ISSN:0975-9646

Privacy and Security Issues in Cloud

Shiv Shankar Barai, Sridhar Mocherla

Computer Science and Engineering, CBIT
Hyderabad, Telangana, India

Abstract— **Cloud computing has become ubiquitous in the corporate world. Its effectiveness in reducing cost while providing a collaborative platform for information processing has helped overhaul the traditional information storage systems and overcome the faults. But, cloud storage involves sharing data with a third-party and security becomes a vital issue. Organizations cannot afford any confidential data to be stolen and in the case that happens, the damage could be irreparable. In this paper, various problems and solutions regarding cloud security are discussed. An overview of cloud security is given before dealing with specific issues dealt by organizations and solutions to those problems.**

Keywords— **Cloud, Encryption, Security, Privacy, Social networks.**

I. INTRODUCTION

Cloud computing is a phrase used to describe a variety of computing concepts that involve a large number of computers connected through a real-time communication network such as the Internet. Instead of storing/processing data, creating and running applications on a local machine, cloud computing removes the tedious tasks and gives the user convenient access to perform his operations. With the increase in internet speeds around the world, the prominence of cloud has rapidly risen. Security of data and applications has become a major issue with the increase in scale of clouds. Cloud Security is a sub-domain of cloud computing that refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing.

Cloud security architecture is effective only if the correct defensive implementations are in place. An efficient cloud security architecture should recognize the issues that will arise with security management. The security management addresses these issues with security controls. These controls are put in place to safeguard any weaknesses in the system and reduce the effect of an attack. While there are many types of controls behind a cloud security architecture, they can usually be found in one of the following categories – Deterrent, Preventive, Corrective and Detective controls.

II. PRIVACY AND SECURITY CONCERNS

In general, the security and privacy issues face by outsourcing data to cloud can be categorized as below:

A. Identity Management

Every enterprise will have its own identity management system to control access to information and computing resources.

B. Physical & Personnel Security

Providers ensure that physical machines are adequately secure and that access to these machines as well as all relevant customer data is not only restricted but that access is documented.

C. Availability

Cloud providers assure customers that they will have regular and predictable access to their data and applications.

D. Application Security

Cloud providers ensure that applications available as a service via the cloud are secure by implementing testing and acceptance procedures for outsourced or packaged application code. It also requires application security measures be in place in the production environment.

E. Privacy

Finally, providers ensure that all critical data (credit card numbers, for example) are masked and that only authorized users have access to data in its entirety. Moreover, digital identities and credentials must be protected as should any data that the provider collects or produces about customer activity in the cloud.

F. Legal Issues

In addition, providers and customers must consider legal issues, such as Contracts and E-Discovery, and the related laws, which may vary by country.

III. CURRENT ISSUES AND PROPOSED SOLUTIONS

From here on, three specific and complex issues faced by organizations when deploying information to the cloud are dealt with in detail. These issues are – *Data ownership and group management, Outsourcing data to a cluster of machines, Outsourcing social networks to cloud.*

A. Data Ownership and Group Management

Data storage is one of the services offered by cloud providers. Consider the practical data application of a company. A company allows its staffs in same group or department to store and share files in the cloud. This relieves the staff from maintaining local storage. However, issues like confidentiality of data, privacy of users are under threat. Cloud providers are generally perceived as untrustworthy by users. Data privacy can be implemented by encrypting data before they are stored in the cloud. Identity privacy must not be absolute as it may be misused. Another feature that must be implemented is multiple-ownership of data i.e. any member of a group can modify

the data that belongs to the group. Groups are also dynamic in nature. Users can leave groups and new users may replace them. Handling of data sharing becomes extremely complex in such situations. To handle all of the conditions mentioned above, a system is proposed for multi-owner data sharing scheme for dynamic groups in cloud, whose features are:

1. Any user in any group can securely share data with others in the untrusted cloud.
2. New users can access data belonging to the group without contacting other participants.
3. Confidentiality of data and anonymity of users are preserved. In case of identity disputes, group manager can reveal the real identities.
4. Effectiveness of scheme is realized by providing extensive simulations.

1) *Terminology:* Bilinear Maps**:** Let G_1 and G_2 be an additive cyclic group and a multiplicative cyclic group of the same prime order q, respectively [11]. Let e :G_1xG_1->G_2 denote a bilinear map constructed with the following properties:
   1. Bilinear: For all a,b ε Z_q^* and P,Q ε G_1, $e(aP,bQ)=e(P,Q)^{ab}$
   2. Nondegenerate: There exists a point P such that $e(P,P)\neq 1$
   3. Computable: There is an efficient algorithm to compute e(P,Q) for any P,Q ε G_1.

