



RESEARCH ARTICLE

Encrypted Query Processing for Improved Protection of Confidentiality

Sinu P S¹, M.Ananthi²

¹PG Scholar, Info Institute of Engineering, India

²Assistant Professor, Department of CSE, Info Institute of Engineering, India

sinu.sreyas@gmail.com

mail2ananthiinfo@gmail.com

Abstract— A log is a collection of record of the events that occurs within an organization containing systems and networks. Logs are being composed of entries which are of its own syntax; each log entry has information that are related to a specific event which has been occurred inside a system or network. Actually, logs are used basically for problems like troubleshooting, but at present logs serve many functions almost in all organizations, for optimizing performance of the system and network, for recording all the actions of users, and for providing useful data for malicious activity investigation. Logs have been in use for containing information that are related to various forms of events that are occurring in the networks and systems. Inside an organization, there are much logs which do contain records that are related to the security of the system; some common examples of these computer security logs are logs that are related to audit that contains the track of user authentication attempts and logs of security device that record the possible types of attacks. In this paper, we focus on the challenges for a secure cloud-based log management service and do propose a framework for doing the above.

Keywords— Cloud computing, logging, privacy, security

I. INTRODUCTION

There are a number of approaches have been proposed for logging information in computing systems. Among these most of the approaches are based on syslog based environment which is the standard for network based logging protocol. The protocol used by syslog is UDP for transferring log information to the log server. Therefore there is no reliable delivery of log messages. The major disadvantage of syslog is it does not protect log records during transmit or at the point-point transfer. Syslog-ng is the replacement for backward compatible with syslog. Some of the features include supporting IPv6 protocol, capability to transfer log messages with reliability using TCP, and filtering the content of logs using regular expressions. Syslog-ng describes encryption of log record using SSL while transmission such that it protects the data from confidentiality and integrity precluding while in transmits. However, syslog-ng does not protect against log data modifications when it is placed at an end-point. The next concept is Syslog-sign which is an enhanced form of syslog that adds origin of authentication, message integrity, replay resistance, message sequencing, and detection of missing messages by using two additional messages—“signature blocks” and “certificate blocks.” Unfortunately, if signature blocks are associated with the log records they get deleted after authentication, which tamper the evidence and forward integrity is only partially fulfilled. Syslog-sign also do not provide any confidentiality or privacy during the transmission of data or at the end-end transmission. Another concept of syslog is Syslog-pseudo which is also an enhancement of syslog that proposes a logging architecture for pseudonymizing log files. The main

idea behind this is that the log records are first processed by the pseudonymizer before being archived. The pseudonymizer filters it out identifying the features from the specific fields that are in the log record and substitutes them with the ones that are carefully crafted with pseudonyms. Therefore, strictly speaking, this protocol do not ensure the correctness of the logs i.e., these log records which are stored in the log are not the same as the ones that are being generated. The another problem with this paper is that when the protocol anonymizes each log record individually it do not protect the log records from the attacks that try to correlate a number of anonymized records. The Anonymous log file anonymizer also performs a similar anonymization for identifying the information therefore by substituting with the default values or by one or more coarse values. However, this problem is with this paper is that if the original values are needed in the investigation, which cannot be restored.

Neither syslog-pseudo nor anonymous log file anonymizer protects the log records from the security attacks i.e., confidentiality and integrity violations and the other end-point attacks. Reliable-syslog aims to implement reliable delivery of syslog messages, which is built on the top of the blocks of extensible exchange protocol (BEEP) that runs over TCP for providing the required reliable delivery service. The Reliable-syslog protocol also allows the device authentication and incorporates its mechanisms to protect the integrity of log messages and to protect the log messages against replay attacks and from the other attacks that affect the log data; however this too do not prevent against confidentiality or privacy preclusions at the end-end transmission or during transmit.

The notion for a forward-integrity log record was proposed by Bellare and Yee for protecting the precompromising of log data from post compromising insertion, deletion, modification, and reordering. Forward integrity is being established by the usage of secret key mechanism that becomes the starting point of a hash-chain. The hash-chain is being generated by a cryptographically strong one-way function where the key is being changed for each and every log record.

Schneier and Kelsey proposed a logging scheme for cloud that relays on forward integrity and assures it. This scheme is mainly based upon the forward-secure message authentication codes and one-way hash chaining similar to that suggested by the Bellare-Yee protocol i.e., if the trusted server is being attacked or being compromised, it breaks the security of the logging scheme.

