Journal of Theoretical and Applied Information Technology
28th February 2026. Vol.104. No.4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DEVELOPING A HYBRID LOG-FILE–BASED COVERT CHANNEL FOR SECURE DIGITAL FORENSIC DOCUMENT TRANSMISSION

TAMER S. FATAYER[1]. SAMY S. ABU-NASER[2], MUSBAH J. MOUSA[3]

[1,3]Al-Aqsa University: Computers and Information Faculty, Gaza, Palestine
[2]Faculty of Engineering and Information Technology, Al-Azhar University, Palestine.
E-mail: [1]ts.fatayer@alaqsa.edu.ps,[2]abunaser@alazhar.edu.ps, [3]mj.mousa@alaqsa.edu.ps

ABSTRACT

Security is a fundamental requirement in Digital Forensic (DF) systems, given the increasing number and sophistication of cyber threats. Among the major challenges in DF is ensuring the secure transmission of Digital Forensic Documents (DFDs) between authorized entities. This paper proposes a hybrid covert channel designed to enhance the confidentiality and un-detectability of DFD transfers by exploiting runtime errors and system log files. The proposed method leverages the Digital Forensic Legal Affairs Server (DFLAS) log file, where original DFDs embedded with covert data are written and subsequently transmitted through a concealed communication channel. The original files remain preserved in the server's log file after transmission, ensuring both integrity and traceability. To achieve robust security, the hybrid covert channel integrates randomization, encryption, and controlled packet delay, collectively reinforcing confidentiality and resistance to detection. The performance evaluation focuses on analyzing time transmission and bandwidth efficiency during exchanging the DFDs. Experimental results show that
the proposed channel achieves high efficiency in covert data transmission, with transmission time and throughput comparable to conventional channels. Additionally, it provides a high level of confidentiality through the use of encryption and randomization techniques. The results show that transferring a 112 MB file requires approximately 4000 ms on the DFLAS side, demonstrating competitive performance compared with existing approaches. The results further indicate that the extraction capacity rate varies according to the applied delay intervals while maintaining practical throughput levels suitable for DF real-world deployment. The achieved throughput of 29.12 MB/sec confirms the method's practicality. The proposed covert channel provides confidentiality, plausibility, and undetectability, making it a promising approach for secure and efficient transmission of digital forensic documents.

Keywords: *Covert channel, Digital forensics, Log file, Undetectability, Encryption*.

1. INTRODUCTION

Security is a fundamental aspect of modern digital forensic (DF) systems. Digital forensics is the scientific process of collecting, analyzing, and preserving digital evidence in a way that maintains its integrity and admissibility in court. As cybercrimes become increasingly complex, DF plays a crucial role in identifying, investigating, and reconstructing digital incidents across various platforms, including computers, smartphones, and networked environments.

Forensic investigators rely heavily on Digital Forensic Documents (DFDs) to record, share, and interpret evidence during investigations. However, as the volume and sensitivity of digital evidence continue to grow, ensuring the secure exchange of DFDs among authorized entities has become a significant challenge. Attackers often target or compromise these documents during transmission, risking evidence manipulation or unauthorized disclosure [1]. Therefore, there is an urgent need for reliable and covert communication mechanisms that ensure confidentiality, integrity, and un-detectability in DF systems.

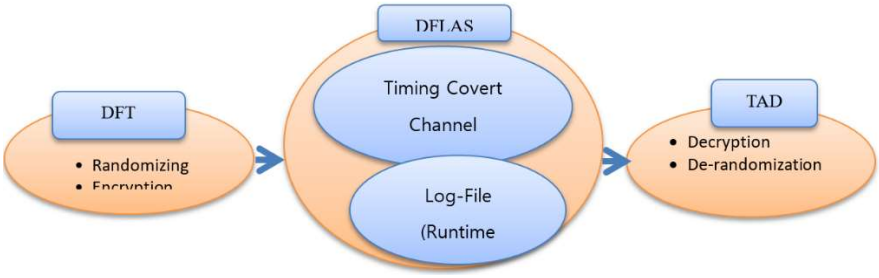

Figure 1: The essential processes involved in developing our timing covert channel

To address this challenge, this paper proposes a hybrid covert channel for the secure transmission of DFDs between forensic entities. The proposed channel leverages runtime errors and system log files to conceal and transmit data, thereby minimizing the likelihood of detection by malicious actors. As illustrated in Figure 1, the framework comprises three main components: the Digital Forensic Team (DFT), the Technical Analysis Department (TAD), and the Digital Forensic Legal Affairs Server (DFLAS). The DFT generates the forensic documents to be securely transmitted to the TAD via a covert channel embedded within the DFLAS.

In this approach, the server is deliberately impersonated and exploited to establish a covert communication channel. It receives an encrypted message, attaches a timestamp, and stores it in a log file generated through a runtime error (such as a division-by-zero event). The TAD subsequently reads the log file, performs decryption and de-randomization, and reconstructs the original message based on the embedded timing patterns. This design ensures that communication remains concealed within normal system behavior, providing plausibility and un-detectability.

The remainder of this paper is organized as follows. Section defines the problem statement, Section 3 outlines the objectives of the study, Section 4 presents the significant and contribution of the study, Section 5 presents background information on covert channels, digital forensics, and log file mechanisms. Section 6 reviews related works on covert data transmission techniques. Section 7 details the proposed hybrid covert channel, including its architecture and implementation. Section 8 discusses the experimental results and evaluates performance in terms of transmission time, extraction rate, and security. Finally, Section 9 concludes the paper and highlights the main findings and future research directions.

2. PROBLEM STATEMENT

In digital communication systems, covert channels represent a critical security concern because they enable unauthorized data exchange in a concealed manner that often evades traditional detection mechanisms. Despite numerous studies exploring storage, timing, and program flow channels independently, existing approaches face significant limitations in achieving a balance between data transmission efficiency, concealment effectiveness, and resilience against detection. Most prior research—such as the exploitation of stack overflow, memory management, or packet-timing manipulation—either suffers from low data transfer rates or lacks robustness against forensic analysis tools and modern intrusion detection systems.

Moreover, current covert channel implementations often rely on predictable or easily detectable system behaviors, such as static memory addresses or fixed communication patterns. These limitations undermine their stealth and restrict their applicability in highly monitored environments. Furthermore, the absence of integrated hybrid models that combine multiple covert communication techniques (e.g., storage, timing, and program flow) reduces the flexibility and efficiency of hidden data transmission.