   Group Signature: Group signatures allow any member of the group to sign messages while keeping the identity secret from verifiers. A group signature scheme is used to provide anonymous access control in the proposed system.

   Dynamic Broadcast Encryption: Broadcast encryption allows a broadcaster to transmit encrypted data so that only a privileged subset can decrypt the data. Dynamic broadcast encryption allows the group managers to dynamically include new members while preserving previously computed information. The dynamic broadcast encryption technique introduced here is based on bilinear pairs.

2) *System Model:* There are major entities involved in the proposed system – Cloud, Group Manager and Group Members. Cloud is operated by CSPs and provides storage capabilities. But the cloud is not trusted by the users. Group manager take charge of system parameters, user registration, revocation of users and revealing the real identity of a disputed data owner. Group members are a set of registered users that store their private data into the cloud server and share them with others in their group.

Fig 1. Overview of the system

3) *Proposed Model:* The various details of the proposed scheme are included under System Initialization, User revocation, File generation, File access and traceability.

System Initialization: The group manager is responsible for system initialization as follows:

1. Generating a bilinear map group system $S=(q_1,G_1,G_2,e)$.
2. Selecting two random elements H, H_0 ε G_1, along with two random numbers $\varepsilon_1,\varepsilon_2 \in Z_q^*$, computing $U=\varepsilon_1^{-1}H$ and $V=\varepsilon_2^{-1}H \in G_1$ such that $\varepsilon_1.U=\varepsilon_2.V$. In addition, the group manager computes H_1, H_2 where $H_1=\varepsilon_1 H_0$ and $H_2=\varepsilon_2 H_0 \in G_1$.
3. Randomly choosing two elements P,G ϵ G_1 and a number $\gamma \in Z_q^*$ and computing W=γ.G and Z=e(G,P), respectively.
4. Publishing the system parameters including (S,P,H,H0,H1,H2, U, V ,W, Y ,Z, f, f1, $Enc_k()$} where f is a one-way hash function: {0, 1}*-> Z_q , f1 is hash function: {0, 1}*-> G_1 and $Enc_k()$ is a secure symmetric encryption algorithm with secret key k.

User Registration: If a user i wants to register with identity ID_i, the group manager randomly selects a number $x_i \in Z_q^*$ and computes A_i, B_i as the following equation:

$$\begin{cases} A_i = \dfrac{1}{\gamma + x_i} \cdot P \in G_1 \\ B_i = \dfrac{x_i}{\gamma + x_i} \cdot G \in G_1. \end{cases}$$

Fig. 2 Equations for parameters

(x_i,A_i,B_i) is the private key.

User revocation:

User revocation is performed by maintain a public Revocation List (RL). The revocation list is characterized by a series of time stamps (t1 < t2 ,... tr). Let ID_{group} denote the group identity. The tuple (A_i, x_i,t_i) represents that user i with the partial private key (A_i, x_i) is revoked at time ti. P1,P2.. Pr and Z_r are calculated by the group manager with the private secret γ as follows

$$\begin{cases} P_1 = \dfrac{1}{\gamma + x_1} \cdot P \in G_1 \\ P_2 = \dfrac{1}{(\gamma + x_1)(\gamma + x_2)} \cdot P \in G_1 \\ P_r = \dfrac{1}{(\gamma + x_1)(\gamma + x_2) \cdots (\gamma + x_r)} \cdot P \in G_1 \\ Z_r = Z^{\frac{1}{(\gamma + x_1)(\gamma + x_2) \cdots (\gamma + x_r)}} \in G_2. \end{cases}$$

Revocation List

ID_{group}	A_1	x_1	t_1	P_1			
	A_2	x_2	t_2	P_2			
	.	.	.	.			
	A_r	x_r	t_r	P_r	Z_r	t_{RL}	$sig(RL)$

Fig 3. Revocation List

In addition, RL is bounded by a signature sig(RL) to declare its validity. The signature is generated by the group manager with the BLS algorithm, i.e. sig(RL)=γf_1(RL). The group manager then migrates RL to cloud for public use.

