Method to secure data in the cloud while preserving summary statistics

Sanchita Barman, Bimal Roy

Indian Statistical Institute

Abstract. In this paper we propose a method which will preserve summary statistics of data organized in a two way table. We have shown that fully homomorphic encryption is a powerful solution. However it has a number of disadvantages which makes it impractical. We have proposed a *Restricted homomorphic encryption* method which uses Pailier encryption and order preserving encryption. This new method can be used for practical purposes owing to it's efficiency in terms of both speed and storage.

Keywords: Fully homomorphic encryption, Paillier encryption, Summary statistics, Order preserving encryption, Restricted homomorphic encryption

1 Introduction

Summary statistics mainly includes mean, median, range and standard deviation. At present, institutes which preserve and handle statistical data store them with themselves. This requires them to use a lot of space and computational power. The existing system does not support variable storage requirement and takes a lot of time for calculating summary statistics. Thus it is very costly and inefficient. We suggest to keep the data in the cloud in an encrypted format in such a way that most of the computations in summary statistics can be done on it without decryption. We propose two ways to solve this problem.The first method uses Fully homomorphic encryption which enables us to encrypt data and perform all the operations of summary statistics without decrypting the ciphertexts. The second method is a "hybrid algorithm"which uses Pailier encryption and Order Preserving encryption designed by Popa, Li and Zelkovich. This method can be termed as *restricted homomorphic encryption.* In this paper we will see why the first method is not enough and how the second method can remove the drawbacks .
We can observe that the operations in summary statistics would mainly need addition multiplication and comparisons. Thus we will focus on the methods which will preserve these operations. Without loss of generality we have assumed that data is stored in an $n \times n$ two way table. These methods are also valid for data tables of higher dimensions. Space required to store the values encrypted by the fully homomorphic encryption method is $n^2.p(number\ of\ bits\ in\ the\ elements)$

whereas that of the hybrid methods is $n^2.p\lceil element/p\rceil$; where p is the public key of the encryption methods used. For our experiment we have taken a 7×7 matrix of 142 bytes. For the fully homomorphic encryption method the space required to store the encrypted values was 727 Kb. We found out max and min of a row or column with a program written in C++ in 2 minutes. We used the same matrix and implemented the hybrid method. It took only 34 Kb of space and less than about a second to find the min or max of a row or column. We will denote the any cloud provider by the term 'server'.

2 Methods used in our paper

We have used the following methods for preserving the summary statistics in a two way table.

2.1 Fully Homomorphic Encryption

This is a ground breaking discovery made by Craig Gentry [1]. His work has been improved quite a lot since his discovery. However none on them could be practical yet. None of the implementations are bootstrapable. In practice polynomials used in statistical calculations do not exceed order three. This is done in order to avoid the usage of more coefficients in the calculation which will in turn affect the accuracy. Hence we can use fully homomorphic encryption for calculating summary statistics. In our paper we have used the homomorphic encryption by Smart and Vercauteren [5] for simplicity. This could be done using any other fully homomorphic encryption.
The public key consists of a prime p and an integer $\alpha mod p$. The private key consists of an integer z . For somewhat homomorphic scheme :-

KeyGen(PK)**:**

```
1  Set the plaintext space to be $P = \{0, 1\}$;
2  Choose a monic irreducible polynomial $F(x)Z[x]$ of degree N;
3  Repeat:-
4  $S(x) \leftarrow B_{\infty,N}(\eta/2)$;
5  $G(x) \leftarrow 1 + 2S(x)$;
6  $p \leftarrow resultant(G(x), F(x))$.;
7  Until $p$ is prime
8  $D(x) \leftarrow gcd(G(x), F(x))$over $F_p[x]$;
9  Let $\alpha \in F_p$ denote the unique root of $D(x)$;
10 Apply the XGCD-algorithm over $Q[x]$ to obtain $Z(x) = \sum_{i=1}^{N-1} z_i * x^i \in Z[x]$ such
   that;
11 $Z(x)G(x) = pmodF(x)$;
12 $B \leftarrow z_0(mod 2p)$;
13 The public key is $PK = (p, \alpha)$, while the private key is $SK = (p, B)$.
```