To address these gaps, this research focuses on creating a hybrid covert channel scenario through the exploitation of a Distributed File System (DFS) log file. The proposed approach integrates storage, timing, and program flow covert communication within a unified framework. By embedding encrypted and noise-mixed data within DFS log entries generated through runtime errors, the system enhances concealment while maintaining efficient and secure data transfer. This hybridization not only improves resistance to detection and analysis but also demonstrates a significant improvement in data transfer time performance compared to traditional covert channel techniques.

The central problem, therefore, is the lack of an effective, resilient, and high-performance hybrid covert communication model that can securely

transmit data through normal system operations without raising suspicion. The proposed method aims to fill this gap by leveraging the natural behavior of log file generation in DFS systems as a covert communication medium.

3. RESEARCH OBJECTIVES

Building on the identified problem, this research aims to design and implement a **hybrid covert** communication model that enhances data concealment, transmission efficiency, and detection resistance within a Distributed File System (DFS) environment. The proposed approach exploits the DFS log file as a communication medium to embed encrypted information using combined storage, timing, and program flow channels.

The specific objectives of this research are as follows:

- **To develop a hybrid covert channel framework** that integrates storage, timing, and program flow mechanisms for secure and undetectable data exchange within DFS log files.
- **To implement an encryption and obfuscation strategy** (using RC6 with noise injection) that enhances data confidentiality and prevents signature-based detection.
- **To exploit runtime errors and log file generation patterns** as a natural and stealthy carrier for covert data transmission.
- **To evaluate the performance of the proposed hybrid model** in terms of data transfer rate, concealment efficiency, and resistance to forensic or intrusion detection analysis.
- **To compare the proposed approach** against existing covert channel methods to demonstrate its superiority in data throughput and detection resilience.

Through these objectives, this research seeks to contribute a novel and practical solution for secure, covert communication that can evade advanced detection mechanisms while maintaining system stability and operational

4. RESEARCH CONTRIBUTION AND SIGNIFICANCE OF THE STUDY

This research makes a significant contribution to the field of covert communication and digital forensics by introducing a hybrid covert channel model that exploits the Distributed File System Log (DFLAS) for secure and undetectable data transmission. Unlike conventional covert channels that rely on either storage or timing alone, the proposed approach integrates storage, timing, and program flow channels to establish a more robust and efficient communication pathway.

The major contributions of this study can be summarized as follows:

- **Novel Hybrid Design:**
  The study presents a hybrid covert channel that combines multiple channel types—storage, timing, and program flow—into a unified framework. This integration enhances the stealthiness and resilience of data transmission against detection techniques.
- **Exploitation of System Log Files:**
  The research introduces a unique technique for embedding covert data into runtime error–generated log files**,** leveraging legitimate system activities to conceal communication without raising suspicion.
- **Enhanced Security through Randomization and Encryption:**
  The use of RC6 encryption coupled with random noise injection ensures that the covert data remains indistinguishable from normal log file entries, increasing confidentiality and detection resistance.
- **Performance Optimization and Evaluation:**
  Comprehensive evaluation demonstrates that the proposed model achieves high data throughput and low detectability**,** outperforming existing methods such as those based on stack overflow or single-channel covert communication.
- **Practical Application in Digital Forensics:**
  By providing a secure and covert communication mechanism between Digital Forensic entities (DFT, DFLAS, and TAD), the proposed approach enhances the integrity, confidentiality, and trustworthiness of sensitive data transfer in forensic operations.

Overall, this research contributes to the advancement of secure covert communication mechanisms and offers a practical, efficient, and stealthy model for protecting sensitive digital evidence within distributed systems. It lays the groundwork for future exploration into adaptive, AI-driven covert communication frameworks that can dynamically respond to evolving detection techniques.

5. BACKGROUND

This section reviews the technical concepts that underpin our work: covert channels, digital forensics (DF), and system log files. We synthesize definitions, key properties, and the motivations that justify a hybrid covert-channel approach for secure Digital Forensic Document (DFD) exchange.

5.1 Covert Channels: Definitions, Taxonomy, and Key Properties

A **covert channel** is any communication path that was not designed for information transfer but can be exploited to pass information between processes or hosts while circumventing security policies. Covert channels are commonly classified along two orthogonal axes: (1) **storage vs. timing**—whether information is embedded in shared storage objects (files, metadata) or in observable timing characteristics (timestamps, inter-event delays); and (2) noisy vs. noiseless / program-flow—whether the channel is affected by uncontrollable system noise or relies on program execution behavior [2]-[3].

Covert channels are evaluated by several fundamental properties [4]-[5]:

- **Un-detectability:** the difficulty of distinguishing covert traffic from legitimate system activity. High un-detectability reduces an adversary's chances of discovery.
- **Robustness:** the channel's resilience to interference, system updates, or active countermeasures.
- **Capacity (covert rate):** the maximum amount of information that can be conveyed reliably per unit time (typically bits/sec).
- **Noise tolerance:** how errors introduced by the environment (scheduling, interrupts, log rotation) affect reliability.

Designers of covert channels trade these properties: for example, maximizing capacity often increases detectability, while emphasizing un-detectability tends to reduce throughput. Hybrid channels—those that combine storage, timing, and program-flow techniques—seek to balance these trade-offs by exploiting complementary strengths (e.g., storage's persistence and timing's stealth). In this work we explicitly combine: (a) a storage covert channel via system log entries, (b) a timing covert channel via controlled timestamps/delays, and (c) a program-flow covert channel using intentionally triggered runtime errors to produce plausible log artifacts.

5.2 Digital Forensics: Requirements and Challenges For Secure Exchange

Digital forensics (DF) is the disciplined process of identifying, collecting, preserving, analyzing, and presenting digital evidence so it remains admissible and trustworthy [6]. DFDs—structured artifacts that document evidence, metadata, chain-of-custody, and analytic results—are central to investigations and must be exchanged between forensic teams, legal authorities, and technical analysts.