File Generation:

The process of storing and sharing a file in the cloud consists of the following operations performs the following operations:

1. Getting the revocation list from the cloud. In this step, the member sends the group identity ID_{group} as a request to the cloud. Then, the cloud responds the revocation list RL to the member.
2. Verifying the validity of the received revocation list. First, checking whether the marked date is fresh. Second, verifying the contained signature sig(RL) by the equation e(W,f_1(RL)) =e(P,sig(RL)). If the revocation list is invalid, the data owner stops this scheme.
3. Encrypting the data file M. The encryption process has two cases:
   a. There is no revoked user in the RL:
      i. Select unique data file identity ID_{data}.
      ii. Choosing a random number $k \in Z_q^*$.
      iii. Computing the parameters C_1,C_2,K,C as the following equation:

$$\begin{cases} C_1 = k \cdot Y \in G_1 \\ C_2 = k \cdot P \in G_1 \\ K = Z^k \in G_2 \\ C = Enc_K(M). \end{cases}$$

Fig 4. Keys for encryptions

   b. There a r revoked users in the RL:
      i. Select unique data file identity ID_{data}.
      ii. Choosing a random number $k \in Z_q^*$.
      iii. Computing the parameters C_1,C_2,K,C as the following equation

$$\begin{cases} C_1 = k \cdot Y \in G_1 \\ C_2 = k \cdot P_r \in G_1 \\ K = Z_r^k \in G_2 \\ C = Enc_K(M). \end{cases}$$

Fig 5. Computing ciphertext

   c. Selecting a random number t and computing f(t). The hash value will be used for data file deletion operation. In addition, the data owner adds f(ID_{data},t) into his local storage.
   d. Message format for uploaded file.

Message Format for Uploading Data

Group ID	Data ID	ciphertext	hash	Time	Signature
ID_{group}	ID_{data}	C_1, C_2, C	$f(\tau)$	t_{data}	σ

Fig 6. Message Format for Uploading Data

   e. Uploading the data shown into the cloud server and adding the ID_{data} into the local shared data list maintained by the manager. On receiving the data, the cloud first checks its validity. If the algorithm returns true, the group signature is valid; otherwise, the cloud abandons the data. In addition, if several users have been revoked by the group manager, the cloud also performs revocation verification. Finally, the data file will be stored in the cloud after successful group signature and revocation verifications.

File Deletion:

This operation consists of following actions:

1. Obtaining the tuple (ID_{data},t) from the data owner's local storage.
2. Invoking group signature algorithm on (ID_{data}, t).
3. Sending (ID_{data}, t) and the signature as a delete request to cloud.
4. Upon receiving the delete request, the cloud calls the signature generation algorithm and the revocation verification algorithm to verify the group signature.
5. The cloud will delete the file if f(t) equals the hash value contained in the file.

File Access:

To access the contents of a shared file, a member has to do the following actions:

1. Users uses his private key to compute a signature s on the message (ID_{group},ID_{data},t). The users sends this message as a request to cloud server. On receiving the request, the cloud server verifies the signature and performs revocation verification. After successful verification, the server returns the data file and RL to server.
2. Checking the validity of revocation list, similar to step 2 of file generation phase.
3. Verifying validity of file and decrypting it. This operation can be divided into 3 cases based on the time stamp t_{data}:
   a. Case 1 ($t_{data}<t_1$). There is no revoked user before data file is uploaded
      i. Check group signature. If false, user stops this operation.
      ii. Using partial private key (A,B) to compute k=e(C_1,A)e(C_2,B)
      iii. Decrypting the ciphertext C with the computed key K.
   b. Case 2 ($t_i<t_{data}<t_{i+1}$). The case indicates that i users have been revoked since data file is uploaded
      i. Check group signature.
      ii. Input A_1,A_2,..A_i to Revocation Verification algorithm. If algorithm returns invalid, stop operation.
      iii. Calculating decryption key K=e(C_1,$A_{i,r}$)e(C_2,B)
      iv. Decrypt ciphertext with key K.
   c. Case 3 ($t_r<t_{data}$). This indicates that r users have been revoked before the data file is uploaded.
      i. Verify group signature using group signature algorithm.
      ii. Input A_1,A_2,...Ar to RV algorithm. If algorithm returns invalid, terminate

operations.

iii. Calculating the decryption key $K=e(C_1,A_{r,r})e(C_2,B)$.

iv. Decrypt the ciphertext C with the key K.

Traceability:

When a dispute occurs on the issue of the data owner, the group manager performs the tracing operation to identify the real identity of the data owner. Given a signature $s=(T_1,T_2,T_3,C,S_1,S_2,S_3,S_4,S_5)$, the group manager employs his private key to compute $A_i=T_3-(\varepsilon_1.T_1+\varepsilon_2.T_2)$. Using A_i , the group manager can identify the real owner from the user list.