Holt improves the Schneier-Kelsey protocol by merging the public verifiability log records. However, this scheme being a public-key based scheme, the overhead is found to be significantly more. None of these three schemes consider the privacy concerns of storing and retrieving log records. In addition to all these three schemes suffer equally from truncation attacks where an attacker deletes the contiguous subset of log records from the very end it is being kept.

II. RELATED WORK

The following have been analysed and studied in order to delegate Log Management. K. Kent and M. Souppaya [1] proposed as: Log management be benefited in an organization in many ways. It helps to ensure that computer security records that are being stored in sufficient detail for an appropriate period of time. Routine log reviews and analysis are beneficial for identifying the following factors: security incidents, policy violations, fraudulent activity, and operational problems which are shortly carried out after they have occurred, and for providing information useful for resolving such problems. Logs can also be useful for performing auditing and forensic analysis, supporting the organization's internal investigations, establishing baselines, and identifying operational trends and long-term problems. Besides the benefits of log management, a number of laws and regulations further compel organizations to store and review certain logs. Most of the organizations face similar log management-related challenges, which do undergo the same underlying problem: effectively having a balanced and a limited amount of log management resources with an ever-increasing supply of log data.

D. New and M. Rose [2] stated in their work as: The BSD Syslog Protocol prescribes a number of service related options and also relate to propagating event messages. This memo also describes the two mappings of the syslog protocol to TCP connections, both which are useful for transferring reliable delivery of event messages. This provides a trivial mapping maximizing backward compatibility and also helps in providing a more complete mapping. Both provide a degree of robustness and security in message delivery that is unavailable to the usual UDP-based syslog protocol, by providing encryption and authentication over a connection-oriented protocol.

M. Bellare and B. S. Yee [3] et al discussed as: Applications include more secure audit logs (e.g., syslogd data) for intrusion detection or accountability, communications security, and authenticating partial results of computation for mobile agents. Computer audit logs contain descriptions of noteworthy events which crashes of system programs, system resource exhaustion, failed login attempts, etc. Many of these events are critical for post-mortem analysis after a break-in. The rst target of an experienced attacker will be the audit log system: the attacker wishes to erase traces of the compromise, to elude detection as well as to keep the method of attack secret so that the security holes exploited will not be detected and by the system administrator. To make the audit log secure, we must prevent the attacker from modifying the audit log data.

J. E. Holt [4] projected as: The popular application Tripwire keeps cryptographic fingerprints of all files on a computer allowing administrators to detect when attackers compromise the system and modify the important system files. But Tripwire is unsuitable for system logs and other files that change often, since the fingerprints creates apply to files in their entirety. Several people have proposed cryptographic systems which allow each new log entry to be fingerprinted, preventing attackers from removing evidence of their attacks from system logs.

B. Schneier and J. Kelsey [5] concentrated on the following: In many real-world applications, sensitive information are kept in logs which are less on an untrusted machine. When an event occurs as an attacker captures this machine, it is guaranteed that the attacker will gain little or no information from the log less and to limit his ability to corrupt the log files. Here the proposed system describes a computationally cheap method for making all log entries generated prior to the logging machine's compromise impossible for the attacker to read and also impossible to undetectably modify or destroy. A computer that uses logs of various kinds of network activity needs to have log entries of an attack undeletable and unalterable, even in the event that an attacker takes over the logging machine over the network. An intrusion-detection system that logs the entry and exit of people into a secured area needs to resist attempts to delete or alter logs, even after the machine on which the logging takes place has been taken over by an attacker.

D. Ma and G. Tsudik [6] discussed as :The need for secure logging is well-understood by the security professionals, including both researchers and practitioners. The ability to efficiently verify all log entries is more important to any application employing secure logging techniques. In this paper, we begin by examining state-of-the-art in secure logging and identify some problems inherent to systems based on trusted third-party servers. They propose a very different approach to secure logging based upon recently developed Forward-Secure Sequential Aggregate (FssAgg) authentication techniques.

