Secure Multi-party based Cloud Computing Framework for Statistical Data Analysis of Encrypted Data

Harsha S. Gardiyawasam Pussewalage†, Pasika S. Ranaweera‡, Vladimir A. Oleshchuk† and Indika A. M. Balapuwaduge†
†Dept. of Information and Communication Technology, University of Agder (UiA), N-4898 Grimstad, Norway
‡Dept. of Electrical and Information Engineering, Faculty of Engineering, University of Ruhuna, Sri Lanka
Email: †{harsha.sandaruwan; vladimir.oleshchuk; indika.balapuwaduge}@uia.no, ‡pasika@eie.ruh.ac.lk

***Abstract*—Secure multi-party computation (SMC) is a paradigm used to accomplish a common computation among multiple users while keeping the data of each party secret from others. Cloud computing is a next generation computing solution which allows its users to use high speed infrastructure and services provided by cloud service providers (CSP) in a cost effective manner. Therefore, deployment of cloud based architecture for SMCs would aid in improving its performance and efficiency. However, cloud based solutions raises concerns over security of users' private data, since data is under the control of an external entity when outsourced to cloud platforms. In this paper we have proposed a Secure Multi-party based Cloud Computing Framework which can ensure security, privacy and anonymity of users' private data. This framework is modeled by considering a scenario which requires outsourcing statistical parameter computation of private sales data of an organization's sales personnel. The results that we have obtained, provides significant evidence of feasibility of multi-party based cloud computing solutions.**

Keywords: ***secure multi-party computation, cloud computing, data security, privacy, anonymity***

I. Introduction

In general terms, a SMC can be defined as a scenario where n parties who are having private inputs x_1, x_2 ..., x_n interested in computing the value of the public function $f(x_1, x_2..., x_n)$ in such a way that at the end of the computation no party is learned any of the private inputs of other parties [1]. The concept of SMC was first introduced by Yao in 1982 through the "millionaire problem" [2] and since then SMCs have being deployed in a variety of applications such as voting systems, auctions, business related private computations and privacy-preserving data mining, etc. Theoretically, a SMC is represented with the existence of a trusted third party (TTP) which does the required user intended multi-party computation. However, this is practically infeasible due to the fact that an external entity cannot be trusted to hand over the private data of users. Therefore, SMC is all about finding appropriate cryptographic protocols that can replace the use of a TTP, to carry out a certain user intended function while ensuring data privacy of users [3].

The term cloud can be seen as an enhancement done to the Internet in terms of hardware virtualization and resource sharing. Cloud computing can be visualized as computing over Internet. More precisely, it is a set of resources and facilities offered to its users economically via the Internet [4]. A cloud makes it possible for its users to access their information in the cloud from anywhere, anytime through the Internet. On the other hand users do not need to worry about the maintenance and availability of resources, due to the fact that it is the responsibility of the CSP. More importantly cloud computing is an on demand service, where users are charged only based on their resource consumption. Because of such benefits, cloud computing has become more and more popular among business entities.

The main issue with most of the traditional SMC protocols is that they incur a significant amount of communication overhead affecting the efficiency of the protocol [5]. As a solution, it is possible to outsource the computations to a CSP which would help to reduce the expenditure as well as the operational overhead. However, the difficulty that we face is how we can enforce the security requirements of a multi-party computation such as data security, privacy and anonymity; when we are dealing with an un-trusted external entity. This issue can be addressed through secure multi-party cloud computing solutions.

The rest of this paper is organized as follows. Related work is explained in Sec. II and then the case that we have formulated is introduced in Sec. III whereas Sec. IV describes the proposed solution. The performance evaluation of the proposed solution is given in Sec. V before the paper is concluded in Sec. VI.