Key security and operational requirements for DFD exchange include [7]-[8]:

- **Confidentiality:** sensitive case data must be protected in transit and at rest.
- **Integrity and provenance:** recipients must be able to verify that documents are unaltered and to trace their origin and handling.
- **Plausible deniability / plausibility:** in hostile environments, transmissions should blend with normal system behavior to avoid drawing attention.
- **Availability and timeliness:** forensic collaboration often requires timely access to data without undue delay.

Current secure transport mechanisms (e.g., TLS, encrypted files) protect confidentiality and integrity but can attract attention in adversarial deployments or be blocked/monitored. Moreover, an attacker with privileged access to forensic infrastructure may attempt to intercept, modify, or disrupt DFD transfers [9]. These concerns motivate covert mechanisms that can hide exchanges within routine system artifacts while preserving forensic requirements such as traceability and verifiability.

5.3 System Log Files As Covert Media

System and application log files record a wide range of routine events—errors, status messages, timestamps, and diagnostics—and are therefore attractive carriers for covert data. Logs are continuously generated, commonly collected centrally, and frequently allowed to traverse network boundaries for monitoring or legal purposes; this ubiquity increases plausibility for hidden messages placed inside them [10]-[11]. Log analysis is an established tool for attack detection and system assessment [12]-[13], which means that covert activity must be carefully designed to avoid anomalous log patterns that would trigger alarms.

Advantages of log-based covert channels:

- **Persistence:** entries remain in storage and can be retrieved later, easing asynchronous extraction.
- **Plausible cover:** embedded data can masquerade as normal error messages or diagnostic fields.
- **Central collection points:** many organizations forward logs to centralized servers (like DFLAS), creating a natural aggregation point for hidden messages.

Constraints and threats:

- **Log sanitization and rotation:** automated log processing may remove or truncate covert data.
- **Anomaly detection:** modern SIEMs may flag unusual frequencies, unexpected message formats, or sudden bursts of error entries.
- **Forensic auditability:** any covert modification must preserve evidence admissibility and avoid corrupting original investigative records.

5.4 Threat Model and Design Rationale

We assume an adversary capable of network monitoring and partial access to forensic infrastructure but **not** full control of both endpoints simultaneously. Our design goals are to deliver confidentiality and un-detectability while retaining forensic traceability. To achieve this we combine encryption (to protect confidentiality), randomization (to avoid repetitive patterns), and timing control (to embed information in timestamps and inter-event delays). The use of runtime errors as a program-flow trigger generates log entries that are plausible within normal server operation, lowering the chance of detection while permitting reliable extraction by authorized recipients.

5.5 Gap Analysis and Contribution

Prior work has explored individual covert-channel modalities (timing-only or storage-only) and log-based concealment techniques, but few studies systematically combine storage, timing, and program-flow channels within an operational DF context. This hybrid approach leverages the persistence and plausibility of logs with the stealth of timing channels and the controllability of program-flow triggers, addressing limitations in capacity, un-detectability, and robustness found in prior methods. Our contribution is a practical, evaluated hybrid covert channel tailored to the secure exchange of DFDs, together with an empirical assessment of throughput, extraction reliability under delay, and security properties.

6. RELATED WORKS

Several studies have investigated covert channel design for secure data transmission across different domains, including operating systems, authentication protocols, and multimedia applications. This section critically reviews relevant literature and highlights how the proposed hybrid covert channel advances the state of the art in terms of confidentiality, un-detectability, and performance.

The authors in [14] developed a storage-based covert channel by exploiting stack overflow vulnerabilities within Linux memory. Their method simulated stack overflow behavior to conceal data but relied on guessing system call addresses [10], which significantly reduced reliability due to memory address randomization in modern Linux kernels. Although their channel demonstrated basic feasibility using audio files as the covert payload, it exhibited limited throughput and weak detection resistance. In contrast, our approach eliminates the dependence on address prediction by embedding covert data in system log files, achieving higher performance and greater stability in data transmission. Table 1 summarizes the performance comparison in terms of data transfer time, illustrating our method's superior efficiency.

Table 1 The time transferring files [14]

Audio file size	Time (Minute)
2.3 KB	1.01
1.2 MB	234

[15] proposed a storage-based covert channel leveraging a challenge–response authentication mechanism. Their system embedded secret bits into authentication nonce after compression, enabling the covert transfer of information during authentication handshakes. While effective for small payloads, their approach suffered from limited throughput, particularly when transmitting larger JavaScript-based data.

The author combined encryption, authentication, and covert channels to improve file transmission security. The study implemented a storage covert channel using bit distribution and prior knowledge agreements to maintain plausibility. The work also evaluated time efficiency in key generation and examined how multiple users could securely access the channel through authentication mechanisms.

Although it enhanced confidentiality, it did not address timing control or randomization, both of which are integrated in our hybrid model to strengthen un-detectability [2].

[16] demonstrated a high-performance covert channel using Java-based runtime errors to encode and transmit covert data. Various encoding schemes—Base64, Hexadecimal, ASCII, Byte, and Huffman—were analyzed to measure transmission throughput. While their design improved performance, it was limited to the Java environment and lacked a hybrid structure combining multiple covert techniques. Our approach, by contrast, fuses **storage, timing, and program-flow channels** to achieve both scalability and stealth.

[17] focused on timing covert channels within Voice over IP (VoIP) systems. By exploiting the voice activity detection (VAD) feature, they embedded secret data during silent periods in voice calls. Although effective in maintaining audio quality, this technique was limited by bandwidth variability and application-specific constraints.

[18] utilized a covert channel for cryptographic key generation, where keys were embedded within transmission timing intervals and later used for encryption and decryption. Their evaluation showed variable key generation times but lacked scalability for large data transfers or forensic document exchange.

[19] applied covert channels to Internet of Things (IoT) networks, implementing a hidden communication layer at the physical layer to enhance data throughput. Despite improvements in data delivery, their method faced bandwidth and transmission distance limitations, making it less practical for large-scale or high-security forensic applications.

Table 2 compares these related studies based on covert channel type, exploiting technique, encryption, randomization, and document type. Our approach uniquely integrates encryption and randomization within a hybrid covert channel that exploits system log files and runtime errors to conceal Digital Forensic Documents (DFDs). Unlike prior studies that focus on a single dimension—such as encryption [2] or timing control [17] our design achieves all key metrics: encryption, randomization, extraction rate with delay, and un-detectability.