D. Dolev and A. Yao [7] et al proposed as: Recently the use of public key encryption was to provide a secure network communication which has received a considerable attention. Such public key systems are usually effective in providing against the passive eavesdroppers, who usually try to tap the lines and try to decipher the message. It has been pointed out, that an improper designed protocol could be vulnerable to an active behaviour like that, one who may impersonate another user or alter the message being transmitted. Several models have been formulated in which the securities of protocols are discussed precisely. Algorithms and characterizations are used in determining protocol security in these models which have been given in. The use of public key encryption was to provide a secure network communication which has received considerable attention.

G. R. Blakley [8] suggested as: certain cryptographic keys, has as a number of components which makes a possible components to compute the secret decoding exponents in an RSA public key cryptosystem 1,5 or the system master key and certain other keys in a DES cryptosystem , 3 are important as they present a dilemma. If too more copies are distributed then one might go astray. If too few copies are made they all might be destroyed. There are two principles for counting incidents.

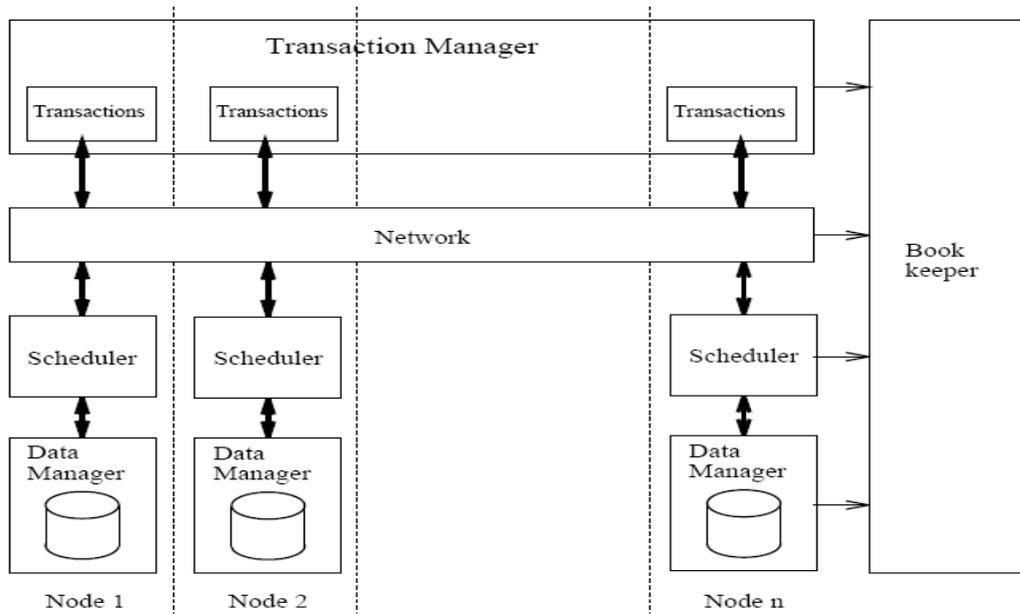
- The first incident is Boole's law of inclusion and exclusion.

Suppose that an organization issues are non-volatile the pieces of information guards and waits with a modest period of time during which incidents can occasionally occur. Let a stand for the number of abnegation incidents, b for the number of betrayal incidents and c for the number of combination incidents. The total number d of incidents is $d=a+b-c$ because a combination incident gets counted twice, once by a and once by b .

- The second principle is that the incidents are so rare, such that the possibility of two separate incidents occurring with the same non-volatile piece of information is usually dismissed on probabilistic grounds as absurd within the range.

A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung [9] discussed the following: Secret sharing schemes were used to protect the secrets by distributing them over different locations. In particular k out of n threshold schemes ,security is assured if the entire life time is secret and therefore its adversary and is restricted to compromise less than k of the n location for long lived and sensitive secret protection is insufficient .Here they propose an efficient proactive secret sharing scheme, where the shares are periodically renewed in such a way that information gained by the adversary in one period of time is useless for attacking the secret after the shares have been renewed.

