Data Lake Architecture for storing and Transforming web server Access Log Files

Dr. P. K. Venkateswar Lal, M. Tech, Ph. D, Professor, Department of CSE, Narayana Engineering College, Gudur

B.Saketha Sai, Department of CSE, Narayana Engineering College, Gudur

Abstract: *This paper presents a novel cloud storage scheme designed to enhance the security and integrity of data in sensor networks. Sensor networks, comprising specialized transducers and a robust communications infrastructure, are vital for monitoring and recording conditions across diverse locations. Leveraging the advantages of cloud computing, such as on-demand self-service and ubiquitous network access, the proposed system introduces a trustworthy audit server to preprocess and upload data on behalf of clients. This approach ensures semi-honest trustworthiness and dynamic data processing in the cloud. The system also develops a strengthened security model to protect against data leakage attacks during the upload phase. Additionally, an efficient verification scheme is presented to ensure remote data integrity in cloud storage. This comprehensive solution aims to provide a secure and reliable framework for cloud storage in sensor networks, addressing critical challenges in data security and integrity.*

Keywords: *Sensor Networks, Data Integrity, Semi-Honest Trustworthiness, Data Leakage Attack.*

I.INTRODUCTION

A sensor network is a collection of specialized transducers that form a communications infrastructure designed to monitor and record various conditions across diverse locations. These networks are pivotal in tracking parameters such as temperature, humidity, pressure, wind direction and speed, illumination intensity, vibration intensity, sound intensity, power-line voltage, chemical concentrations, pollutant levels, and vital body functions. Each sensor node within the network is small, lightweight, and portable, equipped with a transducer, microcomputer, transceiver, and power source. The transducer generates electrical signals based on the physical effects and phenomena it senses. The microcomputer processes and stores this sensor output, while the transceiver handles communication by receiving commands from and transmitting data to a central computer. Power for each sensor node is typically supplied by a battery.Modern sensor networks often feature bi-directional capabilities, enabling not only the collection of data but also the control of sensor activity. Initially developed for military applications like battlefield surveillance, these networks have found widespread use in industrial and consumer applications, including industrial process monitoring and control, machine health monitoring, and environmental tracking.

The purpose of this research is to propose a secure method for transmitting sensor data provenance in cloud-based environments. This system formulates the problem of secure provenance transmission in sensor networks and introduces an in-packet Bloom filter (iBF) provenance-encoding scheme. Additionally, it extends encryption methodologies to detect packet drop attacks by malicious sensor nodes. This approach ensures that only authorized parties can process and verify the integrity of the provenance, thereby maintaining confidentiality, integrity, and freshness of the data.

Provenance tracking is increasingly dependent on digital records, particularly in high-stakes scenarios such as pharmaceuticals and clinical trials. For example, the provenance of pharmaceuticals is meticulously tracked from manufacturing to consumer delivery, and new medical devices and treatments undergo rigorous record-keeping during clinical trials. In a multi-hop sensor network, data provenance enables the base station (BS) to trace the source and forwarding path of individual data packets. Despite the constraints of storage, energy, and bandwidth in sensor nodes, this system addresses these challenges to ensure secure and reliable provenance recording.

This research is geared towards college-level audiences and professionals interested in internal intrusion detection processes. It provides valuable insights into secure data transmission and provenance tracking within sensor networks, making it useful for those studying or working in fields related to IT security, network management, and data integrity.

The proposed system aims to secure the transmission of provenance for sensor data. It includes an in-packet Bloom filter (iBF) provenance-encoding scheme, an extension of encryption methods, and mechanisms to detect packet drop attacks by malicious forwarding nodes. By addressing these aspects, the system ensures the secure and efficient transmission of provenance data in sensor networks.

The system ensures that only authorized parties can process and verify provenance, maintaining data confidentiality, integrity, and freshness. Provenance tracking, critical in industries like pharmaceuticals and clinical trials, relies heavily on accurate and secure digital records. In a multi-hop sensor network, the proposed method allows tracing the source and path of data packets while overcoming storage, energy, and bandwidth constraints of sensor nodes. This system provides a foundational framework for building secure, reliable, and scalable cloud environments for sensor networks.

