential privacy.

Since almost all data at endpoint is encrypted, sharing of private keys to analyst is required so as to perform aggregation and other computations required to perform analysis on provided data. This sharing of private keys is a serious security issue since analyst is not always a trusted party. Although above mentioned tools provide privacy, none of tools we are aware of provides solution to this security issue, thereby potentially compromising confidentiality of entire dataset.

IV. Proposed Solution

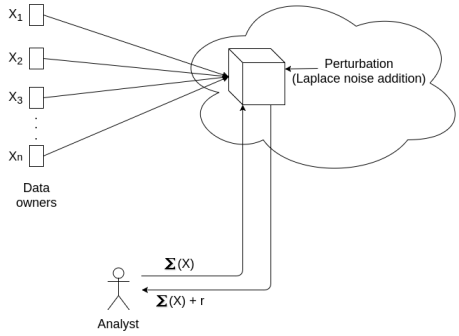

Fig. 1. Existing Architecture

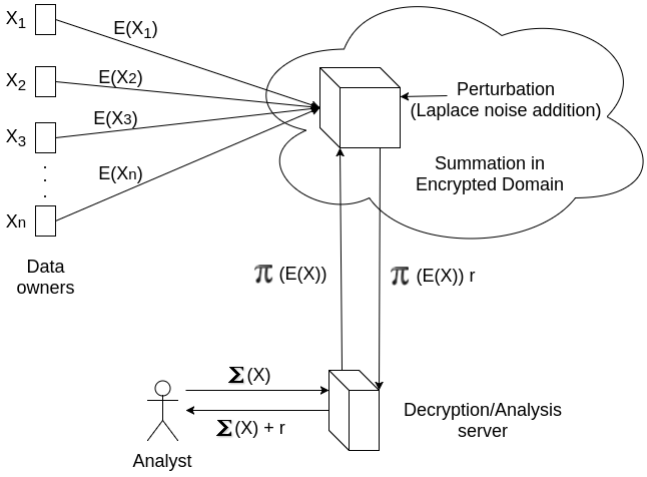

Fig. 2. Proposed Architecture

Figure 1 and 2 shows the system architecture for existing and proposed solution respectively. As per the data flow, Data owners X_1, X_2, ..., X_n share their data for analysis. In the existing setup , the data (X_i) is directly send on the cloud while in the proposed setup, the data is homomorphically encrypted ($E(X_i)$) through the public key and stored on the cloud; the private key lies with the decryption server. Each entity encrypts the data at the client side and thus reduces the overhead for encryption on cloud.

The model then computes the noise to be added using Laplace distribution(as discussed in Section II(E)) using the sensitivity and the privacy parameter ϵ, which is predetermined by the data owners. This noise is homomorphically encrypted using the same public key and the resultant cipher is added to the encrypted data ($E(X_i)$). Thus the data is perturbed in encrypted domain without actually decrypting it.

The analyst is an external entity who does not possess the private keys to data. The analysis is done by firing queries to decryption/analysis server such as an aggregation query ($\sum(X)$) in this case. The decryption/analysis server acts as a mediator which facilitates exchange of queries and the corresponding results between the analyst and the cloud which is not present in existing setup. As per equation (3), homomorphic addition includes multiplication of the cipher data, hence the corresponding query ($\Pi(E(X))$) is sent to the cloud to perform the desired aggregation. The model assumes that the sensitivity(as discussed in Section II(C)) for the desired operation is already known or can be calculated easily through domain knowledge/analysis. For example, in case of a summation query on salary dataset, sensitivity is the maximum salary given to an employee in an organisation(such as a CEO) which can be easily learnt from the annual balance sheet without invading any individual's privacy.

The aggregation query is applied on the perturbed data and the encrypted result with some noise added ($\Pi(E(X)r)$) is then returned to the decryption server. The decryption server decrypts the cipher using private key and the result is sent to the analyst ($\sum(X)$ + r).

In the proposed solution, ϵ-differential privacy is guaranteed to the data owners[9]. Also security i.e confidentiality of the dataset is ensured as the aggregation operation is carried out in encrypted domain and the analyst does not have the access to the private keys.

V. Experimentation and Results

The experimentation and analysis is done over the Employee payroll dataset for all Cook County employees excluding Forest Preserves, indicating amount of base salary paid to an employee during the County fiscal quarter[10]. The aggregation statistics is done on the attribute base salaries of employees which includes other attributes like fiscal quarter, bureau, position id to name a few. During the experimentation, Laplace noise was added to the data for achieving ϵ-differential privacy. Entire implementation has been done in Python 3.5. The Laplace noise was generated using Python's NumPy library which generates noise samples from a Laplacian distribution with given mean and scale. The Homomorphic encryption was done using python's Paillier library which is suitable for Homomorphic addition.

