Secure Aggregation in Large Scale Overlay Networks

Waseem Ahmad, Ashfaq Khokhar
Department of Electrical and Computer Engineering
University Of Illinois at Chicago
Chicago, IL 60607
Email: {wahmad,ashfaq}@ece.uic.edu

Abstract

Overlay networks have been very useful in solving large scale data dissemination problems. In this paper, we consider the case of data gathering which is the inverse of dissemination problem. In particular, we focus on a scenario where an organization or a constellation of organizations is interested in gathering data from large number of nodes spread across the administrative boundaries. Providing individual nodes with full assurance that the privacy of their data won't be compromised is a critical problem in achieving the true benefits of this collaborative process. We provide a novel solution to the problem by employing a threshold homomorphic cryptosystem which allows processing of encrypted data without revealing anything about the underlying private data. The threshold property of the cryptosystem ensures that no single node is able to decrypt the aggregate results. The proposed solution provides excellent scale-up properties while preserving privacy and secrecy of the data even among malicious adversaries.

I. Introduction

Overlay networks have proven very useful in solving large scale dissemination problem as well as in creating dynamic collaborative communities. Data gathering, which may be in the form of aggregation, resemble very closely to the dissemination problem albeit in opposite direction. Whereas in dissemination, an organization or a constellation of organizations may be interested in distributing digital media to an extremely large number of subscribers, in data gathering these organizations are interested in getting some data or a function of it thereof, from a large number of participants. These sorts of scenarios occur frequently in disaster response organizations. Take an example of Center for Disease Control (CDC) which might be interested in getting the statistics about the patients suffering from a specific disease from the hospitals throughout the nation. Hospitals, however, might be reluctant to share this information as it might disclose their mortality rate which might create a bad public perception about them. Moreover, since in most of the overlay networks, peer-to-peer communication model is employed, privacy and secrecy of the individual nodes' data become even more important as the data might reach in the hands of their competitors. In the wake of these arguments, it is essential that a middleware be developed which provides a trusted environment for these individual organizations to confidently share their data for achieving a common useful goal. The data aggregation protocol of this middleware should have following design goals

- *Privacy Preserving:* The aggregation protocol should preserve privacy of individual nodes by means of provably secure cryptographic primitives.
- *Malicious Adversarial Model:* The adversarial model assumed by the secure aggregation protocol should treat each node as a potentially malicious adversary who might want to break into the privacy of other nodes by doing anything he could.
- *Scalable :* The aggregation protocol should scale well with increasing number of participants.
- *Fault Tolerant :* The protocol should be able to handle faults gracefully. More importantly there should be no single point of failure.

TRIUMF[1] is a trusted reliable middleware for fault tolerant applications whose secure aggregation protocol is aimed at meeting above mentioned goals. This paper focuses on the secure aggregation protocol only (for details on TRIUMF architecture see [1]). Following are the key features of the proposed aggregation protocol which enable it to meet the above requirements.

- *Additively Homomorphic Cryptosystem :* The proposed secure aggregation protocol employs an Additively Homomorphic Cryptosystem which is a variant of Damgard and Jurik's generalization of Paillier's cryptosystem [12]. The additive homomorphic property allows combining two cipher-texts such that underlying plain texts are added together. This would allow processing of the encrypted data without revealing anything about the plain-text. This essentially preserves the privacy of individual nodes.
- *"Threshold" Property of Homomorphic Cryptosystem :* One novel feature of the proposed aggregation protocol is the threshold nature of the key generation phase of the Additively Homomorphic Cryptosystem mentioned above. A threshold cryptosystem allows private key to be shared among w servers such that t of them can reconstruct it. Here t is referred to as the threshold value of the cryptosystem and it is usually $\geq \frac{w}{2} + 1$. This means that we can distribute trust among w servers instead of trusting only a single server to do the decryption. This eliminates the possibility of single point of failure and provides un-interrupted decryption as long as number of malicious servers are $\leq t - 1$.

