Partial Homomorphic Encryption for Secure Log Management Using Tor Network

Mr.M.Rathinraj
Assistant Professor, Department of Computer Science & Engineering,
Dr. S.J.S Paul Memorial College of Engineering and Technology, Puducherry – 605 502

Miss. J.Ramya Rajalakshmi
[#2]*Master of Technology, Department of Computer Science & Engineering,*
Dr. S.J.S Paul Memorial College of Engineering and Technology, Puducherry – 605 502

Miss. M. Saranya
[#3]*Master of Technology, Department of Computer Science & Engineering,*
Dr. S.J.S Paul Memorial College of Engineering and Technology, Puducherry – 605 502

Abstract

Logging is closely related to Forensic Computing. Log files helps cyber forensic process in probing and seizing computer, obtaining electronic evidence for criminal investigations and maintaining computer records for the federal rules of evidence. To make the logs useful for the use in court, there is a necessity to prove that the logs have not been modified after being generated. Moreover, since the logs contain confidential information, they must be protected strictly. Therefore a secure logging scheme that ensures the integrity and confidentiality of the logs is needed. Partial Homomorphic Encryption is proposed for implementing secure log management system. Homomorphic Encryption systems are used to perform operations on encrypted data without knowing the private key (without decryption) the client is the only holder of the secret key. Tor Network improves the privacy and security of log data while transmission.

Key Terms: Logging, Log Management, Homomorphic Encryption, Partial Homomorphic encryption, Tor Network.

1. Introduction

Log is a record of events occurring when a user is active and used for statistical purposes as well as backup and recovery. Log files are registered by the operating system or other control program for recording the details about incoming dialogs, error and status messages and certain transaction. Start and stop times of routine jobs may also be recorded. Any program might generate a log file. Logging is the processes of collecting the log files [1]. An application may generate a log that the user can refer to if necessary or that may be helpful in the event of a failure. For example, an FTP program may generate a log file showing the date, time and source and destination paths for each file transferred. In Windows, the log files are stored in (Control Panel/System and Security/Administrative Tools/Event Viewer).In Fedora, the log files are stored in (Computer/Var/Log).

Log management is the process of generating, transmitting, analyzing, storing and disposing of computer security log data. Log management [6] (LM) comprises an approach to dealing with large volumes of computer generated log messages (also known as audit records, audit trails, event-logs, etc). Log management is driven by reasons of security, system and network operations (such as system or network administration) and regulatory compliance. Log management defines what you need to log, how to log it and how long to retain this information. And, it is important to carry out the log management process in a secure manner.

System logs are an important part of any secure IT system. They record significant events happened in the past such as user activity, program execution status, system resource usage, data changes, etc. They provide a valuable view of the past and current states of almost any type of complex system. In conjunction with appropriate tools and procedures audit logs can be used to enforce individual accountability, reconstruct events, detect intrusions and identify problems. Because of their forensic value, system logs are an obvious target for attackers. An attacker who gains access to a system naturally wishes to remove traces of her presence in order to hide attack details or frame innocent users. The first target of an experienced attacker would often be the logging system. To make the audit log secure, the log data must be prevented from modification by the attacker. Secure versions of audit logs should be designed to defend against such damages.

Log data [5] can only be used if it is "correct". Log management can generate reports that are helpful for system maintenance [2], security management and many other purposes. Uses for log data include marketing, forensics and hr accounting. As they identify goals, companies would do well to consider the broader advantages of log management and analysis and look for systems or services that will allow a migration toward a more complete use of log data in the future.

Tor [7] is a network of virtual tunnels that allows people and groups to improve their privacy and security on the Internet. It also enables software developers to create new communication tools with built-in privacy features. Tor allows organizations and individuals to share information over public networks without compromising their privacy. Tor protects the user against a common form of Internet surveillance known as traffic analysis. Traffic analysis can be used to infer who is talking to whom over a public network. Knowing the source and destination of your Internet traffic allows others to track your behavior and interests. Instead of choosing a direct route between the sender and receiver

2. Homomorphic Algorithm

Homomorphic Encryption systems are used to perform operations on encrypted data without knowing the private key (without decryption), the client is the only holder of the secret key. So on decryption, the result of any operation, it is the same as if the calculation is done on the raw data.