Since the encryption is done at the client side, there is no overhead in encrypting and storing data on cloud as the data items are stored parallelly by the endpoints. The model performs privacy preserving transform by computing noise and perturbing the data as soon as the encrypted data is stored on cloud. Thus the privacy preserving transformation incurs little

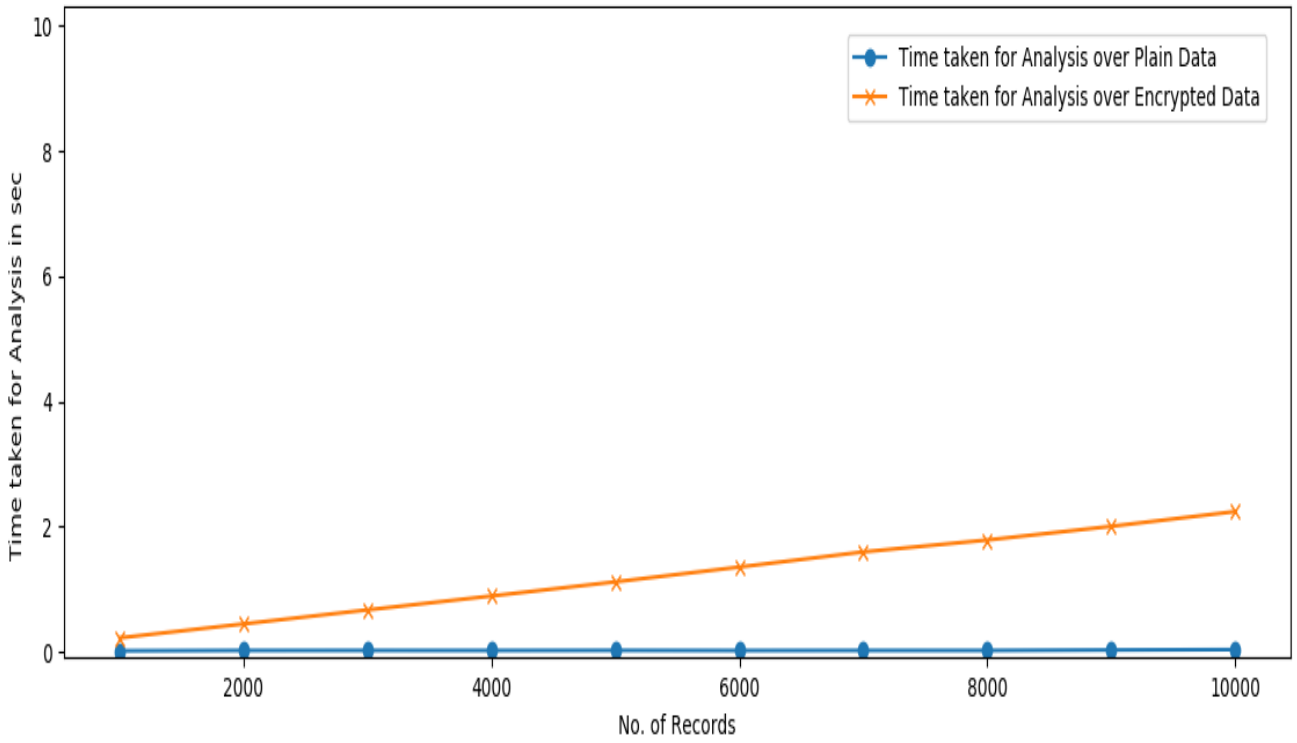

Fig. 3. Time for Analysis Vs No. of Records

Algorithm 1 Proposed model

Require: $S \leftarrow$ Sensitivity, $\epsilon \leftarrow$ Privacy parameter

Generate public key e and private key d
for all data items X_i **do**
    $E(X_i) \leftarrow$ Encrypt with public key e
    Store $E(X_i)$ on cloud
end for
PERTURBATION(S, ϵ, e)
$\sum(X) \leftarrow$ query to decryption/analysis server ▷ from analyst
$\Pi(E(X)) \leftarrow$ query to cloud for homomorphic aggregation
$\sum(E(\bar{X}_i)) \leftarrow$ aggregation operation
$\Pi(E(X)r) \leftarrow$ encrypted result to decryption server
$\sum(X) + r \leftarrow$ decrypt result with private key d
Return result to analyst

procedure PERTURBATION(S, ϵ, e)
    $r_i \leftarrow$ compute Laplace noise from S and ϵ ▷ eq (7)
    for all r_i **do**
        $E(r_i) \leftarrow$ encrypt noise using public key e
    end for
    for all encrypted data items $E(X_i)$ **do**
        $E(\bar{X}_i) \leftarrow E(X_i) + E(r_i)$ ▷ data perturbation
    end for
end procedure

or no overhead as it is done asynchronously with respect to the analysis. Experiments were performed to analyse the overhead incurred in the data analysis due to added confidentiality to the dataset by homomorphic encryption. Figure 3 compares the time required for analysis in existing architecture as per Figure 1 and the time required for analysis in proposed model. The blue plot indicates aggregation on plaintext data which incurs negligible overhead as suggested in the graph. The red plot depicts the time required for analysis on cipher data in the proposed model. This time includes aggregation operation on encrypted data and decryption of the result which is not the case with existing model. As the graph suggests, the added security incurs an overhead of 1 - 2 secs on a system with 4GB of RAM for 10,000 data records which is acceptable for many practical analytical operations. The results obtained in the above experiment are presented in Table 1.

