ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.5, Pages : 254 - 259 (2014)
Special Issue of ICACSSE 2014 - Held on October 10, 2014 in St.Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

Secure sharing in distributed information Management Applications

Thota Beula[1], Eswar Kodali[2]

[1]M.Tech Student, Dept. of CSE St. Ann's College of Engineering and Technology, Chirala, AP, INDIA, thotabeula@gmail.com

[2]Associate Professor, Dept. of CSE St. Ann's College of Engineering and Technology, Chirala, AP, INDIA, kodali_eswar@yahoo.co.in

Abstract: Interaction between entities who may not trust each other is now commonplace on the Internet. This paper focuses on the specific problem of sharing information between distrusting parties. Previous work in this area shows that privacy and utility can co-exist, but often do not provide strong assurances of one or the other. In this paper, we sketch a research agenda with several directions for attacking these problems, considering several alternative systems that examine the privacy vs. utility problem from different angles. We consider new mechanisms such as economic incentives to share data or discourage data leakage and a hybrid of code-splitting and secure multi-party computation to provide various assurances of secrecy. We discuss how to incorporate these mechanisms into practical applications, including online social networks, a recommendation system based on users' qualifications rather than identities, and a "personal information broker" that monitors data leakage over time. We hope that this paper will spark ideas and conversation at ACITA about directions most worth pursuing.

Key words : Access control, information sharing, privacy.

1 INTRODUCTION

The rise of distributed information management (DIM) applications has followed the rise of the Internet. In these applications, users store information on a site for the purpose of sharing it with recipients. Users have incentive to share data; for example, users build social capital when sharing data on a social network like LinkedIn,[1] creating future opportunities for work or collaboration. On the other hand, sharing too much information is dangerous because the recipients of the information, or the system itself (assuming it is not controlled by the information owner), may have incentive to share sensitive data with eavesdroppers that a user has not authorized to view his data. For example, Alice could report to Bob's former employer that he has joined a company in violation of his former employer's IP agreement, having discovered this information on LinkedIn.

2 LITERATURE REVIEW

The rise of distributed information management (DIM) applications has followed the rise of the Internet. In these applications, users store information on a site for the purpose of sharing it with recipients.

2.1 Online social networks

An online social network (OSN) is an application through which users can easily share certain types of information, such as personal expertise and interests, noteworthy professional or personal events, photos, or messages. One popular OSN is Facebook, which has upwards of 500 million active users, 50% of which log in at least once per day [1]. LinkedIn is another popular social network specializing in professional interactions, e.g., for finding employment and making business connections. The Pentagon has recognized that OSNs are valuable for military personnel, recently reversing a ban on the services [2] and launching its own social media hub to better facilitate intra-organization interactions [3].

In the terminology of distributed information management applications that we introduced above, the site is the OSN and users and recipients are OSN participant members. Participants are attracted to OSNs because they make it easy to organize information and share it among a select group, where such information sharing would have been tedious or non-scalable otherwise. Several OSNs serve as a platform for third-party

Applications with which their users can interact, and these interactions are often customized based on a participant's personal data: one popular Facebook application is a calendar that tracks the birthdays of a user's friends [4].

Most OSNs are free to users, adding to their appeal, and generate revenue from advertising. Users' personal data, all of which is available to the site, is of particular interest, since it can be used to display targeted ads. For example, based on a user's status posts and profile information the site can attempt to select ads a certain user is more likely to be interested in; better selection leads to more profit.

To summarize, participants have incentive to share data with their friends and associates and with the site and third-party apps to improve their interaction experience. Third-party appli-cations also often rely on advertising and thus want to attract as many users as possible to their own sites; customizing the experience based on user data is thus useful. The site has incentive to attract as many users as possible and to collect as much data as possible to increase its revenue. Indeed, researchers have observed that the terms-of-use agreement for Facebook is quite draconian [5]; it requires that users "not provide any false personal information on Facebook", "keep [their] contact information accurate and up to date", and grant Facebook a non-exclusive license to any content a user posts.

