International Journal of Current Research and Modern Education (IJCRME)
ISSN (Online): 2455 - 5428
(www.rdmodernresearch.com) Volume I, Issue I, 2016

SECURE & EFFECTIVE COLLISION TOLERANT SCHEDULING FOR UNDERWATER WSN ROUTING

M. Vijayalakshmi* & M. Rajakumar**

* PG Scholar, Department of Master of Computer Applications, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

** Associate Professor & Head, Department of Master of Computer Applications, Dhanalakshmi Srinivasan Engineering College, Perambalur, Tamilnadu

Abstract:

Large numbers of conclusions in privacy-preserving have been obtained by researchers, most of which are based on an assumption that each party is semi-honest. Internet involves and enables sharing of data but identity of the shared data owner is to be preserved. This is called maintaining anonymity that is whistle blower with increasing server storages like cloud this is a problem area. In particular, a party is deemed semi-honest when that the party follows the protocol properly with the exception that it keeps a record of all its intermediate computation results and then tries to deduce further information in addition to the protocol result. Moreover, researchers also assume that every party does not collude or share its record with any other party. However, reliance on such assumptions proves problematic in that a party cannot trust any party without taking substantial risk that their protection of their private information.

Privacy-preserving data mining is about preserving the individual privacy and retaining as much as possible the information in a dataset to be released for mining. The perturbation approach and the k-anonymity model are two major techniques for this goal. The k-anonymity model assumes a quasi-identifier (QID), which is a set of attributes that may serve as an identifier in the data set. In the simplest case, it is assumed that the dataset is a table and that each tuple corresponds to an individual.

The privacy may be violated if some quasi-identifier values are unique in the released table. The assumption is that an attacker can have the knowledge of another table where the quasi-identifier values are linked with the identities of individuals. Therefore, a join of the released table with this background table will disclose the sensitive data of individuals. Consider a scenario in which a large number of parties, such as many small retail stores, collaborate to compute some secure function of their inputs. For example, they want to get some information regarding market conditions. In such a scenario, some of the parties involved may be enticed to sneak a peek at their competitor's private records. This scenario with four parties, numbered 1, 2, 3 and 4, collaborating to compute some secure function of their inputs, 1X,2X, 3X and 4X respectively. Assume that party 1 is a common competitor of party 2, 3 and 4. Then these parties may thus choose to unite to form a coalition after the execution of the protocol to deduce the privacy information of party 1 from the messages received from party 1 (m2, m3, m4) and the final result. Methods for assigning and using sets of pseudonyms have been developed for anonymous communication in mobile networks. The methods developed in these works generally require a trusted administrator, as written, and their end products generally differ from ours in form and/or in statistical properties.

Introduction:

The availability of huge numbers of databases recording a large variety of information about individuals makes it possible to discover information about specific individuals by simply correlating all the available databases. Although confidentiality and privacy are often used as synonyms, they are different concepts: data

confidentiality is about the difficulty (or impossibility) by an unauthorized user to learn anything about data stored in the database. Usually, confidentiality is achieved by enforcing an access policy, and possibly by using cryptographic tools. Privacy relates to what data can be safely disclosed without leaking sensitive information regarding the legitimate owner. The problems of data confidentiality and anonymization have been considered separately. However, a relevant problem arises when data stored in a confidential, anonymity-preserving database need to be updated. The operation of up-dating such a database, e.g. by inserting a tuple containing information about a given individual, introduces two problems concerning both the anonymity and confidentiality of the data stored in the database and the privacy of the individual to whom the data to be inserted are related: (i) Is the updated database still privacy-preserving? and (ii) Does the database owner need to know the data to be inserted? Clearly, the two problems are related in the sense that they can be combined into the following problem: can the database owner decide if the updated database still preserves privacy of individuals without directly knowing the new data to be inserted.