II. Related Work

The requirement for the distributed computing systems emphasized with the advancement of Information and Communication Technology (ICT) has led to the introduction of cloud computing concept which provides a high speed infrastructure for the users with low maintenance and high availability. However, users are concerned about the confidentiality and integrity of data in the cloud servers [6]. Therefore, the adoption of cloud computing techniques has been greatly

inhibited due to the issues of data security, privacy and anonymity associated with them [7]. The researchers have proposed different approaches to overcome the drawbacks in cloud computing. A security protocol has been proposed in [5], to assure data protection while ensuring better performance under normal circumstances. Reducing the usage of sensitive information is another scenario which has been considered to avoid misusing and stealing of user data [8][9]. It is also important to keep the identity of the user anonymous [10].

Issues with data confidentiality of cloud users have tempted the requirement for encrypting the user data before sending it to cloud. Solutions based on homomorphic encryption schemes have been introduced [11]. Homomorphic encryption schemes such as Paillier and RSA systems support operations like addition and multiplication on encrypted data [11]. Furthermore, two additive symmetric key homomorphic schemes called as iterated hill cipher (IHC) and modified Rivest scheme (MRS) has been suggested in [12]. Wireless sensor network (WSN) based applications exemplifies the requirement of encrypted data based computations due to the fact that such computations are not feasible within the sensor [13]. Method introduced in [13] aggregates the sensor data and forwards it to an entity with high computational power. Similar method with public key based scheme to ensure user privacy has been proposed in [14]. Paper [15] has presented a method to carry out combination of additive and multiplicative operations on encrypted data using Paillier and RSA cryptosystems.

Though there has been a significant improvement in the areas of cloud security and encrypted data processing, multi-party based cloud computing solutions have not yet being developed. Such solution is proposed in [1] where they have introduced a secure multi-party cloud computing framework (SMCC) which allows multiple users to perform any common computation of their interest in the cloud. Even though the SMCC method and security protocols introduced in [1] ensures data security and identity anonymization of users, it has not been validated against practical scenarios.

III. Case Description

The considered case involves sales management in a particular organization called 'ABC'. The management of the organization is interested in analyzing mean, variance, standard deviation, skewness and kurtosis of the daily sales values of their employees for the purpose of sales related decision making. It is of organization's best interest to outsource the multi-party statistical computations to a CSP while outsourcing the analytical work to an external analyzer due to the lack of computational resources. Furthermore, security and privacy of user inserted sales data becomes a crucial aspect due to the engagement of external parties invoking the necessity of an efficient security protocol. We are going to consider this case for presenting the secure multi-party based cloud computing solution throughout this paper.

IV. Secure Multi-party based Cloud Computing Framework

Fig. 1 illustrates the architecture of the secure multi-party based cloud computing framework that we propose to address the case introduced in the previous section. Framework consists of four entities: proxy server, cloud server, analyzer, and parties who are taking part in the computations. The main functionality of the cloud server is to perform the required statistical computations upon the reception of user data while the function of the proxy server is to hide the identity of each of the users to provide identity anonymization. Furthermore, the analyzer is the external party which receives the statistical parameters of user sales data which is used for analytical purposes. The functionalities of each of these entities are illustrated in the following subsections while explaining how a secure computation can be achieved with the proposed framework. Moreover, we have used the following cryptographic keys in the proposed framework.

- RSA public and private key pair for proxy server, cloud server and the analyzer for authentication purposes.
- We have used extended ElGamal public key encryption scheme (EEES) [16] to encrypt private sales data of users. Therefore, we generate EEES keys at the analyzer end and send the parameters required for encrypting the data (El_para) to the users via the authentication process. El_para includes integer values p, g, N, y and k. p is a large secure prime value and $N = p \cdot q$ where q is also a large secure prime value. Furthermore, g represents a root of $GF(p)$ and $y = g^x \ mod \ p$ where x denotes the private key of the encryption scheme. k is a positive number selected by users when encrypting the data [16].
- We have used two 3DES keys to encrypt messages conveyed between the entities during computation time.