2.2 Collaborative reviewing

Another class of application that must balance the pri-vacy/utility trade-off is what we call an information hub. Pop-ular examples of such systems are Slashdot [6] and Reddit [7]. The users of information hubs post snippets or links to articles they think might be of interest, and recipients respond with comments about the posting. Some sites bubble highly-favored posts to the top, incentivizing clever and interesting content from users. The hosting service itself has incentive to acquire as many readers as possible, usually to increase ad revenue.Information hubs are valuable to the defense commu-nity. The U.S. intelligence community uses Intellipedia [8], a classified variant of Wikipedia [9], for organizing infor-mation of inter-agency interest. STRATCOM's 4-star blog aims to serve a similar purpose, encouraging contributions outside the chain of command [10]. Here there are similar incentives: posts/articles are rated directly or indirectly and posters/authors can be commentators and vice versa. This encourages posting of high-quality information in Reviewers may sometimes wish to remain anonymous. This is for the same reason the scientific community uses anonymous peer-reviewing today—to protect a reviewer who provides a negative opinion from backlash. At the same time, anonymity (without further assurance) may reduce the credibility of the review. Thus there is incentive to share some information so that others can judge the review, but not enough information to identify the reviewer. Another problem is weighing a reviewer's self-proclaimed expertise. Users would like corroboration of these claims. Such corroboration could come from other users who may agree to the user's expertise, or it may come from artifacts, such as published papers (or program committee memberships) in the declared area of expertise. The reviews themselves add to their own credibility the more they stand alone in justifying their conclusions.

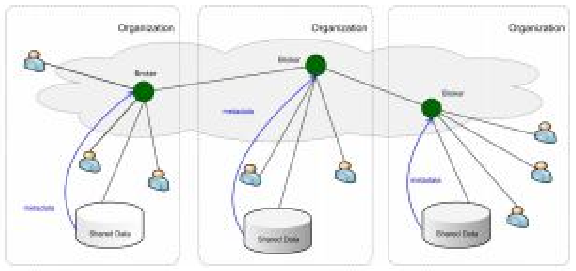

Fig 1. An overview of the IBS infrastructure**.**

3.2 Sensor Networks

Operators of sensor neworks must also balance the trade-off between sharing certain data (measurements and aggregates) while protecting others (device locations). An adversary that learns the locations of sensors can interfere with measurements or capture the sensors [11].

Sensors networks do not exactly fit the DIM model explored in this paper. Specifically, the users (sensors) cooperate with the site (the operator), which determines the policies for sharing with recipients (third-parties that wish to use sensor data). However, they can still benefit from the techniques we discuss. As an example, consider two operators of distinct networks, N_1 and N_2, located near each other. N_1 wishes

to use readings from N_2 to augment its own data collection. For sharing to be successful, each must take precautions. At the very least, N_2 must aggregate (or otherwise mask) its data so that N_1 cannot determine the location of the sensors. N_2 may wish to further limit the data it shares so that N_1 cannot use it to attain a tactical or competitve advantage over N_2. On the other hand, N_1 must ensure that N_2 is not sharing bogus data. It can do so by, for instance, ensuring that readings from sensors closest to N_2's sensors are similar. It can also compare aggregated statistics (i.e., the average temperature over the past week is the same in both networks) for consistency.

3 DEVELOPING SHARING POLICIES

3.1 Valuing information

Indicators of whether to share or not when DIM users consider whether to share information or not, their concerns will be abstract. For example, if a user Bob shares his religious beliefs on an OSN, he may encourage communication from like-minded people (whom he may not otherwise realize are like-minded) and develop stronger relationships with them. On the other hand, he may also tickle the prejudices of others and elicit a negative reaction. The question Bob must answer is: will the benefit outweigh the cost?