This paper builds an algorithm for sharing simple integer data on top of secure sum. The sharing algorithm will be used at each iteration of the algorithm for anonymous ID assignment (AIDA). This AIDA algorithm, and the variants that we discuss, can require a variable and unbounded number of iterations. Finitely-bounded algorithms for AIDA increase a parameter in the algorithm will reduce the number of expected rounds. However, our central algorithm requires solving a polynomial with coefficients taken from a finite field of integers modulo a prime. That task restricts the level to which can be practically raised.

Modules Description:

Network Setup:

In this module the network is setup with a server. Multiple clients are registered and each one is capable of data sharing. Thus each client is a peer and hence is essentially distributed. An identity is assigned but this is not anonymous. This is the base id. The client has to register himself and then upload the relevant data. The client login details are stored in the network database i.e. SQL Server. This is called as the account in the cloud server.

All the fields for the client registration Service providers register themselves with Service Registry, Service consumers discover or find provider for required services from service registry. Then, service consumer binds the service provider to execute the service request by the provider of services. Grid can be used as a layer through which service providers, service consumers or grid clients, grid manager or administrators communicate. XML format is used for platform agnostic communications with open standards for published web services in request / response mode using Simple Object Access Protocol (SOAP) or Web Services Definition Language.

Routing of Messages:

In routing, a set of encrypted layers are responsible for encoding routing information, thereby provides the cloud of routing onions comprised of several routers or nodes. The routing layer provides protection of privacy by concealing the identity of the sender and recipient of a message. It also protects the underlying message through encryption during inter-router traversals in a network. It also provides a strong degree of unlink ability, so that an eavesdropper cannot easily determine both the sender and receiver of a given message at a given time, thereby ensuring anonymity to the greatest extent.

However, an eavesdropper or attacker with the ability to monitor an under loaded router in a network might be able to trace the path of a message through the network and intrude. The routing networks become vulnerable to such intrusion or intersection attacks and predecessor attacks. Intersection attacks are caused due to failing or leaving nodes in a network. Predecessor attacks are facilitated through session tracking of the infected node.

A service requester requests for a service. This request is received by a designated Edge node. The edge node extracts the identification of the service requester and generates a dynamic token (ticket) based on internally generated encryption technique similar to our dynamic token generation algorithm. It then finds out a core node in the onion routing layer to get the desired service. The edge nodes store the sender's details and then find the suitable core node in an onion routing framework.

The core node then seeks for the subsequent core nodes in onion routing methodology and establishes a path to flow the message to the respective Service Provider or a set of Service Providers. The nodes may have multiple roles of a broker, collator, sender on behalf of some other service requester etc. The message flow is done in an encrypted mode. After the service provider(s) complete the processing, the response is sent back to the service requester back in the same path as that established during request sending. The broker, on its own or others in the cloud collates responses from several service providers, if applicable. The collation may require authenticating the ticket, which is all along flowing with the messages. The final results are sent back to the edge node. Edge node in turn sends responses back to the requester. The intermediate nodes may not be available and therefore the encrypted message may get lost. This situation can be better handled by the introduction of an Authentication Server in the Routing Grid framework, where every time a node gets a message registers the sender and receiver information along with the dynamically generated ticket.

Data Sharing:

Either a client requests data or calls for submission of data to the application like a voting or opinion or a document. All these involve data to be shared by the user. While collating responses from multiple service providers, the master service provider or broker has to open individual sections of the form. But, the individual service provider is supposed to open the portion of the form designated for its own filling up and not to intervene with anyone else's area. This requires that the data document be multi-parted and have some means to protect portions to undesired service provider. One level of concern is that the XML content, which may contain information of multiple heterogeneous service providers, may get exposed to one service provider, a node on the grid. This may have a breach of privacy between multiple service providers. Many a times, privacy is closely resembled with anonymity that demands the need of being unidentified or unobserved while transacting over public domain such as web or other public realm. Adequate level of privacy needs to be achieved through controlled disclosure of identity and associated information. Anonymity can ensure achievement of privacy needs. In general, anonymous message transmission requires that the transacting message would not carry any information about the original sender and intended receiver. This is the pre step before the AIDA is applied.