Recent research on information leakage has primarily focused on confining sensitive data to a single node within a network, limiting data sharing within an enterprise. This research proposes a novel approach to address these limitations by enabling secure data sharing and detecting potential leaks. Traditional methods like Tightlip and Capizzi et al.'s approaches confine data within single machines, which restricts data dissemination necessary for various applications. This system enhances these methodologies by allowing secure and efficient data sharing across the network while maintaining stringent security measures to prevent data leaks.

II. PROPOSED SYSTEM

Sensor networks are composed of specialized transducers equipped with a communications infrastructure, designed to monitor and record environmental and system parameters across diverse locations. These networks capture data on a variety of conditions, such as temperature, humidity, pressure, wind direction and speed, illumination intensity, vibration intensity, sound intensity, power-line voltage, chemical concentrations, pollutant levels, and vital body functions. Each node in a sensor network is typically small, lightweight, and portable, featuring a transducer, microcomputer, transceiver, and power source. The transducer converts physical phenomena into electrical signals, which are then processed and stored by the microcomputer. The transceiver facilitates communication by receiving commands from and transmitting data to a central computer, while the power source, often a battery, sustains the sensor node's operations.

Modern sensor networks support bi-directional communication, enabling not only data collection but also the control of sensor activities. Initially developed for military applications like battlefield surveillance, these networks have found applications in various industrial and consumer sectors, including industrial process monitoring, machine health monitoring, and environmental management.

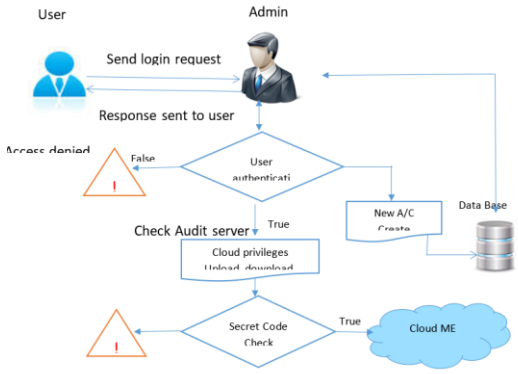
This research introduces a new cloud storage scheme designed to ensure the retrievability and integrity of data in sensor networks. The proposed system leverages a trustworthy audit server to preprocess and upload data on behalf of clients, enhancing semi-honest trustworthiness and supporting dynamic data processes in the cloud. A strengthened security model is developed to safeguard against data leakage during the upload phase

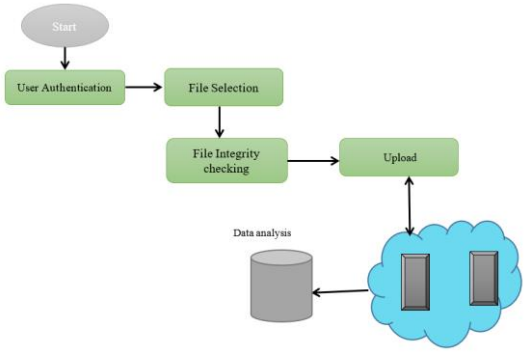
of an integrity verification scheme. Additionally, the system presents an efficient verification scheme for ensuring remote data integrity in cloud storage.