To answer such questions, users would like to gather evi-dence that support a decision to share or not to share. Such evidence could take many forms:

Positive vs. negative: a decision to share can cause events that are either good or bad for the user.

Observed vs. provided: a user may be alerted to the ram-ifications of a decision either through observed evidence (e.g., losing friends in an OSN) or through soliciting re-actions directly from users (e.g., a posted article received many "thumbs-down" votes).

In-band vs. out-of-band: the application by which the information is shared (e.g., the OSN) can gather the evidence, or it may be acquired via some means outside the system.

Trustworthy vs. untrustworthy: The reputation of the evidence provider (whether a system, individual, or oth-erwise) may be taken into account when using it to assess value.

Creating policies that incorporate evidence requires evaluat-ing various forms of evidence. A first step is to identify useful evidence by observation of existing DIM use.[2] By gathering data of user activities over time, we can create a taxonomy of evidence, its sources, and effects in real-world usage to help guide user policies. We can also observe a user's behavior over time. If a user's activity increases, then we can infer that some positive value is being received from interaction. By observing many users' activities, we can identify events that correlate with various levels of site usage.

3.2 Economics-based metrics

To encourage productive sharing and discourage illicit in-formation release, we can apply ideas from markets. Placing monetary value on sensitive data enables a broader range of policies. When a user shares data with a site, they agree upon a price for the data. The price does not have to be a fixed value— it can represent recurring payments, for example, $1 per month that the user allows the site to use the data. Compensating users for their data gives them some recourse if data is leaked.

This approach is similar to the existing Internet economic model built around advertising. Sites share their resources (the attention of readers) with a third-party (the advertiser) by placing an ad on their site and receive compensation in return. The type of ad and amount of compensation varies depending on the composition and size of the audience. Advertisers pay the site either a fixed price for a given amount of space on a Web page, or on a per-impression/per-click model that depends on how often the ad is displayed or clicked on. We borrow concepts from this model and show how they apply to users sharing their resources (data) with a third-party (a hosting site).

Marketplaces and pricing: There are many possible meth-ods of data valuation. We envision a "data marketplace," much like a stock ticker, that reports on the current "going rate" of various pieces of information. When a user shares data with a site, the two parties enter into an agreement on what the data is worth and how the user will be compensated for it (possibly only in the case of leakage).

[2]One problem with such a data-based study on Facebook in particular is that user data is proprietary, and storing it, even when it is accessible, is a violation of the Facebook license agreement. The MySpace user data [12] archive may be useful in this regard

Payment schemes: We present three different possible pay-ment schemes: a one-time payment upon data transfer, a one-time payment upon data leakage, and a recurring payment (possibly depending on usage). One-time payment upon data transfer. In this scheme, the site pays the user the current going rate for each piece of data that is shared. Once the site has the data, it assumes ownership and may share the data with whomever it chooses (as it has no incentive to not share the data). One advantage of this system is that no moni-toring system is required to detect leakage. However, care must be taken to ensure the user-provided information is correct—a user could easily create many accounts and share bogus data to collect multiple payments.

4 ENFORCING POLICIES

4.1 THE PRINCIPLE OF LEAST SHARING

One way to avoid unauthorized release of information is not to share it in the first place. For example, rather than send private data to OSN servers which perform ad selection, an alternative would be for the user to run the OSN's ad selection algorithm on locally-stored data and provide the OSN with the result. If the choice of ad does not identify the user, the OSN learns less about the user but can still serve targeted ads.

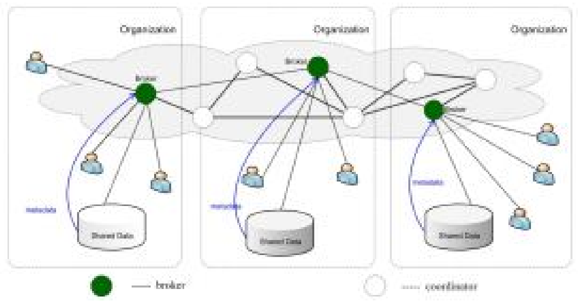

Fig 2. The architecture of PPIB.

