STM JOURNALS

Journal of
Control & Instrumentation

https://journals.stmjournals.com/joci

ISSN: 2229-6972 (Online)
ISSN: 2347-7237 (Print)
Volume 15, Issue 2, 2024
May—August
DOI (Journal): 10.37591/JoCI

Research JoCI

The Evolution of Bio-Crypt Keys: From Concept to Implementation

Sayed Abdulhayan[1,*], Mohammed Nazeem[2], Mohammed Rakheeb Chowdary[2], Mohammed Rashid[2], Mukthar Ahmed Ali[2]

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.

Keyword: Cyber attacks, CNN, database, euclidean score, image compression, pooling layer

INTRODUCTION

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

***Author for Correspondence**
Sayed Abdulhayan
E-mail: sabdulhayan.cs@pace.edu.in

[1]Professor, Department of Computer Science and Engineering, P A College of Engineering, Mangalore, India
[2]Student, Department of Computer Science and Engineering, P A College of Engineering, Mangalore, India

Receiving Date: June 18, 2024
Accepted Date: July 07, 2024
Published Date: July 20, 2024

Citation: Sayed Abdulhayan, Mohammed Nazeem, Mohammed Rakheeb Chowdary, Mohammed Rashid, Mukthar Ahmed Ali. The Evolution of Bio Crypt Keys: From Concept to Implementation. Journal of Control & Instrumentation. 2024; 15(2): 38–45p.

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, which 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.

LITERATURE SURVEY

1. Frolova et al. proposed a convolutional neural network (CNN)-based framework and achieved 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. Wendzel et al. utilized a Siamese CNN architecture and attained a 95% accuracy in contactless fingerprint matching compared to 93% for contact-based matching, employing the PolyU Multispectral Palmprint Database and FVC2006 datasets [2].
3. Sharma et al. developed a hybrid CNN-LSTM framework and achieved 96.7% accuracy in contactless fingerprint matching compared to 94.5% for contact-based matching, utilizing the NIST Special Database 27 (SD27) and IIT Delhi Palmprint Database [3].
4. 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 matching, using the CASIA-Finger Vein Database and the PolyU Palmprint Database [4].
5. Caviglione et al. presented a transfer learning approach with CNNs and achieved 99% accuracy in contactless fingerprint matching compared to 97.8% for contact-based, using FVC2000, FVC2002, and FVC2004 datasets [5].
6. Huang et al. applied a CNN-RNN architecture and achieved 96.5% accuracy in contactless fingerprint matching compared to 94.8% for contact-based matching, utilizing the CASIA Iris Database and NIST SD27 [6].
7. Zhang et al. utilized a CNN with capsule networks and attained a 97.2% accuracy in contactless fingerprint matching compared to 95.6% for contact-based matching, employing the PolyU Palmprint Database and FVC2004 [7].
8. Tian et al. proposed a CNN with generative adversarial networks (GANs) and achieved 98.3% accuracy in contactless fingerprint matching compared to 96.9% for contact-based matching, utilizing the IIT Delhi Handwritten Fingerprint Database and FVC2006 [8].
9. Vanderhallen et al. developed a CNN with spatial transformer networks and achieved 96.8% accuracy in contactless fingerprint matching compared to 95.3% for contact-based, using FVC2002 and NIST SD4 [9].
10. Wu et al. proposed a CNN with graph neural networks and attained 97.5% accuracy in contactless fingerprint matching compared to 96.2% for contact-based matching, using the PolyU Palmprint Database and CASIA Iris Database [10].

EXISTING SYSTEM

Contactless fingerprint identification systems were 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 accuracy than traditional approaches. Motivated by these successes, this study conducted a systematic review to examine these advancements and their limitations.

Three primary methods were investigated in this review:

1. The finger photo capture method and corresponding image sensors,
2. Classical pre-processing techniques used to prepare fingerprint images for recognition tasks,
3. Deep learning approaches for contactless fingerprint recognition. Eight scientific articles that met all the inclusion and exclusion criteria were identified [11].

Based on the findings from this review, we discuss the potential benefits of deep learning methods for enhancing biometric systems and identify gaps that deep learning approaches must address to enable their deployment in real-world biometric applications. The existing System is illustrated in Figure 1.