A. Authentication and Key Exchange

In the proposed framework, cloud server and analyzer are external entities. Therefore, it is necessary to mutually

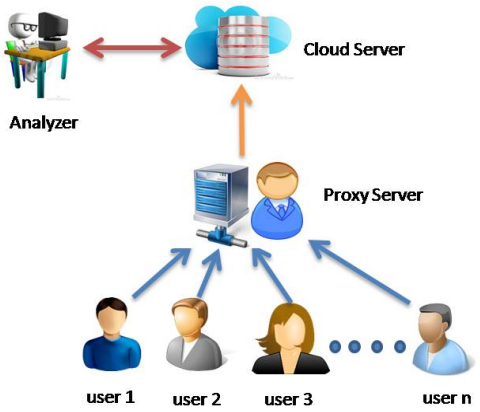

Fig. 1. Proposed Secure Multi-party based Cloud Computing Framework

authenticate cloud server and analyzer as well as cloud server and the proxy which is the exit point of the organization's network. Let us assume that RSA public keys of each entity are known to all other entities. Fig. 2 represents the procedure for establishing authentication between the cloud server and the analyzer. In Fig. 2 RSA signing and encryption procedures are denoted with usual notations [ ] and { } respectively. The authentication process is initiated with analyzer sending an authentication request. After receiving it, cloud server generates a response including the current timestamp (TS), the secret key phrase (KP_{DES_CA}) to generate the 3DES key to be shared with the analyzer (K_{DES_CA}) and send it to the analyzer after signing with RSA private key ($[KP_{DES\_CA}, TS]_{Cloud}$) and then encrypting with the public key of the analyzer ($\{[KP_{DES\_CA}, TS]_{Cloud}\}_{Analyzer}$). Then, at the analyzer end, the received message is decrypted accordingly and check whether the TS is within the defined clock skew to authenticate the cloud server. If authenticated, analyzer attains KP_{DES_CA} and generates K_{DES_CA}. In order to authenticate itself to the cloud server, analyzer generates a message by incrementing received TS value by one and attaching El_para which is signed and encrypted by RSA private key of analyzer and public key of the proxy server respectively. Among the values in El_para; p, g, N and y are public values while k is a positive number usually selected by users when encrypting the data. Therefore, k becomes a security critical parameter for the encryption scheme. Hence, we are sending El_para in an encrypted environment. Thereafter, the complete message is signed and encrypted with RSA private key of the analyzer and RSA public key of the cloud server respectively. After receiving this message at the cloud server, it decrypts the complete message and extracts the value of TS. Finally, it checks whether the received value is equals to the value of the original TS sent to the analyzer incremented by one. If it is verified, analyzer is authenticated to the cloud server whereas the encrypted El_para are retrieved and saved to be sent to the proxy server.

We have also adopted a similar approach to authenticate proxy server to the cloud server. Hence, at the end of the authentication of proxy server and cloud server, we are able to share the 3DES symmetric key K_{DES_PC} between them while

Fig. 2. Mutual Authentication between Cloud server & Analyzer

El_para will be saved at the proxy in order to be transferred to the users.

It is also important for us to authenticate users with the proxy server to make sure that only the valid users are allowed to take part in the computing process. In order to accomplish this, we have created a file in the proxy server which includes secure hash algorithm (SHA-1) hashes of username and password pairs of valid users. When a user logs into the system, SHA1 hashes of the entered username and password are transferred to the proxy server in order to be validated. If the login information is validated, El_para will be forwarded to the user end in order to encrypt the values that needed to be sent for the computations.

B. User Data Encryption

Each user can start sending private data for computations after being successfully authenticated into the system. When data is entered by a user party, first the data is encrypted with EEES by using received El_para. This encryption induces the required homomorphic properties into user data allowing required statistical computations to be carried out on the encrypted data at the cloud server. Furthermore, Eqn. 1 represents the encrypted result (A, B) for a plaintext message M when encrypted with EEES.

$$(A, B) = (g^k \bmod p, (y^k\ (M + r \times p) \bmod N) \bmod p) \quad (1)$$

It is important to note that (g, k, y, p, N) represents El_para and therefore common for all users whereas r is a positive integer randomly selected by each user. Eqn. 1 also depicts that component A of the ciphertext is independent of M, and would be common for all the users. After encryption, the resulting ciphertext values of user sales data are transmitted to the proxy server.