The idea of homomorphic computation is to encrypt some numbers, perform algebraic operations like add and "multiply on Ciphertext, then decrypt the result and find it to be exactly the same as if corresponding "+" and "*" operations were applied to the plaintexts. In other words, a homomorphic cryptosystem enables cryptographically secure computations in an untrusted environment. Homomorphism decouples the ability to perform computations from the necessity to view the data as clear text. This allows owners of sensitive data to manipulate encrypted secret data while it resides in an insecure location or to outsource computations on secret data to an untrusted third party.

3. Partial Homomorphic encryption for Log Management

An encryption is homomorphic, if: from Enc (a) and Enc (b) it is possible to compute Enc (f (a, b)), where f can be: +, ×, $\oplus$ and without using the private key. According to the operations that allow assessing on raw data Homomorphic Encryption has been distinguished. The additive Homomorphic encryption (only additions of the raw data) is the Pailler and Goldwasser-Micalli cryptosystems and the multiplicative Homomorphic encryption (only products on raw data) is the RSA and El Gamal cryptosystems. Different cryptosystems support varying levels of homomorphism. Table 3.1 summarizes the different partially Homomorphic Algorithm and its operations.

Table 1: Partially Homomorphic algorithms

Cryptosystem	Homomorphic Operations
RSA	Multiplication mod n
Elgamal	Multiplication, Exponentiation
Paillier	Addition, Subtraction, Multiplication
Goldwasser-Micali	XOR
Benaloh	Addition, Subtraction
Naccache-Stern	Addition, Subtraction, Multiplication

For managing the collected logs [3], it may be outsourced to the Third Party. The Third Party may be responsible for auditing the log data, provides space for storing the log data for future use etc.So, our proposed algorithm keeps the log data secure while we outsource it to the Third Party. We have chosen RSA Algorithm which is Partially Homomorphic in nature for encrypting the log file before giving it to the Third Party .RSA Algorithm is also called as Multiplicative Homomorphic Algorithm.

4. Description

A Homomorphic algorithm is Multiplicative, if

$$\text{Enc}(x \otimes y) = \text{Enc}(x) \otimes \text{Enc}(y)$$

EXAMPLE RSA CRYPTOSYSTEM

RSA is a cryptosystem, which is known as one of the first practicable public-key cryptosystems and is yet widely used for secure data transmission.

In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem.

Working with a public-key encryption system has mainly three phases:

•Key Generation: Once a receiver wants to receive secret messages he/she creates a public key (which is published) and a private key (kept secret). The keys are generated in a way that conceals their construction and makes it 'difficult' to find the private key by only knowing the public key.

•Encryption: A secret message to any person can be encrypted by his/her public key (that could be officially listed like phone numbers).

•Decryption: Only the person being addressed can easily decrypt the secret message using the private key.

5. Implementation and justification of RSA as Partially Homomorphic Algorithm

In this section, we explained the RSA Algorithm and provided an example for justifying it as Partially Homomorphic.

Key Generation:

Step-1: Choose two large primes, p and q, randomly and independently of each other. However, be sure that they are very large so that their product will be of sufficient size. Also, the primes are to be kept secret.

Step-2: Compute the product n = pq. The product, N, is to be made public and is known as the modulus.

Step-3: Compute the totient $\phi(n) = (p-1)(q-1).$ The totient function is Euler's phi function (see notes section).

Step-4: Choose an integer e (which is to be made public), where $1 < e < \phi(N),$ such that $\gcd(\phi(n), e) = 1,$ or equivalently, e is co-prime or relatively prime to $\phi(N).$ In general, if $\gcd(a,b) = 1,$ then a and b are relatively prime and $a \equiv b(\bmod \phi(N))$ is a congruence. Note that this can be found using the Euclidean Algorithm.

Step-5: Compute the private key, d, which is the multiplicative inverse of $e\ (\bmod \phi(n)),$ i.e., find an integer d with $de \equiv 1\ (\bmod \phi(N)).$ In general, solve for d in the equation $1 = ed \bmod (p-1)(q-1).$ The existence of d follows from the fact that if given two integers, a and b, where the $\gcd(a,b) = 1,$ then b is invertible (mod a) and b has a multiplicative inverse in Z_n . This can also be found with the Euclidean Algorithm.