Furthermore, by utilizing the Digital Forensic Legal Affairs Server (DFLAS) log file as a covert storage medium, our system securely handles DFDs of various sizes while maintaining confidentiality, plausibility, and reliability.

While prior studies have demonstrated the feasibility of covert channels in various environments, many suffer from practical limitations when applied to real-world digital forensic scenarios. Storage-based approaches, such as stack overflow exploitation, are increasingly unreliable due to modern memory randomization techniques, while timing-based channels often trade throughput for stealth. Program-flow–only approaches, although efficient, typically lack robustness when deployed outside controlled environments.

The reviewed literature shows that few studies combine multiple covert channel modalities within a single operational framework, particularly in the context of digital forensics. This gap highlights the need for hybrid designs that balance capacity, robustness, and undetectability, which directly motivates the approach proposed in this research.

Table 2:Related works comparative analysis with our work

Paper#	Covert channel type	Exploiting techniques	Encryption	Randomization	Object	Extraction rate with delay
Our work	Hybrid (storage, timing and program-flow)	Our hybrid covert channel relies on taking advantage of the DF LegalAffairs Server log file also using runtime error to store covert data into log file	yes	yes	DFD (doc & txt)	yes

Table 2:Related works comparative analysis with our work

OverCovert: Using Stack-Overflow Software Vulnerability to Create a Covert Channel [14]	storage-based covert channel	Using Linux memory randomization to create a storage covert channel by manipulating stack overflows.	No	No	audio files	no
Challenging channels: Encrypted covert channels within challenge-response authentication [15]	Storage -based covert channel	Modifying nonces in challenge-response authentication to carry covert information using compression.	Yes	No	JavaScrip t file	yes
SecureCommunicat ion Using Cryptography and Covert Channel [2]	Storage -based covert channel	Uses pre-agreement knowledge between entities, bit distribution plausibility, and covert message encoding techniques.	Yes	No	Table for user informati on (name, id)	no
Creating and Developing a High-Throughput Covert Channel via Program Execution[16]	Progra m-flow covert channel	explores the use of several encoding methods, including ASCII, Byte, Hexadecimal, Base64, and Huffman coding.	No	No	Video, Text and audio	no
VoIP network covert channels to enhance privacy and information sharing [17]	timing covert channel	exploits the Voice Activity Detection (VAD) feature used to save bandwidth by suspending the transmission of packets during speech pauses.	No	No	web pages	no
LoPhy: A Resilient and Fast Covert Channel Over LoRa PHY [19]	Storage -based covert channel	They implemented their channel over the IoT physical layer.	No	No	Bit rate	Yes

7. CREATING A HYBRID COVERT CHANNEL SCENARIO VIA EXPLOITING A DFS LOG FILE.

7.1 Research Methodology and Protocol

This study follows a design-and-evaluate experimental methodology. First, a hybrid covert channel architecture was designed by integrating storage, timing, and program-flow channels within a Distributed File System (DFS) environment. Second, a proof-of-concept implementation was developed involving three entities: the Digital Forensic Team (DFT), the Digital Forensic Legal Affairs Server (DFLAS), and the Technical Analysis Department (TAD). Third, controlled experiments were conducted using varying file sizes, delay intervals, and log generation times to evaluate performance metrics, including transmission time, extraction rate, and throughput. Finally, the results were analyzed and compared with existing covert channel approaches to assess effectiveness and limitations.

The methodology of creating a hybrid covert channel that allows DFT to transmit DFD to TAD via DFLAS is shown in Fig. 2. Storage, timing, and program flow are channels that make up our hybrid covert channel. We kept the secret information in the DFLAS log file in the storage channel. Since the data is transmitted and received in accordance with a certain time, it is regarded as a timing channel. The control of computer code execution is channeled through runtime errors, it is regarded as a program flow channel.

The covert channel is developed based on three entities, which are DFT, DFLAS, and TAD. DFT is in assigned to preparing the forensic documents that will be transmitted to TAD over the DFLAS covert channel.

The following steps need to be taken on the DFT side:

- The DFT software side has launched the a ctual file.
- Next, we separated the actual file into 4k b locks.
- We add a random two-byte to each block t o introduce noise.
- Next, the RC6 encryption technique was u sed to encrypt the block, including noise. with a lot of noise mixed in, the encrypted message would be indistinguishable from other traffic.
- The encrypted message is sent to the DFL AS.

The following are the responsibilities of DFLAS:

- Acquired every block from DFT and adde d a timestamp to it.
- Catching and writing to the DFLAS log fi le by exploiting runtime errors (division b y zero).

The procedures to be performed on the TAD side are as follows:

- Connecting to the DFLAS side to catch th e log file, where this step is considered an assumption.
- We divided the log file into blocks; each b lock size is 4152.
- Then decrypted the block, including noise and timestamp, using the RC6 encryption algorithm.
- We remove a noise by removing the rando m two bytes.
- Then getting the actual data depending on the timestamp.