Encrypt(M, PK)**:**

```
1 Parse $PK$ as $(p, \alpha)$;
2 . If $M \in \{0, 1\}$ then abort;
3 $R(x) \leftarrow B_{\infty,N}(\mu/2)$;
4 $C(x) \leftarrow M + 2R(x)$;
5 $c \leftarrow C(\alpha)(modp)$;
6 Output $c$.
```

Decrypt(c, SK)**:**

```
1 Parse $SK$ as $(p, B)$;
2 $M \leftarrow (c - cB/p)(mod2)$;
3 Output $M$ .
```

Add(c_1, c_2, PK)**:**

```
1 Parse $PK$ as $(p, \alpha)$;
2 $c3 \leftarrow (c_1 + c_2)(modp)$;
3 Output $c_3$ .
```

Mult(c_1, c_2, PK)**:**

```
1 Parse $PK$ as $(p, \alpha)$;
2 $c3 \leftarrow (c_1 * c_2)(modp)$;
3 Output $c_3$ .
```

In order to make the scheme fully homomorphic Smart et all introduces a concept of recrypt which takes the “dirty ciphertext ”as input and produces“cleaner output”. They also modify their KeyGen algorithm accordingly.

KeyGenF(PK)**:**

```
1 Generate $s_1$ uniformly random integers $B_i$ in $[p, ..., p]$ such that there exists a
  subset $S$ of $s_2$ elements with $\sum_{j \in S} B_j = B$ over the integers;
2 Define $sk_i = 1$ if $i \in S$ and 0 otherwise. Notice that only $s_2$ of the bits $\{sk_i\}$ are
  set to one;
3 Encrypt the bits $sk_i$ under the somewhat homomorphic scheme to obtain
  $c_i \leftarrow$Encrypt$(sk_i, PK)$;
4 The public key now consists of $PK = (p, \alpha, s_1, s_2, \{c_i, B_i\}_{i=1}^{s_1})$.
```

2.2 Pailier Encryption

There exists quite a number of additive homomorphic encryption methods. The Pailier encryption[3] is most efficient among them. It is an extension of the Okamoto-Uchiyama scheme. It is based on the Decisional composite residuosity assumption: Given $N = pq$, it is hard to decide whether an element in $\mathbb{Z}_2^*$ is an N-th power of an element in $\mathbb{Z}_{N^2}^*$. No adaptive chosen-ciphertext attacks recovering the secret key is known.
If k is the security parameter. Two k-bit primes p and q are chosen uniformly at random $N = pq$. $g \in B$ is a random base.To encrypt a message $m < N$, one chooses a random value $r \in \mathbb{Z}_N^*$ and computes the ciphertext as:

$$c = g^m r^N mod N^2 \tag{1}$$

When receiving a ciphertext c, it is checked whether that $c < N^2$. If yes,the message m is retrieved as:

$$m = L(c^{\lambda(N)} mod N^2)/L(g^{\lambda(N)} mod N^2) mod N \tag{2}$$

It has useful properties like multiplication of encrypted message results in the addition of the original plaintext mod n:

$$D[E(m_1) \times E(m_2) mod n^2] = m_1 + m_2 mod n \tag{3}$$

Also raising a ciphertext to a constant power results in the constant multiple of the original plaintext:

$$D[E(m)^k mod n^2] = k \times m mod n \tag{4}$$

2.3 Order Preserving Encryption

An Order Preserving Encryption can be considered as a Partial Homomorphic Encryption where the operation is order comparison. Range queries are made often in summary statistics. Hence it might be helpful if we can preserve the sort order while keeping all the other informations intact in the ciphertext. We have used a very recent scheme of order preserving encryption by Raluca Ada Popa, Frank H. Li, and Nickolai Zeldovich [4]. It is the first order preserving scheme which achieves ideal security, IND-OCPA. Database can be efficiently maintaied since update, insertion, deletion, and query can be done without decryption.
We have modified and used their scheme to preserve summary statistics. In this section we will only talk about the parts that we have adopted. They construct a binary search tree based on the sort order of the elements which they call the *OPE tree.* They encrypt the the elements in the data using any standard IND-CCA2 symmetric key encryption scheme or a deterministic encryption scheme whose security property is that of a pseudo-random function. They store this tree in an untrusted server. The server cannot construct the tree by itself since it does not know the underlying plaintexts.
Tree Construction and *OPE encoding*:- The path of the element from the root denotes it's order. The right edge is labeled as "1 "and each left edge is labeled as "0 ". The path of each node is padded with 1000... to achieve the same length for every *OPE encoding.* The length of the *OPE encoding* is $1 + h$ where h is the height of the binary search tree. Thus we have:
OPE encoding = [path]1000...
Popa et al have ensured that the tree maintains a logarithmic tree height. Thus their scheme requires to re-balance occasionally. They have presented a technique by which encoding updates can be performed in an *one pass* over the required values. We can adopt this technique in case of any maintenance required for the database.
The following is an algorithm for tree traversal for a particular value in their scheme :-

OPE tree traversal for a value v:

```
   Input: A plaintext $m$
   Output: Whether the corresponding ciphertext $c$ is present
 1 The client asks the server for the root of the OPE Tree $c\prime$;
 2 The client decrypts $c\prime$ and obtains $m\prime$;
 3 if If $m < m\prime$ then
 4     The client tell the server 'right';
 5 end
 6 else
 7     The client tell the server 'left';
 8 end
 9 The server returns the next ciphertext $c\prime\prime$ based on the above instruction;
10 The algorithm is terminated if $m$ is found or the server arrives at an empty spot
   in the tree .
```

3 Methods preserving summary statistics

3.1 Method using fully homomorphic encryption

We have used fully homomorphic encryption scheme by Smart and Vercauteren. We have however avoided the lattice dimensions 2048 and 8192 since they are proven to be insecure[2].

Setup. The client will generate the public and the secret keys. It uses the secret key to encrypt the plaintext and keep them in the server. The server has the public key and hence it can do any operations on the ciphertext as desired by the client . The client can thus use the computational power of the cloud i.e the server compute any summary statistics . The server returns the resultant ciphertext value. The client can decrypt it to get the desired answer.

How does it work. Every Computable Function has a circuit version. A *Fully Homomorphic Encryption* can handle every boolean circuit. So any operation in summary statistics can be executed using this method.

Security. We have seen that in somewhat homomorphic encryption scheme as defined by Smart et al. decryption works only as long as the cypher text noise is within certain bounds say r. These bounds depend mainly on the parameter N. In Perl's implementation the security parameter is $r = 2^v/(2 * \sqrt{N}) = 2^{384}/16$. For a detailed discussion of the implementation please refer to The rest of the implementation relies on fully homomorphic scheme. Hence this method is secure.

Observation. This method is very accurate in their result. It is perfectly secure since the server only gets the ciphertexts and all the operations can be done

without decrypting the ciphertexts. Hence any computation apart from the decryption can be done by the server. Thus the client do not need to store anything else apart from the private key and if it is concerned about the authentication , then it can save the Merkle hashes of the ciphertexts. Summary statistics does not involve computation of a high degree polynomial [three is maximum] in practice, since it increases the number of atributes and hence lowers the accuracy.
However this method has it's drawbacks too. We have implemented this method on a machine which has RAM of size 7.7 Gb, Intel i5 processor with a speed of 3.6 GHz. My input file was of 148 bytes and the output file was of 727 KB. This exponential increase in file size can render this method completely impractical if space is costly. Also my program took about 2 minutes to be executed. Thus as the theory had rightly suggested, fully homomorphic encryption will take a lot of time to be executed.
Thus we have suggested a different method in which we have addressed these issues.