The public key consists of the modulus, N, and the public exponent, also referred to as the encryption exponent, e.The private key consists of, again, the public modulus, N, and the private exponent, d, which is also called the decryption exponent. The value of d must be kept secret, as well as the values of p, q, and $\phi(n).$

Encryption:

The public key numbers is transferred to the person that wants to send us their message.
Then, encryption is done by,
c = m^e mod n
c is our encrypted Message.

Decryption:

In order to decrypt the message private key n and d is needed.**m= c^d mod n**

RSA cryptosystem realize the properties of the multiplicative Homomorphic encryption for example if we assume that two ciphers C_1, C_2 corresponding respectively to the messages m1, m2. The client sends the pair (C_1, C_2) to the Cloud server, the server will perform the calculations requested by the client and sends the encrypted result ($C_1 \times C_2$) to the client. If the attacker intercepts two ciphers C_1 and C_2, which are encrypted with the same private key he/she will be able to decrypt all messages exchanged between the server and the client. Because the Homomorphic encryption is multiplicative i.e. the product of the ciphers equals the cipher of the product.

Suppose we have two ciphers C_1 and C_2 such that:

C_1	=	**m_1^e mod n**
C_2	=	**m_2^e mod n**
$C_1.C_2$	=	**$(m_1m_2)^e$ mod n**

Example: The application of RSA multiplicative Homomorphic encryption on two messages m1 and m2.

Let, for p = 3, q = 5, e = 9 and d = 1 with block size = 1
Two messages m1 and m2 and their ciphers C1 and C2 respectively, obtained using the RSA encryption.
m1 = 539625
C1 = 00 05 00 03 00 09 00 06 00 02 00 05
m2 = 216491
C2 = 00 02 00 05 00 06 00 04 00 09 00 01

The ciphers are converted into binary system, table 2 and 3 represents the conversion of cipher blocks into Binary System.
The binary multiplication of the ciphers block by block is represented in table 4.

If we decrypt the cipher $C_1 \times C_2$ with the private key, we get:

C_1C_2 = 00 10 00 03 00 04 00 05 00 04 00 02 00 04 00 01 00 08 00 05

m_1m_2 = 10 0 3 5 4 2 4 1 8 5

Table 2: The blocks of C1 in binary system

Blocks of C_1	Binary No.
00 05	00 0101
00 03	00 1001
00 09	00 1001
00 06	00 0110
00 02	00 0010
00 05	00 0101

Table 3: The blocks of C2 in binary system

Blocks of C_2	Binary No.
00 02	00 0010
00 01	00 0001
00 06	00 0110
00 04	00 0100
00 09	00 1001
00 01	00 0001

We are multiplying $m_1 \times m_2$ block by block.

m_1 = 5 3 9 6 2 5

m_2 = 2 1 6 4 9 1

m_1 m_2 = 10 3 54 24 18 5

The resultant value is exactly the same raw message obtained by multiplying $m_1 \times m_2$.

Table 4: Binary multiplication of ciphers block

00 0101×00 0010 = 00 1010	00 10
00 1000×00 0011 = 00 11000	00 24
00 1001×00 0110 = 00 110110	00 54
00 0110×00 0100 = 00 11000	00 24
00 0010×00 1001 = 00 10010	00 18
00 0101×00 0001 = 00 0101	00 05

6. System Architecture

Log Generators are machines which generates Logs. An Organization may contain n number of systems, each system generates logs. On the whole, these systems are termed as Log Generators. The logs generated by the Log Generators are collected in the centralized area called Logging Client.

An Organization may need a third party for temporary storage of data (Eg. Cloud is also a third party model which provides many services to their clients), Auditing etc.

Initially, the raw log data are collected by the Logging Client. Then, the logs are encrypted by the logging client using partial homomorphic encryption algorithm before outsourcing it to the third party.