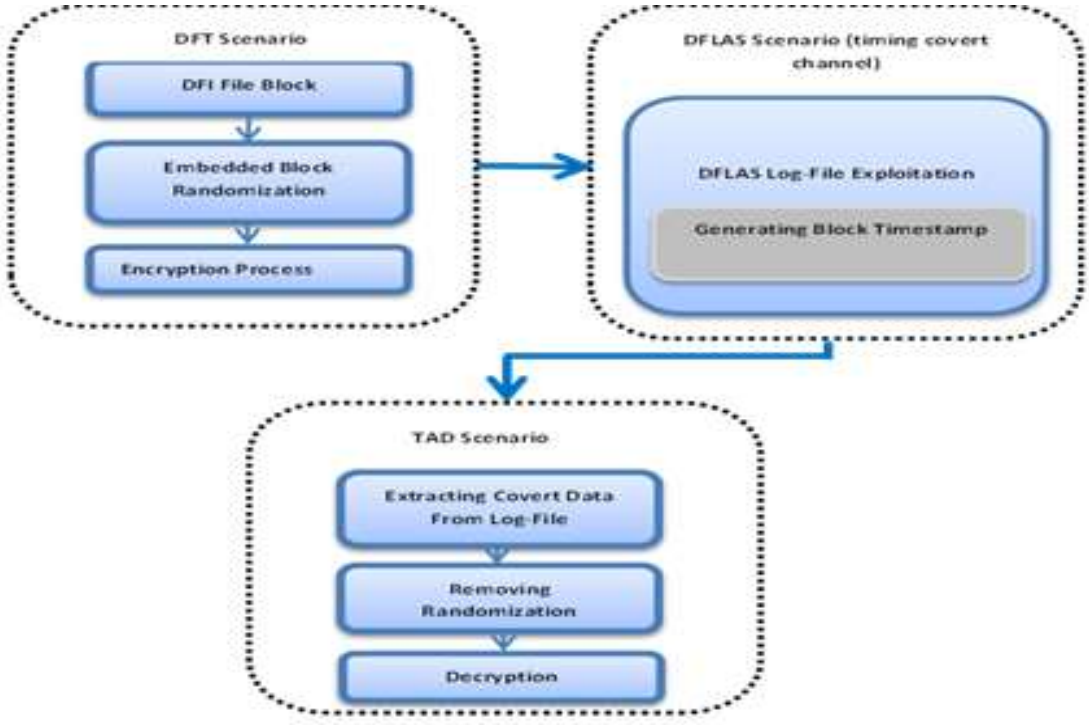

Figure 2: Developing a timing covert channel scenario via exploiting a DFS log file

8. EXPERIMENTAL RESULTS ANALYSIS

8.1 Experimental Metrics

We evaluate the time performance of DFT, DFLAS, and TAD for developing an undetectable covert channel. Additionally, we investigated the extraction rate of log files using different time delays. We use a proof-of-concept technique using a laptop with the following features: Intel(R) Core (TM) i3 CPU @ 2.00GHz, 250GB SSD hard disk, 2GB RAM, and the OS is Windows 10. Table 3 indicates main processes for creating and obtaining covert data on each side. As a result, we are tracking how long it takes the three entities (DAD, DELAS, and DFT) to complete their tasks.

Table 3: The processes for creating and obtaining covert data on each side.

DFT	DFLAS	TAD
• Encryption • Randomization • Sending to DFLAS	• Receiving from DFT. • Creating covet channel I. Generating timestamp for each block II. Embedded timestamp with block III. Using run time errors to store block in log file.	the time needed for: • Connection to system log file. • Retrieving covert data • Decryption • Derandomization

8.2 Experimental Results

The results of our research are summarized and illustrated in Table 4 and Figs. 3, 4, 5, and 6. The time required to transfer DFD (KB size) across the covert channel is demonstrated in Fig. 3 The log file sizes, which vary from 56 KB to 448 KB, are shown by the line flowchart x-axis in Fig. 3. The original file transmission time is shown on the y-axis. We assess the time in DFT, DFLAS, and TAD, which are considered to be the three components required to produce a covert channel. The timings needed for DFT to send 448 KB to TAD via DFLAS are 60 ms, 99 ms, and 182 ms, respectively. It is obvious that the majority of time is spent in creating the covert channel (DFLAS) is spent in creating the covert channel (DFLAS) through timestamps embedded and runtime errors. For evaluating the effect on time performance, we enlarged the file size. The time required to transfer DFD (MB size) across the covert channel is demonstrated in Fig. 4. The log file sizes, which vary from 14.1 MB to 112 MB, are shown by the line flowchart x-axis in Fig. 4. The original file transmission time is shown on the y-axis. The timings needed for DFT to send 112MB to TAD via DFLAS are 4019 ms, 5616 ms, and 4033 ms, respectively. It is obvious that when the file increased, the TFD needed more time analyzing and retrieving data. Additionally, as file size increased the DFT and DFLAS time are approximately same.

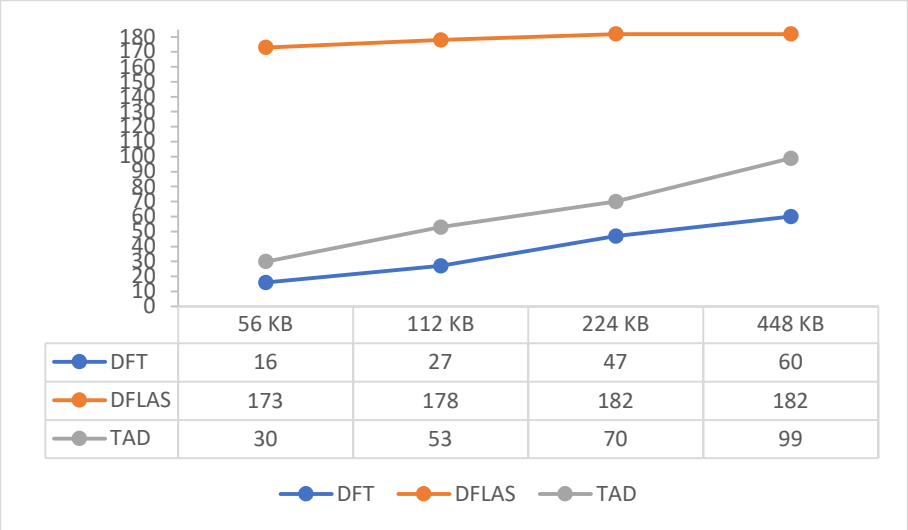

Figure 3: Depicts the time required to transfer KB DFD size in a line chart.

Figure 4: Depicts the time required to transfer MB DFD size in a box plot

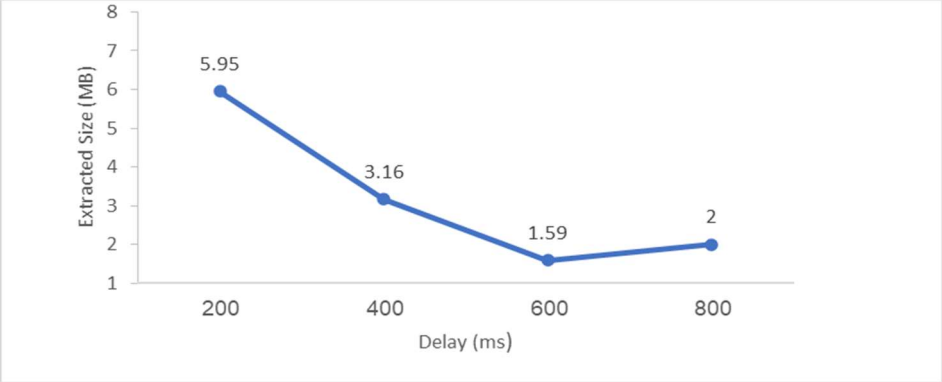

Figure 5: Depicts the actual extracted date during several delay rations in a line Chart