K. Nørvag, O. Sandsta [10] et al concentrated on simulating distributed database systems as a much difficult system, where there are many factors which may influence the results. This paper includes architectural options as well as distribution of workload and data. In this paper they present a DBsim simulator and some simulated results. The DBsim simulator architecture is easily extendible, and it is easy for changing parameters and configuration. The simulated results in this paper is a comparison of both performance and response times for algorithms such as two concurrency control , timestamp ordering and two-phase locking algorithms. The simulations have been running in more number of nodes, network types, data declustering and workloads. The results shown is a mix of small and long transactions, the throughput generated is significantly higher for a system when compared with a timestamp ordering scheduler than for a system with a two-phase locking scheduler. There are only short transactions, and therefore the performance of the two schedulers are almost identical when compared with results. The Long transactions are treated more fairly by a two-phase locking scheduler, because a timestamp ordering scheduler has higher abort rate for long transactions. When Compared to a centralized database schema distributed database system is much more complex in nature. The simulated model used in the centralized simulator is flexible in nature to simulate the most important aspects of a distributed database system.



2.1 The architecture of the simulator.

This is the model simulator for a distributed page-server based object-oriented database system. This simulator is modular and extendible, and in this paper they have given results from simulations with two different scheduler strategies. In further work for the DBsim simulator can include extensions that could make the simulator more suitable for simulation of algorithms for object-oriented databases. Therefore, much more can be done with both the simulation model and the simulator. This includes adding new schedulers to the system, e.g., other versions of the two-phase locking scheduler, like wound-wait and wait-die. In a real system, replication is used for increased reliability and performance. This could also be integrated into this framework. More information about the data structure is put into the model, facilitating the use of more advanced scheduling algorithms. One particularly interesting scheduler strategy is multi granularity locking, which is a) easy to implement and b) efficient, and able to give a more considerable improved performance.

III. MATERIALS AND METHODS

The Log Generators, Logging Client or Logging Relay, Logging Cloud and Log Monitor .Create those file and connect to form a cloud.

The protocol starts with the logging client randomly generating three master keys— A_0 and X_0 for ensuring log integrity, and K_0 for ensuring log confidentiality. These keys need to satisfy the requirements of the chosen proactive secret-sharing scheme. It then embarks upon preparing the log records. Log data arrives at the logging client as a series of messages L_1, L_2, \dots, L_n . Each L_i contains a group of log records generated by a log generator. We assume that these messages are transmitted to the logging client over an authenticated network. The logging client uploads prepared log records in batches of n . The value n is determined randomly at the beginning of each log batch preparation.

1. Before any log data arrives at the logging client, the logging client creates a special first log entry $L_0 = \{\text{TS, log-Initialization, } n\}$. It then encrypts this log entry with the key K_0 and computes the message authentication code $\text{MAC}_0 = H_{A_0} [E_{K_0}[L_0]]$ for the encrypted entry with the key A_0 . The client adds the resulting first log entry for the current batch— $\{E_{K_0}[L_0], \text{MAC}_0\}$ —to the log file.
2. The logging client then computes new set of keys $A_1 = H[A_0]$, $X_1 = H[X_0]$ and $K_1 = H[K_0]$, securely erases the previous set of keys and waits for the next log message to arrive.
3. When the first log message L_1 arrives, the logging client creates a record $M_1 = L_1 \parallel H_{A_0} [E_{K_0} [L_0]]$. It encrypts M_1 with the key K_1 , and creates a message authentication code for the resulting data as $\text{MAC}_1 = H_{A_1} [E_{K_1}[M_1]]$. It also computes an aggregated message authentication code $\text{MAC}'_1 = H^{n}_{X_1}[\text{MAC}_0 \parallel \text{MAC}_1 \parallel n]$. The log batch entry is $\{E_{K_1} [M_1], \text{MAC}_1\}$. It then creates the next set of keys $A_2 = H[A_1]$, $X_2 = H[X_1]$ and $K_2 = H[K_1]$ and securely deletes A_1 and K_1 .
4. For every new log data L_i that the logging client subsequently receives, it creates log file entries $\{E_{K_i} [M_i], \text{MAC}_i\}$, where $M_i = L_i \parallel \text{MAC}_{i-1}$ and $\text{MAC}_i = H_{A_i} [E_{K_i} [M_i]]$. It also creates the aggregated message authentication code $\text{MAC}'_i = H^{n-i+1}_{X_i} [\text{MAC}_{i-1} \parallel \text{MAC}_i] \parallel n - i + 1$. Once MAC'_i has been generated, MAC'_{i-1} is securely deleted. The client finally creates new keys $A_{i+1} = H[A_i]$, $X_{i+1} = H[X_i]$ and $K_{i+1} = H[K_i]$ and securely erases the keys A_i , X_i and K_i .
5. After the client creates the last log entry M_n for the current batch from the last log data L_n , it creates a special log close entry $LC = \{E_{K_{n+1}}[\text{TS, log-close} \parallel \text{MAC}_n], H_{A_{n+1}} [E_{K_{n+1}}[\text{TS, log-close} \parallel \text{MAC}_n]]\}$, and an aggregated message

authentication code MAC'_{n-1} . It then securely erases the three keys used in this step and uploads the resulting log batch and aggregated message authentication code to the logging cloud as one unit.