- *Scalable Communication Primitives:* This paper focuses on a hierarchical aggregation framework which suits to the nature of the problem we are addressing. Ideally, gossip based communication protocols are considered extremely scalable for large scale networks[2, 14]. But there are two problems associated with employing gossip based protocol in our aggregation framework which are as follows
  - Gossip based protocols are useful when all the nodes require the end result of computation. This might not be the case in the applications that we are considering. For example, in the CDC example, individual hospitals need not know the total number of patients suffering from a specific disease across the nation.
  - Even in the wake of above argument, it would have been worthwhile to implement a gossip based communication protocol for varying nature of applications. But the problem that we face is that the gossip basically works on the basic principle of variance reduction. This means that each two nodes will modify their local values such that the difference between their values is essentially zero. For example, if we want to take sum over the values of all nodes, then every two nodes who are involved in the gossip, will take average of their values and replace their local values with this average. At the end, a stable average value is multiplied by total number of nodes to get the eventual sum. Now problem with variance reduction is that it requires more than just addition of plain texts. For example, in case of average it requires an addition and a division operation. This would require our cryptosystem to be both additive and divisive/multiplicative homomorphic. There has been some proposals on homomorphic protocols which are both divisive/multiplicative and additive [8] but they are known to have severe security limitations and have been broken with relative ease [3]. In the absence of any variance reduction mechanism, we will have to make sure that a value from a node is added only once in the entire aggregate. For this to happen, we will have to enforce a topology on the network and hierarchical topology is known to have good scalability properties.

  The hierarchical communication framework that we are employing is natural to the kind of applications that we are envisioning. For example, CDC national head quarter can initiate a request to build an overlay network comprising all the hospitals in the country. The first step for that would be to contact its regional head quarters to start contacting all the hospitals in their own areas. In this way a hierarchical network comes into life where entire network is divided into clusters of geographically proximate nodes representing hospitals. Each cluster has a master node being represented by regional head quarter [1]. The communication protocol starts by having individual nodes sent their values to the local master node. Since all nodes may not be connected to the master node directly, a multi-hop connection scheme just like in any overlay network is employed. The multi-hop scheme works such that each intermediate node combines the encrypted value that it holds with the encrypted value that it receives (note that this combining of the cipher-text amounts to the additions of the plain-texts in the additively homomorphic cryptosystem that we employ). After this combining operation, the nodes forwards the resulting cipher-text to the next node in hierarchy.

The work in this domain has suffered from following shortcomings(details in Section VII)

- Most of the work done in secure aggregation domain assume a semi-honest (honest but curious model) which is unpractical in large scale distributed computing.
- No algorithm, to the best of our knowledge, has been able to guarantee full privacy of individual nodes while still not suffering from single point of failure problem.

As mentioned above, our protocol solves the above mentioned problem. Details will follow in the coming sections.

Organization: The rest of the paper is organized as follows. Section II describes the working of the proposed secure aggregation framework as well as outlines the assumptions behind the construction of the protocol. Section III describes the theoretical details of the proposed modification of Jurik et al's Threshold Additively Homomorphic Cryptosystem[13]. Section IV defines the spanning tree construction procedure for our hierarchical aggregation framework. Section V presents a complexity analysis of the proposed protocol. Section VI describes the simulation results which evaluate the protocol. Section VII outlines the related work. Section VIII draws conclusions and presents future line of our research action.

II. Proposed Protocol for secure Aggregation

Before we delve deeply into the working of the proposed protocol, we will enumerate the set of assumptions that are made. These assumptions are as follows.

1) *Network Model.* We assume that our network is composed of a set of N nodes $P_1, P_2, ..., P_N$. These nodes are connected through insecure point-to-point channels and are modeled as polynomial time randomized Turing machines [5]. Moreover, these N nodes are divided into clusters of size $n = \frac{N}{w}$ each. Nodes $S_1, S_2, ..., S_w$ are the masters of clusters $C_1, C_2, ..., C_w$. Master nodes also are connected through insecure point-to-point channels but have also access to dedicated broadcast channel.
2) *Adversarial Model :* We assume that the adversary can corrupt up to $\leq \frac{N}{2}$ nodes and corrupted nodes can collude as well. This means that it is possible that the adversary can corrupt up to $t-1$ master nodes. The computational power of

[1]Note that geographical clustering is only one way to partition the nodes. Other types of clustering are also possible based on the application requirements (this is why we are terming it as an 'overlay network')

the adversary is modeled by a probabilistic polynomial time Turing machine. Moreover, the adversary is assumed to be static i.e. it chooses the corrupted players at the beginning of the protocol. There are pro-active secret sharing schemes [18] which can cater for the dynamic adversaries. Our protocol, at this time, is not designed for dynamic adversaries but it is straightforward to adopt a proactive secret sharing scheme in our protocol.