The extraction of covert data between TAD and DFLAS from the log file is depicted in Fig. 5, Fig. 6 and Table 4. The time at which the log file is generated and the delay ratio are the two parameters used in the extraction process. The time delay interval for generating a log file varies from 200 ms to 800 ms, as shown by the line flowchart x-axis in Fig. 5. The extracting information rate (actual data) is shown on the y-axis. In Table 4 and Fig. 5 we fixed the log file generating time in this experiment and varied the delay intervals. TAD can extract approximately 5.95 MB, 3.16 MB, and 1.59 MB from the 114 MB

Table 4: Relation between extracted actual data with delay time

File Size (MB)	Log file Size (MB)	Extracted Size (MB)	Delay (ms)
112	114	5.95 3.16 1.59 2.00	200 400 600 800

We fixed the delay intervals (delay (200 ms)) and changed the log file generating time (T1 to T9) as depicted in Fig. 6. where T denotes the moment the log file-generating process occurs or the log file's timestamp is inserted. TAD can extract approximately 5.95 MB, 3.16 MB, and 1.59 MB from the 114 MB log file using 200 ms, 400 ms, and 600 ms, respectively. It clear that as log file is increased the extraction covert data size is increased. Additionally, a crucial factor in modifying the extraction ratio is the log file generation time. The extraction ratio varied roughly between [88MB-125MB] when utilizing various T and fixed log file sizes which is 457MB.

Journal of Theoretical and Applied Information Technology
28th February 2026. Vol.104. No.4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

8.3 Differences From Prior Research

Unlike existing covert channel studies that focus on a single communication mechanism—such as storage-based or timing-based channels—this research introduces a hybrid covert channel that integrates storage, timing, and program-flow techniques within a unified digital forensic framework. Prior works exploiting memory manipulation, protocol fields, or application-layer timing patterns often suffer from limited throughput, reduced robustness, or high detectability in monitored environments.

In contrast, the proposed approach uniquely exploits runtime-error–generated log files within the Digital Forensic Legal Affairs Server (DFLAS) as a covert carrier. This design choice enhances plausibility, since log files and runtime errors are part of normal system behavior, and improves undetectability compared with memory- or protocol-based channels. Furthermore, unlike approaches that lack encryption or randomization, this work integrates RC6 encryption and noise injection, resulting in stronger confidentiality and resistance to forensic and intrusion detection analysis. Experimental results confirm that the proposed method achieves higher throughput and more stable performance than many previously reported covert channel techniques.

8.4 Bandwidth Measurements

We measured the bandwidth of the covert channel using Tsai and Gligor's methods [20]-[21] and added the additional time required for encryption and noise as depicted in Eq.1:

$$B = \frac{b}{T_R + T_E + T_N + T_S + T_{CS} + T_{log}} \quad \textbf{[Eq.1]}$$

b: number of bits transmitted per second between DFT and DFLAS

TS: time for reading data from the DFT channel and sending it to DFLAS.

to create a covert channel between DFT and DFLAS

TR: Receiving data from DFT.

TN: Time for performing noise process.

TE: time for performing the encryption process.

TCS: time for a context switch between DFT and DFLAS.

Tlog: time for writing on the log file (during exploiting run time errors) and embedded timestamp.

Since it encompasses all operational procedures, developing a covert channel is mostly calculated on the DFALS side. For example, as depicts in Fig. 4, it takes 4033 ms to send 112 MB in order to generate the cover data in the DFALS log file. Consequently, B = 112 MB / 4033 ms = 939,524,096 bits / 4.033 s ≈ 232,959,111 bits/second.

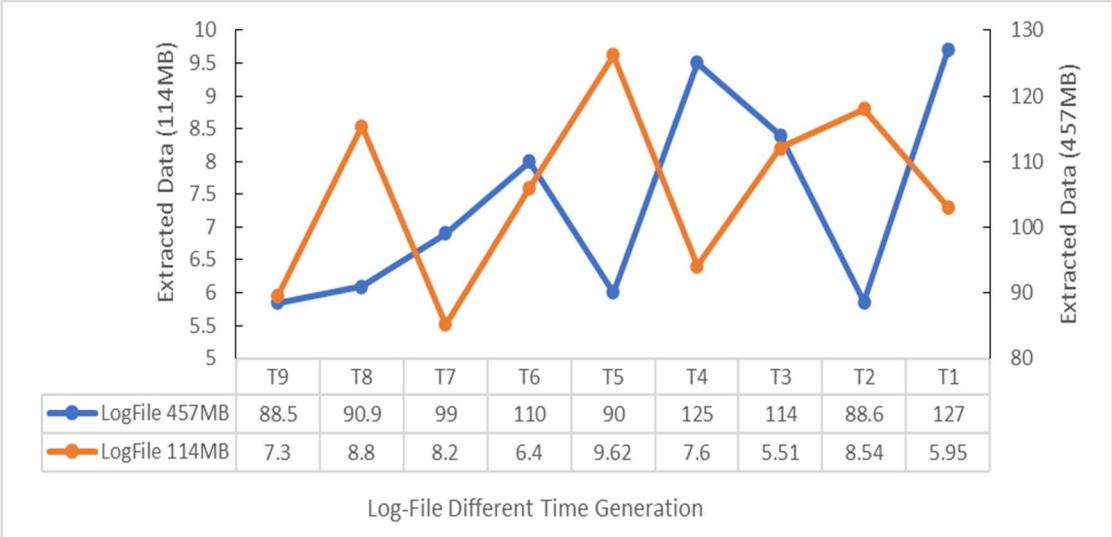

Figure 6: Actual extracted date for each time generation of the log file with varying log file sizes

TAD is in charge of data recovery and covert data extraction. Instead of 112 MB, the file size is now 114 MB, which is the size of the log file. 956,301,312 bits/5.616s ≈ 170281572 bits/second = B=114MB/5616ms. Fig. 7 shows how the DFLAS throughput performance and data size are related. The y-axis shows throughput rate, and the x-axis shows various log file sizes, from 56 KB to 448 KB. The throughput performance over a range of data sizes, from 14.1 MB to 112 MB, is shown in Fig. 8. The y-axis shows the throughput Figs., and the x-axis shows the data sizes. The throughput is almost consistent during the trend's first plateau, which occurs between 14.1 MB (180.58 million) and 28.2 MB (180.86 million). But at 56.4 MB (214.57 million), there is a noticeable increase, and it keeps going up at 112 MB (232.96 million).