C. Proxy Server Functionality

The main idea of having a proxy server is to eliminate the traceability of messages sent from each user towards the cloud server. At the beginning of a computation, proxy server waits for encrypted data from all the users. After receiving all of them, it creates a new message by including all the encrypted values of users $(A, B_1), (A, B_2), (A, B_3)..., (A, B_n)$ and the number of users (n). Then a hashed message authentication code (HMAC) of the preceding message is generated with SHA1. After that HMAC is appended to the message along with the current TS. Finally, the whole message is encrypted with the 3DES key, K_{DES_PC} and forwarded to the cloud server to begin the computational process. The structure of the message sent from proxy server is illustrated in Fig. 3.

D. Cloud Server and Analyzer Functionality

After cloud server is authenticated to both analyzer and proxy server, it waits for encrypted user data from the proxy server. When it receives such a data set, then cloud server decrypts it with 3DES key, K_{DES_PC} and separates the three components in the received message which are TS, HMAC

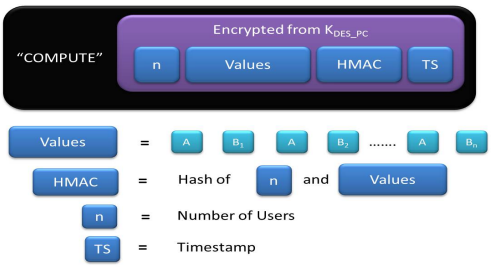

Fig. 3. Structure of the Frame sent from Proxy to Cloud Server

and the original message consisting with encrypted user data and number of users. Firstly, cloud server checks the received TS is within the defined clock skew to make sure that it is not a replayed frame. Then, if it is verified a HMAC is created with the original message part and check whether the computed and received HMACs are matching. If so, the private data of users are accepted and otherwise a retransmission request will be sent to the proxy server.

Let's Consider that encrypted private data of users are successfully acquired by the cloud server. Then it starts the statistical computations by first computing the encrypted summation (A_{Sum}, B_{Sum}). By using the additive homomorphism of EEES, it is possible to express (A_{Sum}, B_{Sum}) as;

$$A_{Sum} = A \tag{2}$$

$$B_{Sum} = B_1 + B_2 + B_3 + ... + B_n \tag{3}$$

Then, cloud server generates a new message by including the values A_{Sum}, B_{Sum} and the number of users n. We have to send the encrypted summation with number of users since EEES does not support homomorphism for division in order to compute the encrypted mean at the cloud server. After that, HMAC of the message is created and it is appended to the tail of the message along with the current TS as shown in Fig. 4. Finally, the complete frame is encrypted with 3DES key, K_{DES_CA} and forwarded to the analyzer.

After receiving the data frame with encrypted summation at the analyzer end, it segments the message accordingly by decrypting with the 3DES key, K_{DES_CA}. Then, the validity of the frame is verified through TS and HMAC as explained previously. After that, the encrypted summation (A_{Sum}, B_{Sum}) is decrypted with the analyzer's private key

Fig. 4. Structure of the Summation Data Frame sent from Cloud to Analyzer

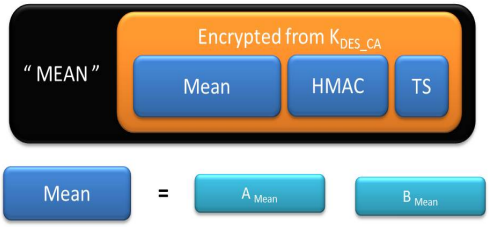

Fig. 5. Structure of the Message which carries Encrypted Mean Value to Cloud

(x) of EEES as given in Eqn. 4. Finally, the decrypted result (M_1) is divided by n to obtain the mean value of users' private sales data.

$$M_1 = B_{Sum}\ ((A_{Sum})^x)^{-1}\ mod\ p \tag{4}$$

$$Mean\ value\ of\ user\ inserted\ sales\ data = \frac{M_1}{n} \tag{5}$$

In order to compute the rest of the statistical parameters variance, standard deviation, skewness and kurtosis; it is necessary to obtain the mean value which is encrypted using EEES. Therefore, after recovering the mean value by the analyzer, it re-encrypts the mean value using El_para which gives us the encrypted components A_{Mean} and B_{Mean}. Furthermore, the analyzer generates a HMAC for the new encrypted mean and appends it to the encrypted mean. Finally, the current TS is also attached to the tail of the message and forwards it to the cloud server after encrypting the complete frame with the shared 3DES symmetric key, K_{DES_CA} as shown in Fig. 5.