3.2 Hybrid method using Restricted Homomorphic Encryption

The previous method generates a huge space expansion for creating the ciphertext. Thus although the first method is computationally accurate, it is economically not practical. Hence we can look at a different method which will not produce such huge ciphertexts and at the same time serve our purpose.
This method will not preserve all the operations for summary statistics . We observe that addition is the most used operation in summary statistics. Thus we will choose an encryption method which is additive homomorphic. Range querries is another common requirement. Thus we suggest the use of an Order Preserving Encryption .
We have used Paillier encryption system for additive homomorphism and the Order preserving encryption method by Popa, Li, and Zeldovich. A similar model was also used in database encryption .The client will encrypt the data and upload it on the server . The server will time to time send back the ciphertext as according to the need of the client and also do the operations as asked by the client. We will deviate from the model proposed by Zelcovich to suit our purpose in some places.
We will also prove that this method is secure for deploying in a third party server like the cloud. Beside preserving summary statistics it can also be used for carrying out database operations like insertion, deletion and update which we will not cover in this paper.

Setup. The server has the $n \times n$ table, in which each cell has three values :-

- The data encrypted by Pailler encryption method.Let us call it P-ciphertext.
- *OPE encoding* according to the row to which the element belongs. Let us call this *OPE_col*.

– *OPE encoding* according to the column to which the element belongs. Let us call this *OPE_row*.

The client encrypts the message by Paillier encryption. It does not reveal both the public and secret keys used for Paillier encryption to the server. Both the keys always remains with the institute which handles the data. The secret key can be shared with some authenticated client outside the institute who are only be able to decrypt the ciphertexts. The idea is to only allow the institute to encrypt data and do the computations which the server would not be able to do. The ciphertext consists of three parts. The *OPE encoding*s are computed according to that of Popa et al. The server can compute both *OPE_col* and *OPE_row* by traversing the tree as done in the method by Popa et al. The decimal conversion of this gives us the OPECode. The path of the node can be easily recovered from the OPECode. However unlike the scheme by Popa et al in our case we need not re-balance the tree since the sort order remains the same without it. The first element in any row or column in out data table is the root of the tree.
Since we are not revealing any of our keys we will not require to authenticate the elements like Popa et al.

How does it work. The server stores the $n \times n$ table containing the ciphertexts. Since the public key for Paillier Encryption, (n, g) is not revealed to the server it does the multiplication and exponent operation and returns the result without doing the $modn^2$ operation . The client does the rest of the operation. The result will remain the same. Thus operations which require addition and difference like mean, weighted mean, and range can be calculated without decrypting the ciphertext. If multiplication is required [like for calculating standard deviation] the client calls back the ciphertext corresponding to the region it wants to do operation . Exploiting the property of Pailier encryptions, it just has to decrypt one of the ciphertext if two corresponding plaintext has to be multiplied. Thus the number of decryptions will be considerably reduced, hence reducing the compelxity of the algorithm.
Range queries is most frequent requirement for summary statistics. This can be done easily since the order of the underlying plaintext is known. We make a binary search tree of the rows and the columns of plaintext matrix to find the OPECode and hence the order of the plaintext. The following are some common operations of summary statistics which are usually used. However any other operation like finding the trend or measure of shape of the distribution can be found with this method.

Range of a row or column: For the particular row or column the element with the highest and the lowest OPECode can be obtained The corresponding P-ciphertexts say c_h and c_l are obtained from the table. The client can find out the multiplicative inverse for one of the P-ciphertexts and send it to the server. Without loss of generality let us assume that the inverse for c_l is obtained say c_l^{-1}. Since this requires the the public key for the Paillier encryption the client cannot find out the inverse. The server then multiplies c_h and c_l^{-1} and send it to the authenticated client to decrypt the answer.