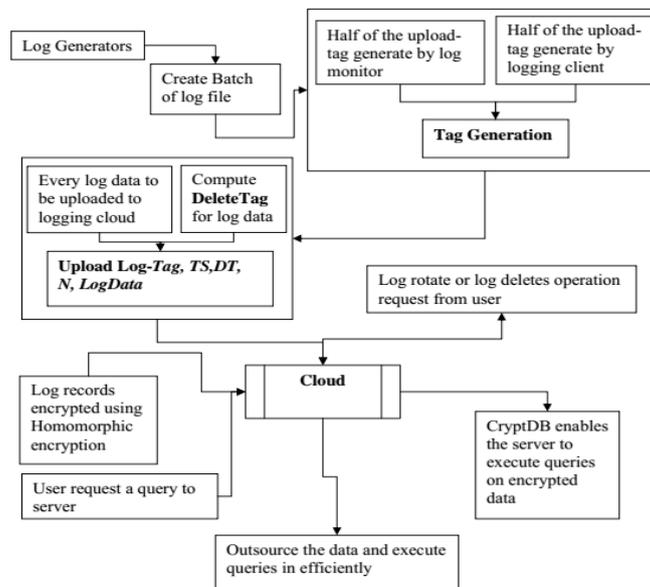
Tor is free software for enabling online anonymity. Tor directs Internet traffic through a free, worldwide volunteer network consisting of more than three thousand relays to conceal a user's location or usage from anyone conducting network surveillance or traffic analysis. Using Tor makes it more difficult to trace Internet activity, including "visits to Web sites, online posts, instant messages and other communication forms", back to the user and is intended to protect users' personal privacy, freedom, and ability to conduct confidential business by keeping their internet activities from being monitored.



Fig 3.1 - A Tor non-exit relay with a maximum output of 239.69 KB/s

IV. PROPOSED SYSTEM

The main function of CryptDB's proxy is to store a secret master key MK, the database schema, and the current encryption layers of all columns. The DBMS server gets to see an anonymized schema (where the table and column names are being replaced by the opaque identifiers), encrypted user data, and some auxiliary tables used by CryptDB. CryptDB also helps the server with CryptDB-specific user-defined functions (UDFs) which enables the server to compute it on cipher texts for certain operations.



4.1 The Overall Architecture of a System

Processing a query in CryptDB involves four steps:

1. The application issues a query, which the proxy intercepts and rewrites: it anonymizes each table and column name, and, using the master key MK, encrypts each constant in the query with an encryption scheme best suited for the desired operation.
2. The proxy checks if the DBMS server should be given the keys to adjust the encryption layers before executing the query, and if it is so, issues an UPDATE query at the DBMS server which invokes a UDF for adjusting the encryption layer to the appropriate columns.
3. The proxy helps in forwarding the encrypted query to the DBMS server, where the query is being executed using the standard queries.
4. The DBMS server helps in returning the (encrypted) query result, where the proxy decrypts and returns the decrypted content to the application. Homomorphic encryption (HOM) is a secure probabilistic encryption scheme (IND-CPA secure), which allows the server to perform the required operations on the encrypted data with the final result being decrypted at the proxy.

Advantages:

- Dynamically adjusting the encryption level is done by using much of encryption for minimizing the information that are revealed to the untrusted DBMS server.
- Highly secure encryption schemes.
- Reduce the communication overhead that happens between a log monitor and the logging cloud which is needed to answer queries on logs.

V. RESULTS AND DISCUSSIONS

Fig.5.1 shows to understand the amount of information that would be revealed to the adversary in practice, we examine the steady-state onion levels of different columns for a range of applications and queries. To quantify the level of security, we define the MinEnc of a column to be the weakest onion encryption scheme exposed on any of the onions of a column when onions reach a steady state (i.e., after the application generates all query types, or after running the whole trace). In the graph X-axis represents the files range from 0 to 9 and Y-axis represents security level of the system. From the results our proposed system is efficient than the existing techniques.