At reception of the above message, cloud server will first decrypt it with K_{DES_CA} and acquire A_{Mean} and B_{Mean} components after validating TS and HMAC of the received data frame. In order to compute the statistical parameters variance, standard deviation, skewness and kurtosis it is necessary to obtain the deviations of the encrypted user sales data from the encrypted mean. These values can be calculated through the subtraction property which is derived from the additive homomorphism of EEES. If the encrypted mean deviation of i^{th} user's sales data is given by V_i;

$$V_i = (A_i, (B_i - B_{Mean})), \tag{6}$$

where (A_i, B_i) denotes the EEES encrypted sales data of i^{th} user. After that we need to obtain 2^{nd}, 3^{rd} and 4^{th} powers of the values V_i for all i. In order to achieve that, we use the property of homomorphism on powers of values possess by EEES [16]. Therefore, we can write the j^{th} power of V_i, (V_i^j) as;

$$V_i^j = (A_i^j, (B_i - B_{Mean})^j) \tag{7}$$

Thereafter, we can obtain $\sum_{i=1}^{n} V_i^2$, $\sum_{i=1}^{n} V_i^3$, $\sum_{i=1}^{n} V_i^4$ by using Eqn. 2 and Eqn. 3. Finally, all theses values are forwarded to the analyzer after attaching the HMAC, current TS and encrypting the complete message with K_{DES_CA}.

When the above mentioned data frame is received at the

analyzer, it will first decrypt the data frame with K_{DES_CA} and acquire the encrypted components $\sum_{i=1}^{n} V_i^2$, $\sum_{i=1}^{n} V_i^3$, $\sum_{i=1}^{n} V_i^4$ after validating the TS and HMAC which were appended to the received message. Then, the received encrypted components are decrypted using the relation given in Eqn. 4. Let us consider that decrypted components of $\sum_{i=1}^{n} V_i^2$, $\sum_{i=1}^{n} V_i^3$, $\sum_{i=1}^{n} V_i^4$ are denoted by V_1, S_1 and K_1 respectively. Then we can determine the variance, standard deviation, skewness and kurtosis of user inserted data from the equations given below.

$$Variance = \frac{V_1}{n} \tag{8}$$

$$Standard\ deviation\ (S.D.) = \sqrt{\frac{V_1}{n}} \tag{9}$$

$$Skewness = \frac{S_1}{(S.D.)^3} \tag{10}$$

$$Kurtosis = \frac{K_1}{(S.D.)^4} \tag{11}$$

After the completion of a computing session, analyzer sends a message to all the users through the proxy server to initiate the next computing session. Moreover, it is not necessary to carry out mutual authentication between entities again at the start of a new session. However, to ensure the freshness, we flush the existing keys after consecutive 10 consecutive sessions. Then, it is necessary to carry out the mutual authentication process again before starting a computing session.

V. Performance Analysis

An experimental setup has been formulated to analyze the performance of the proposed framework. Fig. 6 illustrates the setup along with the system configuration information of the PCs and the server. In order to implement this setup, we have compiled a program in Java where each entity is executing as a separate program. Values to be computed are inserted through interfaces of user programs while analyzer program will display the results at the conclusion of a computing session.

A. Definitions of Measured Parameters

- Encryption Time (T_e):Time taken to encrypt user inserted value at a user program using EEES.
- Entity Process Time (T_{EP}):Time duration that encrypted data are handled at proxy server, cloud server and analyzer until the conclusion of a computing session.
- Total Process Time (T_{TP}) :Total time duration that user data is handled at all the entities user, proxy server, cloud server and analyzer.
- Transmission Delay (T_{TD}): Collective time taken for transmission of system messages between entities between initiation and conclusion of a computational session.
- Total Time (T_T): Time duration between the initiation and the conclusion of computing session.