3) *Generation of secret Shares :* Our secure aggregation protocol employs a fully distributed key reconstruction and subsequent decryption without any trusted dealer. There is however a one time key generation process which is done off-line before the network deployment. This key generation is under taken by the authority wishing to get the eventual aggregate result and is a one time process only. This authority generates a public/private key pair. It then distributes the public key to all nodes at the time of deployment while the private key is shared among the master nodes using Shamir's Secret Sharing scheme [15]. The same assumption is held by the threshold cryptosystems proposed in [9,5].
4) *Communication and Computation Rounds :* Although the proposed protocol is totally asynchronous. For ease of analysis, we divide the protocol into synchronized communication and computation rounds (the same assumption is held by aggregation protocols proposed in [2,14]). One communication round of the proposed framework is a pull-push protocol whereby each node pulls encrypted value from its child and pushes the combined result on to its parent. One computation round corresponds to a call to either of the encryption or combining operations of the cryptosystem. Note that decryption is only performed at the end of protocol execution and it only involves the master nodes.

A. Working of the Proposed Protocol

The proposed framework works as follows

- The entire network is decomposed into clusters of nodes based on application specific proximity measure.
- Each cluster is assigned a master node. The master can be selected based on some leader election algorithm but in most cases application provider will be able to pre-define such master nodes.
- Using additively homomorphic cryptosystem, a Public Private Key pair is generated. Using secret sharing scheme of [15], the private key is divided among the master nodes of the clusters.
- Each cluster's master node initiates a spanning tree building process (a variant of self-stabilizing distributed spanning tree construction algorithm proposed in [6]). Similar spanning tree is induced on the master nodes of individual clusters such that each master node is the member of that spanning tree.
- In a recursive manner, each node performs a communication and a computation round such that it calls the combine operation of the cryptosystem on the encrypted local value and the encrypted value of its child. The communication round starts by feeding the value pulled from the child node to the computation round and ends by pushing the result of the computation round on to the parent node. Note that by the end of the computation round, the local value of each node is set to the combined value. After first communication round, Each node will indulge in the subsequent communication and computation rounds only if there is a change in its local value.
- After one complete tree traversal, the master node will have the aggregate value over the set of nodes which were present at the time of construction of spanning tree. The self stabilizing distributed spanning tree construction protocol of [6] ensures that communication goes smooth even in the case of node failures. The new nodes who were not present at the time of the construction of the tree might have to wait till they are included in the tree.

III. Threshold Additively Homomorphic Cryptosystem

The cryptosystem employed by our secure aggregation protocol is slightly modified version of the one proposed by Jurik et al [13] for efficiency and practicality reasons. Basically, we differ from their secret sharing scheme in that our protocol employs secret generation process which works modulo a prime while their cryptosystem does it module a composite number which runs into problems as mentioned later. The novelty of our scheme is to maintain the underlying security and homomorphic properties of their cryptosystem. The modified cryptosystem works as follows.

Key Generation: Key Generation starts by finding two large primes p and q that satisfy $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are primes and different from p and q. The length of these primes is the security parameter given to the key generation process. Also we set the RSA modulus as $n = pq$. Like in [13], we decide on $s > 0$ such that the plain-text space will be $\mathbb{Z}_{n^s}$. Another prime l is picked which would serve as a modulus for our secret sharing scheme such that $n^{s+1} < l$. Also we set $g = n + 1$. Let λ be the least common multiple of $p - 1$ and $q - 1$. By the chinese remainder theorem, choose d such that $d \bmod n \in \mathbb{Z}^*_{n^s}$ and $d = 0 \bmod \lambda$. We use Shamir's secret sharing scheme [15] to generate the private key shares to be divided among w servers. For this we make the polynomial $f(X) = \sum_{i=0}^{t} a_i X^i \bmod l$ by picking a_i (for $0 < i \leq t$) as random values from $\{0, ..., l\}$ and $a_0 = d$. The secret share of the i'th server will be $s_i = f(i)$ for $1 \leq i \leq w$. A verification key v_i is associated with each server i such that $v_i = g^{s_i} \bmod n$. The public key is (n, s, l) and $\{s_1, s_2, ..., s_w\}$ is the set of private key shares of master nodes $\{1, 2, ..., w\}$.