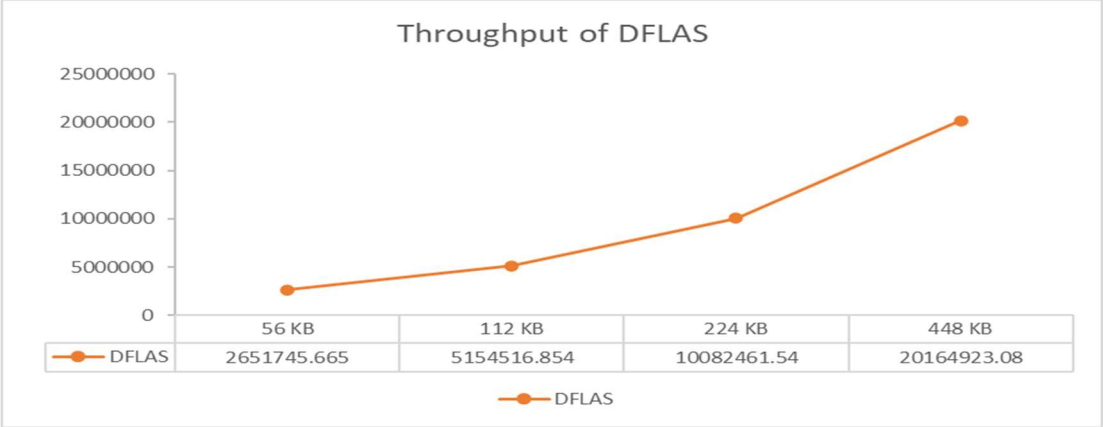

Figure 7: Throughput of DFLAS using DFD KB size

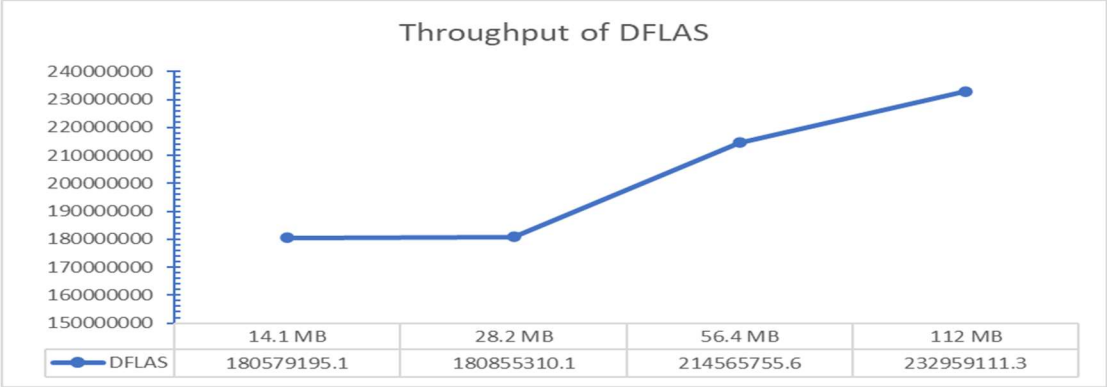

Figure 8: Throughput of DFLAS using DFD MB size

Figs. 7 and 8 summarize and show the outcomes of our covert channel throughput. The throughput of transferring DFD (KB size) across the covert channel is shown in Fig. 7. In DFLAS, we evaluate the throughput. Sending 448 KB has a throughput of 20164923.08. The throughput of transferring DFD (MB size) across the covert channel is shown in Fig. 8. Sending 112 MB has a throughput of 232959111.3.

8.5 Covert Channel Discussion and Undetectably

In Table 5, we explored the measurement performance of several studies that used covered channels in sending and receiving data between entities. The metrics parameters are time performance, throughput, and paper limitations. We mean by authentication that the authorized entities are not allowed to use the covert channel. In our case, we measure the time performance for transferring information between DF entities. We measure the throughput performance for sending data between DFT and DFLAS entities as future work. Time and throughput performance is acceptable. But we need to achieve integrity. It's clear that our covert channel is flexible and not limited in sending information because we used log files and run-time errors.

Table 5. Covert channel measurement comparative analysis

Paper#	Time	Throughput	Authentication	Limitations
Our works	We measure the time performance for sending data between DF entities.	We measure the throughput performance for sending data between DFT and DFLAS entities	As future work.	Time and throughput performance is acceptable. But we need to achieve integrity.
OverCovert: Using Stack-Overflow Software Vulnerability to Create a Covert Channel [14]	Time performance is computed for transferring audio file.	Throughput not computed	Not computed	The limitation in time performance due to the fixed size of Linux memory
Challenging channels: Encrypted covert channels within challenge-response authentication [15]	Time performance is computed for transferring web pages file.	Is measured	Not computed	The limitation in throughput and time performance due to modification in the challenge-response protocol is difficult.
Secure Communication Using Cryptography and Covert Channel [2]	The time of handling multiuser.	Not computed	computed	The time required to agree on shared data not computed. Also, the used pre-agreement is fixed
A key-agreement protocol based on the stack-overflow software vulnerability [20]	Time performance is computed to measure the time for agreeing on keys between two entities	Throughput is not computed to measure the bandwidth for transferring keys between client and server.	computed	The time required to agree on 16 bits is approximately 5 seconds.

We were able to achieve hybrid covert channel un-detectability for a variety of reasons:

- Confidentiality and stealth. To make sure the channel remains concealed from detection, we employed an encryption procedure. Our covert channel is a typical channel with very normal traffic patterns.
- Randomization.

  a. With a block size of 4KB, we incorporated 2-byte randomization into every block of the original DFD.

  b. In order to create randomization in the covert data extraction, we employed multiple times for the generation log file. Additionally, this makes the log file's content randomized.
- Plausibility: Since the log file and runtime errors are a component of DFLAS, our covert channel is observed as normal. The attacker tries to pass the log file, which is already on the network.