Aida – Secure Sum Algorithm:

N nodes wish to compute and share only the average of a data item, without revealing the value of this data item for any member of the group. This is called as the secure sum and it is used sparingly. Each node chooses random values. Each "random"

value is transmitted from node to node. The sum of all these random numbers is desired total. Each node totals all the random values received. Now each node simply broadcasts this message to all other nodes.

All pairs of nodes have a secure communication channel available; a simple, but resource intensive, secure sum algorithm can be constructed. Suppose that a coalition of the nodes seeks to garner information about the private data of the other nodes. Since the parties are semi-honest, the possibly useful outside information available to a coalition seeking to garner the private information of other parties, consists of only the random numbers in the messages received from those parties, the partial sums and the total. The secure sum algorithm just given remains N-private even when the input data are known as a multiset to all parties. We term this input permutation collusion resistance as the coalition knows the data held by the remaining parties only as a multiset.

The correctness of the computation in the multiparty case is confirmed by the results computed locally. In term of the efficiency, the total running time is illustrated. For example, the running time of a computation of SPoS is about 0.769 seconds in the case of 10 parties and 1024 bits size key. From this one can see that the running time is proportional to the number of parties if the key size is fixed. One now compares the efficiency of the method with the proposed by viewing the experiment result of both. Since the experiment was done under the similar environments it can directly compare the result with optimal units. The most time-consuming operation in this is the last step of the secure-logarithm based protocol, while other operations' computational time can be ignored; hence the time is almost not related to the number of parties.

In other words, the efficiency of that method is almost equal to the secure-logarithm-based protocol. However, the computational time of the secure-logarithm-based protocol is quite long. As mentioned in one computation with the 1024 bits key takes about 3.54 seconds, whereas the computational time of SRoS in the case of 20 parties and the 1024 bits key is about the same of, yet the security level our method achieves is much higher. Note that computing one secure ratio of summations needs two times of the SPoS protocol. If one reduces the security level of SPoS to 1-private with the method introduced in 6.3, the running time will be reduced to less than 0.2 seconds.

The WAP protocol proposed contains two times of encryptions, two times of decryptions and two times of communications, which is similar to the 2-party case in our method. Thus the efficiencies of both methods are similar. Mert's OMP protocol is an extension of the WAP protocol. With OMP, each party runs the WAP protocol with the next party in order. Then the efficiency is similar to SPoS in the case of multi-party.

Decode:

The broadcast is unique and hence the algorithm offers the maximum security against privacy. The communications requirements of the algorithms depend heavily on the underlying implementation of the chosen secure sum algorithm. Suppose that our group of nodes wishes to share actual data values from their databases rather than relying on only statistical information as shown in the previous section. That is, each member of the group of nodes has a data item which is to be communicated to all the other members of the group. However, the data is to remain anonymous, collusion resistant method for this task using secure sum as our underlying communication mechanism. Our data items are taken from a, typically finite, field. In the usual case, each will be an integer value and will be the field where is a prime number satisfying for all.

Thus, arithmetic will typically be performed using modulus, but other fields will also be used.

Existing System:

In a distributed environment multiple parties on a network to jointly carry out a global computation by sharing data with each other. The problem of sharing privately held data so that the individuals who are the subjects of the data should not be identified. Existing systems in secure multiparty computation, allows multiple parties on a network to jointly carry out a global computation that depends on data from each party while the data held by each party remains unknown to the other parties.

A secure computation function widely used in the literature is secure sum that allows parties to compute the sum of their individual inputs without disclosing the inputs to one another. This function is popular in data mining applications and also helps characterize the complexities of the secure multiparty computation. This again depends on data from each party while the data held by each party remains unknown to the other parties. This allows parties to compute the sum of their individual inputs without disclosing the inputs to one another. Hence the method is not suited for distributed computing.

Disadvantages:

The existing methods allowing agencies to opt-out of a secure computation based on the results of the analysis, should they feel that those results are too informative about their data. Secure communication is different since there may be disruptions in communications and hence the data owner may also be compromised. There are possibilities of collision to happen in communications. Distributed database access nullifies the anonymity. Also the system uses a trusted central authority and is suitable for that type of computation only. Also computational overheads and communication costs are huge. Tradeoffs between the data to be shared the owner and the anonymity may happen in bandwidth in huge data shares.