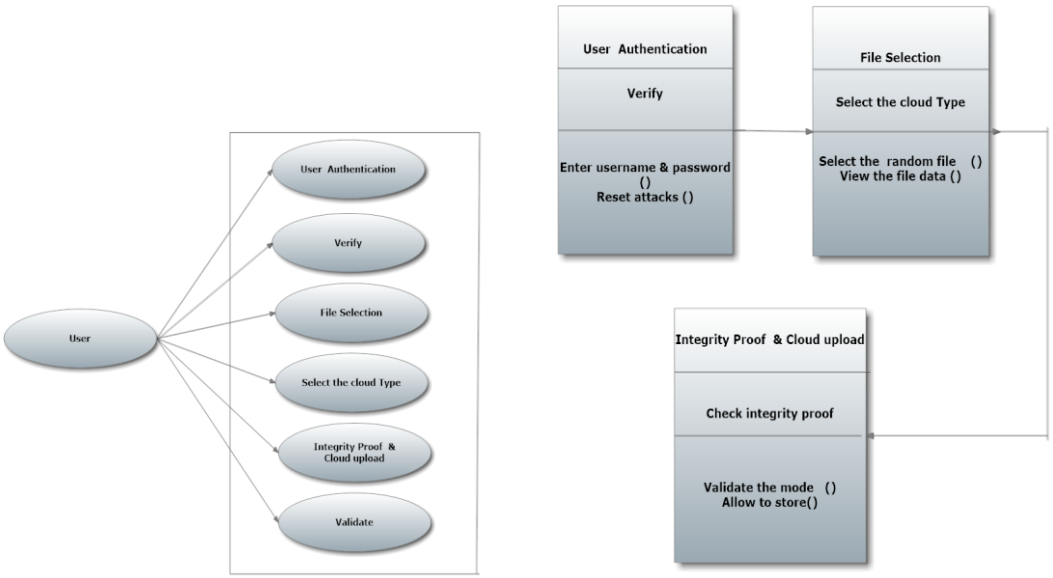
Advantages

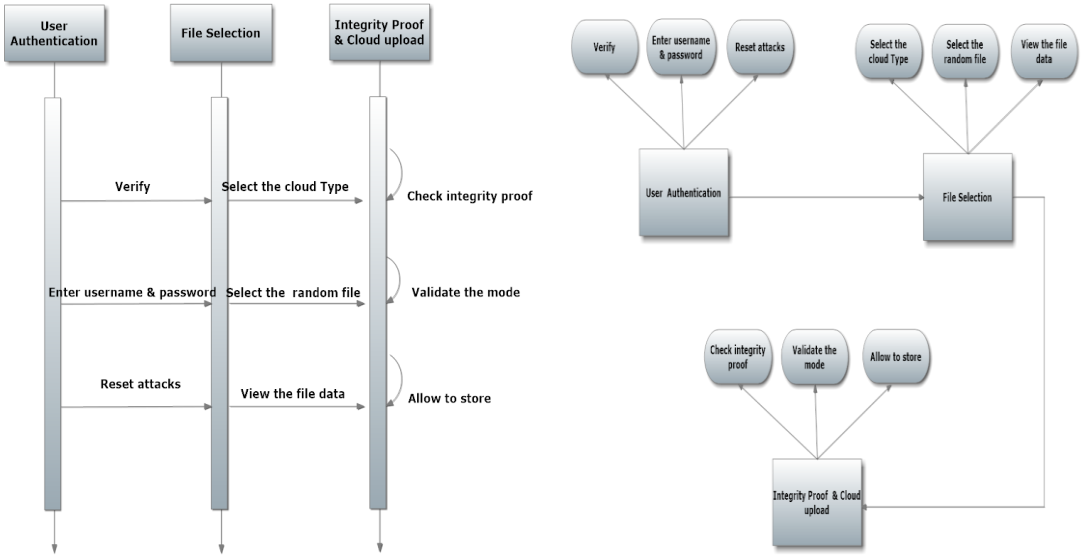
- **Low Computation Cost:** The system reduces the computational burden on users, allowing data processes to work efficiently.
- **Minimal Storage Requirements:** The Cloud Audit Server (CAS) does not need high storage capacity.
- **Security Against Reset Attacks:** The system is proven secure against reset attacks within the strengthened security model, supporting efficient public verifiability and dynamic data operations.