B. Outsourcing Data to a Cluster of Machines

In the problem privately outsourced computation client wishes to delegation of a function f on a private input x to an untrusted worker without the latter learning anything about x and f(x). The problem of privately outsourcing computation to a cluster of machines which occurs when the computation is beyond the capabilities of single machine, e.g., to analyze large scale networks, is considered. Here the computations are performed by large scale clusters of workers.

This problem is addressed by introducing the notion of parallel homomorphic encryption scheme introduced in [1]. Parallel Homomorphic Encryption is an encryption technique that support computation over encrypted data via evaluation algorithms that can be efficiently executed in parallel. The delegated PHE technique hides the function being evaluated. MapReduce model of parallel computation shows how to construct PHE schemes that supports various MapReduce operations on encrypted datasets. The PHE schemes are two new constructs of randomized reductions for univariate and multivariate polynomials. Here the privacy of the input is guaranteed even if the adversary sees all the client's queries.

Randomized reductions for univariate polynomials is information-theoretically secure and is based on permutation polynomials, whereas our reduction for multivariate polynomials is computationally secure under the multi-dimensional noisy curve reconstruction assumption.

For our purposes, the cluster of workers are considered as a system composed of w workers and one controller. Given some input, the controller generates n jobs (where typically $n >> w$) which it distributes to the workers. Each worker executes its job in parallel and returns some value to the controller who then decides whether to continue the computation or stop it.

In this work, the client encrypts x and sends the ciphertext and f to the controller. Using the ciphertext, the controller generates n jobs that it distributes to the workers who execute their jobs in parallel. When the entire computation is finished, the client receives a ciphertext which it decrypts to recover f(x). In delegated PHE the function f is hidden. It includes an additional token generation algorithm that takes as input f and output a token T that reveals no information about f but that, nonetheless, can be used by the evaluation algorithm to return an encryption of f(x).

Applications of PHE: It is used for the setting of outsourced computation where a weak computational devices wishes to make use of the resources of a more powerful server. Using this scheme a client can encrypt a large database during an offline phase and then have the workers evaluate different functions on its data during the online phase. The client does not need to know the functions it want to evaluate during the online phase at the time of encryption.

Parallel computation: The specifics of how the computation and communication between processors are organized leads to particular architectures, each having unique characteristics in terms of computational and communication complexity. This has motivated the design of several architecture independent models of parallel computation, including NC circuits, the parallel RAM (PRAM) and Dyrad Models. Arithmetic PHE scheme yields an NC-parallel HE scheme for any function f in NC.

Applications to cloud based cluster computing: A MapReduce algorithm is run by an execution framework that handles the details of distributing work among the machines in the cluster, balancing the workload so as to optimize performance and recovering from failures. The most popular framework is Hadoop which is open source and used by hundreds of large organizations including Amazon, EBay, Facebook, Yahoo, Twitter and IBM.

Recent trend in cluster computing has been to make use of cloud infrastructures such as Amazon's Elastic MapReduce, Cloudera's Hadoop distribution and Microsoft Azure Hadoop service. With such services, a client can run a MapReduce algorithm on massive datasets in the cloud. While these services allow clients to take advantage of all the benefits of cloud computing, they require the client to trust the provider with its data.

Using an MR-parallel HE scheme a client can maintain the confidentiality of its data while utilizing the processing power of a third party MapReduce cluster. Of course, the client must bear the costs of encryption and decryption which for massive datasets can represent a non-trivial amount of work. But this cost is dominated by the amount of work that is outsourced. All of our constructions, encryption can be performed in a streaming manner. This means that even if the data is very large, it can still be encrypted by a weak client albeit rather slowly.

1) *Terminology:* MapReduce: It offers a simple interface to design and implement parallel algorithms and an execution framework that handles the details of distributing work among the machines in the cluster, balancing the workload so as to optimize performance and recovering from failures. A MapReduce program is composed of a Map() procedure that performs filtering and sorting and a Reduce() procedure that performs a summary operation.

Homomorphic Encryption: Homomorphic encryption is a secure probabilistic encryption scheme allowing the server to perform computations on encrypted data with the final result decrypted at the proxy. An encryption scheme is Homomorphic if it supports computation on encrypted data in addition to the standard encryption and decryption algorithms it

also has an evaluation algorithm that takes as input an encryption of some message x and a function f and returns an encryption of f(x) .

Randomized Reductions: Randomized reductions transforms the input in to n pairs of output such that upon decrypting the output we get original message. RR guarantees that no information about the input can be obtained.