Proposed System:

This system uses efficient algorithms for assigning identifiers to the nodes of a network. The IDs are anonymous using a distributed computation with no central authority. Such IDs can be used as part of schemes for sharing or dividing communications bandwidth, data storage, and other resources anonymously and without conflict. Here importance is not given to the communication system hence the costs are less. An anonymous id is assigned to the user and the user's identity is anonymously preserved. It is possible to use secure sum to allow one to opt-out of a computation beforehand on the basis of certain rules in statistical disclosure limitation. Hence automatically during a transaction computation is secure indirectly and is done so in an anonymous manner. The required computations are distributed without using a trusted central authority. This assignment of serial numbers allows more complex data to be shared. This results in nil computational overheads the reason being that it is distributed. Also no tradeoffs happen in bandwidth communication as the id is allotted to the user.

Advantages:

The advantages are obvious with no id being visible to any person at any time of the transaction. This makes the data owner oblivious to the transaction but hides from any faults. The method is not cryptographic hence it does not hog memory space. Also a combination can be used for different variations. The methods developed in these works do not require a trusted administrator, as written, and their end products generally differ from the proposed inform and/or in statistical properties.

System Implementation:

- ✓ Data Aggregation
- ✓ Symmetric Key Cryptosystem
- ✓ Data collector Module

Data Aggregation:

The process of redefining data into a summarization based on some rules or criteria. Aggregation may also encompass de-normalization for data access and retrieval. Data aggregation combines the original order of data and collect fields in different table.

Symmetric Key Cryptosystem:

Symmetric key cryptosystem, using our encryption and decryption. First, for the same level of security, the symmetric key cryptosystem requires a smaller secret key length. Furthermore, the encryption and decryption of the symmetric key cryptosystem is much faster. We require that the cryptosystem be semantically secure, AES-128 under the cipher block chaining counter mode meets this requirement.

Data Collector Module:

Collector collects the detail from user using shaking and shuffling activities. And retrieve the information using data aggregation method. Collector after collect the data forward to admin.

System Maintenance:

File Attribute Generation:

All the attributes for the data files are generated here by the resource owner. This is then sent to the cloud to be stored in a specified format with the attributes and data. For each data file the owner assigns a set of meaningful attributes which are necessary for access control. Different data files can have a subset of attributes in common. Each attribute is associated with a version number for the purpose of attribute update as we will discuss later. Cloud Servers keep an attribute history list AHL which records the version evolution history of each attribute and PRE keys used. In addition to these meaningful attributes, we also define one dummy attribute, denoted by symbol AttD for the purpose of key management. AttD is required to be included in every data file's attribute set and will never be updated. The access structure of each user is implemented by an access tree. Interior nodes of the access tree are threshold gates. Leaf nodes of the access tree are associated with data file attributes. For the purpose of key management, we require the root node to be an AND gate (i.e., n-of-n threshold gate) with one child being the leaf node which is associated with the dummy attribute, and the other child node being any threshold gate. The dummy attribute will not be attached to any other node in the access tree. Fig.1 illustrates our definitions by an example. In addition, Cloud Servers also keep a user list UL which records IDs of all the valid users in the system.

Results and Conclusion:

Whenever there is a user to be revoked, the data owner first determines a minimal set of attributes without which the leaving user's access structure will never be satisfied. Next, he updates these attributes by redefining their corresponding system master key components. Cloud Servers are able to compute a single PRE key that enables them to update the attribute from any historical version to the latest version. This property allows Cloud Servers to update user secret keys and data files in the "lazy" way as follows. Once a user revocation event occurs, Cloud Servers just record information submitted by the data owner as is previously discussed. If only there is a file data access request from a user, do Cloud Servers re-encrypt the requested files and

update the requesting user's secret key. Here the leaving user (i.e. the users' key to be revoked) is ascertained. The attributes are updated minus this user who is going to be revoked.