Encryption: The plaintext $m \in \mathbb{Z}_{n^s}$. Given the plaintext m, choose a random $r \in \mathbb{Z}^*_n$ and let the ciphertext be $E_{s,pk}(m, r) = c = g^m r^{n^s} \bmod n^{s+1}$.

Share Decryption : The i'th server will compute $c_i = c^{2\Delta s_i}$, where c is the ciphertext and broadcasts it to all the other master

nodes/servers. Along with this will be a zero knowledge proof that $\log_g(v_i) = \log_{c^{4\Delta}}(c_i^2)$ which will convince the other servers that it has indeed raised c to his secret exponent s_i.

Share Combining: If each server has number of verified shares $\geq t+1$, then it can combine them into the result as follows. Suppose we define $b_j = \Delta \prod_{1 \leq k \leq t, k \neq j} \frac{k}{k-j} \mod l$ Then we have $m' = \prod_{j=1}^{t} c_j^{2b_j} \mod n^{s+1}$. The value of m' will have the form $m' = c^{4\Delta^2 \sum_{j=1}^{t}(b_j s_{i_j})}$ Since $d = \sum_{j=1}^{t}(b_j s_{i_j}) \Rightarrow m' = c^{4\Delta^2 d}$ or $m' = (n+1)^{4\Delta^2 m} \mod n^{s+1}$ Here m is the desired plaintext. We apply the algorithm of [13] on m' and multiply the result by $(4\Delta^2)^{-1}$ to find the value of m.

Homomorphic Property of the Cryptosystem: To show that the cryptosystem is additively homomorphic, consider two messages $m1$ and $m2$ which are encrypted using the same public key pk such that $c_1 = E_{(s,pk)}(m1, r1)$ and $c_2 = E_{(s,pk)}(m2, r2)$ then $c_1 c_2 = g^{m_1} g^{m_2} r_1^{n^s} r_2^{n^s} = g^{m_1+m_2} r^{n^s}$ where $r = (r_1 r_2) \in \mathbb{Z}_n^*$ so $c_1 c_2 = E_{(s,pk)}(m1 + m2, r)$

IV. Spanning Tree Construction

As mentioned above, our protocol depends upon a spanning tree construction algorithm proposed in [6]. The algorithm assume unique identifiers given to each node. The node with the minimum identifier eventually plays the role of the root in the spanning tree. In this algorithm, nodes which are part of some spanning tree expect to receive "power" from the root of the tree where "power" refers to continuous flow of certain messages, one per round. The basic idea is that only legal roots may be the source of the power and the fake roots are forced out to make a new tree. Whenever a node receives power from a node with smaller identifier (than the one it is currently attached to), it attaches itself to that node's tree. In an asynchronous network, such as ours, the power supply idea is implemented using different types of messages. Nodes use periodic exchange of "weak messages" to synchronize their state while "strong" messages are used to carry power. This algorithm is known to stabilize in $O(n)$ rounds without any knowledge of n [6].

V. Complexity Analysis

As mentioned above, for ease of analysis, we divide the protocol execution into well defined computation and communication rounds. The computation cost is due to cryptographic operations. The main component of the encryption is fast exponentiation while that of combine operation is multiplication of large integers. The complexity of both the operations is roughly the same. The communication cost is taken to be the number of messaging rounds required. During each round there can be multiple messages but they will be on different channels.