Figure 4 shows the plot for utility of data against the privacy parameter when ϵ-differential privacy is applied on plain data as well as encrypted data for various values of ϵ ranging from 0 to 2.5. The plot demonstrates that the utility of data decreases as the privacy increases, which is consistent with the utility-privacy tradeoff in Differential Privacy approach on plaintext.

VI. Conclusion And Future Scope

The paper proposes a complete framework for implementing differential privacy practically on a secure cloud without de-

TABLE I
TIME TAKEN FOR ANALYSIS OVER PLAIN DATA AND ENCRYPTED DATA

Number of records	Analysis time for plain data	Analysis time for encrypted data
1000	0.02045	0.227043
2000	0.025951	0.449804
3000	0.026272	0.67365
4000	0.026653	0.895583
5000	0.027069	1.121422
6000	0.027913	1.357723
7000	0.028099	1.599434
8000	0.029916	1.788925
9000	0.033267	2.007628
10000	0.039398	2.243297

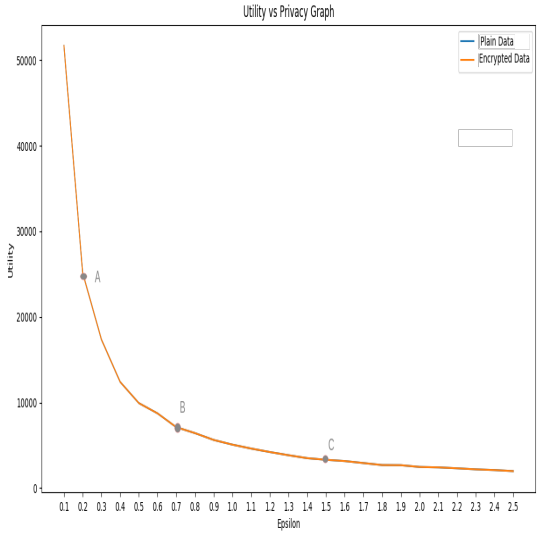

Fig. 4. Utility vs Privacy Graph

crypting the data. Experimental results show that the overhead incurred due to added security is 1 - 2 secs on a system with 4GB of RAM for 10,000 data items which is acceptable in many practical analytical applications such as Human Resource and various archived data analysis.

Figure 4 shows three points 'A', 'B' and 'C' which can be used to decide the privacy parameter ϵ as per the utility need. Value of ϵ after point 'C' can be used to provide more privacy since utility reaches the saturation point. Value of ϵ at point 'A' can be used where need of utility is an important factor and can't be compromised to a larger extent.

The proposed work can be extended further to provide differential privacy for other complex operations such as multiplication. This can be achieved if homomorphic multiplication libraries are available which provide multiplication of encrypted objects efficiently. Also multiplicative differential privacy can be adopted which might give better utility results for same perturbation.

REFERENCES

[1] C.Forbes, M.Evans, N.Hastings and B.Peacock.Statistical Distributions, 4th edition, 2011.

[2] F. McSherry, "Privacy integrated queries", Communications of the ACM, vol. 53, no. 9, p. 89, 2010.

[3] Johnson, N., Near, J. and Song, D. Towards Practical Differential Privacy for SQL Queries. [online] Arxiv.org. Available at: https://arxiv.org/abs/1706.09479, 2018.

[4] Medium. Uber Releases Open Source Project for Differential Privacy. [online] Available at: https://medium.com/uber-security-privacy/differential-priva cy-open-source-7892c82c42b6 , 2018.

[5] C. Dwork. Differential Privacy. In Automata, Languages and Programming, volume 4052 of Lecture Notes in Computer Science, pages 1-12. Springer Berlin Heidelberg, 2006.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In Theory of Cryptography, volume 3876 of Lecture Notes in Computer Science, pages 265-284. Springer Berlin Heidelberg, 2006.

[7] Craig Gentry. Computing arbitrary functions of encrypted data. Commun. ACM, 53(3):97-105, March 2010.

[8] Paillier P. (1999) Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern J. (eds) Advances in Cryptology EUROCRYPT 99. EUROCRYPT 1999. Lecture Notes in Computer Science, vol 1592. Springer, Berlin, Heidelberg.

[9] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. Theoretical Computer Science, 9(3-4):211-407, 2013.

[10] Cook County of Illinois, Finance and Administration, Employee payroll data, Fiscal Year 2016. https://catalog.data.gov/dataset/employee-payroll

[11] S.Shahani, J.Abraham, R.Venkateswaran. Distributed Data Aggregation with Privacy Preservation at Endpoint. *International Conference on Management of Data (COMAD)*, 2017

[12] A. F. Karr, C. N. Kohnen, A. Oganian, J. P. Reiter, and A. P. Sanil. A Framework for Evaluating the Utility of Data Altered to Protect Confidentiality. *The American Statistician Journal*, 60(3):224232, 2006.

[13] B.Sosinsky. Cloud Computing Bible, 2011.