9. CONCLUSION AND FUTURE WORKS

This research proposed a hybrid covert channel architecture that combines storage, timing, and program-flow mechanisms to securely transmit Digital Forensic Documents (DFDs) between forensic entities. The study successfully achieved its stated objectives by demonstrating that exploiting runtime-error–generated log files within DFLAS can provide a plausible, undetectable, and efficient communication medium.

Experimental results confirmed that the proposed approach delivers competitive transmission time and high throughput, with the transfer of a 112 MB file requiring approximately 4000 ms and achieving a throughput of 232.9 Mbps. Compared with prior studies, the proposed method offers improved scalability and robustness, particularly for large forensic documents. However, the approach assumes controlled access to system log files and may be affected by aggressive log sanitization or

advanced SIEM-based anomaly detection, which represent limitations and threats to validity.

Future research will focus on enhancing integrity verification, authentication between DF entities, and adaptive timing mechanisms to further improve resilience against evolving detection techniques.

REFERENCES:

[1]. Singh, A., Ikuesan, R. A., & Venter, H. (2022). Secure storage model for digital forensic readiness. IEEE Access, 10, 19469–19480. https://doi.org/10.1109/ACCESS.2022.3148842

[2]. Fatayer, T. (2020). Secure communication using cryptography and covert channel. Journal of Computer and Network Security. https://doi.org/10.5772/intechopen.82580

[3]. Carrara, B., & Adams, C. (2016). A survey and taxonomy aimed at the detection and measurement of covert channels. In Proceedings of the 2016 ACM Workshop on Information Hiding and Multimedia Security (pp. 115–126). https://doi.org/10.1145/2909827.2930800

[4]. Jadhav, M. V., & Kattimani, S. L. (2011). Effective detection mechanism for TCP based hybrid covert channels in secure communication. In 2011 International Conference on Emerging Trends in Electrical and Computer Technology (pp. 1123–1128). IEEE. https://doi.org/10.1109/ICETECT.2011.5760288

[5]. Kaur, M., & Singh, H. (2017). A review on covert timing channels & their applications. International Advanced Research Journal in Science, Engineering and Technology, 4(6), 181–185.

[6]. Sharma, B. K., Joseph, M. A., Jacob, B., & Miranda, B. (2019). Emerging trends in digital forensic and cybersecurity: An overview. In 2019 Sixth HCT Information Technology Trends (ITT) (pp. 309–313). IEEE. https://doi.org/10.1109/ITT48889.2019.9075101

[7]. Bhatele, K. R., Jain, S., Kataria, A., & Jain, P. (2020). The fundamentals of digital forensics. In B. Gupta & D. Gupta (Eds.), Handbook of research on multimedia cyber security (pp. 165–175). IGI Global. https://doi.org/10.4018/978-1-7998-2701-6.ch008

[8]. Abudu, O., Adams, A., Onyenaucheya, O., & Erinfolami, S. (2024). Digital forensics in cybersecurity.

[9]. Modi, K., Shukla, K., & Fiadufe, F. (2024). Forensic investigation and analysis of malware in Windows OS. International Journal of Electronic Security and Digital Forensics, 17, 169–182. https://doi.org/10.1504/IJESDF.2025.143477

[10]. Frude, N. (1987). Log files and listing files. In A guide to SPSS/PC+ (pp. 197–212). Palgrave Macmillan. https://doi.org/10.1007/978-1-349-09709-8_11

[11]. Duy, P. T., Do Hoang, H., Thu Hien, D. T., Ba Khanh, N., & Pham, V.-H. (2019). SDNLog-Foren: Ensuring the integrity and tamper resistance of log files for SDN forensics using blockchain. In 2019 6th NAFOSTED Conference on Information and Computer Science (NICS) (pp. 416–421). IEEE. https://doi.org/10.1109/NICS48868.2019.9023852

[12]. Nadeem, S. F., & Huang, C.-Y. (2018). Data visualization in cybersecurity. In 2018 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 48–52). IEEE. https://doi.org/10.1109/CSCI46756.2018.00017

[13]. Skopik, F., Landauer, M., & Wurzenberger, M. (2022). Blind spots of security monitoring in enterprise infrastructures: A survey. IEEE Security & Privacy, 20(6), 18–26. https://doi.org/10.1109/MSEC.2021.3133764

[14]. Fatayer, T. S., Khattab, S., & Omara, F. A. (2011, February). OverCovert: Using stack-overflow software vulnerability to create a covert channel. In 2011 4th IFIP International Conference on New Technologies, Mobility and Security (pp. 1–5). IEEE. https://doi.org/10.1109/NTMS.2011.5720625

[15]. Schmidbauer, T., Keller, J., & Wendzel, S. (2022, August). Challenging channels: Encrypted covert channels within challenge-response authentication. In Proceedings of the 17th International Conference on Availability, Reliability and Security (pp. 1–

10). ACM. https://doi.org/10.1145/3538969.3543801

[16]. Alhelal, A., & Al-Khatib, M. (2024). Creating and developing a high-throughput covert channel via program execution. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3375113

[17]. Saenger, J., Mazurczyk, W., Keller, J., & Caviglione, L. (2020). VoIP network covert channels to enhance privacy and information sharing. Future Generation Computer Systems, 111, 96–106. https://doi.org/10.1016/j.future.2020.04.013

[18]. Fatayer, T. S., Khattab, S., & Omara, F. (2010). A key-exchange protocol based on the stack-overflow software vulnerability. In 2010 IEEE Symposium on Computers and Communications (ISCC) (pp. 411–416). IEEE. https://doi.org/10.1109/ISCC.2010.5546530

[19]. Liu, B., Gu, C., He, S., & Chen, J. (2024). LoPhy: A resilient and fast covert channel over LoRa PHY. IEEE/ACM Transactions on Networking, 7–16. https://doi.org/10.1109/TNET.2024.3398814

[20]. Tsai, C.-R., & Gligor, V. D. (1988). A bandwidth computation model for covert storage channels and its applications. In Proceedings of the 1988 IEEE Symposium on Security and Privacy (pp. 108–121). IEEE.

[21]. Tsai, C.-R. (1987). Covert-channel analysis in secure computer systems (Doctoral dissertation). University of Maryland, College Park.