We assume that there are total w servers and N general nodes in the network and t is the threshold value of the cryptosystem. The network is partitioned into clusters such that one server node falls in each cluster and each cluster has $n = \frac{N}{w}$ nodes. Now the communication cost can be broken into the cost of setting up the spanning tree $(O(n))$, combining cipher-text values over the tree $O(n)$, validating the share decryptions for all the w servers and combining the valid shares $(O(wt))$. On the other hand, the cost of cryptographic operations is the cost of assigning a node to relevant cluster and subsequent encryption of local value $(O(1))$, cost of combining values over each cluster tree $(O(n))$ and the cost of verification of share decryption and subsequent combining $(O(wt))$ and the decryption of the eventual aggregate $(O(1))$. So the overall cost function can be described as $C_{overall} = C_{comm} + C_{crypto}$ where $C_{comm} = w + 2n + wt$ and $C_{crypto} = n + wt$ so we could write the cost function as

$$C_{overall} = (w + 2n + wt)_{comm} + (n + wt)_{crypto} \tag{1}$$

A. Finding out the Optimal Number of Master Nodes for a given value of N

Our theoretical as well as experimental results reveal that the by having more servers (with private key shares) beyond certain value deteriorates the performance. Below is the mathematical analysis to find out the optimal number of servers given the total number of nodes in the network.

From Eq., solving for the communication cost $C_{comm} = w + 2n + wt$ so $\Rightarrow C_{comm} = w(1 + \frac{w}{2} + 1) + \frac{2N}{w} \quad since \quad t = \frac{w}{2} + 1, n = \frac{N}{w}$ $C_{comm} = (\frac{w^2}{2} + 2w + \frac{2N}{w})$ Now lets take the derivative to find the optimal value $\frac{dC_{comm}}{dw} = w + 2 - \frac{2N}{w^2} = 0$ $\Rightarrow w^3 + 2w^2 - 2N = 0$ Solving the above cubic equation yields $w = \left\lceil \sqrt[3]{\frac{-20 \pm \sqrt{291344}}{54}} + 2N - \frac{16}{27} - \sqrt[3]{\frac{-20 \pm \sqrt{291344}}{54}} - \frac{2}{3} \right\rceil$

Similarly for C_{crypto} we have $C_{crypto} = \frac{N}{w} + w(\frac{w}{2} + 1)$ and for optimal value $\frac{dC_{crypto}}{dw} = w^3 + w^2 - N = 0$. From our experimental results, we will see that communication and encryption costs are directly proportional to each other. Therefore, we will simply select the value of w which incurs minimum communication cost (which will also correspond to the minimum encryption cost)

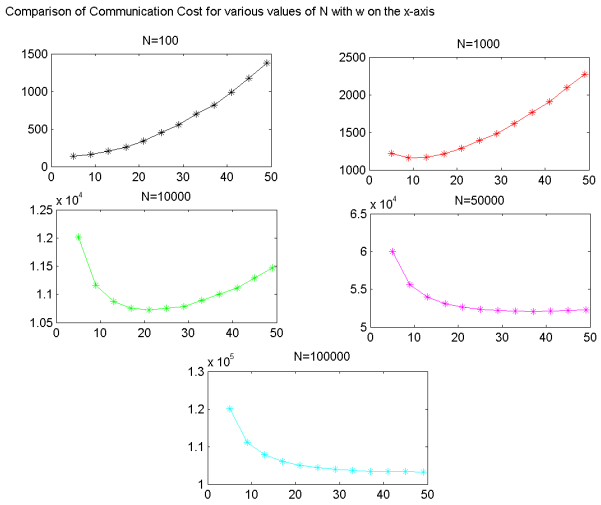

Fig. 1. Comparison of Communication Cost for various values of N with w on the x-axis

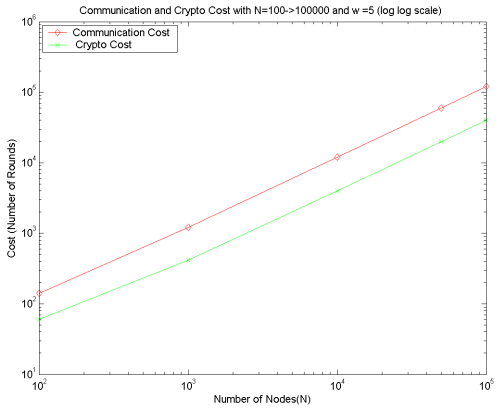

Fig. 2. Communication and Crypto cost with N varying from 100 to 100000 and w = 5

