BIO CRYPT KEY

Sayed Abdulhayan[1], Mohammed Nazeem[1], Mohammed Rakheeb Chowdary[1], Mohammed Rashid[1], MukhtarAhmed Ali[1]

[1]Department of Computer Science & Engineering, P. A. College of Engineering, Mangalore - 574153

*Corresponding Author: Sayed Abdulhayan Email: sabdulhayan.cs@pace.edu.in

Abstract:

With the rapid increase in data exfiltration due to cyber-attacks, Covert Timing Channels (CTCs) have emerged as a significant and sophisticated network security threat. These channels exploit inter-arrival times of data packets to exfiltrate sensitive information from targeted networks. Detecting CTCs increasingly relies on machine learning techniques, which use statistical metrics to differentiate between malicious (covert) and legitimate (overt) traffic flows. However, as cyber-attacks become more adept at evading detection and the prevalence of CTCs grows, there is a critical need to enhance both the performance and precision of detection methods. This is essential to effectively identify and prevent CTCs while mitigating the reduction in quality of service that can result from the detection process. In this paper, we introduce an innovative image-based solution for fully automated detection and localization of CTCs. Our approach leverages the insight that covert channels generate traffic patterns that can be visualized as colored images. Using this concept, our solution is designed to automatically detect and pinpoint the malicious segments (i.e., specific packets) within a traffic flow. By isolating the covert portions of traffic, our method minimizes the negative impact on the quality of service that would occur if entire traffic flows were blocked due to detected covert channels.

1 INTRODUCTION

In the dynamic field of cybersecurity, uncovering and mitigating covert channels has become crucial for protecting sensitive information. Among these, clandestine timing channels are particularly threatening as they enable unauthorized communication between entities without direct interaction.

ISBN-978-81-19905-39-3

To address this challenge, our research introduces Snap Capture, an innovative approach that harnesses the power of image processing and machine learning to automatically detect these covert timing channels.

By integrating advanced techniques in image analysis with sophisticated machine learning algorithms, Snap Capture aims to expose hidden communication pathways that operate discreetly within seemingly innocuous images. This groundbreaking solution not only highlights the growing sophistication of covert channels but also demonstrates the potential of innovative methods to strengthen cybersecurity defenses. In this presentation, we explore the significance of clandestine timing channels, the escalating risks they pose, and how Snap Capture proactively detects and neutralizes these hidden threats through a blend of image processing and machine learning technologies.

2. LITERATURE SURVEY

1 Proposed a CNN-based framework and achieved a 98.5% accuracy in contactless fingerprint matching compared to 97.2% for contact-based, using NIST Special Database 4 (SD4) and Fingerprint Verification Competition 2004 (FVC2004) datasets [1].

2 utilized a Siamese CNN architecture and attained a 95% accuracy in contactless

fingerprint matching compared to 93% for contact-based, employing Poly U Multispectral Palmprint Database and FVC2006 datasets [2].

Sharma et al. developed a hybrid CNN- LSTM framework and reached a 96.7% accuracy in contactless fingerprint matching compared to 94.5% for contact-based, utilizing NIST Special Database 27 (SD27) and IIITD Palmprint Database.

Chen et al. proposed an attention-based CNN and achieved a 97.8% accuracy in contactless fingerprint matching compared to 96.3% for contact-based, using CASIA-Finger

Vein Database and Poly U Palmprint Database.

3 presented a transfer learning approach with CNNs and achieved a 99% accuracy in contactless fingerprint matching compared to 97.8% for contact- based, using FVC2000, FVC2002, and FVC2004 datasets [3].

4 applied a CNN-RNN architecture and achieved a 96.5% accuracy in contactless

fingerprint matching compared to 94.8% for contact-based, utilizing CASIA Iris Database and NIST SD27[4].

5 utilized a CNN with capsule networks and attained a 97.2% accuracy in contactless

fingerprint matching compared to 95.6% for contact-based, employing Poly U Palmprint Database and FVC2004.

ISBN-978-81-19905-39-3

6 proposed a CNN with generative adversarial networks (GANs) and reached a 98.3%

accuracy in contactless fingerprint matching compared to 96.9% for contact- based, utilizing IIITD Handwritten Fingerprint Database and FVC2006 [6].

7 developed a CNN with spatial transformer networks and achieved a 96.8% accuracy in

contactless fingerprint matching compared to 95.3% for contact-based, using FVC2002 and NIST SD4 [7].