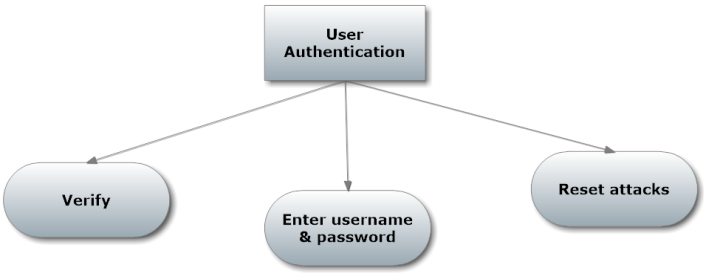
System Architecture **Flow Diagram**











Modules Description

Client Phase: This module focuses on verifying the cloud user's authentication. Upon system entry, users validate their identity using an ID and cloud password. New users must register, providing personal details and a secret code, which is encrypted for security using the AES algorithm. If trust in the user is not established, the process is aborted. The system supports efficient public verifiability and dynamic data operations while proving secure against reset attacks.

User Access Privilege in Cloud: This module defines user privileges in the cloud, offering services like data upload, download, and dynamic data deletion. Users can upload data upon meeting integrity proof requirements, download data after verifying security codes, and delete data securely from the cloud.

Integrity Proof for User Service Access: The cloud server reduces client burden by maintaining files and sequentially monitoring user accounts, dynamically updating them after each action. The system employs an Integrity Verification algorithm for multiple challenge-responses, ensuring data correctness and supporting efficient public verifiability and dynamic data operations.

Cloud Audit Server Phase: This module manages and verifies user data actions, ensuring secure and efficient data handling, while supporting efficient public verifiability and dynamic data operations.

Testing of Product: System testing ensures the system operates accurately and efficiently before live operation. It involves executing a program to find errors, with successful tests identifying undiscovered errors. Various tests, including unit testing, integration testing, white box testing, black box testing, validation testing, user acceptance testing, and output testing, are conducted to ensure system correctness, security, and usability.

- **Unit Testing:** Tests each module separately during programming.
- **Integration Testing:** Ensures overall system performance through top-down and bottom-up approaches.
- **White Box Testing:** Uses control structures of procedural design to drive test cases.
- **Black Box Testing:** Validates external behavior of the system against specified functions.
- **Validation Testing:** Ensures software functions as expected by the customer.
- **User Acceptance Testing:** Tests for user acceptance and satisfaction.
- **Output Testing:** Verifies the system produces required outputs in specified formats.

System Implementation

Implementation refers to installing the software in its real environment to satisfy intended users. Key factors for successful implementation include user awareness of system benefits, building user confidence, and providing proper guidance. The system must be user-friendly, with operational documentation provided to make users comfortable.

User Training

Effective user training ensures that individuals involved in the system are confident in their roles. Training covers computer awareness and application software usage, explaining system philosophy, screen flow, design, help features, and error handling. Training may vary across user groups and hierarchy levels.

System Maintenance

System maintenance is crucial for adapting to changes in the system environment. Maintenance involves corrective, adaptive, perceptive, and preventive activities to ensure the system remains functional and up-to-date. Corrective maintenance addresses errors, adaptive maintenance handles environmental changes, perceptive maintenance involves enhancing system capabilities, and preventive maintenance improves future maintainability and reliability.

Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES) is a symmetric key encryption algorithm that uses the same key for both encryption and decryption. This necessitates that all operations performed during encryption must be reversible during decryption using the same key. AES focuses on various invertible operations to achieve this.

Key Components of AES

1. Key Derivation and XOR Combination:

- A string is derived from the key.
- The plaintext or emerging ciphertext is combined with this string using the XOR operation.
- During decryption, the same XOR operation is applied to reverse this combination.

2. Mixing Operations: These operations shuffle the data around to achieve diffusion, spreading the plaintext information across the ciphertext.

3. Translation (Substitution) Operations: These operations replace one piece of data with another using substitution boxes (S-boxes). S-boxes are predefined tables that define the replacement strings for small portions of the ciphertext.