VI. Evaluation

The cryptosystem is implemented in Java and the simulations were performed in a Simulator adopted from PeerSim [2] to handle BigInteger values (plaintexts and ciphertexts are represented as BigInteger in our system). The simulations were carried out with values of N from the set $\{100, 1000, 10000, 50000, 100000\}$ while the values for w were selected from the set $\{1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49\}$. Our first set of experiments were aimed at measuring the communication cost (number of communication rounds) for a fixed value of N while varying w. The results of this experiment shown in Fig.1 clearly indicate that the communication cost is the least for an optimal value of w for each value of N. This optimal value confirms our theoretical analysis in the last section. We also note that benefits of having more servers become more evident when the value of N becomes sufficiently large. We should also clarify that the true benefit of the threshold cryptography is in distributing the trust among a number of servers instead relying fully on a single server. This makes our protocol resilient to benign and malicious failures. Here benign failures refer to system crashes because of internal (system) errors or network failures while malicious failures refer to the crashes due to adversarial interference from outside. Similarly our second set of experiments attempt to measure the cryptographic operations cost. Due to space limitation, we have not included the graphs for cryptographic operations cost but the curves for cryptographic operations cost is identical to the communication cost curves. As indicated in our theoretical analysis, we conclude that by reducing one of cryptographic cost or the communication cost of the other. This is justified by our theoretical analysis as well, because both of these costs are directly proportional to a linear combination of n and wt. Fig.2 plots the communication and crypto cost for $w = 5$ case such that N varies from 100 to 100000. The graph is shown in $loglog$ scale. The graph shows that crypto and communication costs change linearly with increasing values of N indicating the scalable nature of the protocol.

VII. Related Work

Threshold Homomorphic Cryptosystems: Jurik et al [13] proposed a generalization of Paillier's Cryptosystem[12] which is supposed to be threshold in nature and has homomorphic properties. But their cryptosystem runs into problems as it tries to generate the the private key shares modulo a composite number. Share generation and reconstruction modulo a composite number is considered impractical as it fails to insure a unique reconstruction of the original key. Our scheme on the other hand adopts a systematic way to introduce shared key generation modulo a prime such that both security and homomorphic properties of the underlying cryptosystem remain in tact. There are plenty of scheme for secret sharing especially in the discrete log problem domain [5, 16, 17]. Pedersen's verifiable secret sharing scheme has been used by many [17]. Gennaro et al discussed some of the shortcomings of Pedersen's scheme [5] and other protocols based on that scheme. They went on to propose their own scheme [5] which is free of attacks possible on Pedersen's scheme.

Secure Aggregation: There has already been some work done in secure aggregation domain. Bobi et al [7] first used additively homomorphic cryptosystem of Paillier to propose a secure sum protocol for their privacy preserving association rule mining algorithm. Their solution however relied on $k - privacy$ assumption i.e. their solution allows a node to have the aggregate result of at least $k-$nodes where $k << n$. Our protocol on the other hand does not suffer from such limitation. Moreover, their scheme is not based on 'threshold cryptography' that would leave it susceptible to single point of failure problem. Another distinguishing factor is the adversarial model employed. Their first protocol assumed an honest but curious adversarial model where an adversary would try to break into the privacy but would follow the protocol honestly. They later engineered a solution to the malicious adversary scenario as well but the solution lacked the mathematical justification as is provided by the zero knowledge proofs employed by our scheme. Clifton et al [11] provided a secure sum solution for their privacy preserving data mining protocol. Their scheme works as follows. All the nodes are connected in a ring. A master node is selected which adds a random value to its local value and transmits it to the next node in the ring. This process goes on till he aggregated value reaches back to the master node who then removes the random value and pushes the eventual sum to the next node in the ring. They assume semi-honest adversary model which is un-realistic in many distributed computing scenarios. Moreover, randomization based privacy protection is not satisfactory in many cases as it reveals the plane of the underlying data.