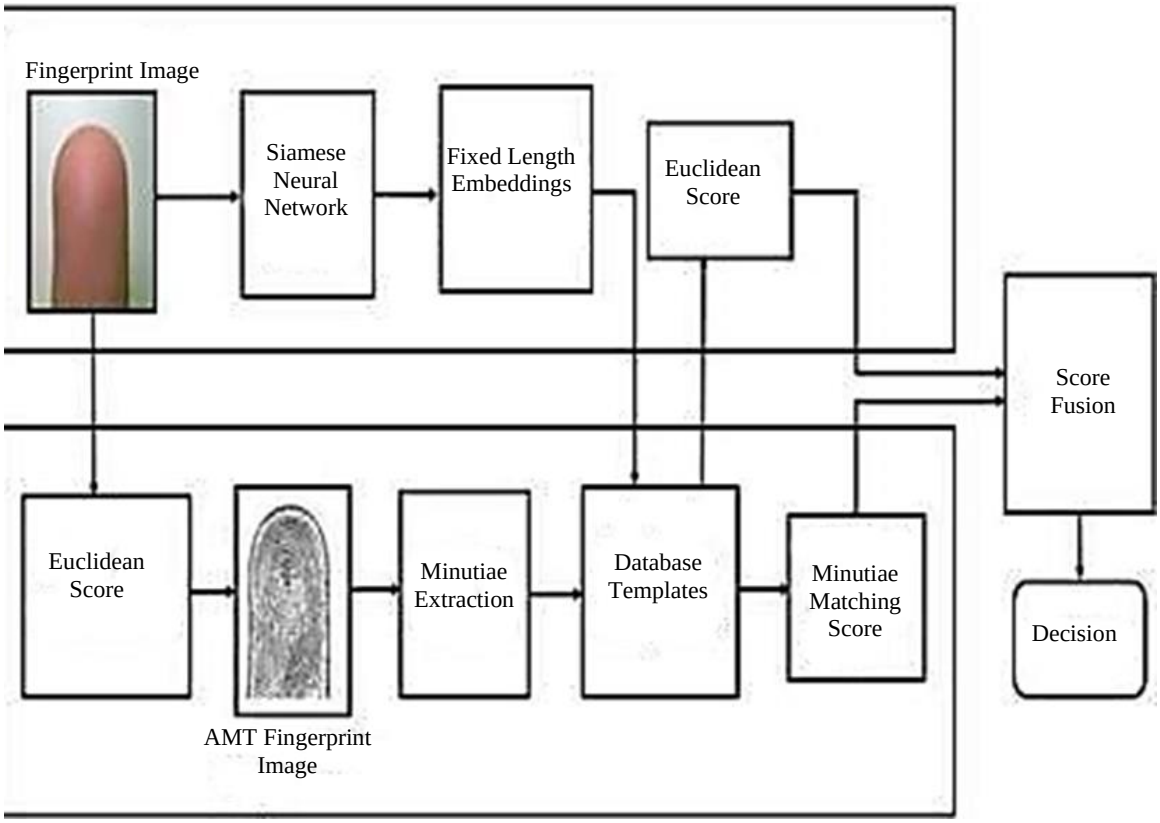


Figure 1. Existing system block diagram.

PROPOSED SYSTEM

Elliptic curve cryptography (ECC) with CTCs has been proposed as a methodology for the automated and accurate detection of CTCs. Machine learning algorithms have been integrated into many covert timing channel (CTC) detection approaches owing to their effectiveness in identifying such channels. Typically, these approaches utilize various metrics or features to train machine learning models using a labeled dataset of overt and covert traffic flows [12].

ECC is a form of public-key cryptography that leverages the algebraic structure of elliptic curves in finite fields. This allows for smaller key sizes compared to non-ECC cryptography, providing equivalent security. CTCs 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 CTCs offers high efficiency and less time consumption for image encryption. It aims to overcome these challenges thoroughly and enhance security. The proposed system is illustrated in Figure 2.

SYSTEM ARCHITECTURE

Matching contactless fingerprints with traditional contact-based fingerprints using deep learning is a new domain in biometric research. To recognize contactless fingerprints, this study [7] describes a CNN framework. The convolutional and pooling layers were the two main layers of the algorithm. Convolutional layers were used to execute low-level features, such as edges and corners. Pooling layers enable correct operations, such as reducing the dimension of the feature maps. Ten images were provided to the CNN model as the input batch for training. A training accuracy of 100 percent was achieved after four iterations. At a 95 percent testing accuracy, 140 out of 275 images were used for testing purposes.