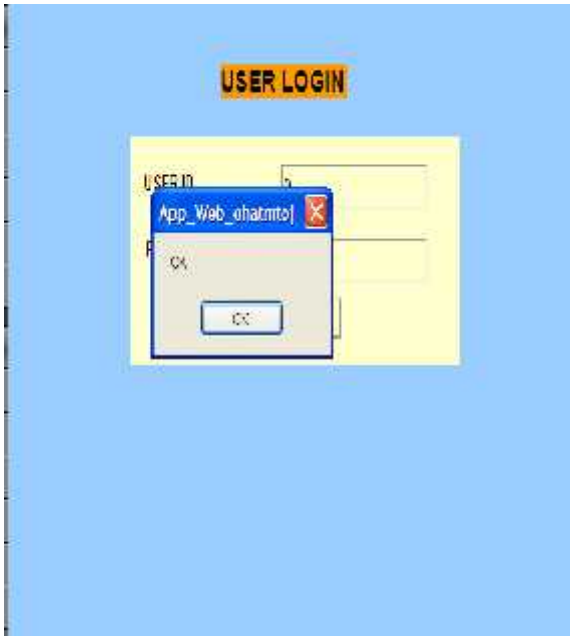
Key Reset:

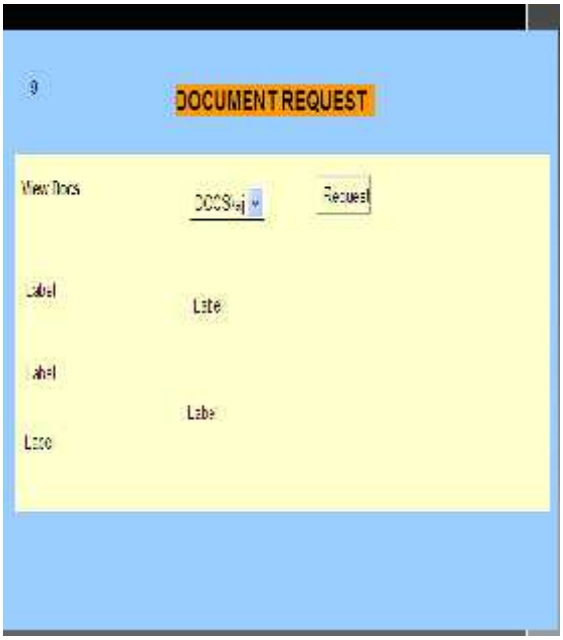
The key is reset for all the users except the revoked user. The data owner determines the minimal set of attributes, redefines MK and PK for involved attributes, and generates the corresponding PRE keys. The owner then sends the user's ID, the minimal attribute set, the PRE keys, the updated public key components, along with his signatures on these components to Cloud Servers, and can go off-line again. Cloud Servers, on receiving this message from the data owner, remove the revoked user from the system user list UL, store the updated public key components as well as the owner's signatures on them, and record the PRE key of the latest version in the attribute history list AHL for each updated attribute. AHL of each attribute is a list used to record the version evolution history of this attribute as well as the PRE keys used. Every attribute has its own AHL. With AHL, Cloud Servers are able to compute a single PRE key that enables them to update the attribute from any historical version to the latest version. This property allows Cloud Servers to update user secret keys and data files in the "lazy" way as follows. Once a user revocation event occurs, Cloud Servers just record information submitted by the data owner as is previously discussed. If only there is a file data access request from a user, do Cloud servers re-encrypt the requested files and update the requesting user's secret key. This statistically saves a lot of computation overhead since Cloud Servers are able to aggregate" multiple update/re-encryption operations into one if there is no access request occurring across multiple successive user revocation events.