4.2 Prior work

Prior work has taken two basic approaches to solving the privacy preserving computation problem, computation splitting

(CS) and secure multiparty computation (SMC). The first is exemplified by Jif/Split [15], [16] and its descendant, Swift [17], and the second by Fairplay [18], [19]. We describe each in turn and then consider generalizations of these ideas that we think are worth exploring.

Computation splitting We can protect x_1; :::; x_n and y_1; :::; y_m by splitting the computation between client and server, using static information flow analysis [15]. Essentially both sides will declare policies about their data, including declassifications that allow some information about the data to be shared in particular ways to the other party. For example, rather than releasing all of a credit card number to the remote party, the declassification may release only its last four digits. The static analysis ensures that any such declassifications take place before any secret data is sent to the remote party. Code that operates on non-secret data can be placed on either the client or server, as the code is not considered secret (we will consider altering this assumption shortly).

In the end, both parties learn the result of the computation and the secret values declassified by the other side. The declassified values may be implied by the result, or may convey additional information. As degenerate cases: If there are no x_1; :::; x_n to be protected, we can ship the entire code to the client to perform locally; conversely, if the client is willing to share the entirety of y_1; :::; y_m, then these can all be shipped to the server, and the computation can be performed there.

Secure multiparty computation Fairplay [18] requires no declassifications, but rather uses an interactive protocol and cryptographic techniques to protect data as it is shipped back and forth between client and server to compute F . It is more flexible than Jif/Split in that it ensures less information is released to the remote party, but also more expensive. Fairplay was originally developed for two-party computations, but has been generalized to n-way computations in FairplayMP [19].

4.3 Keeping F hidden from the client

Suppose the server wants to hide some or all of F . For example, suppose F is the OSN's ad selection algorithm. If the algorithm is very effective the OSN would prefer to keep it hidden from competitors. Most reliably, the entire computation would occur at the server, and as the computation proceeds, it would send read requests to the client, asking for bits of data it needs. However, this approach could reveal more data than the client desires. For example, suppose the site wishes to determine the average age of a user's friends. We might do this by defining F (u) for a given user u as follows:

```
n = 0; c = 0;

foreach f 2 friends(u)

n = n + 1; c = c + age(f); return c=n
```

Consider three possible ways this code could be computed between client and server:

1) F runs entirely at the server. It must query the client for each of u's friends f, so that it can average the friends' ages.

2) F runs entirely on the client. Thus the user releases far less data—just the average age of his friends, not the number of friends or their identities. However, running F at the client reveals the function the server wishes to compute—if u just provides his friends list to the server, he does not know what about his friends the site finds interesting, protecting the site's algorithm.

3) As a middle ground the server could ask for the num-ber of u's friends and their ages. Doing so does not reveal what the site will do with this, i.e., compute the average—the server could be computing the median, finding the maximum or minimum, etc.

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.5, Pages : 254 - 259 (2014)
Special Issue of ICACSSE 2014 - Held on October 10, 2014 in St.Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

Which division of computation is used is a matter of policy. The programmer could label portions of F as server-private, and the CS or SMC algorithm could ensure that none of that code runs at the client. For example, labeling all of F as private results in the first case, above; labeling none results in the second; labeling only the expression c=n results in the third.

A research challenge is to reconcile the constraints on code location imposed by privacy requirements of the client's data and the privacy requirements of the code. Moreover, even if

"all" the code runs at the server, the client can infer properties of the algorithm by observing individual read requests that it makes. We must ask: at a high level, what can the client reason about the overall form of F based only on seeing the portion of its computation that runs at the client? How do we describe knowledge of a particular function, based on the computation?