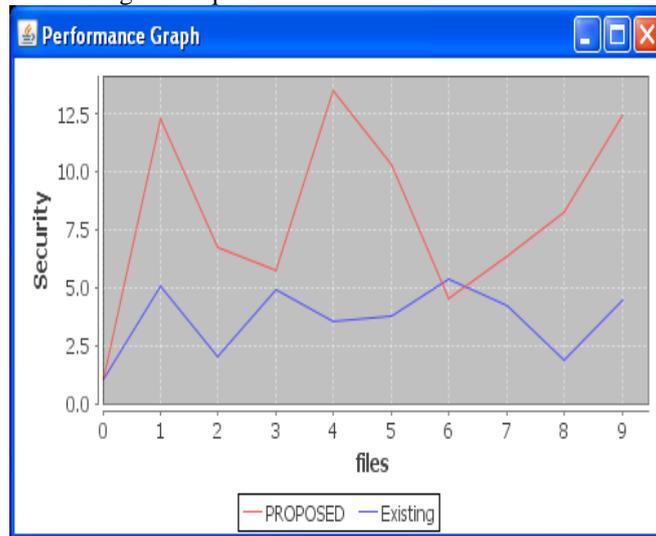


Fig.5.1 Security Graph

The Fig.5.2 shows the computation time of existing and proposed algorithm. In that graph X-axis represents the files range from 0 to 9 and Y-axis represents the computation time for run a algorithm in varies from 0 to 100 seconds. From the results the proposed algorithm runs faster than the existing algorithm.

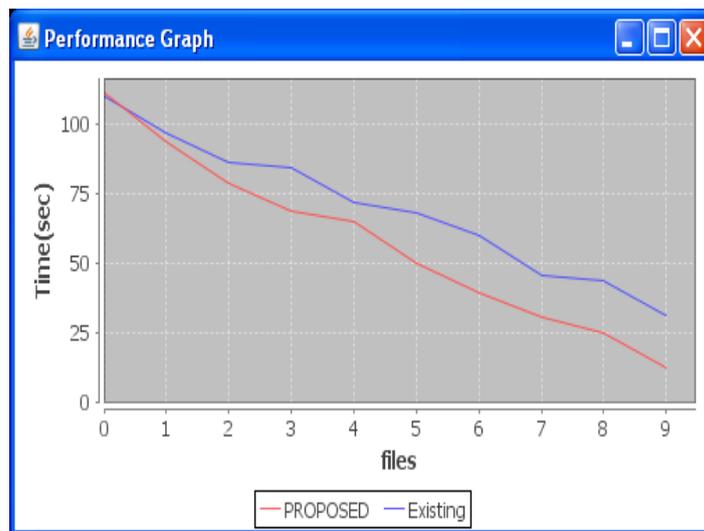


Fig 5.2. Time Graph

VI. CONCLUSIONS

Here we propose a homomorphic encryption schemes for encrypting log records. In that case the logging cloud can execute some queries on the encrypted logs without excluding confidentiality or privacy. A system that provides a practical and strong level of confidentiality in the facing two significant threats which confronts database-backed applications: curious DBAs and arbitrary compromises of the application server and the DBMS. CryptDB meets its aims using three ideas: running queries efficiently over encrypted data by using a novel encryption strategy, dynamically helps in adjusting the encryption level by using more number of encryption schemes to minimize the information that are being revealed to the untrusted DBMS server, and by chaining the encryption keys to user passwords in such a way that it allows only authorized users to gain access to the encrypted data.

The current implementation of the logging client is built by loosely coupling it with the operating system . In the future, we plan to refine the log client implementation as tightly integrated with the OS such that it replaces the current logging process.