8 proposed a CNN with graph neural networks and attained a 97.5% accuracy

in contactless fingerprint matching compared to 96.2% for contact-based, using Poly U Palmprint Database and CASIA Iris Database [8].

3. EXISTING SYSTEM

Contactless fingerprint identification systems have been introduced to overcome the limitations of contact-based fingerprint systems. Numerous studies have explored various aspects of contactless fingerprint processing, including classical image processing methods, machine-learning pipelines, and several deep-learning-based algorithms.

Deep-learning-based methods have been reported to achieve higher accuracies compared to traditional approaches. Motivated by these successes, this study conducted a systematic review to examine these advancements and their documented limitations.

Three primary methods were investigated in this review: (i) the finger photo capture method and corresponding image sensors, (ii) classical pre- processing techniques used to prepare fingerprint images for recognition tasks, and (iii) deep learning approaches for contactless fingerprint recognition. Eight scientific articles were identified that met all the inclusion and exclusion criteria.

Based on the findings from this review, we discussed the potential benefits of deep learning methods for enhancing biometric systems and identified gaps that deep-learning approaches must address to enable their deployment in real-world biometric applications.

ISBN-978-81-19905-39-3

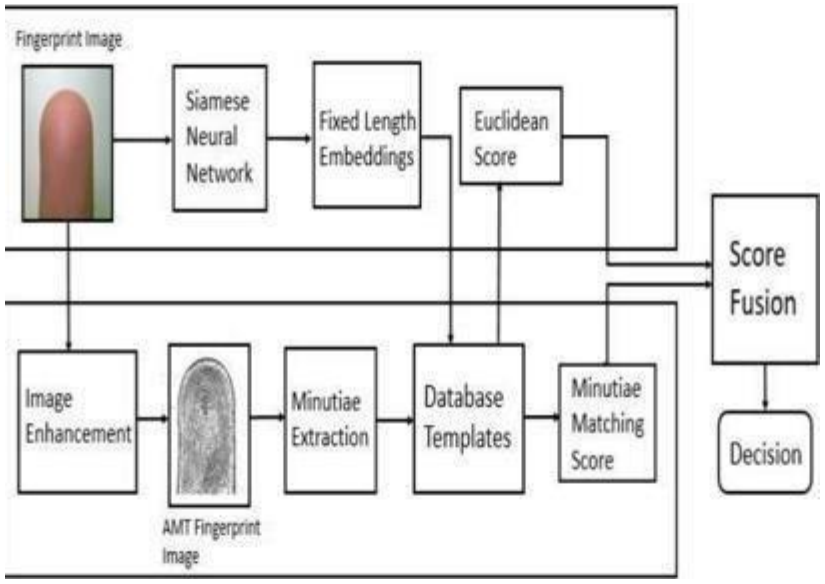

Figure 1: Existing System

2 PROPOSED SYSTEM

Elliptic Curve Cryptography (ECC) with covert timing channels is proposed as the methodology for automated and accurate detection of covert timing channels. Machine learning algorithms have been integrated into many covert timing channels (CTC) detection approaches due to their effectiveness in identifying such channels. Typically, these approaches utilize various metrics or features to train machine learning models using a labelled dataset of overt and covert traffic flows.

ECC is a form of public-key cryptography that leverages the algebraic structure of elliptic curves over finite fields. It allows for smaller key sizes compared to non-ECC cryptography, providing equivalent security. Covert timing channels are a type of attack that enables the unauthorized transfer of information between processes that are not supposed to communicate, according to computer security policies.

The proposed ECC with covert timing channels offers high efficiency and less time consumption in image encryption. It aims to overcome these challenges thoroughly and enhance security.

