ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

Data Privacy using Multiparty Computation based on Homomorphic Encryption Schemes

V.Satish Kumar[1], MD.Fayaz[2]

Assistant Professor, Dept. of CSE, Dr. K.V.S.R Women's Engineering College, Kurnool, Andhra Pradesh, India[1,2]

ABSTRACT: Secure Multiparty Computation (SMC) permits a collection of users wants to decide some methods on their various inputs whereas keeping these inputs encrypted throughout the computation. In several situations, however, outsourcing these computations to untrusted server is fascinating, in order that the server will perform the computation instead of the users. But some of the solutions that are existing are inefficient, largely depends on user interaction, or a same key is required to encrypt the inputs —drawbacks creating the use in follow terribly restricted. For avoiding all these drawbacks we are proposing a construction: it is systematic, it doesn't need any user interaction whatever (except for knowledge up- and download), and it permits evaluating any dynamically chosen function on inputs encrypted under various public keys. Two non-colluding untrusted servers will together perform the computation by measure the cryptographic protocol. The cryptographic protocol mostly secure in the semi-honest model. We are exploring the interface of the result with two real-world situations with different domains: Privacy-Preserving Face Recognition and Private Smart Metering. Finally, we have a tendency to provide the performance analysis of our general construction to spotlight its practicability.

KEYWORDS: Secure Multiparty Computation, Semi-Honest Model, Outsourcing Computation, Homomorphic Encryption.

I. INTRODUCTION

Now-a-days central web servers play a key role, because most of the online communication being done by these web servers to process the huge amounts of private data. Social services, online auctions and cloud services are some of the examples come under the online services for utilizing this paradigm [2, 4, 9]. Regarding data privacy in these schemas, there are number of concerns are raised in the recent decade. The topic of Secure Multiparty Computation gets the importance to deal the privacy threats to delicate data.

Only the computation of the results was shown to the user even though there is an interaction between participating parties, the data is not shown. We used homomorphic encryption technique to find the solutions in this paper. Most of the encryption techniques support only restricted homeomorphisms, because Secure Multiparty Computation solutions based on interactions.

In order to perform more complicated actions, cryptography steps are required, that need parties holding a secret key, to be online. These collaborative natures of the protocols greatly obstruct promotion. Consider, for example, suppose there is an application scenario to aggregate the data to compute global sales statistics where a number of clients wanted to encrypt the individual input data and propel it to a server. In this case it is not possible to image that all the clients are online and able to ease the server to perform the computations. Same problem occurs for other frameworks as well, like privacy preserving smart metering: initial encryptions parties cannot perform further computations due to single meters have to encrypt the data and transmit to the server for further processing. From this we can also exhibit another key feature which are used in practical problems: every member in the group are able to use its real pair of public and private keys , instead of deciphering all the input data with the same public key for all SMC solutions.

Hence, to compute an encrypted data with distinguish public keys, a systematic SMC solution is required that collaborates with the clients as much as feasible [7]. Briefly we are examining the following scenario in this paper:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

1. A set of n inconsistent distrusting clients say P1,……..,Pn, every client having its own public and private key pairs, store the encryptions into the server C after encrypting the data with their individual public keys.
2. Let chose a function to compute the client's data over the server C, such that all the inputs and the results remain private.
3. The server C requires the capability to compute the functions without any connection to the clients, because clients are not always online. In specific, this also effects the client's extraction of results.
4. The server grasps nothing at all because clients can extract the result once they are in online.

In some respects, our model of secure computation resembles that of instance hiding [3]. In the instance hiding scenario, a computationally limited party A that holds input x uses a computationally unlimited party C in order to compute f(x), without C learning anything about x (except for its length, and the value of f(x)). Thus in both models, party C must not learn the input. However, in each model C is used for a different reason. In our secure computation model, parties A and B each have the power to compute f, but they need C because neither one of them holds the whole input. Our model is most interesting and has potential cryptographic applications if f is an easily computable function. In the instance hiding model, party A holds the input, and cannot compute f(x) because of lack of computational power. Hence [12] analyze their model in cases that f is not known to be computable in polynomial time. Their main result is that NP-hard problems, such as SAT, do not have instance hiding schemes unless the polynomial hierarchy collapses. The same result holds for our secure computation model, if we restrict A and B to random polynomial time computations.