Mean and weighted mean: The server multiplies the P-ciphertexts of the required row or column. For weighted mean the P-ciphertexts are raised to the power of their corresponding weights before multiplying them together. The server the send the result to the client who will do the $modn^2$ operation and decrypt to get the result.

Standard Deviation: The server and the client find the mean of the row or column. The client finds the inverse of P-ciphertext of the mean. The server returns the multiplication of the P-ciphertexts and the inverse of the mean to the client. The client has decrypt the P-ciphertexts and do the rest of the operations to find the standard deviation.

Correctness and Security. In the Order Preserving encryption proposed by Popa et al they have used a deterministic encryption scheme or a IND-CCA2 symmetric encryption scheme . They have shown that their scheme is IND-OCPA secure. We will show that our adaptation of their method is secure in a certain standard that we will just describe.
A scheme is IND-OCPA (indistinguishability under orderer plaintext attacks) when the adversary which notices interaction between the server and the client guesses the security parameter by less than a probability of half. In our case we have not used any security parameter for producing the OPECodes which actually reveals the order. We will mostly depend on the security of the Pailier encryption method. We are not revealing either the public key or the private keys. Thus it is not possible for anyone else except the client to either encrypt or decrypt . Our proof is similar to that done by Popa et al, only that we will use it in our setting on the modified scheme that we have suggested.
Let us consider an adversary Adv and two sequence of values Adv asks $v = (v_1, v_2.....v_n)$ and $w = (w_1, w_2....w_)$. The view of the Adv consists of the information the server receives. We will consider two cases when the client chooses to encrypt v and the case when it chooses to encrypt w. We will show that the view of the two cases in the view of Adv is same.
We will proceed inductively as the number of values to be encrypted. The base case is when no values are encrypted which is also seen by the Adv. Now consider that after i encryptions , Adv has the same information distribution for both the cases . We need to show that it remains same after the next step as well. At the i+1 th step there are two possibilities:-
The first possibility is that the encryption of v_i is in the ciphertext table. Then the encryption of w_i is also in the corresponding ciphertext table, since both the sequence have the same order relation. So now what the Adv sees is a table which consists of ciphertexts and the same pattern of OPECodes. Now since Pailler encryption is IND-CPA secure it cannot distinguish between the encryption of the sequences it has used. So no information is leaked other than the pattern of the ciphertexts.
The second possibility is that the encryption of the sequences are not in the

ciphertext table. Now since both the sequences have the same order relation the path taken by them will be the same. So the only information obtained by the client is the path and nothing else.
Having said this, we should also mention the security breach this scheme suffers from. Pailler encryption is IND-CPA secure. However if an adversary asks the client to encrypt two values of it's choice, it can definitely distinguish between the encryptions of the two corresponding plaintexts. Thus in order to prevent this attack, we will not reveal any of the keys. Also care must be taken to prevent exposure to such attacks.

Observation. As we can see that not all the operations can be done without decrypting the ciphertext in this method. Since Pailier encryption is preserves only addition , we will need to decrypt the ciphertext whenever multiplication will be needed. However due to the self binding properly of Pailier encryption , the underlying plaintext can be multiplied with a known constant without decryption. Thus we can easily compute weighted mean and mean without decryption.
The main advantage lies in the size of the ciphertext. We took the same file as we did in the previous method as the input. The size of the output is 34KB which is lesser than that of the previous method. Also the time taken to do the computations will be less , since the complexity will depend on the number of values in the database rather than the number of their bits.
Thus if the client keeps a buffer space with itself for multiplication ,this method can be more advantageous for deploying in the cloud.

References

1. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
2. Chunsheng Gu. Cryptanalysis of the smart-vercauteren and gentry-halevi's fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:328, 2011.
3. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
4. Raluca A. Popa, Frank H. Li, and Nickolai Zeldovich. An ideal-security protocol for order-preserving encoding. *IACR Cryptology ePrint Archive*, 2013:129, 2013.
5. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. *IACR Cryptology ePrint Archive*, 2009:571, 2009.