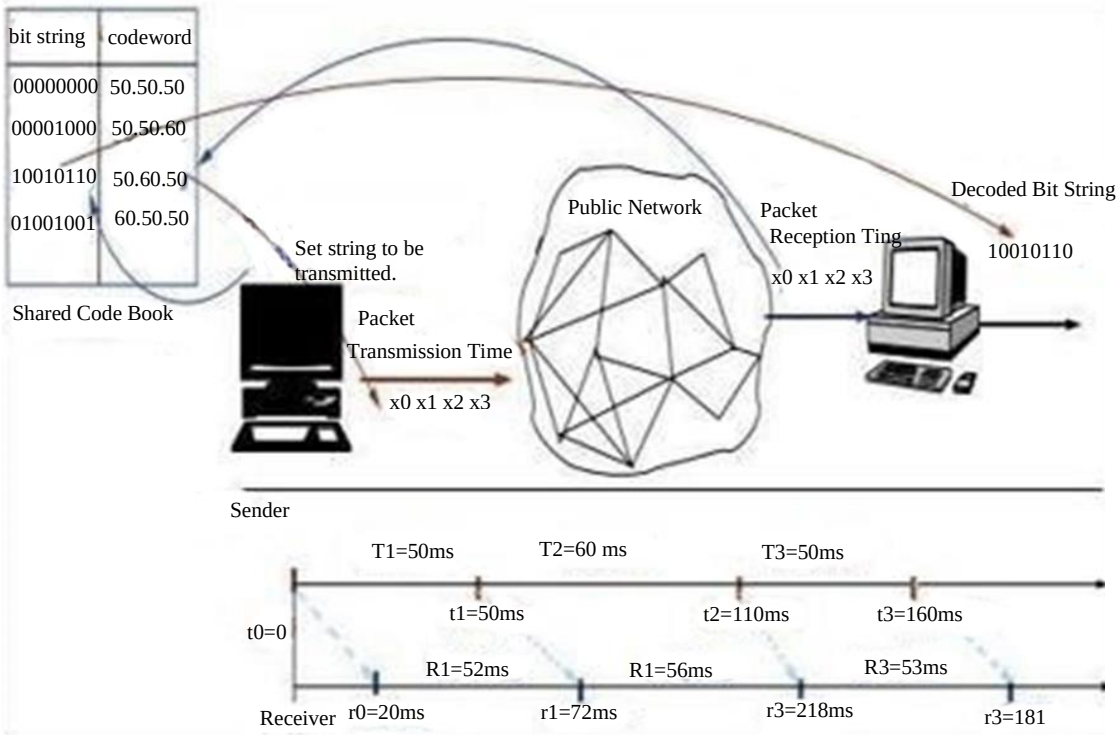
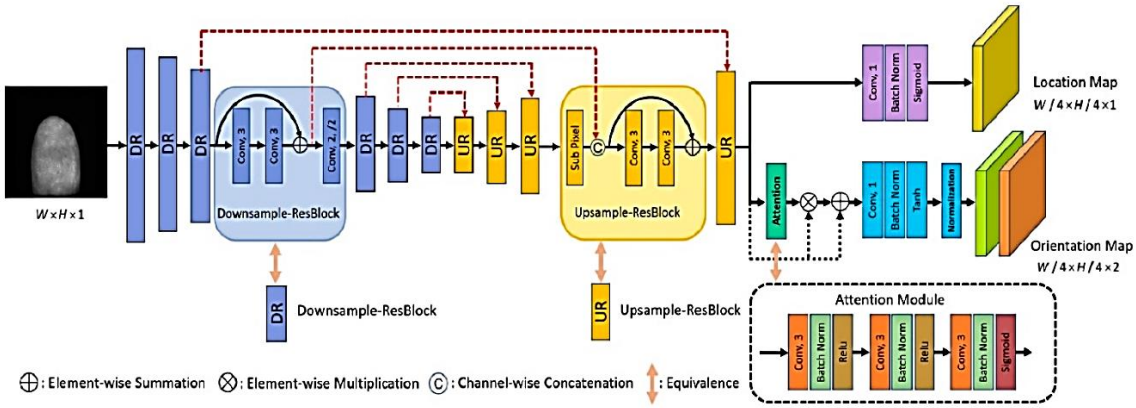


Figure 2. Proposed system.