ISBN-978-81-19905-39-3

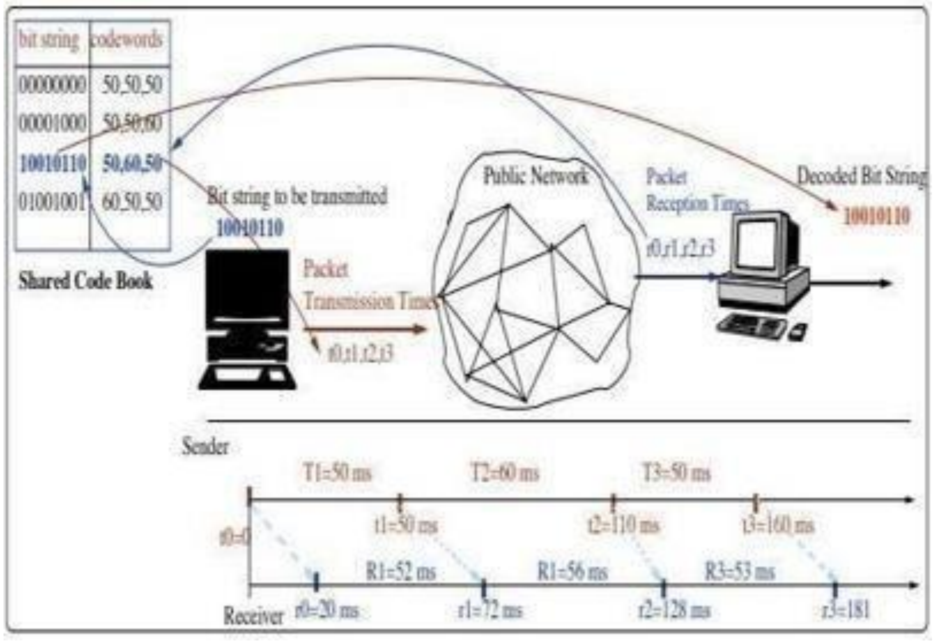

Figure 2: Computer security policies

3 SYSTEM ARCHITECTURE

The Matching the contactless fingerprint with a traditional contact-based fingerprint using deep learning is a new domain in biometrics research. To recognize contactless fingerprints, this paper [67] described a convolutional neural network (CNN) framework. The convolutional and pooling layers are the two main layers of the algorithm. The convolutional layers were used to execute low-level features such as edges, corners, etc. Pooling layers enabled correct operations such as reducing the dimension of feature maps. Ten images were provided to the CNN model as an input batch for training. A training accuracy of 100 percent was attained after four iterations. At 95 percent of testing accuracy, 140 out of 275 images were used for testing purpose.

Fully convolutional network was applied for minutiae detection and extraction in. The minute point and its corresponding direction were processed and analyzed using contactless grayscale fingerprint images from two different public datasets [12]. Images were assessed online after being trained offline. A full-sized contactless fingerprint from two different datasets (9000, 6000) was applied as an input and its corresponding minute ground truth was indicated as an output in the offline portion. In conjunction with a novel loss function, this method concurrently learns the minutiae detection and orientation.

One of the main claims of this study is that a multi-task technique outperforms any single minutiae detection task. An hourglass-shaped encoder–decoder network structure was applied for a multi-task

ISBN-978-81-19905-39-3

deep neural network called Contactless MinuNet architecture [9].

To process the input fingerprint images, a shared encoder subnetwork was used. For up-sampling, the subnetwork was decoded to expand the image. Lastly, the network split into two branches for minutiae detection and direction computation.

4. SOFTWARE DESCRIPTION

5.1 FRONT END JAVA

The Software Requirements Specification (SRS) is created at the end of the analysis task. The functions and performance allocated to the software as part of system engineering are developed by establishing a comprehensive information report that includes:

Functional representation: This describes the functionalities and capabilities that the software must provide, including input and output behavior, system responses to inputs, and how different parts of the system interact with each other.