While hiding the code would seem to make the splitting problem harder, it actually can make it easier in some cases. In particular, the static analysis performed by CS algorithms is designed under the presumption that the client will know when it could be receiving server-private values (or functions of them) because it knows the code. If we assume that some or all of the code is unknown, the client may not know the difference between a query $y_1 < 5$ and $y_1 < x_1$ (where x_1 is a server-private value that could be 5), and thus will not realize it is learning something about private server data.

5 Related Work

5.1 Preliminaries

1) XML Data Model and Access Control: The eXtensi-ble Mark-up Language (XML) has emerged as the de fact standard for information sharing due to its rich semantics and extensive expressiveness. We assume that all the data sources in PPIB exchange information in XML format, i.e.take XPath [30] queries and return XML data. Note that the more powerful XML query language, XQuery, still uses Path to access XML nodes. In XPath, predicates are used to eliminate unwanted nodes, and test conditions are contained within square brackets "[ ]". In our study, we mainly focus on value-based predicates. To specify the authorization at the node level, fine-grained access control models are desired. We adopt the 5-tuple access control policy that are used in the literature [31], [27], [29].The policy consists of a set of access control rules (ACR) ={subject, object, action, sign, type}, where (1) subject is the role to whom the authorization is granted; (2) object is a setof XML nodes specified by an XPath expression; (3) action is operations as "read", "write", or "update"; (4) sign ∈ {+, −}refers to access "granted" or "denied", respectively; and (5)type ∈ {LC, RC} denotes "local check" (i.e., applying autho-rization only to the attributes or textual data of the contextnodes) or "recursive check" (i.e., applying authorization to althe descendants of the context node). A set of example rulesare shown below:

Example 2. Example ACRs:

R1:{role1,/site//person/name, read, +, RC}

R2:{role1,/site/regions/asia/item, read, +, RC}

R3:{role2,/site/regions/asia/item, read, +, RC}

R4:{role2,/site/regions/*/item/name, read, +,RC}

R5:{role2,/site/regions/*/item[location="USA"]/description,read,+, RC} ✷

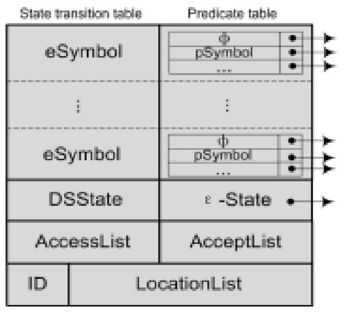

Fig. 3. Data structure of an NFA state.

Existing approaches for enforcing access control policy can be classified as engine-based [32], [25], [28], [33], [34],view-based [35], [36], [31], [37], [38], pre-processing [26], [27], [39], [40], [41], and post-processing [42] approaches. In particular, we adopt the Non-deterministic Finite Automaton(NFA) based approach as presented in [29], which allows access control to be enforced outside data servers, and independent from the data. The NFA-based approach constructs NFA elements for four building blocks of common XPathaxes (/x, //x, /*, and //*) so that XPath expressions, as combinations of these building blocks, can be converted toan NFA, which is used to match and rewrite incoming XPathqueries. Please refer to [29] for more details on the QFilterapproach.

2) Content-based Query Brokering: Indexing schemes havebeen proposed for content-based XML retrieval [43], [44],[45], [46]. The index describes the address of the data serverthat stores a particular data item requested in an user query.Therefore, a content-based index rule should contain thecontent description and the address. In [9], we presented acontent-based indexing model with index rules in the form ofI = fobject, locationg, where (1) object is an XPath expressionthat selects a set of nodes; and (2) location is a list of IPaddress of data servers that hold the content. When an userqueries the system, the XPath query is matched with the object field of the index rules, and the query will be sent to the data server specified by the location field of the matched rule(s). While other techniques (e.g. bloom filter [7], [6]) can be used to implement content-based indexing, we adopt the model in [9] in our study since it can be directly integrated with the NFA-based access control enforcement scheme. The integrated NFA that captures access control rules and index

rules is a content-based query broker (QBroker).