The instance hiding scenario has been extended to the multiple-oracle scenario, with the surprising result that any function has an instance hiding scheme with polynomially many oracles [4]. The main open question regarding instance hiding is whether a constant number of oracles suffice. It is interesting to note that the secure computation model relates to this question. If in our model, all functions (within a given complexity class) can be securely computed with polynomially many shared random bits and polynomial message size (and no restriction on the computational power of A and B), then all functions have a 3-oracle 2-round instance hiding protocol. (The verifier splits its input into the sum of two random inputs. In the first round, one oracle gets to compute MA and another gets to compute MB. In the second round, the third oracle computes the function, based on the messages MA and MB.) Our model is also related to the problem of fault tolerant distributed computing [5, 8]. Most notably, our efficient protocol for computing any NLOGSPACE function can be used in -order to extend the result of [3], that any function in (algebraic) NC1 can be computed efficiently in a fault tolerant manner in a constant number of rounds. (Details are omitted from the current abstract.) There are several natural extensions to our model. In one of them, the input is distributed among k players, who want Carol to compute f for them. Our results extend to the multiparty case. Each party's inputs correspond to some subset of the variables in the polynomial representation. There is a designated party receiving output that learns only the output of the polynomial evaluation while all other parties receive no output. We assume a broadcast channel and that the private keys for the threshold encryption scheme distributed in a preprocessing stage.

II. RELATED WORK

Previous papers on SMC protocols were concerned with interactive solutions where all parties are actively involved in computing an arbitrary function on their respective inputs in a privacy-preserving manner [15]. Since we strive for a non-interactive solution, these constructions are not applicable in our scenario. To reduce computational costs at the clients' side, SMC has been considered in the client/server model (as we do) [16], but again with interaction of the clients during and/or after the computations. Furthermore, give a solution with minimal interaction of the clients, while relying on two non-colluding servers (as in our construction). Their solution, however, is mostly of theoretical interest both for efficiency reasons and because clients are bound to encrypt their private inputs under a single public key that is shared between the two servers (so clients do not have individual private keys).

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

III. MODELS FOR SECURE COMOUTATION

A. A SIMPLE PRIVACY-PROTECTED SIGNAL PROCESSING ALGORITHM

Let us start by introducing a simple yet representative signal processing algorithm to expound how cryptographic techniques are used to achieve privacy protection. The algorithm we use exemplifies two operations commonly encountered in signal processing, specifically 1) the weighted linear combination of input samples, as in linear filtering and signal transformations, and 2) the comparison of some processed result to a threshold, an operation reminiscent of signal quantization and classification. Two parties are involved in the signal processing operation: the party denoted by *A(lice)*, which owns the privacy-sensitive signal $x(i)$, say some recorded biometrics or medical signals; and the party *B(ob)*, which has the signal processing algorithm $f(.)$, say some access control algorithm or diagnosis algorithm. Alice is interested in the result $f(x(i))$, but does not wish to reveal $x(i)$ to Bob because of privacy concerns. Bob, on the other hand, cannot or does not wish to reveal (essential parameters of) his algorithm $f(.)$ to Alice for computational or commercial reasons. An example would be the intricate details of a commercially successful service such as search engines. It is roughly known which data is used in producing search results, but the exact function involved is not publicly known. This setup is typical of the examples mentioned in the section "Need for Privacy Protection in Signal Processing," and it will also play an important role when we discuss more elaborate privacy-protected signal processing operations in later sections. The toy example that we will use to convey the main ideas in privacy-protected signal processing is the following. Bob owns an algorithm that processes two signal samples $x(1)$ and $x(2)$ to obtain a binary classification result C