AES Rounds A single set of mixing, substitution, and XOR operations is known as a round. The number of rounds in AES depends on the key size. The standard AES key sizes and corresponding number of rounds are:

Key Sizes versus Rounds			
	Key Block Size *(Nk words)*	**Plaintext Block Size** *(Nb words)*	**Number of Rounds** *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES operates on a fixed block size of 4 words (128 bits). Although AES was initially designed to support multiple block sizes, the final specification uses a constant block size of 128 bits. This block size is kept as a named constant in the algorithm specification to allow for potential future changes.

AES is a robust and efficient encryption standard widely used for securing data. Its structure of key derivation, mixing, substitution, and XOR operations ensures strong security while maintaining performance. The use of a fixed block size and variable key lengths allows AES to be adaptable for various security needs, making it a versatile choice for encryption.

Note: In This Project we using AES-128 key size for re-encryption process

For the *Mathematical Cryptograph Notations*:

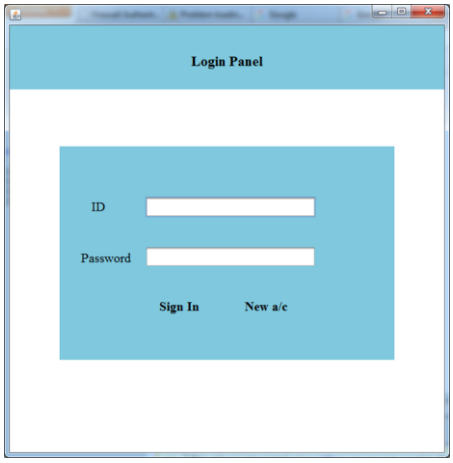
- The input data block is broken into a 4x4 byte array (128-bit key)
  The initial subkey (derived from the cipher key via key schedule [8]) is XOR'd to the byte array by an AddRoundKey() operation (Nb = block size)
- For each encryption “round”/iteration to n-1 (128-bit, n=10; 192-bit, n=12; 256-bit, n=14):
- Each array byte is substituted using a SubBytes() S-box [7] operation
- A ShiftRows() cyclic shift operation is done on the last three rows of the byte array

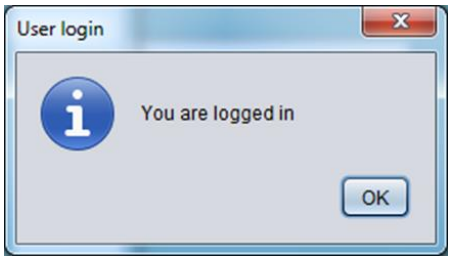
The primary encryption module calling every other function:

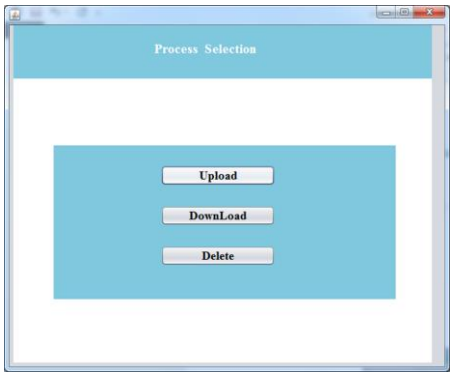
```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
        byte state[4,Nb]
        state = in
        AddRoundKey(state, w[0, Nb-1])
        for round = 1 step 1 to Nr–1
                SubBytes(state)
                ShiftRows(state)
                MixColumns(state)
                AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
```

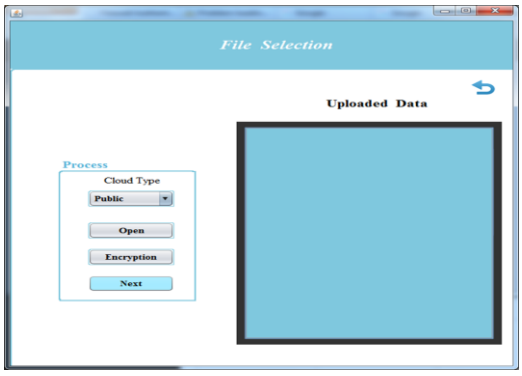
```
        end for
        SubBytes(state)
        ShiftRows(state)
        AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
        out = state
    end
```