Example 3. Example index rules:

I1:f/site/people/person/name, 130.203.189.2gI2:f/site/regions//item[@id>"100"],135.176.4.56g I3:f/site/regions/samerica/item[@id>"200"],195.228.155.9gI4:f/site //namerica/item/name, 135.207.5.126gI5:f/site/regions/namerica/item/location,74.128.5.9 1g 2QBroker is an NFA constructed in a similar way as QFilter[29]. Fig. 3 shows the data structure of each NFA state in QBroker, where the state transition table stores the child nodes specified by the XPath expression as the child states in eSymbol. The binary flag DSState indicates that the state is a

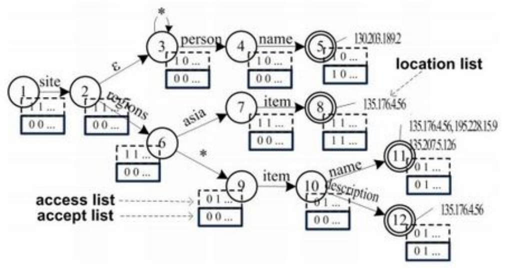

Fig. 4. The state transition graph of the QBroker that integrates index rules with ACRs.

5.2 Automaton Segmentation

In the context of distributed information brokering, multiple organizations join a consortium and agree to share the data within the consortium. While different organizations may have different schemas, we assume a global schema exists by aligning and merging the local schemas. Thus, the access control rules and index rules for all the organizations can be crafted following the same shared schema and captured by a global automaton, the global QBroker. The key idea of the automaton segmentation scheme is to logically divide the global automaton into multiple independent yet connected segments, and physically distribute the segments onto different brokering servers.

1) Segmentation: The atomic unit in the segmentation is an NFA state of the original automaton. Each segment is allowed to hold one or several NFA states. We further define the granularity level to denote the greatest distance between any two NFA states contained in one segment. Given a granularity level k, for each segmentation, the next i 2 [1; k] NFA states will be divided into one segment with a probability 1=k. Obviously, a larger granularity level indicates that each segment contains more NFA states, resulting in a smaller number of segments and less end-to-end overhead in distributed query processing. On the contrary, a coarse partition is more likely to increase the privacy risk. The tradeoff between the processing complexity and the privacy requirements should be considered in deciding the granularity level. As privacy protection is of the primary concern of this work, we suggest a granularity level

2. To reserve the logical connection between the segments after segmentation, we define heuristic segmentation rules: (1) multiple NFA states in the same segment should be connected via parent-child links; (2) no sibling NFA states should not be put in the same segment without the parent state; and (3) the "accept state" of the original global automaton should be put in separate segments. To ensure the segments are logically connected, we change the last states of each segment to be "dummy" accept states, which point to the segments holding the child states in the original global automaton.

Algorithm 1 The automaton segmentation algorithm: *deploySegment()*

```
Input: Automaton State $S$
Output: Segment Address: $addr$
 1: for each symbol $k$ in $S.StateTransTable$ do
 2:     $addr$=$deploySegment(S.StateTransTable(k).nextState)$
 3:     $DS$=$createDummyAcceptState()$
 4:     $DS.nextState \leftarrow addr$
 5:     $S.StateTransTable(k).nextState \leftarrow DS$
 6: end for
 7: $Seg = createSegment()$
 8: $Seg.addSegment(S)$
 9: $Coordinator = getCoordinator()$
10: $Coordinator.assignSegment(Seg)$
11: return $Coordinator.address$
```

Fig 5. The algorithm of deploy segment

5.3 Deployment:

We employ physical brokering servers, called coordinators, to store the logical segments. To reduce the number of needed coordinators, several segments can be deployed on the same coordinator using different port number to distinguish them. Therefore, the tuple ¡coordinator, port¿ uniquely identifies a segment. For the ease of the presentation, we assume each coordinator only holds one segment in the rest of the article. After the deployment, the coordinators can be linked together according to the relative position of the segments they store, and thus form a tree structure, where the