Screenshots:

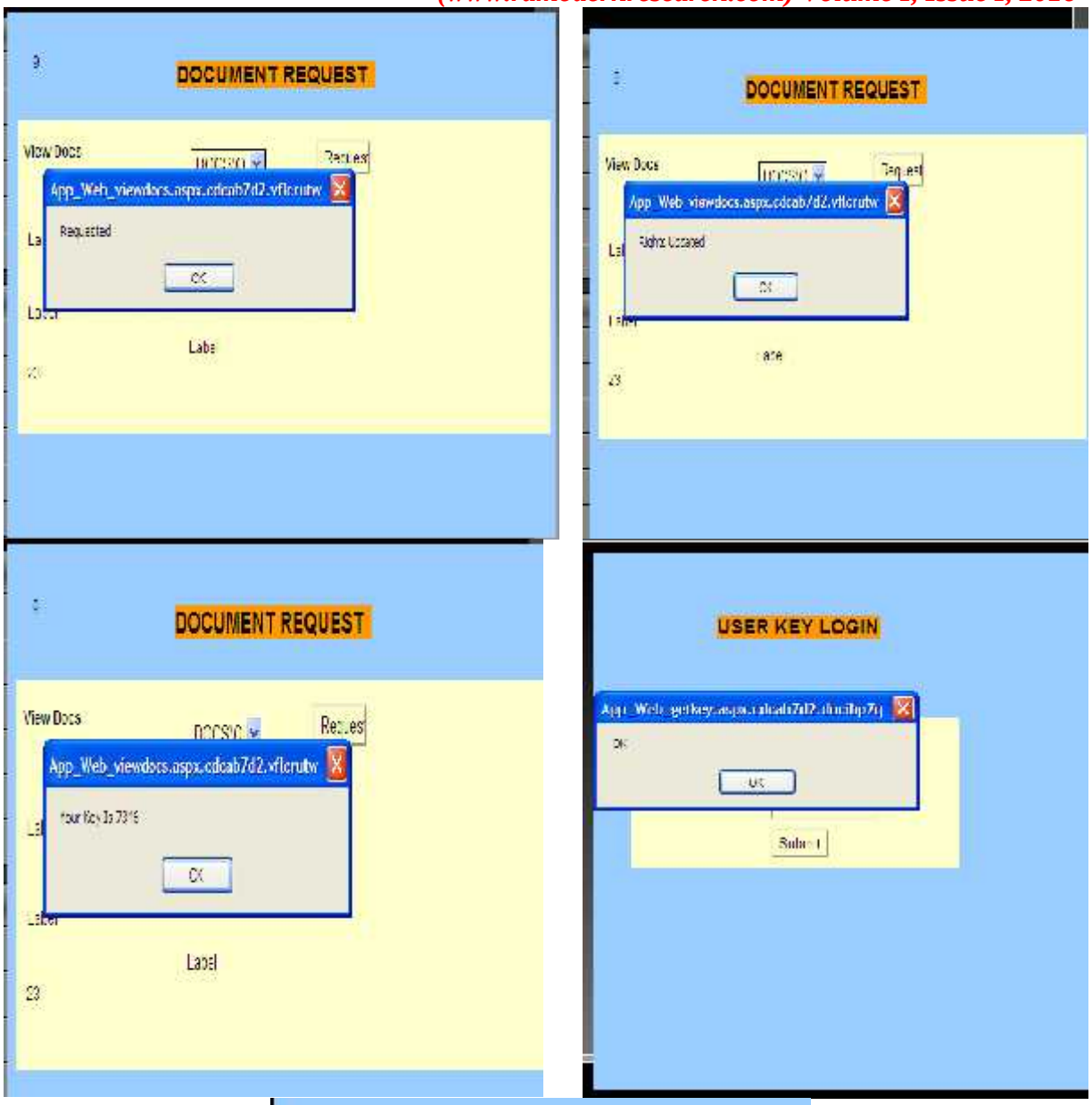

NEW USER REGISTRATION
UserName
Password
DateOfBirth
Address
City
MobileNo
EMailID
Insert Cancel