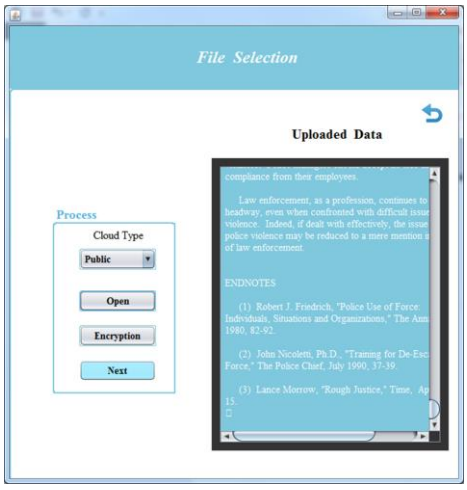
III.RESULTS AND SNAPSHOTS

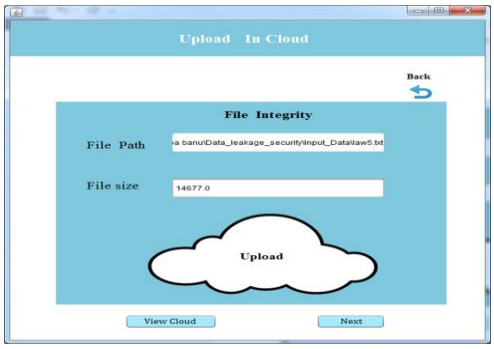


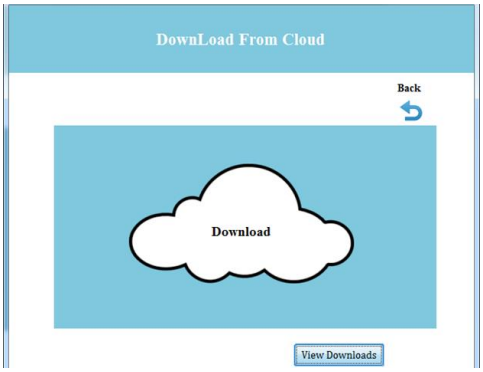


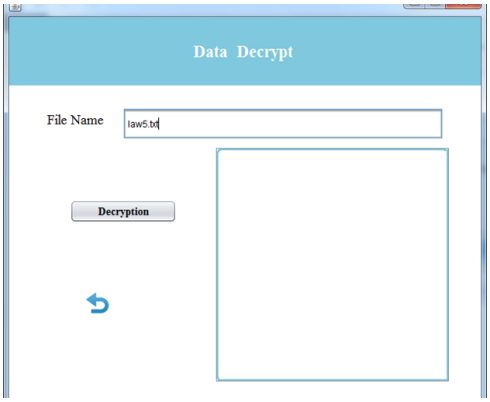


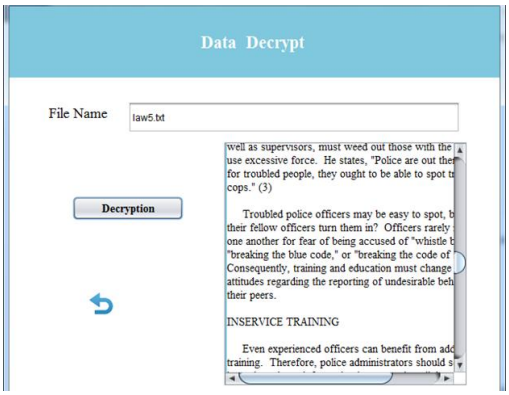












.CONCLUSION

Preventing sensitive data from being compromised remains a significant and practical research challenge, especially in the context of cloud computing. This project has implemented and evaluated a privacy-preserving data-leak detection system that allows data owners to safely deploy locally or delegate traffic inspection tasks to cloud providers without exposing sensitive data. By addressing the challenge of reducing the computational burden on users, the system ensures that private information is safeguarded, even in an untrusted environment. In cloud computing, where users store their data files on cloud servers, preventing unauthorized access and realizing secure resource sharing are critical. Traditional access control methods often assume a secure domain shared by data owners and storage servers, with the server being fully trusted. However, this assumption is no longer valid in cloud environments where service providers can be attacked by malicious entities. The

proposed system tackles these issues by ensuring that access control can be realized on encrypted data while maintaining the confidentiality of user data.