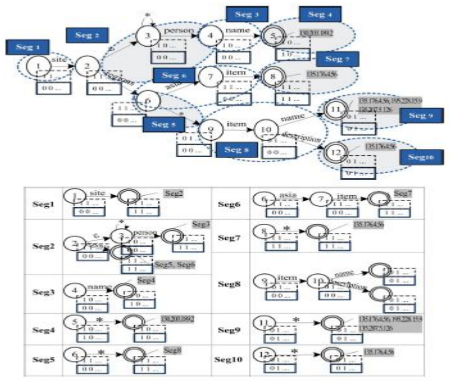

Fig . 6 An example to illustrate the automaton segmentation scheme: (a) divide the global automaton with granularity level of 1; (b) the segments are linked to form a tree structure.

6 SECURITY ANALYSES

6.1 Data confidentiality

The outsourced data are kept confidential. The data owner creates an encrypted version of the data file. The encryption of a file is done using a secret key K generated by owner, where it can be accessed by only owner and authorized users.

6.2 Integrity and newness

Integrity and newness properties are preserved using techniques like hashing and lazy revocation. Where hashing techniques preserve the integrity of the data and enables the parties to detect corruption. Lazy revocation will ensure the data newness. It allows the owner to revoke the right of some users for accessing the outsourced data and also allows the revoked users to read unmodified data blocks. However the updated blocks must not be accessed by such users.

6.2 Detection of data corruption

Due to hashing techniques that are followed in this scheme at all the parties the data cannot be corrupted on cloud servers without being detected. During the data access phase of the proposed scheme, the authorized user receives the encrypted file from the CSP and FH_{TTP} from the TTP. The authorized user computes a hash for the received file and compares it with one received form the TTP. If both the computed hash and hash received from the TTP are not matched then file has been corrupted on the server. For violating data integrity without being detected the CSP hast to send a file F' which is not the original file uploaded by the owner but

ISSN 2278-3091

International Journal of Advanced Trends in Computer Science and Engineering, Vol.3 , No.5, Pages : 254 - 259 (2014)
Special Issue of ICACSSE 2014 - Held on October 10, 2014 in St.Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

their hashes must match. Due the the one way nature of the hashing technique the CSP cannot generate such a file.

6.3 Enforcement of access control

The owner creates a secret key K only authorized users will know. With which they decrypt the file to read data, and thus the access control is achieved in the proposed scheme.

6.4 Cheating detection of dishonest party

If the owner falsely accuses the CSP regarding data integrity, the TTP performs cheating detection. In this procedure TTP retrieves the encrypted file from the CSP and computes hash and compares with stored hash, if they match then owner/ user is the dishonest party, else CSP is the dishonest party.

7 CONCLUSIONS

This paper has presented an overview of the problem of securely sharing information among mutually-distrusting par-ties in a distributed system, along with directions of research toward solving the problems that arise. By examining online social networks in particular and community-based review and sensor network applications in general, we describe factors that should be taken into consideration when crafting information sharing policies, namely evidence of reactions to prior sharing efforts and economic incentives to abide by policy. We also discuss mechanisms for enforcing such policies, with a focus on collaborative computation, beginning with the ideas of secure multiparty computation and computation splitting. We propose that these mechanisms be augmented with means to keep sensitive code hidden, to ensure that collaborative computation is consistent and efficient, and to quantitatively track knowledge about private information that can be inferred from the results of computations. We believe we have sketched a rich agenda for research ahead at the cross-section of pro-gramming languages, cryptography, economics, and systems.