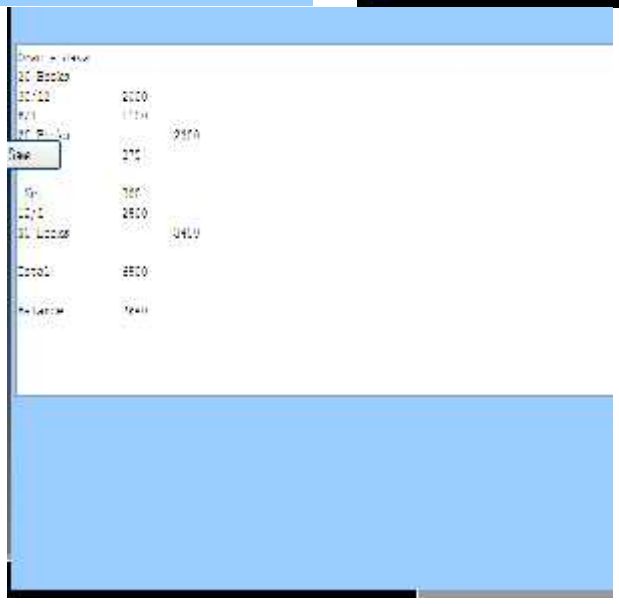

USER LOGIN
USERID
Password
Submit


USER LOGIN
USERID
App_Web_ohatmtoj
OK


DOCUMENT REQUEST
View Docs
Request
Label
Label


DOCUMENT REQUEST
App_Web_ohatmtoj
OK
Label


DOCUMENT REQUEST
View Docs
App_Web_viewdocs.aspx.cdcab7d2.vflcrutw
Requested
OK
Label
DOCUMENT REQUEST
View Docs
App_Web_viewdocs.aspx.cdcab7d2.vflcrutw
OK
DOCUMENT REQUEST
View Docs
App_Web_viewdocs.aspx.cdcab7d2.vflcrutw
OK
Label
USER KEY LOGIN
OK




ADMIN HOME
Logout
View Data Owners
View Users
View Requests
View Documents


VIEW DATA OWNERS
OwnerId OwnerName City EMailId ContactNos Password


VIEW USERS
UserId UserName Password Role DateOfBirth Address City MobileNo EMailID

Conclusion:

Thus the proposed random secure sum AIDA cryptograph algorithm is foolproof in allocating ID to users and the anonymous identity is maintained in the user and the data owner is not compromised under any circumstances thus providing ample proof for the sets of users in multiparty environments. Even under difficult situations the communications and bandwidth is not affected in any manner. So unlike cryptographic measures and traditional systems AIDA proves to be secure for distributed architecture keeping the user safe from prying persons under attack in different segments.

Future Enhancements:

In future the scheme may be extended as a web service so that any interconnected user of the network can utilize it to the maximum without the need to implement the code. Also mobile web services are an area of interest for future extensions to AIDA.

References:

1. Sarbanes–Oxley Act of 2002, Title 29, Code of Federal Regulations, Part 1980, 2003.
2. White Paper—the Essential Guide to Web Analytics Vendor Selection, IBM [Online]. Available: http://measure.coremetrics.com/corem/getform/reg/wp-evaluation-guide
3. A. Shamir, "How to share a secret," Commun. ACM, vol. 22, no. 11, pp. 612–613, 1979.
4. Friedman, R. Wolff, and A. Schuster, "Providing k-anonymity in data mining," VLDB Journal, vol. 17, no. 4, pp. 789–804, Jul. 2008.
5. F. Baiardi, A. Falleni, R. Granchi, F. Martinelli, M. Petrocchi, and A. Vaccarelli, "Seas, a secure e-voting protocol: Design and implementation," Comput. Security, vol. 24, no. 8, pp. 642–652, Nov. 2005.
6. D. Chaum, "Untraceable electronic mail, return address and digital pseudonyms," Commun. ACM, vol. 24, no. 2, pp. 84–88, Feb. 1981.
7. Q. Xie and U. Hengartner, "Privacy-preserving matchmaking for mobile social networking secure against malicious users," in Proc. 9th Ann. IEEE Conf. Privacy, Security and Trust, Jul. 2011, pp. 252–259.
8. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in Proc. 19th Ann. ACM Conf. Theory of Computing, Jan. 1987, pp. 218–229, ACM Press