Our security model differs from traditional models in both the verification and data updating processes. Specifically, our scheme requires tags to be authenticated by the client during each protocol execution, rather than being pre-calculated or stored by the client. This approach enhances security by mitigating the risk of unauthorized access and data leaks.

In conclusion, this project contributes to the field of cloud security by providing a robust solution for preventing data leaks and unauthorized access. The proposed system ensures that sensitive data remains secure, even in the presence of potential threats, making it a valuable advancement in cloud computing technologies and applications.

V.REFERENCES

[1] Adobe Systems Incorporated. Adobe Flash Player http://www.macromedia.com/software/flash/about, 2008.

[2] R. Anderson and F. Petitcolas. On the Limits of Steganography. IEEE Journal of Selected Areas in Communications, 16(4):474-481, 1998.

[3] K. Borders and A. Prakash. Web Tap: Detecting Covert Web Traffic. In Proc. of the 11th ACM Conference on Computer and Communications Security (CCS), 2004.

[4] K. Borders and A. Prakash. Towards Quantification of Network-Based Information Leaks Via HTTP. In Proc. of the 3rd USENIX Workshop on Hot Topics in Security, 2008.

[5] S. Brand. DoD 5200.28-STD Department of Defense Trusted Computer System Evaluation Criteria (Orange Book). National Computer Security Center, 1985.

[6] S. Cabuk, C. Brodley, and C. Shields. IP Covert Timing Channels: Design and Detection. In Proc. of the 11th ACM Conference on Computer and Communications Security (CCS), 2004.

[7] S. Castro. How to Cook a Covert Channel. hakin9, http://www.gray-world.net/projects/ cooking_channels/hakin9_cooking_channels_en.pdf, 2006.

[8] J. Gailly and M. Adler. The gzip Home Page. http://www.gzip.org/, 2008.

[9] J. Giles and B. Hajek. An Information-Theoretic and GameTheoretic Study of Timing Channels. IEEE Transactions on Information Theory, 48:2455–2477, 2003.

[10] M. Handley, V. Paxson, and C. Kreibich. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In Proc. of the 10th USENIX Security Symposium, 2001.

[11] M. Kang, I. Moskowitz, and D. Lee. A Network Version of the Pump. In Proc. of the 1995 IEEE Symposium in Security and Privacy, 1995.

[12] G. Malan, D.Watson, F. Jahanian, and P. Howell. Transport and Application Protocol Scrubbing. In Proc. of the IEEE INFOCOM 2000 Conference, 2000.

[13] S. McCamant and M. Ernst. Quantitative Information Flow as Network Flow Capacity. In Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2008.

[14] Mozilla. The Firefox Web Browser. http :// www .mozilla .com /firefox /, 2008.

[15] Mozilla. SpiderMonkey (Javscript-C) Engine. http :// www .mozilla .org /js /spidermonkey/, 2008.

[16] A. Myers, N. Nystrom, L. Zheng, and S. Zdancewic. Jif: Java information flow. http://www.cs.cornell.edu/jif, 2001.

[17] R. Richardson. CSI Computer Crime and Security Survey. http :// i.cmpnet.com/ v2.gocsi.com/pdf/CSISurvey2007.pdf, 2007.

[18] RSA Security, Inc. RSA Data Loss Prevention Suite. RSA Solution Brief, http://www.rsa.com/products/EDS/sb/DLPST_SB_1207- lowres.pdf, 2007.