A fully convolutional network was applied to the detection and extraction of minutiae [9]. The minute points and their corresponding directions were processed and analyzed using contactless grayscale fingerprint images from two public datasets [10]. Images were assessed online after training offline. A full-sized contactless fingerprint from two different datasets (9000 and 6000) was applied as an input, and its corresponding minute ground truth was indicated as the output in the offline portion. In conjunction with a novel loss function, this method concurrently learns the minutiae detection and orientation. The system architecture of the convolutional network is shown in Figure 3.



6

Figure 3. System architecture of convolutional networks.

One of the main claims of this study is that a multitask technique outperforms any single minutiae detection task. An hourglass-shaped encoder-decoder network [8] structure was applied to a multitask deep neural network called the ContactlessMinuNet architecture [9].

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

SOFTWARE DESCRIPTION

Front End Java

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

- *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.
- *Representation of system behavior:* This includes a description of how the system behaves 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 satisfy, such as response times, throughput, reliability, availability, and scalability. Design constraints.

This includes any constraints to which the software design must adhere, 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 is an application or an 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 architecture as illustrated in Figure 4.

As a platform-independent environment, Java can be slightly slower than the 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 while maintaining portability.

Client/Server

A server is any device or software that has resources to share. There are various types of servers.

- *Compute servers:* These provide computing power and resources.
- *Print servers:* These manage a collection of printers, allowing them to be shared across the 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.

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

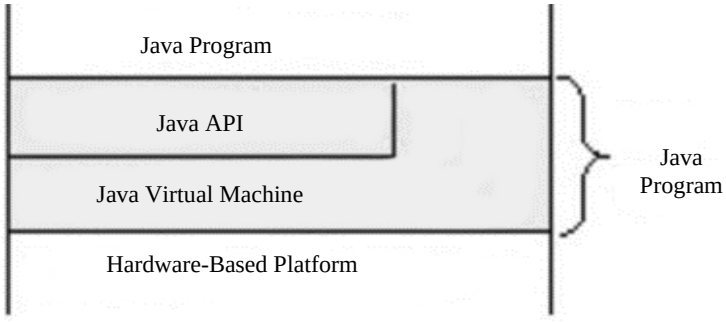


Figure 4. Hardware architecture.

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

Reserved Sockets

Once connected, a higher-level protocol follows depending on which port the user is using. Transmission Control Protocol (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).

This list goes on to other well-known ports reserved for various protocols. Each protocol determines how a client should interact with the port. The protocol defines the rules and structure of the data exchanged between the client and the server over a specific port. This allows different types of applications and services to communicate effectively over the internet using the TCP/IP Protocol Suite.

Java and the Net

Java supports TCP/IP by extending an already established stream I/O interface. It supports both the TCP and User Datagram Protocol (UDP) protocol families.

TCP is used for reliable, stream based I/O across the network. This ensures that data packets are delivered in sequence without errors, making it suitable for applications that require reliable communication, such as file transfer and web browsing (, and) supports a simpler, faster point-to-point datagram-oriented model. UDP does not guarantee the delivery or order of packets, making it faster, but less reliable than TCP. It is commonly used for real-time applications, such as video streaming, online gaming, and voice-over IP (VoIP). Java's support for both TCP and UDP allows developers to select an 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.

InetAddress

The `InetAddress` class is used to encapsulate both the TCP/IP address and the domain name associated with that address. Users interact with this class using the name of the IP host, which is more convenient and understandable than its IP address. The `InetAddress` class abstracts away the details of the actual IP number.

For 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 applications can resolve hostnames to IP addresses1 and vice versa, enabling network communication that is both flexible and transparent to the user. The Inet Address class had no visible constructors. Users use one of the available factory methods to create an InetAddress object. Factory methods are conventions whereby static methods in a class return an instance of that class. This is performed instead of overloading a constructor with various parameter lists, which can make the results clearer. Three commonly used `InetAddress` factory methods are: Static InetAddress getLocalHost() throws unknown host exceptions.