2) *Proposed Scheme:* MapReduce Parallel homomorphic Encryption using Randomized Reductions involves MapReduce Algorithm, PHE algorithm and Randomized Reduction algorithm.

Homomorphic Encryption: Let F be a family of n-ary functions. A F-homomorphic encryption scheme is a set of four polynomial-time algorithms HE = (Gen, Enc, Eval, Dec) such that Gen is a probabilistic algorithm that takes as input a security parameter k and outputs a secret key K; Enc is a probabilistic algorithm that takes as input a key K and an n-bit message m and outputs a ciphertext c; Eval is a (possibly probabilistic) algorithm that takes as input a function f ∈ F and n encryptions $(c_1,...,c_n)$ of messages $(m_1,...,m_n)$ and outputs an encryption c of $f(m_1,...,m_n)$; and Dec is a deterministic algorithm takes as input a key K and a ciphertext c and outputs a message m.

Map Reduce Algorithm: A MapReduce algorithm Π = (Parse, Map, Red, Merge) is executed on a cluster of w workers and one controller as follows. The client provides a function f and an input x to the controller who runs Parse on (f, x), resulting in a sequence of input pairs $(l_i, v_i)_i$. Each pair is then assigned by the controller to a worker that evaluates the Map algorithm on it. This results in a sequence of intermediate pairs $\{(\lambda_j, \gamma_j)\}_j$. Note that since the Map algorithm is stateless, it can be executed in parallel.

MR – Parallel HE: A private-key MR-parallel F-homomorphic encryption scheme is a tuple of polynomial-time algorithms PHE = (Gen, Enc, Eva, Dec), where (Gen, Enc, Dec) are as in a private-key encryption scheme and Eval = (Parse, Map, Red, Merge) is a MapReduce algorithm, as explained in [2]. More precisely we have:

$K \leftarrow Gen(1_k)$: is a probabilistic algorithm that takes as input a security parameter k and that returns a key K.

$c \leftarrow Enc(K,x)$:is a probabilistic algorithm that takes as input a key K and an input x from some message space X, and that returns a ciphertext c. We sometimes write this as $c \leftarrow Enc\ K(x)$.

$\{(l_i, v_i)\}_i \leftarrow Parse(f, c)$: is a deterministic algorithm that takes as input a function f ∈F and a ciphertext c, and that returns a sequence of input pairs.

$(\lambda j,\gamma j)j \leftarrow Map(l,v)$: is a (possibly probabilistic) algorithm that takes an input pair (l,v) and that returns a sequence of intermediate pairs.

$(\lambda,z) \leftarrow Red(\lambda, P)$: is a (possibly probabilistic) algorithm that takes a label λ and a partition P of intermediate values and returns an output pair (λ,z).

$c_0 \leftarrow Merge(\lambda_t,z_t)_t$: is a deterministic algorithm that takes as input a set of output pairs and returns a ciphertext c_0.

$y \leftarrow Dec(K,c_0)$: is a deterministic algorithm that takes a key K and a ciphertext c0 and that returns an output y. This is sometimes written as $y \leftarrow Dec\ K(c_0)$.

We say that PHE is correct if for all k ∈N, for all f ∈F_k, for all K output by $Gen(1_k)$, for all x ∈ X, for all c output by $Enc_K(x)$, $Dec_K(Eval(f,c))= f(x)$.

3) *Procedure:* Let HE = (Gen, Enc, Eval, Dec) be a public-key -homomorphic encryption scheme and let RR = (Scatter, Recon) be a C-universal (t,n)-local randomized reduction from f to g such that Recon belongs to F. Consider the multi-use MR-parallel C-homomorphic encryption scheme PHE = (Gen,Enc, Eval, Dec), where PHE.Eval =(Parse, Map, Part, Red, Merge), defined as follows:

- $Gen(1^k)$: compute (pk; sk) ← $HE:Gen(1^k)$. Output K = (sk, pk).
- Enc(K, x): for all i belongs to [#x], compute (s_i, st_i) Scatter(xi) and e_i $HE:Enc_{pk}(sti)$. Output c =
- (pk, s1,…,$s_{\#x}$,e_1,…,$e_{\#x}$).
- Parse(f, c): for all i belongs to [#x] and j belongs to [n], set $l_{i,j} := i$ and $v_{i,j} := (f, pk, s_i[j], e_i)$. Output $(l_{i,j}, v_{i,j})_{i,j}$.
- Map(l,v): parse v as (f, s, e) and compute a ← $HE.Enc_{pk}(g(s))$. Output n:=l and w:= (a, e).
- Red(n, P): parse P as (ar, er)r and compute z ← HE:Eval(Recon, er, (ar)r). Output (n,z).
- $Merge((n_t,z_t)_t)$:output $c' := (z_t)_t$.
- Dec(K, c'): for all i belongs to [#c'], compute $y_i := HE:Dec_{sk}(z_i)$. Output y = (y1,….., $y_{\#c0}$).