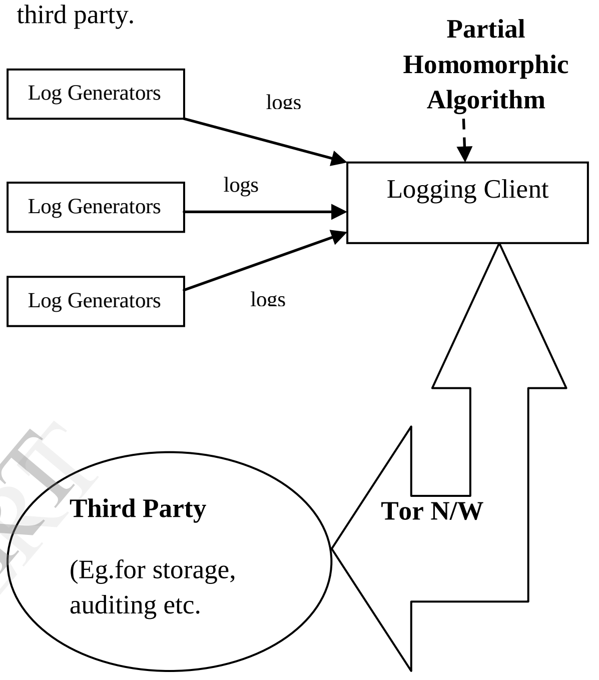

Figure 1: System Architecture for Proposed System

7. Performance Evaluation

For evaluating the performance of the proposed system, two criteria have been chosen,

•Effect of encryption of log records over Overall Performance.

•Effect of Tor Network over Performance.

To calculate the Time Overhead, two log preparation settings have been prepared.

Table 5: Effect of encryption of log records over Overall Performance

Log Settings	Time(sec)	Security of Logs
Without Encryption	53	No security
With Encryption	79	Secured

Table 4: Effect of Tor Network over Performance

Log Settings	**Time using Tor(sec)**	**Time without Tor**
Without Encryption	53	49
With Encryption	79	83

*The time overhead may differ according to the size of the log data, system configuration and network bandwidth.

8. Conclusion

This paper has described a novel and innovative framework called Partial Homomorphic Algorithm. We have proposed an effective mechanism for secure logging .It provides confidentiality, intergrity and privacy of log data. The aim of homomorphic encryption is to ensure privacy of data in communication and storage processes such as the ability to delegate computations to untrusted parties which fulfils the both the secure logging and secure auditing needs.

9. References

[1] Indrajit Ray, Kirill Belyaev, Mikhail Strizhov, Dieudonne Mulamba and Mariappan Rajaram,” Secure Logging As a Service—Delegating Log Management to the Cloud”, *IEEE Systems Journal*, vol. 7,no. 2,pp. 323-334, 2013.

[2] Karthik Nagaraj, Charles Killian and Jennifer Neville,” Structured Comparative Analysis of Systems Logs to Diagnose Performance Problems”, *9th USENIX Symposium on Networked Systems Design and Implementation*, pp. 26-29, 2012.

[3] Mayank Saxena ,Nikhil Kumar Singh ,Satyendra Singh Thakur and Parmalik kumar,” A Review of Computer forensic & Logging System”, *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 1, pp. 1- 6, 2012.

[4] R Accorsi,” Towards a secure logging mechanism for dynamic systems”, *Proceedings in the 7th IT Security Symposium*, pp. 1-7, 2005.

[5] Arasteh, Ali Reza, et al. "Analyzing multiple logs for forensic evidence" *International Journal of Digital Forensics & Incident Response on digital investigation*, vol. 4, pp 82-91, 2007.

[6] Mayol Arnao Reinaldo, Nunez Luis A and Lobo Antonio, “An Approach to Log Management: Prototyping a Design of agent for Log Management “, *International Journal on Networking and Internet Architecture* ,vol. 1112, no. 795, pp. 1- 9, 2011.

[7] Tor Project, Inc. (2011, Sep.) Tor: Anonymity Online.

[8] Abad C, Taylor J, Sengul C, Yurcik W, Yuanyuan Zhou and Rowe K, "Log correlation for intrusion detection: a proof of concept," *19th Annual Computer Conference on Security Application*., vol. 8, no. 12, pp.255, 264, 2003.