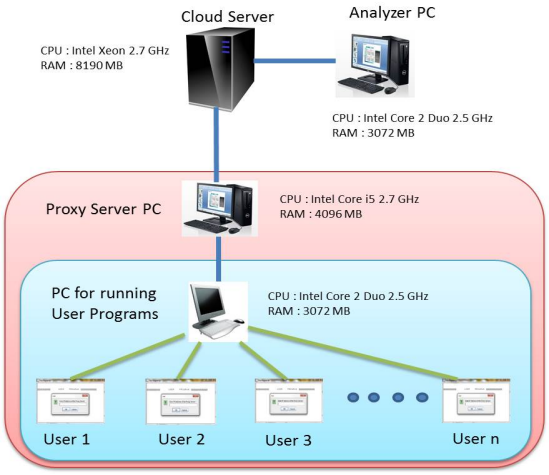

Fig. 6. Experimental Setup

Therefore, we can write;

$$T_{TP} = T_{EP} + T_e \times n \tag{12}$$

$$T_T = T_{TP} + T_{TD} \tag{13}$$

B. Test Results

In order to evaluate the performance of the implemented framework, we have to understand the variation of T_e, T_{EP}, T_{TP} and T_T as a function of number of users and size of the user input respectively. Tests carried out for T_e suggested that, it exhibits an exponential variation with the size of the prime numbers associated with EEES. This fact suggests that even though increasing the size of the prime values might strengthen the security, efficiency of the system will be degraded. Since we are considering sales values of users as the messages to be encrypted, prime size of 64 bits is quite adequate for our application. Hence, all the experiments were carried out with 64 bits as the prime size of the encryption scheme.

Fig. 7 shows the variation of T_e with the size of the user input. Even though the user input is varied from 4 bits to 64 bits, fluctuation of T_e is only limited to the range of 17 ms to 21 ms. Therefore, we can conclude that input data size does not influence T_e when the prime size is fixed. Furthermore, average T_e taken for a single user is approximately 20 ms.

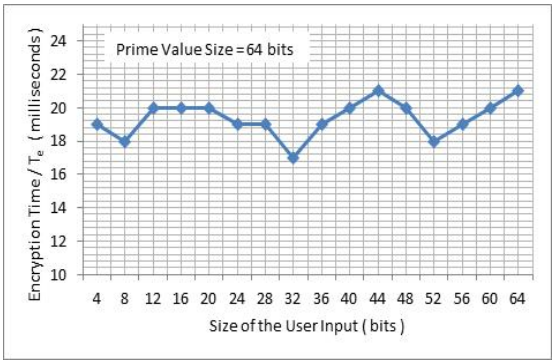

Fig. 7. Variation of Encryption Time as a function of User Input Size

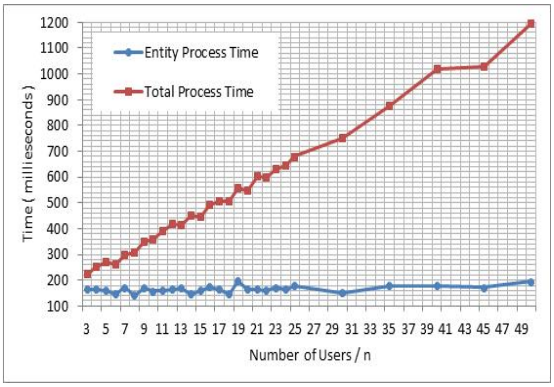

Fig. 8. Variation of T_{EP} & T_{TP} as a function of n

Fig. 8 illustrates the variation of T_{EP} and T_{TP} with n. We can clearly observe that T_{EP} is independent of n. However, T_{TP} exhibits a linear accumilation with n due to the fact that, total T_e is linearly increasing with n.