C. Outsourcing Social Networks to Cloud

Storage of Social network data is one of the important services of cloud. Today, many companies publish social networks to a third party, e.g., a cloud service provider. Hence, preserving privacy when publishing social network data is important, with reasons explained in detail in [4]. Social networks model social relationships with a graph structure using nodes and edges, where nodes model individual social actors in a network, and edges model relationships between social actors. The relationships between social actors are often private, and directly outsourcing the social networks to a cloud may result in unacceptable disclosures. For example, publishing social network data that describes a set of social actors related by sexual contacts or shared drug injections may compromise the privacy of the social actors involved. A naive approach is to simply anonymize the identity of the social actors before outsourcing. However, an attacker that has some knowledge about a target's neighbourhood, especially a one-hop neighbourhood and the structure of the graph, can still re-identify the target with high confidence. This type of an attack is termed as 1*-neighbourhood attack. To generate an anonymized social network which is secure from such an attack, we propose a heuristic indistinguishable group anonymization (HIGA) scheme.

1) *Terminology:* 1-neighborhood graph: A graph showing a target actors one hop neighbours, i.e., the neighbours that are directly connected to the node under consideration, using edges.

1*-neighborhood graph: A 1-neighborhood graph where the degrees of the one hop neighbors are known.

1*-neighbourhood attack: An attack where the attacker is assumed to know the degrees of the target's one-hop neighbours, in addition to the structure of the 1-neighborhood graph.

Node Indistinguishability: Nodes u and v are indistinguishable if an observer cannot decide whether or not G∗ u = G∗ v in the original graph G, by comparing G∗ u and G∗ v in an anonymized graph G.

Group Indistinguishability: For a group of nodes g = {v|v ∈ V (G)} and |g|⩾k if for each pair of nodes {u,v |u,v ∈ g}, u and v are indistinguishable in the published graph G, group g is an indistinguishable group.

Probabilistic Indistinguishability: A published social network achieves probabilistic indistinguishability, if all nodes {v|v ∈ V (G)} can be classified into m ⩾ 1 groups, where each group has the property of group indistinguishability.

2) *System Model*: We consider a system that consists of a publisher, a cloud service provider, an attacker, and many users. The publisher, such as Facebook or Twitter, outsources a social network to a cloud. In our system, a social network is modelled as an undirected and unlabeled graph. The node identities are assumed to be removed.

The attacker has certain background knowledge about the target and he tries to re-identify the target by analysing the outsourced social network. To protect the privacy of the social actors in the network from the attacker, the publisher anonymizes the network before outsourcing.

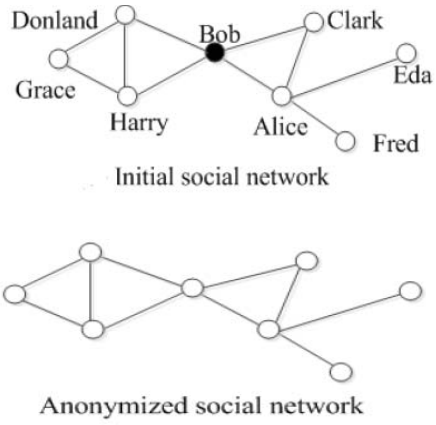

Fig 7: Social network anonymization

3) *Proposed Scheme*: To generate an anonymized social network which is secure from 1*-neighbourhood attack, we propose a heuristic indistinguishable group anonymization (HIGA) scheme.