VIII. Conclusions and Future Research

We have developed a secure aggregation protocol which employs a threshold additively homomorphic cryptosystem and is designed to take care of malicious adversarial scenarios. The threshold homomorphic cryptosystem is a modification of the one proposed by Jurik et al [13] in order to make the secret sharing process more efficient and practical. We have used a hierarchical aggregation framework as this is natural to the kind of applications we aim to address. We however, are looking into add a divisive/multiplicative homomorphic element to our cryptosystem so that we are able to perform variance reduction process required for gossip base protocols. We have fully implemented the cryptosystem and simulated it on a large scale network to evaluate its performance. We have also given mathematical foundation to calculating the optimal number of key shares given the total number of nodes N in the network. To the best of our knowledge, this is the first completely secure, practical and efficient realization of a threshold homomorphic cryptosystem in a large scale network.

References

[1] Waseem Ahmad, Ashfaq Khokhar: *TRIUMF:A Trusted Middleware For Fault Tolerant Collaborations*. University of Illinois at Chicago Tech Report TR-MSL0786. URL : `http://www2.uic.edu/~wahmad1/papers/triumf.pdf`

[2] Mrk Jelasity, Alberto Montresor, and Ozalp Babaoglu: *Gossip-based aggregation in large dynamic networks*. ACM Transactions on Computer Systems, 23(3):219252, August 2005

[3] Jung Hee Cheon and Hyun Soo Nam : *A Cryptanalysis of the Original Domingo-Ferrer's Algebraic Privacy Homomophism*, Cryptology ePrint Archive, Report 2003/221, 2003, URL: `http://eprint.iacr.org/`,

[4] Y. Desmedt: *Some Recent Research Aspects of Threshold Cryptography*. In E. Okamoto, G. Davida and M. Mambo, editors, Information Security, Proceedings (Lecture Notes in Computer Science 1396), pp. 158-173. Springer-Verlag, September 1997, Tatsunokuchi, Ishikawa, Japan.

[5] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, Tal Rabin: *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems*. EUROCRYPT 1999: 295-310

[6] Yehuda Afek, Anat Bremler: *Self-Stabilizing Unidirectional Network Algorithms by Power-Supply* , Chicago Journal of Theoretical Computer Science,1998(3),December 1998.

[7] Bobi Gilburd, Assaf Schuster, Ran Wolff: *Privacy-Preserving Data Mining on Data Grids in the Presence of Malicious Participants*. HPDC 2004: 225-234

[8] Josep Domingo-Ferrer: *A provably secure additive and multiplicative privacy homomorphism*, Lecture Notes in Computer Science, vol. 2433, pp. 471-483, sep. 2002.

[9] I. Damgard and M. Koprowski :*Practical Threshold RSA Signatures Without a Trusted Dealer*,Technical report, Aarhus University, BRICS, November 2000.

[10] Chris Clifton, Murat Kantarcioglu, AnHai Doan, Gunther Schadow, Jaideep Vaidya, Ahmed K. Elmagarmid, Dan Suciu: *Privacy-preserving data integration and sharing*. DMKD 2004: 19-26

[11] Murat Kantarcioglu, Chris Clifton: *Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data*. IEEE Trans. Knowl. Data Eng. 16(9): 1026-1037 (2004)

[12] P. Paillier: *Public-Key Cryptosystems based on Composite Degree Residue Classes*, Advances in Cryptology - EUROCRYPT 99, LNCS volume 1592, pp. 223-238. Springer Verlag, 1999.

[13] I. Damgard, and M. Jurik: *A Generalisation, a Simplification and some Applications of Paillliers Probabilistic Public-Key System, Public Key Cryptography (PKC 2001)*, LNCS 1992, pp. 119-136. Springer Verlag, 2001.

[14] David Kempe, Alin Dobra, and J. E. Gehrke. *Computing Aggregate Information using Gossip*. In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003). Cambridge, MA, October 2003.
[15] Adi Shamir: *How to share a secret*, Communications of the ACM, 22(1), pp612613, 1979.
[16] T.Pedersen : *A threshold cryptosystem without a trusted party*. In Advances in Cryptology, EUROCRYPT'91, pages 522-526. LNCS No.547.
[17] T.Pedersen : *Non-interactive and information-theoretic secure verifiable secret sharing*. In Advances in Cryptology, EUROCRYPT'91, pages 522-526. LNCS No.576.
[18] Stanislaw Jarecki: *Proactive secret sharing and public key cryptosystems*. Master's thesis, MIT, September 1995