$$C = \{0 \text{ if } h1x(1) + h2x(2) < T, \quad 1 \text{ otherwise}\}$$

B. SECURITY MODEL

In this simple example, the value of the signal samples $x(1)$ and $x(2)$ are private to Alice and hence held secret from Bob. The linear weights $h1$, $h2$, and threshold T are private to Bob. The classification result $C! \{0, 1\}$ should be private to Alice. In other words, Bob must be unaware not only of the input signal $x(i)$, but also of the output result C.
At this point, we need to make two security model assumptions. The first is that Bob plays his role correctly and always executes $f(x(i))$ in a correct manner, without attempting to disrupt C. Bob's possible attacks concentrate on obtaining the input signal $x(i)$ or the intermediate results, such as the value of $h1x(1) + h2x(2)$, or C. Under this assumption, Bob is called an honest-but-curious or a semihonest party [13]. The second assumption is that Alice is not able to submit unlimited processing requests to Bob, because this would enable her to learn the values of $h1$, $h2$, and T by trial and error, which is commonly known as a sensitivity attack [14]. Sensitivity attacks are usually not treated in cryptography, and they are considered to be outside of the attacker model.

C. PUBLICLY VERIFIABLE COMPUTATION

The goal of verifiable computation (VC) is to provide means for a weak client to efficiently verify the correctness of the results returned for outsourced computation jobs (i.e. doing less work than the job itself). We extend this definition with the following two properties: public delegation and public verifiability, which are useful for many practical scenarios. Public delegation allows decoupling the party who provides the function to be evaluated and the party who has the input for the computation. Public verifiability enables anyone to verify the correctness of the returned results. To illustrate the importance of these two properties, we can consider the scenario where we outsource the evaluation of an analysis function for the lab tests of hospital patients. In this case public delegation enables the doctor to specify the function that will be evaluated and the lab assistant to provide the input for the computation. The public verifiability property makes it possible that both the doctor and the patients can verify the output of the analysis function on the test results. We show how to construct a verifiable computation scheme, which satisfies both of these properties, from attribute-based encryption. Our solution does not use expensive primitives such as fully homomorphic encryption or probabilistically checkable proofs (PCPs), which underlie existing VC solutions.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

D. ATTACKER MODEL

Proofs of security critically rely on assumptions about the capabilities and intentions of adversaries. First, in public key cryptography it is always assumed that the attacker has restricted computational power. An attacker is therefore not able to break the hard mathematical problem on which the used cryptosystem relies (see "Private Key and Public Key Cryptography"). We also assume that, when needed, the keys are generated and certified by a third trusted party (a certification authority) prior to execution of the protocols, and the public keys are available to all users in the system.

A second assumption concerns the intentions of adversaries. An important assumption in the section "A Simple Privacy-Protected Signal Processing Algorithm" was that Bob is a curious but- honest adversary participating in the computation. This attacker model describes Bob as a party that will follow all protocol steps correctly, but who is curious and collects all input, intermediate, and output data in an attempt to learn some information about Alice. A much more aggressive attacker model for Bob is the malicious adversary participating in the computation. In this case, Bob's intentions are to influence the computations such that Alice obtains a possibly incorrect answer. In the given example, Bob can easily influence the outcome by ignoring Alice's values "*x*(1), and "*x*(2),, and using some fictive input values *x*u (1) and *x*u (2). Bob encrypts these values using Alice's public key *PK*, and the processing proceeds as explained earlier. In fact, if the communication between Alice and Bob is not secured using traditional (private key) cryptographic techniques, even a malicious outsider adversary, not participating in the computation, might influence the result by actively capturing Alice's encrypted input and replacing it with some encrypted bogus signal values. Even though outsider adversaries are important in real-world applications, the focus encrypted signal processing is on the protection of privacy toward adversaries that participate in the computing process. Achieving security against malicious adversaries is a hard problem that has not yet been studied widely in the context of privacy-protected signal processing. There are two likely reasons for this.

First, it can be shown that any protocol that is secure against a curious-but-honest adversary can be transformed into one that is secure against malicious adversaries. The transformation requires proving the correctness of all intermediate computation steps using cryptographic techniques known as commitment schemes [15] and zero-knowledge proofs [14]. For instance, Alice could prove that she knows a certain value of an encrypted number without revealing that value itself. The drawback is that commitment schemes and zero-knowledge proofs are known to be notoriously computationally demanding and they increase significantly the number of interactive protocols between Alice and Bob. Loosely speaking, the protocol slowdown is in the order of a factor of ten [12], [13].