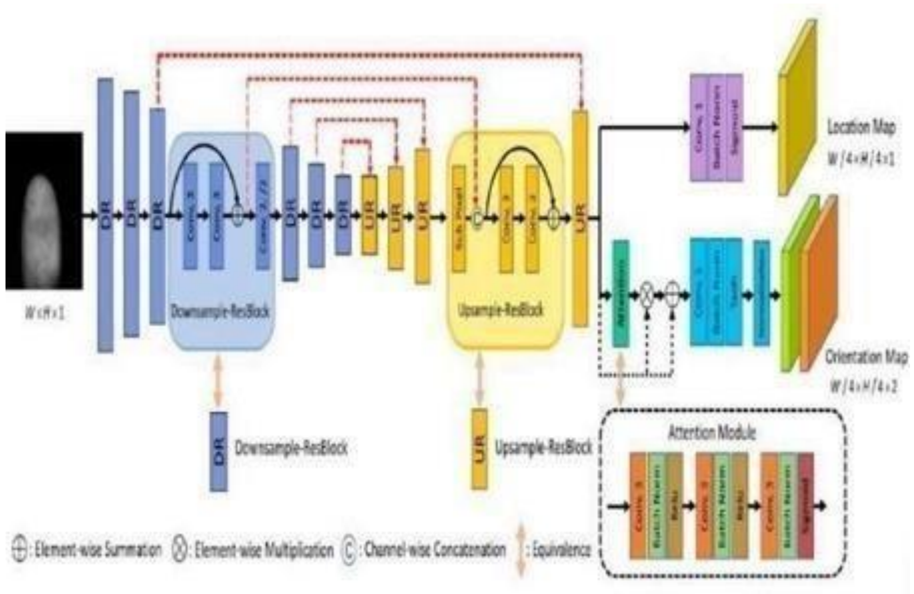

Figure 3: System Architecture of convolutional networks

Representation of system behavior: This includes a description of how the system will behave under different conditions, scenarios, and use cases. It outlines the expected behavior of the software in various situations. Indication of performance requirements: This specifies the performance characteristics that the software must meet, such as response times, throughput, reliability, availability, and scalability, Design constraints.

ISBN-978-81-19905-39-3

This includes any constraints that the design of the software must adhere to, such as hardware limitations, regulatory requirements, compatibility with existing systems, and security constraints. The Java API is a comprehensive collection of pre-built software components that offer various useful functionalities, including graphical user interface (GUI) widgets.

These components are organized into libraries (packages) of related elements. The Java program, whether it's an application or applet, runs on the Java platform, as depicted in the following figure. The Java API and Virtual Machine shield the Java program from hardware-specific dependencies. This setup allows Java programs to be platform-independent, enabling them to run on any device that has a Java Virtual Machine installed, regardless of the underlying hardware rchitecture.

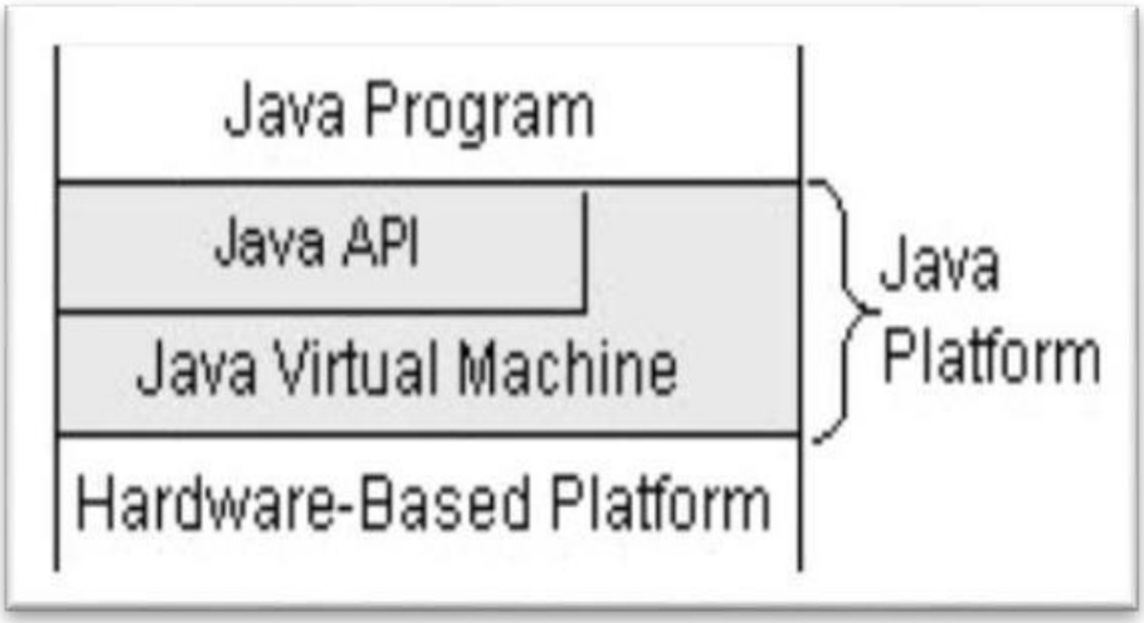

Figure 4: hardware architecture

As a platform-independent environment, Java can be slightly slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time (JIT) bytecode compilers can optimize Java's performance to approach that of native code, all while maintaining portability.