Our basic idea consists of four key steps:

1. Grouping
2. Testing
3. Anonymization
4. Randomization

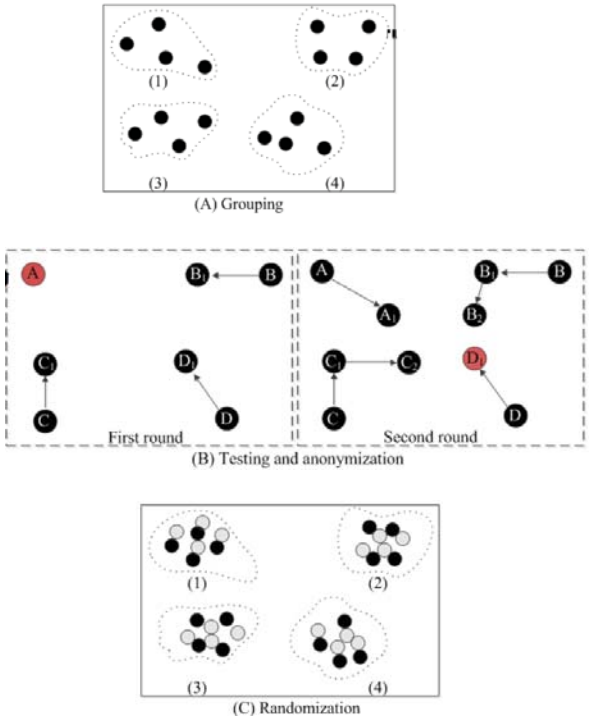

Fig 8: Analogue of the HIGA scheme

The various details of the proposed scheme are explained below:

Grouping:

We classify nodes whose 1*- neighbourhood graphs satisfy certain metrics into groups, where each group size is at least equal to k. We group nodes by using the following metric: number of one-hop neighbours, in-degree sequence, out-degree sequence, total number of edges, and betweenness.

Although other metrics, e.g., closeness centrality and local clustering coefficient, also can be used for grouping, we only consider the above metrics. The concepts of "number of one-hop neighbours" and "total number of edges" are easily understood. Therefore, we only provide the definitions for the other metrics.

For a node v ∈ V (G) whose 1*-neighbourhood graph Gv* = (Gv,Dv), where Gv = (Vv,Ev), we have the following definitions:

In-degree sequence. $I_v = \{|E_u^+|\}_{u \in V_v}$, *where* $E_u^+ = \{(u,w)|w \in V_v\}$, *and* $|E_u^+|$ *is the number of edges in* E_u^+.

Out-degree sequence. $O_v = \{|E_u^-|\}_{u \in V_v}$, *where* $E_u^- = \{(u,w)|w \notin V_v\}$, *and* $|E_u^-|$ *is the number of edges in* E_u^-.

Betweenness. $B_v = |V_v^*|/|V_v^+|$, *where* $V^* = \{\langle u,w\rangle | u,v \in V_v \wedge (u,w) \notin E_v\}$, *and* $V_v^+ = \{\langle u,w\rangle | u,v \in V_v\}$.

Fig 9: Neighbourhood sequences

For a node v, the in-degree sequence is a sequence of in-degrees for v's one-hop neighbours, where the in-degree for v's one-hop neighbourhood u is the number of edges that are connected between u and other v's one-hop neighbours; the out- degree sequence is a sequence of out-degrees for v's one-hop neighbours, where the out-degree for v's one-hop neighbourhood u is the number of edges that are connected between u and the nodes outside v's 1-neighborhood graph; the betweeness is the ratio of the number of paired nodes whose shortest path must go through node v to the total number of node pairs in v's 1-neighborhood graph.

Testing:

Random walks are used for testing. The random walk (RW) is known as a useful tool to obtain the steady state distribution for a graph, referred to as the topological signatures, which provide the foundation for the approximate matching. Specifically, given graph G = (V (G), E (G)), where V (G) = {u_1, u |V (G)|}. A RW on G allows the probability $p_{uj}(t)$ of a node $u_j \in V(G)$ being located at time t to be computed with Eq. 1:

$$p_{u_j}(t) = \sum_{u_i \in V(\mathcal{G})} \frac{1}{|V(\mathcal{G})|} \cdot (1-d) \cdot p_{u_i}(t-1) + \sum_{u_i \in N(u_j)} \frac{1}{|N(u_i)|} \cdot d \cdot p_{u_i}(t-1) \quad (1)$$

Fig 10: Probability of locating a node at a time

where |V (G)| is the number of nodes in G, |N(uj)| is the number of one-hop neighbours for node uj, and d is the damping factor which defines the probability of directly jumping or traversing. For Eq. 1, the first part relates to the probability of moving from node ui to uj by jumping directly, while node ui is not a one-hop neighbour of uj; the second part relates to the probability by traversing the edge from u to uj, while node ui is a one-hop neighbour of uj. Therefore, the probability distribution on all nodes in G, denoted as a vector p (t) = [pu1 (t), pu|V (G)|(t)], can be calculated with Eq. 2:

$$\mathbf{p}(t) = \frac{(1-d)}{N} \cdot \mathbb{I} + d \cdot \mathbf{W} \cdot \mathbf{p}(t-1) \quad (2)$$

Fig 11: Probability distribution function of the nodes

Where W = (θA)', A being the adjacency matrix of the graph and θ the diagonal matrix whose diagonal element is 1 |N(ui)| ; I is a vector whose entries are all equal to one.