Second, the objective of privacy-protected signal processing is not to enforce correct computations on the service provider's side, as they already do that in nonprivacy-protected settings. Therefore, the malicious adversarial model might simply be an unrealistically aggressive scenario for many signal processing applications.

A third aspect of the attacker model describes whether and, if so, how parties involved in the computation might collude with each other. They could, for instance, exchange pieces of information that they collected to infer privacy-sensitive information. To illustrate collusion attacks, consider the slightly modified (and in the eyes of cryptographic experts, ridiculously simple) toy example where Alice has the private signal value *x*(1) and another party, called Charles, has the private signal value *x*(2). If Alice and Charles make use of the same public private key pair, then collusion between Bob and Charles will leak *x*(1) to Charles as he can decrypt the value *x*(1),. A seemingly obvious solution to make such collusion impossible is to have Alice and Bob use different public-private key pairs. Unfortunately, we can then no longer exploit the additive homomorphic property and (4) no longer holds.

The fourth and final aspect we address is the secrecy of the algorithm that Bob uses. As we mentioned in the section "A Simple Privacy-Protected Signal Processing Algorithm," Alice should be prohibited from repeatedly sending arbitrary input signals because she can infer critical parameters from the outputs of the algorithm using sensitivity attack [10]. For instance, by sending the input ("*x*(1), "*x*(2),) =("1,, "0,) Alice learns whether *h*1 T. Furthermore, the secrecy of Bob's algorithm can be guaranteed only if certain algorithmic parameters cannot be inferred directly from the input-output relation. Let us take the example where Bob carries out a convolution with a filter whose impulse response he wishes to keep secret. If Bob provides the filter output directly to Alice, he completely reveals his algorithm. After all, Alice simply sends Bob a signal with a delta impulse, and obtains the impulse response of the (supposedly secret) filter as output. Hence, although in some cases there might be a need for the secure evaluation of a secret algorithm, the algorithm's inherent properties might make secrecy a meaningless concept. Since sensitivity attacks are primarily related to algorithm properties, they are typically not considered part of the attacker model in cryptography.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

E. SECURE DATA SHARING WITH ENCRYPTED SEARCH

We explore solutions for secure search from a different perspective compared to traditional settings for MPC. We start with strict efficiency requirements motivated by speeds of available hardware and what is considered acceptable overhead from practical point of view, and we adopt relaxed definitions of privacy, which still provide meaningful security guarantees while allowing us to meet the efficiency requirements. We design security architecture and implement a system for data sharing based on encrypted search. Our protocol combines ideas of Bloom filters, a construction for a new private key deterministic encryption scheme and a new primitive called re-routable encryption. We evaluate the performance of our system and its practical usability with tests over tens of gigabytes of data, in which our implementation achieves only 30% overhead compared to the running time for SQL queries on the same database. The latencies for the document retrieval in our system are of the order of the time required for file transfer over SSH. Further, we propose a modification of the protocols that trades o↵ a relaxation of the privacy guarantees for the opportunity of another implementation optimization (bit slicing) that brings several orders of improvement in the search time.

IV. PERFORMANCE

We implemented and tested the simplex protocol using our Java libraries for secure computation. We measured the running time of the protocol for five processes (parties) running on different PCs (Intel Core Duo, 1.8 GHz) with full mesh interconnection topology. The experiments were carried out in an isolated network for two settings: Ethernet LAN with 100 Mbps links and WAN with 10 Mbps links in Fig. 1 and Fig. 2. The average round-trip time of the WAN was 40ms. The LAN experiments show the protocol performance for low network delay, while the WAN experiments show the effects of higher network delay and lower bandwidth.

Figure 3 shows the running time of an iteration for log(q) = 288 bits, k = 2f = 80 bits, and linear programs of several sizes: m = n = 25, m = n = 50, m = n = 100. To reduce the number of rounds, all the shared random bits needed by iteration (for comparisons and reciprocal) are generated in parallel by an initial precomputation phase. Moreover, the running time can be reduced by executing the precomputation in parallel with the previous iteration.