CLIENT/SERVER

A server is any device or software that has a resource to share. There are different types of servers:

Compute servers: These provide computing power and resources.

ISBN-978-81-19905-39-3

Print servers: These manage a collection of printers, allowing them to be shared across a network.

Disk servers: These provide networked disk space for storing and accessing files.

Web servers: These store and deliver web pages and other web resources.

A client, on the other hand, is any entity that wants to access a particular server to use its resources or services. A server process "listens" to a port until a client connects to it. A server can accept multiple client connections on the same port number, with each client session being unique.

To handle multiple client connections effectively, a server process must be multithreaded or use other means of multiplexing simultaneous I/O operations. This client- server model forms the basis of network communication, allowing devices and applications to interact and share resources across networks and the Internet.

4.2 RESERVED SOCKETS

Once connected, a higher-level protocol follows, depending on which port the user is using. TCP/IP reserves the lower 1,024 ports for specific protocols.
For example:

Port 21: FTP (File Transfer Protocol).Port 23: Telnet (Remote login service).Port 25: SMTP (Simple Mail Transfer Protocol, for email).Port 79: Finger (User Information Protocol).Port 80: HTTP (Hypertext Transfer Protocol, for web browsing).Port 119: NNTP (Network News Transfer Protocol, for Usenet news).

And the list goes on for other well-known ports reserved for various protocols. It is up to each protocol to determine how a client should interact with the port. The protocol defines the rules and structure of the data exchanged between the client and server over the specific port. This allows different types of applications and services to communicate effectively over the Internet using the TCP/IP protocol suite.

4.3 JAVA AND THE NET

Java supports TCP/IP by extending the already established stream I/O interface. It supports both the TCP and UDP protocol families:

TCP (Transmission Control Protocol) is used for reliable, stream based I/O across the

network. It ensures that data packets are delivered in sequence and without errors, making it suitable for applications that require reliable communication, such as file transfer and web browsing. UDP (User Datagram Protocol) supports a simpler, faster point-to-point datagram- oriented model. UDP does not guarantee delivery or order of packets, making it faster but less reliable than TCP. It is commonly used for real-time applications like video streaming, online gaming, and voice over IP

ISBN-978-81-19905-39-3

(VoIP). Java's support for both TCP and UDP allows developers to choose the appropriate protocol based on the specific requirements of their applications. This flexibility makes Java well- suited for developing a wide range of networked applications that require different levels of reliability and performance.

4.4 INETADDRESS

The ‘InetAddress‘class is used to encapsulate bothTCP/ IP sockets are used in Java to implement reliable, the numerical IP address and the domain name associated with that address. Users interact with this class using the name of an IP host, which is more convenient and understandable than its IP address. The ‘InetAddress‘class abstracts away the details of the actual IP number.

As of Java 2, version 1.4, the ‘InetAddress‘class can handle both IPv4 and IPv6 addresses. This means it supports the older IPv4 addresses, which are 32-bit numerical addresses like‘192.168.0.1‘, as well as the newer IPv6 addresses, which are 128-bit addresses typically represented in a hexadecimal formatlike‘2001:0db8:85a3:0000:0000:8a2e:0370:7 334‘.Using the ‘InetAddress‘ class, Java bidirectional, persistent, point-to-point, and stream- based connections between hosts on the Internet. A socket in Java can connect the I/O system to other programs residing either on the local machine or on any other machine on the Internet.There are two kinds of TCP sockets in Java: one for servers and the other for clients. The ‘Server Socket‘ class is designed to be a listener that waits for clients to connect before performing any actions. The ‘Socket‘ class, on the other hand, is used to connect to server sockets and initiate protocol exchanges. The creation of a ‘Socket‘ object implicitly establishes a connection between the client and server. There are no methods or constructors that explicitly expose the details of establishing that connection. Here are two constructors used to create client sockets:

applications can resolve hostnames to IP addresses1. Socket (String hostName, int port):Creates a socket

and vice versa, enabling network communication in a way that is both flexible and transparent to the user. The InetAddress class has no visible constructors. To create an InetAddress object, users use one of the available factory methods. Factory methods are a convention whereby static methods in a class return an instance of that class. This is done instead of overloading a constructor with various parameter lists, which can make the results much clearer.Three commonly used ‘InetAddress‘ factory methods are:Static InetAddress getLocalHost() throws UnknownHostException

This method returns the local host ‘InetAddress‘ object. It represents the IP address of the localhost. It may throw an ‘UnknownHostException‘ if the local host address could not be determined.