Anonymization:

Suppose that there are m groups, g1,...,gm, where each group size is assumed to be at least equal to k. For each group, if any pair of nodes are not approximately matching, we use a heuristic anonymization algorithm to make the 1-neighbourhood graphs approximately match as follows: Initially, the candidate group set (CGS) consists of m groups. We sort groups in descending order of the number of neighbours, pick the first one as the processing group, and remove it from CGS. For each node in the processing group, we construct its 1-neighborhood graph, and use RW to calculate related topological signatures. Then, for any pair of nodes u and v, we use Eq. 5 to calculate the cost of matching their 1-neighborhood graphs. For any pair of nodes, if this cost is smaller than a threshold value α, we choose the next grouping in CGS as the processing group and do it again.

Otherwise, we modify 1-neighborhood graphs of the nodes in the processing group as follows: We first choose a random node u in the group as the group seed. For any other node v in this group, if the related cost cost(Gu,Gv) is larger than α, we approach the structure of Gu to that of Gv with probability q, and approach the structure of Gv to that Gu with probability 1−q. This process will continue until, for any pair of nodes in the processing group, the cost for matching their 1-neighborhood graphs is equal to or smaller than α. The anonymization process may be recursive, since some changes may impact the groups that have been processed previously. However, due to the power-law node distribution, and the small world phenomenon, this process will rapidly stop.

To approach the structure of Gv = (Vv,Ev) to that of Gu = (Vu,Eu), we first obtain the optimal matching of nodes in two graphs. In the optimal matching, for any pair of nodes x ∈ Vv and w ∈ Vu, if cost(x,w) > α , we make u' s connections the same as those of v. During the approaching process, we make sure that the structure of Gu will not be modified.

Randomization:

Consider a graph G = (V (G), E (G)) and a randomization probability p. We first randomly remove p(|E(G)|) edges from G, and then for two nodes that are not linked, we add an edge with probability p. The key problem lies in determining p to randomize the graph.

IV. CONCLUSION

We address the issues *Data ownership and group management, Outsourcing data to a cluster of machines, Outsourcing social networks to cloud* by the above mentioned techniques.

To address data ownership and group management, we design a secure data sharing scheme for dynamic groups in an untrusted cloud. In this scheme, a user is able to share data with others in the group without revealing identity privacy to the cloud. Additionally, the scheme supports efficient user revocation and new user joining. More specially, efficient user revocation can be achieved through a public revocation list without updating the private keys of the remaining users, and new users can directly decrypt files stored in the cloud before their participation.

By using Parallel Homomorphic Encryption we can ensure the data integrity and data error localization. PHE schemes can be used to evaluate several functionalities such as keyword search, set membership testing and disjunctions.

Using Heuristic Indistinguishable Group Anonymization (HIGA) scheme to anonymize social networks is an effective security measure and that it doesn't compromise query answer accuracy

REFERENCES

[1] Sena Kamara and Mariana Raykova, "*Parallel Homomorphic Encryption"*, Microsoft Research, 2013.

[2] Sowmya.D, "*Cryptographic cloud storage with Hadoop Implementation"*, IOSR Journal of Computer Engineering, 2013.

[3] Xuefeng Liu, Yuqing Zhang, Boyang Wang, and Jingbo Yan, Mona: "*Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud"*, IEEE Transactions On Parallel And Distributed Systems, 2013.

[4] Guojun Wang, Qin Liu, Shuhui Yang and Jie Wu, "*Outsourcing Privacy – Preserving Social Networks to a Cloud"*, IEEE transactions on social networking, 2013.

[5] "Can Homomorphic Encryption be Practical?", *Microsoft Research, 2013.*

[6] R. Lu, X. Lin, X. Liang, and X. Shen, "*Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing*," Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010

[7] B. Zhou and J. Pei, "*Preserving privacy in social networks against neighborhood attacks*," in Proc. of IEEE ICDE, 2008

[8] Henri Gilbert, *"Advances in Cryptology -EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques", French Riviera, May 30 -June 3, 2010. Proceedings, volume 6110 of Lecture Notes in Computer Science*. Springer, 2010