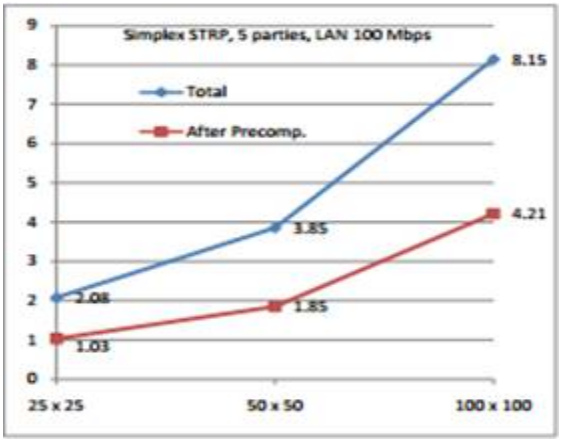

Fig. 1. The average round-trip time of LAN with 100 Mbps

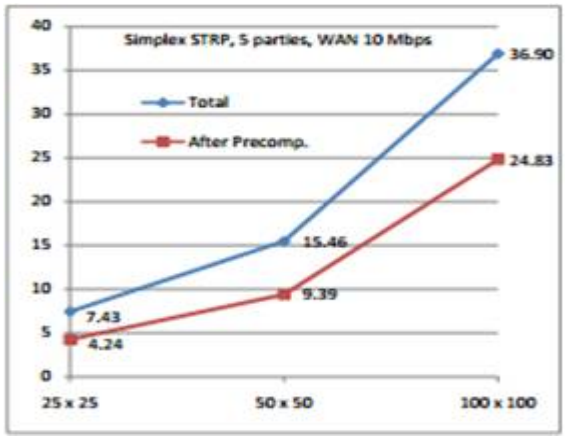

Fig. 2. The average round-trip time of WAN with 10 Mbps

	LAN			WAN		
	25 × 25	50 × 50	100 × 100	25 × 25	50 × 50	100 × 100
Precomputation	1.05	2.00	3.93	3.19	6.08	12.08
Select pivot column	0.22	0.37	0.65	0.67	1.44	2.52
Select pivot row	0.52	0.83	1.40	1.82	3.06	5.41
Update the tableau	0.29	0.65	2.17	1.75	4.88	16.90

Fig. 3. Running time (seconds) for secure simplex iterations

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

A simplex protocol for this variant of the algorithm can easily be obtained by adapting the protocols. The main difference is the update of the tableau using secure integer arithmetic instead of fixed-point arithmetic. This protocol is more efficient than the variant in [16] (e.g., 2m + n comparisons instead of 3m+n and 2mn multiplications for the update of the tableau instead of 3m(n + m)). However, they are both affected by the growth of the values in the tableau, which can reach thousands of bits for linear programs with tens of variables and constraints [15]. The experiments showed a large increase of the running time for pivot selection and precomputation (comparisons) and for the update of the tableau (large shares) even for log(q) = 1024 bits.

The tests show that our approach offers an important performance gain and suitable accuracy for secure linear programming. The main performance bottleneck is the secure comparison. Our comparison protocol [12] provides statistical privacy and performs most of the computation in a small field, hence with low overhead.

By encoding binary values in small fields and using efficient share conversions, the amount of data exchanged is reduced from O(k 2) bits (when integer and binary values are encoded in the same field) to O(k). The simplex algorithm used by the protocol needs only 2m + n comparisons (instead of 3m+n when using the large tableau to select the pivot) and fixed-point arithmetic can reduce their input bit-length by a factor of 10 with respect to integer pivoting. Nevertheless, most of the precomputation time and pivot selection time shown in Fig. 3 is due to comparisons. Further performance improvement would require secure comparison with sublinear complexity; currently, in the multiparty setting, this can be achieved only by trading off privacy for efficiency.