Fig. 9 depicts that both T_{EP} and T_{TP} are independent of the size of the user input. However, T_{TP} curve is elevated approximately by 200 ms than T_{EP}. This shift accounts for the total T_e of 10 users who participated in the computing session.

According to Eqn. 13, T_T is dependent upon T_{TD}. T_{TD} is affected by number of factors such as bandwidth, distance or hop count between the entities and system configuration of the entities. Fig. 10 shows the variation of T_{EP} and T_T when the number of users are varied from 5 to 50 at two instances where the hop count (HC) between proxy server and cloud server is 1 and 17 respectively. As we can observe form Fig. 10, T_{TP} values when HC is 1 and 17 varies in the same way. T_T at HC 1 exhibits a similar behavior since the associated T_{TD} is approximately 4 ms and does not impose any significant effect on T_T. However, T_T when HC = 17 is elevated by 900 ms from T_T at HC = 1. This elevation represents the increment in T_{TD} since the distance between proxy server and cloud server is increased. This fact proves that T_T is affected by T_{TD}.

The results acquired so far suggests that, both T_{EP} and T_{TP} are independent of the network parameters and only depend on the specifications of the PCs and server used in

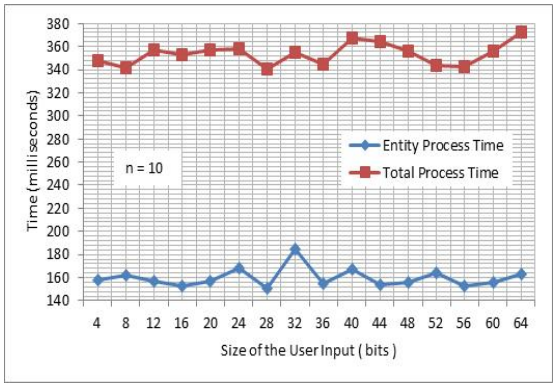

Fig. 9. Variation of T_{EP} & T_{TP} as a function of User Input Size

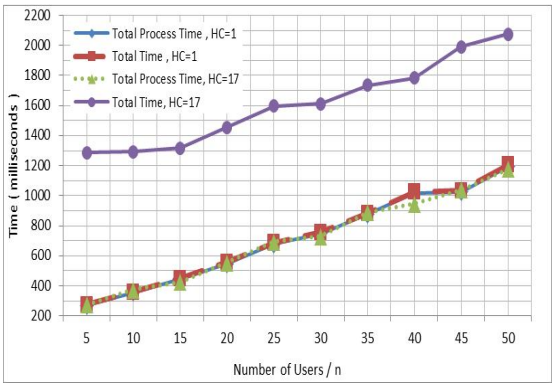

Fig. 10. Variation of T_{EP} & T_T as a function of n

the experimental setup. Moreover, average T_{EP} of the system is approximately 162.53 ms. Hence, we can derive;

$$Average\ T_{TP} = Average\ T_{EP} + (Average\ T_e \times n) \quad (14)$$

$$Average\ T_{TP} = (162.53 + 20 \times n)\ ms \quad (15)$$

The maximum T_T represented in Fig. 10 is slightly higher than 2s and T_{TP} is approximately 1.2s when HC is 17 and $n = 50$.

Under practical circumstances, the encryption process of user data is a parallel process. Hence, T_{TP} of Eqn. 12 could be decremented drastically. That would result the timing measurements of equations 12,13,14 and 15 to be reduced than the obtained results. Therefore, these experimental results verify that the proposed system is capable of operating efficiently under practical circumstances.

VI. Conclusions

In this paper we have proposed a secure multi-party based cloud computing framework for outsourcing statistical parameter computations on users' private data. The inclusion of proxy server in the framework ensures that cloud server will not have any idea about the ownership of each encrypted data component received at the cloud. Hence, user data anonymization is achieved. Furthermore, data privacy is also guaranteed due to the fact that user data are encrypted with EEES while computations are also carried out on the encrypted data. We have also used concepts of TS and HMAC to make the security framework withstand against replay attacks and possible integrity violations. The performance evaluation that we have presented in Sec. IV provides evidence for the efficiency of the proposed framework. Therefore, we can conclude that cloud environments can be successfully deployed to improve the efficiency of multi-party computations while enforcing the security requirements of user parties.