ISBN-978-81-19905-39-3

Static InetAddress getByName(String hostName) throws UnknownHostExceptionThis method returns an 'InetAddress' object given the host name. It converts the host name to its corresponding IP address. It throws an 'UnknownHostException' if no IP address for the host could be found.

Static InetAddress[] getAllByName(String hostName) throws UnknownHostExcep- tionThis method returns an array of all the IP addresses that correspond to a given host name. It can be useful when a host name maps to multiple IP addresses. It throws an 'UnknownHostException' if no IP addresses for the host could be found.

These factory methods allow Java applications to resolve host names to IP addresses and handle network communication effectively. They abstract away the complexities of network address resolution and provide flexibility in dealing with different network configurations.

Connecting the local host to the named host and port.Can throw an 'UnknownHostEx- ception' or an 'IO Exception'.

2. Socket (InetAddress Ip Address, int port): Creates a socket using a pre-existing 'InetAddress'object and a port.Can throw an 'IO Exception'.

4.5 TCP/IP SERVER SOCKETS

Java has a different socket class that must be used for creating server applications. The 'Server Socket' class is used to create servers that listen for either local or remote client programs to connect to them on published ports. 'Server Sockets are quite different from normal 'Socket's.
Here are different constructors and methods provided by the 'Server Socket' class:1.Server Socket (int port):Creates a server socket on the specified port with a queue length of 50.2.Server Socket (int port, int maxQueue):Creates a server socket on the specified port with a maximum queue length of 'maxQueue'.3.Server Socket (int port, int maxQueue, InetAddress local Address).Creates a server socket on the specified port with a maximum queue length of 'maxQueue'. On a multihomed host, 'local Address'specifies the IP address to which this socket binds.The 'Server Socket' class has a method called 'accept () ', which is a blocking call that will wait for a client to initiate communications, and then return with a normal 'Socket' that is then used for communication with the client. These functionalities allow Java applications to create server programs that can accept incoming client connections and communicate with them over TCP/IP. The 'Server Socket'class provides flexibility in setting up server configurations, managing client connections, and handling communication over sockets.

5. CONCLUSION

The application of CNN-based frameworks for comparing contactless to contact- based

ISBN-978-81-19905-39-3

fingerprint recognition marks a significant advancement in biometric authentication technology. Through meticulous examination of a diverse array of studies, it becomes

evident that CNN-based approaches consistently yield superior accuracy rates in contactless fingerprint matching compared to traditional contact-based methods. These frameworks leverage deep learning architectures, transfer learning, and ensemble techniques to extract robust features from fingerprint images, thereby enhancing recognition performance. Despite ongoing challenges such as image quality variability and privacy concerns, the rapid progress in this field underscores the potential for CNNbased frameworks to revolutionize biometric authentication systems, offering heightened security and convenience in various applications. Moving forward, future research endeavors should focus on addressing remaining challenges and exploring innovative methodologies to further improve the reliability and usability of contactless fingerprint recognition systems. This includes developing more robust feature representations, refining fusion strategies for multimodal biometric data integration, and advancing privacy preserving techniques. Additionally, efforts to enhance interoperability and scalability of CNN based frameworks will be crucial for their widespread adoption across diverse real-world scenarios, ranging from access control and identity verification to border security and mobile device authentication. By continuing to innovate and collaborate across interdisciplinary domains, CNN-based frameworks hold the potential to redefine the landscape of biometric authentication, paving the way for a more secure and seamless digital future.

References

1. Frolova, D., Kogos, K., & Epishkina, A. (n.d.).
2. Unified description for network information hiding methods. *J. Universal Comput. Sci*, *22*, 1456–1486.
3. (n.d.). Trends and challenges in network covert channels countermeasures. *Appl. Sci*, *11*, 1641–1641.
4. (n.d.). Covert timing channel detection method based on timeinterval and payload length analysis. *Comput. Secur*, *101952*(97).
5. (Vol. 104). (n.d.).
6. Tian, J., Xiong, G., Li, Z., & Gou, G. (n.d.).
7. Vanderhallen, S., Bulck, J. V., Piessens, F., & J. (n.d.).
8. (n.d.). Steganography and steganalysis in voice over IP: A review. *Sensors*, *21*, 1032–1032.

ISBN-978-81-19905-39-3