V. CONCLUSION AND FUTURE WORK

Due to existence of non-colluding untrusted servers, we presented a systematic computation method for which it doesn't requires any user interaction and it allows the implementation of arbitrary functions on inputs that are encrypted under different public keys. We have experimental results by showing our protocol to be secure in the semi-honest model and highlighted its practicability. Two application scenarios, one on privacy-preserving face recognition and one on private smart metering underlined the applicability of our construction to the real-world. We consider the extension to the malicious adversarial model as future work. Furthermore, we are planning to experiment with certain functionalities used in other application scenario, e.g., computations on medical health records. To this end, it seems useful to integrate our work into tools for automating SMC.

REFERENCES

1. Andreas Peter, Erik Tews, and Stefan Katzenbeisser, —"Efficiently Outsourcing Multiparty Computation under Multiple Keys", IEEE Transactions on Information Forensics and Security Vol: 8, No: 12 Year 2013.
2. Musa J.D "Meassurement and Management of Software Reliability" proceedings of the IEEE, VOL. 68, NO. 9, September 1980.
3. Smrithy V , Rekha, V.Adinarayanan, Anurag Maherchandani, Sneha Aswani, "Bridging the Computer Science Skill Gap with Free and Open Source Software" International Conference on Engineering Education (ICEED 2009),Kuala Lumpur, Malaysia, IEEE, VOL 56, NO.7, 2009.
4. Li M. and C.Smidts,"A Ranking of Software Engineering Measures based on Expert Opinion," IEEE Transactions on Software Engineering, vol. 29, pp, 811-24, 2003.
5. Goel A.L. and K.Okumoto, "Time-Dependent Error-Detection Rate Model for software and other performance measures," IEEE Trans. Reliability, vol. 28, pp. 206-211, 1979.
6. Yamada S. and S.Osaki, "Software Reliability Growth Modeling: Models and Applications," IEEE Trans. Software Eng., vol. 11, pp.1, 431- 1, 437, 1985.
7. A.C.Yao, "protocol for secure computations," in proceedings of the 23rd annual IEEE symposium on foundation of computer science, pp. 160-164, Nov.1982.
8. R. Sheikh, B. Kumar and D. K. Mishra, "Changing Neighbors k- Secure Sum Protocol for Secure Multi-party Computation," Accepted for publication in the International Journal of Computer Science and Information Security, USA, Vol.7 No.1, pp. 239-243, Jan. 2010.
9. C. Clifton, M. Kantarcioglu, J.Vaidya, X. Lin, and M. Y. Zhu, "Tools for Privacy-Preserving Distributed Data Mining," J. SIGKDD Explorations, Newsletter, vol.4, no.2, ACM Press, pages 28-34, Dec. 2002.
10. D. K. Mishra, M. Chandwani, "Extended Protocol for Secure Multiparty Computation using Ambiguous Identity," WSEAS Transaction on Computer Research, vol. 2, issue 2, Feb. 2007.
11. Yehuda Lindell and Benny Pinkas, "Secure Multipart Computation for privacy-preserving data mining". The Journal of Privacy and Confidentiality, Vol. 1, pp. 59-98, 2009.
12. Gayatri Nayak and Swagatika Devi, "A survey on privacy preserving data mining: approaches and techniques". International Journal of Engineering Science and Technology (IJEST), Vol. 3, march 2011.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 5, May 2017

13. Chandwani M., Mishra D.K., "A Research review on secure multi-party computation," Invited paper in Samadhan - the Solution – A Journal of Perspective Managers," Vol. 10, pp. 3-20, Jul-Dec 2007.
14. Jajodia S., "Database security and privacy," ACM computing surveys (CUSR), Vol. 28 No. 1, pp. 129-131, Mar 1996.
15. Ayswarya R Kurup, Simi Lukose (2014), "Security Enhanced Privacy Preserving Data sharing With Random ID Generation", IJSRE Volume 2 Issue 8, 2014.
16. Larry A. Dunning, And Ray Kresman (2013), "Privacy Preserving Data Sharing With Anonymous ID Assignment", IEEE Transactions on Information Forensics and Security, Vol. 8, NO. 2,2013.