References

[1] N. Maheshwari, and K. Kiyawat, "Structural Framing of Protocol for Secure Multiparty Cloud Computation", in *Proceedings of* 5^{th} *Asia Modelling Symposium*, IEEE, Kuala Lampur, Malaysia, May 2011.

[2] A. C. Yao, "Protocols for Secure Computations", in *Proceedings of Annual Symposium on Foundations of Computer Science*, vol.0 , pp. 160-164, Nov. 1982.

[3] R. Oppliger, "Contemporary Cryptography", *Artech House Computer Security Library*, Norwood, 2005.
[4] F. Shaikh, S. Haider "Security Threats in Cloud Computing," in *Proceedings of International Conference for Internet Technology and Secured Transactions (ICITST)*, IEEE, Abu Dhabi, UAE, Dec. 2011.
[5] Q. Ma, L. Xiao, I.-L. Yen, M. Tu, and F. Bastani, "An Adaptive Multiparty Protocol for Secure Data Protection", in *Proceedings of* 11^{th} *International Conference on Parallel and Distributed Systems*, IEEE, Fukuoka, Japan, July 2005.
[6] S. Chakraborty, S. Sehgal, and A. Pal, "Privacy Preserving E-negotiation Protocols based on Secure Multi-party Computation", in *Proceedings of SoutheastCon.*, IEEE, Fort Lauderdale, USA, April 2005.
[7] S. Bleikertz, M. Schunter, C. W. Probst, D. Pendarakis, and K. Eriksson, "Security Audits of Multi-tier Virtual Infrastructures in Public Infrastructure Clouds", in *Proceedings of the Workshop on Cloud Computing Security Workshop (CCSW)*, ACM, New York, USA, Oct. 2010.
[8] S. Pearson, Y. Shen, and M. Mowbray, "A Privacy Manager for Cloud Computing", in *Proceedings of the* 1^{st} *International Conference on Cloud Computing*, Springer-Verlag, Berlin, Germany, 2009.
[9] M. Mowbray, and S. Pearson, "A Client-based Privacy Manager for Cloud Computing", in *Proceedings of the* 4^{th} *International ICST Conference on Communication System Software and Middleware*, ACM, New York, USA, Jun. 2009.
[10] D. Mishra, and M. Chandwani, "Anonymity Enabled Secure Multi-party Computation for Indian BPO", in *Proceedings of Region 10 Conference - TENCON*, IEEE, Taipei, Republic of China, Nov. 2007.
[11] M. Tebaa, S. El Hajji, and A. El Ghazi, "Homomorphic Encryption Method applied to Cloud Computing", in *Proceedings of National Days of Network Security and Systems (JNS2)*, IEEE, Marrakech, Morocco, Apr. 2012.
[12] A.-F. Chan, "Symmetric-key Homomorphic Encryption for Encrypted Data Processing", in *Proceedings of International Conference on Communications (ICC)*, IEEE, Dresden, Germany, Jun. 2009.
[13] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient Aggregation of Encrypted Data in Wireless Sensor Networks", in *Proceedings of the* 2^{nd} *Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous)*, IEEE, San Diego, USA, Jul. 2005.
[14] J. Wei, S. Guo, and Q. Xu, "Secure Homomorphic Aggregation Algorithm of Mixed Operations in Wireless Sensor Networks", in *Proceedings of International Conference on E-Business and Information System Security (EBISS)*,IEEE, Wuhan, China, May 2009.
[15] W. Luo, and X. Li, "A Study of Secure Multi-party Statistical Analysis", in *Processdings of International Conference on Computer Networks and Mobile Computing (ICCNMC)*, IEEE, Shanghai, China, Oct. 2003.
[16] G. Xiang, B. Yu, and P. Zhu, "Algorithm of Fully Homomorphic Encryption", in *Proceedings of* 9^{th} *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, IEEE, Sichuan, China, May 2012.