REFERENCES

[1] "Facebook statistics," http://www.facebook.com/press/info.php? statistics, May 2010.
[2] C. Weber, "Pentagon allows social networking on military computers," Politics Daily (on-line), Mar. 2010, http://www.politicsdaily.com/2010/ 03/02/pentagon-allows-social-networking-on-military-computers/.
[3] "Dod social media hub," http://socialmedia.defense.gov, May 2010.
[4] "Birthdaycalendar," http://www.facebook.com/BirthdayCal.
[5] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Per-sona: an online social network with user-defined privacy," in SIGCOMM, 2009.
[6] "Slashdot," http://slashdot.org, May 2010.
[7] "Reddit," http://www.reddit.com, May 2010.
[8] Reuters, "U.S. intelligence unveils spy version of Wikipedia," October 2006.
[9] "Wikipedia," http://en.wikipedia.org, May 2010.
[10] "Four-star blogging at STRATCOM," http://www.defenseindustrydaily. com/fourstar-blogging-at-stratcom-0239/, Mar. 2005.
[11] M. Anand, E. Cronin, M. Sherr, M. Blaze, Z. Ives, , and I. Lee, "Sensor network security: More interesting than you think," in HotSec, 2006.
[12] D. Worthington, "Myspace user data for sale," PC World on-line, Mar. 2010, http://www.pcworld.com/article/191716/myspace user data for sale.html.
[13] D. Dukes, "Facebook causes Barrow teacher's firing," Fox 5 Atlanta (online), Nov. 2009, http://www.myfoxatlanta.com/dpp/news/facebook+ causes+barrow+teacher's+firing+111009.
[14] M. Srivatsa, S. Balfe, K. G. Paterson, and P. Rohatgi, "Trust management for secure information flows," in CCS, 2008.
[15] S. Zdancewic, L. Zheng, N. Nystrom, and A. C. Myers, "Untrusted hosts and confidentiality: Secure program partitioning," in SOSP, 2001.
[16] L. Zheng, S. Chong, A. C. Myers, and S. Zdancewic, "Using replication and partitioning to build secure distributed systems," in Security and Privacy, 2003.
[17] S. Chong, J. Liu, A. C. Myers, X. Qi, K. Vikram, L. Zheng, and X. Zheng, "Secure web applications via automatic partitioning," SIGOPS Oper. Syst. Rev., vol. 41, no. 6, pp. 31–44, 2007.
[18] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay—a secure two-party computation system," in SSYM, 2004.
[19] A. Ben-David, N. Nisan, and B. Pinkas, "FairplayMP: a system for secure multi-party computation," in CCS, 2008.
[20] M. Backes, B. Kopf,¨ and A. Rybalchenko, "Automatic discovery and quantification of information leaks," in Security and Privacy, 2009.
[21] M. R. Clarkson, A. C. Myers, and F. B. Schneider, "Quantifying information flow with beliefs," J. Comput. Secur., vol. 17, no. 5, pp. 655–701, 2009.
[22] S. Hansell, "AOL removes search data on vast group of web users," New York Times, Aug. 2006.
[23] N. Prize, http://www.netflixprize.com/.
[24] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in Security and Privacy, 2008.
[25] P. Cousot and N. Halbwachs, "Automatic discovery of linear restraints among variables of a program," in POPL, 1978, pp. 84–96.
[26] P. Cousot and R. Cousot, "Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints," in POPL, 1977, pp. 238–252.
[27] P. Golle, "Revisiting the uniqueness of simple demographics in the us population," in WPES. New York, NY, USA: ACM, 2006, pp. 77–80.
[28] C. Dwork, "Differential privacy," in ICALP, 2006.

AUTHORS:

T.BEULA received the B.TECH degree in Computer Science & Engineering from JNTU Kakinada in 2012 & Pursing her M.TECH in Computer Science & JNTU Kakinada..

Eswar Kodali is presently working as an Associate Professor in Department of Computer Science and Engineering, in St. Ann's College of Engineering and Technology, Chirala. He guided many UG an PG students. He has more than 10 years of excellence in teaching. He published various.