1. This method returns the local host `InetAddress` object. This represents the IP address of a local host. It may throw an `UnknownHostException` if the local host address cannot be determined.
2. Static InetAddress getByName(String hostName) throws UnknownHostExceptionThis method returns an `InetAddress` object given the hostname. The hostname is converted to its corresponding IP address. It throws an Unknown Host Exception` if no IP address can be found for the host.
3. Static Inet Address [] get All By Name (String hostName) throws UnknownHostExceptionThis method returns an array of all the IP addresses that correspond to a given hostname. This can be useful when host names map to multiple IP addresses. It throws an `UnknownHostException` if no IP address for the host can be found.

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

TCP/IP Client Sockets

IP sockets are used in Java to implement reliable, numerical, 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 types of TCP sockets in Java: servers and 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. No methods or constructors explicitly expose the details of establishing a connection. Two constructors are used to create client sockets:

1. *Socket (String hostName, int port):* Creates a socket connecting the local host to the named host and port.Can throw an `UnknownHostException` 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.'

TCP/IP Server Sockets

Java has a different socket class, which must be used to create 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. There 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. ' In a multihomed host, a local address specifies the IP address to which the 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 via TCP/IP. The `Server Socket` class provides flexibility in setting up server configurations, managing client connections, and handling communication over sockets.

CONCLUSION

The application of CNN-based frameworks for comparing contactless and contact-based fingerprint recognition marks a significant advancement in biometric authentication technology. Through meticulous examination of a diverse array of studies, it is evident that CNN-based approaches consistently yield superior accuracy rates in contactless fingerprint matching compared with 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 CNN-based frameworks to revolutionize biometric authentication systems, offering heightened security and convenience in various applications. Future research endeavors should focus on addressing the remaining challenges and exploring innovative methodologies to further improve the reliability and usability of contactless fingerprint recognition systems. This includes the development of more robust feature representations, refining fusion strategies for multimodal biometric data integration, and advancing privacy-preserving techniques. Additionally, efforts to enhance the interoperability and scalability of CNN-based frameworks are 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 have 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. Traffic normalization for covert channel protecting. In: Proceedings of the IEEE Conf. Russian Young Researchers Electr. Electron. Eng. 2021; 2330–3. DOI: 10.1109/ElConRus51938.2021.9396163.
2. Wendzel S, Mazurczyk W, Zander S. Unified description for network information hiding methods. J Univ Comput Sci. 2016;22:1456–86.
3. Singh H, Sharma RK, Singh VP. Online handwriting recognition systems for Indic and non-Indic scripts: a review. Artif Intell Rev. 2021;54:1525–79. DOI: 10.1007/s10462-020-09886-7.
4. Liang K, Chen J, He T, Wang W, Singh AK, Rawat DB, Song H, Lyu Z. Review of the open datasets for contractless sensing. IEEE Internet Things J. 2024;.
5. Caviglione L. Trends and challenges in network covert channels countermeasures. Appl Sci. 2021;11:1641. DOI: 10.3390/app11041641.
6. Han J, Huang C, Shi F, Liu J. Covert timing channel detection method based on time interval and payload length analysis. Comput Secur. 2020;97:101952.
7. Zhang L, Huang T, Rasheed W, Hu X, Zhao C. An enlarging-the-capacity packet sorting covert channel. IEEE Access. 2019;7:145634–40. DOI: 10.1109/ACCESS.2019.2945320.
8. Tian J, Xiong G, Li Z, Gou G. A survey of key technologies for constructing network covert channels. Secur Commun Netw. 2020;:1–20.
9. Vanderhallen S, Van Bulck J, Piessens F, Mühlberg JT. Robust authentication for automotive control networks through covert channels. Comput Netw. 2021;193:108079. DOI: 10.1016/j.comnet.2021.108079.
10. Wu Z, Guo J, Zhang C, Li C. Steganography and steganalysis in voice over IP: a review. Sensors. 2021;21:1032. DOI: 10.3390/s21041032. PubMed: 33546240.
11. Mileva A, Velinov L, Hartmann S, Wendzel S, Mazurczyk W. Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels. Comput Secur. 2021;104:102207.
12. McLoone M, McCanny JV. System-on-chip architectures and implementations for private-key data encryption. Springer Science & Business Media; 2003 Dec 31.