REFERENCES

- [1] Dieudonne Mulamba, Indrajit Ray, Mariappan Rajaram, Mikhail Strizho "Secure Logging As a Service—Delegating Log Management to the Cloud" IEEE systems journal, vol. 7, no. 2, june 2013.
- [2] K. Kent and M. Souppaya. " Guide to Computer Security Log Management", NIST Special Publication 800-92. sep.2006
- [3] PCI Security Standards Council. *Payment Card Industry (PCI) Data Security Standard— Requirements and Security Assessment Procedures Version 2.0* Oct 2010.
- [4] Sarbanes-Oxley Act 2002." *A Guide to the Sarbanes-Oxley Act*" Sep. 2002.
- [5] C. Lonvick, "The BSD Syslog Protocol", Request for Comment RFC 3164, Internet Engineering Task Force, Network Working Group, Aug. 2001.
- [6] D. New and M. Rose, "Reliable Delivery for Syslog", Request for Comment RFC 3195, Internet Engineering Task Force, Network Working Group, Nov. 2001
- [7] M. Bellare and B. S. Yee, "Forward integrity for secure audit logs," Dept.Comput. Sci., Univ. California, San Diego, Tech. Rep., Nov. 1997.
- [8] Bellare, R. Canetti, and H. Krawczyk." Keying hash functions for message authentication". In N. Kobnitz, editor, *Advances in Cryptology: Crypto '96 Proceedings*, volume 1109 of *Lecture*.
- [9] D. Ma and G. Tsudik, "A new approach to secure logging," *ACM Trans. Storage*, vol. 5, no. 1, pp. 2:1–2:21, Mar. 2009.
- [10] Swanson, M., Guttman, B.:" Generally accepted principles and practices for securing information technology systems". In: NIST 800-14. 1996
- [11] Bellare, M., Yee, B.: "Forward integrity for secure audit logs". In: Technical Report, Computer Science and Engineering Department, University of San Diego. Nov. 1997
- [12] Bellare, M., Yee, B.: "Forward-security in private-key cryptography". In: Proc. of CT-RSA'03.

- [13] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inform. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.
- [14] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. Nat. Comput. Conf.*, p. 313 Jun. 1979.
- [15] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *Proc. 15th Ann. Int. Cryptology Conf.*, pp. 339–352. Aug. 1995
- [16] K. Nørøvåg, O. Sandst^oa, and K. Bratbergsengen, "Concurrency control in distributed object oriented database systems," . *Symp. Adv. Databases Inform. Syst.*, , pp. 32–32 Sep. 1997.
- [17] B. Schneier and J. Kelsey, "Security audit logs to support computer forensics," *ACM Trans. Inform. Syst. Security*, vol. 2, no. 2, pp. 159–176.
- [18] R. Anderson, "Robustness Principles for Public Key Protocols," *Advances in Cryptology \CRYPTO '95*, Springer-Verlag, pp. 236 ,247 1995.
- [19] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
- [20] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. Nat. Comput. Conf.*, Jun. 1979, p. 313.
- [21] R. Ostrovsky and M. Yung, "How to withstand mobile virus attack," in *Proc. 10th Ann. ACM Symp. Principles Distributed Comput.*, Aug. 1991, pp. 51–59.
- [22] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *Proc. 15th Ann. Int. Cryptology Conf.*, Aug. 1995, pp. 339–352.
- [23] I. Teranishi, J. Furukawa, and K. Sako, "*k*-times anonymous authentication(extended abstract)," in *Proc. 10th Int. Conf. Theor. Appl. Cryptology Inform. Security*, LNCS 3329. 2004, pp. 308–322.
- [24] D. L. Wells, J. A. Blakeley, and C. W. Thompson, "Architecture of an open object-oriented database management system," *IEEE Comput.*, vol. 25, no. 10, pp. 74–82, Oct. 1992.
- [25] K. Nørøvåg, O. Sandst^oa, and K. Bratbergsengen, "Concurrency control in distributed object oriented database systems," in *Proc. 1st East-Eur. Symp.*
- [26] R. Droms, *Dynamic Host Configuration Protocol*, Request for Comment RFC 2131, Internet Engineering Task Force, Network Working Group, Mar. 1991.
- [27] Project: AN.ON—Anonymity Online. (2012, Mar.). *JAP Anonymity and Privacy* [Online]. Available: <http://anon.inf.tu-dresden.de/index-en.html>
- [28] Ultrareach Internet Corporation. (2012, Mar.). *Ultrareach—Privacy, Security, Freedom* [Online]. Available: <http://ultrasurf.us/index.html>
- [29] Global Internet Freedom Consortium. (2012, Mar.). *FreeGate* [Online]. Available: <http://www.internetfreedom.org/FreeGate>