Optimising Security, Power Consumption and Performance of Embedded Systems

Parisa Raimi

School of Computer Science and Electronic Engineering (CSEE)

University of Essex

This dissertation is submitted for the degree of

Doctor of Philosophy

October 2023

This thesis is a heartfelt tribute to my beloved parents, Talaat and Rahim. Their unwavering love and unyielding faith in me have been the driving forces behind my academic journey. Although they departed in 2023 and couldn't witness my attainment of a doctorate, while their absence at this milestone pains me, their memory propels me forward. I believe that their enduring blessings will continue to guide and inspire me on my future journey.

Acknowledgement

"All our dreams can come true if we dare to pursue them. In my opinion, hard work and determination are the essential traits of a successful researcher."

I am grateful to the many people who supported me in various ways during my Ph.D. journey. First of all, I praise and thank God for his countless blessings and for giving me the strength to accomplish this research during tough times.

This thesis will always be incomplete without expressing gratitude to my supervisor, Dr. Amit Singh. I would like to express my sincere appreciation to him for his patience, kindness, and invaluable support throughout the entire process. His guidance, tutelage, and motivation were instrumental to the successful completion of this endeavour, and I am forever indebted to him.

I am grateful to my husband and my sisters and brothers for their unwavering patience and support over the years. Without their encouragement, love, and understanding, none of this would have been possible. Their presence in my life has been a constant source of inspiration and motivation, and I am deeply grateful for their unending support.

Last but not least, I am grateful to the University of Essex for providing me with a conducive environment to carry out my studies.

"In loving memory of my dear brother's wife (Leila), whose unwavering love and everlasting contributions to my life, especially during my Ph.D. journey, fills my heart with gratitude."

Abstract

Increased interest in multicore systems has led to significant advancements in computing power, but it has also introduced new security risks due to covert channel communication. These covert channels enable the unauthorized leakage of sensitive information, posing a grave threat to system security. Traditional examples of covert channel attacks involve exploiting subtle variations such as temperature changes and timing differences to clandestinely transmit data through thermal and timing channels, respectively. These methods are particularly alarming because they demand minimal resources for implementation, thus presenting a formidable challenge to system security. Therefore, understanding the different classes of covert channel attacks and their characteristics is imperative for devising effective countermeasures.

This thesis proposes two novel countermeasures to mitigate Thermal Covert Channel (TCC) attacks, which are among the most prevalent threats. In the first approach, we introduce the Selective Noise-Based Countermeasure. This novel technique disrupts covert communication by strategically adding a selective noise (extra thread) to the temperature signal to generate more heat and change its pattern. This intervention significantly increases the Bit Error Rate (BER) to 94%, thereby impeding data transmission effectively. Building upon this, the second strategy, termed Fan Speed Control Countermeasure, dynamically adjusts fan speed to reduce system temperature further, consequently decreasing the thermal signal frequency and shutting down any meaningful transmission. This methodology achieves a high BER (98%), thereby enhancing system security.

Furthermore, the thesis introduces a new threat scenario termed Multi-Covert Channel Attacks, which demands advanced detection and mitigation techniques. To confront this emerging threat, we propose a comprehensive two-step approach that emphasizes both detection and tailored countermeasures. This approach leverages two distinct methodologies for implementation, with the primary goal of achieving optimal performance characterized by high BER and low power consumption.

In the first method, referred to as the double multi-covert channel, we employ two distinct frequency ranges for the timing and thermal covert channels. Through extensive experimentation, we demonstrate that this approach yields a high BER, providing a formidable challenge to various defense strategies. However, it is noteworthy that this method may potentially lead to overheating

issues due to the increased operational load. Alternatively, our second method, the single multi-covert channel, employs a single frequency range for data transmission. Notably, this approach addresses the overheating concerns associated with the double multi-covert channel, thereby reducing power consumption and minimizing the risk of system overheating. The experimental results presented in this thesis demonstrate the efficacy of the proposed strategies. By adopting a two-different approach, we not only enhance detection capabilities but also mitigate potential risks such as overheating. Our findings contribute significantly to the ongoing discourse on covert channel attacks and offer valuable insights for developing robust defense mechanisms against evolving threats.

By providing insights into both traditional and emerging covert channel threats in multicore systems, this thesis significantly contributes to the field of multi-embedded system security. The proposed countermeasures demonstrate tangible security improvements, while the exploration of multi-covert channel attacks sets the stage for detection and defense strategies.

Contents

List of Figures

List of Tables

Abbreviations

AES	Advanced Encryption Standard
ARM	Advanced RISC Machines
ASICs	Application-Specific Integrated Circuits
ASLR	Address Space Layout Randomization
AVR	Advanced Virtual RISC
BER	Bit Error Rate
CPA	Correlation Power Analysis
CPD	Change Point Detection
DPA	Differential Power Analysis
DTS	Digital Thermal Sensors
DoS	Denial-of-Service
ECC	Elliptic Curve Cryptography
ECC	Error Correcting Code
EDAC	Error Detection and Correction
EMI	Electromagnetic Interference
EMC	Electromagnetic Compatibility
EMEA	Electromagnetic Emanation Analysis
EMFI	Electromagnetic Fault Injection
EMFS	Electromagnetic Fault Sensing
EMP	Electromagnetic Pulse
FPGAs	Field Programmable Gate Arrays
HSM	Hardware Security Module
HSMs	Hardware Security Modules
IDS	Intrusion Detection Systems

IDS	Intrusion detection systems
IoT	Internet of Things
IP	Internet Protocol
MES	Manufacturing Execution System
MITM	Man-in-the-Middle
PER	Packet Error Rate
QoS	Quality of Service
RAM	Random Access Memory
RCE	Remote Code Execution
RISC	Reduced Instruction Set Computer
RSAA	Rivest–Shamir–Adleman Algorithm
RTOS	Real-Time Operating Systems
SCAs	Side-Channel Attacks
SEM	Scanning Electron Microscopes
SPA	Simple Power Analysis
SSL	Secure Sockets Layer
STA	Simple Thermal Analysis
SoC	System-on-a-Chip
TCC	Thermal Covert Channel
TLS	Transport Layer Security
TPMs	Trusted Platform Modules
XOR	Exclusive-OR

Chapter 1

Introduction

1.1 Background

In recent years, the design objectives for numerous electronic systems have shifted towards prioritizing security, power efficiency, and performance, particularly in embedded systems. Embedded systems have become integral to various real-life applications, ranging from low-end devices like wireless handsets and networked sensors to high-end systems such as network routers and web servers. As the utilization of embedded systems continues to grow across diverse fields, addressing security becomes imperative. Balancing the enhancement of performance and security in embedded systems while maintaining low power consumption remains a significant scientific challenge for researchers. Ensuring the security of embedded systems requires the implementation of various security measures at different levels within the system's architecture. These measures include secure boot [150], data encryption, access control, secure communication protocols [151], and regular firmware updates [152]. In addition to these measures, regular security audits and testing, including automated and manual techniques such as fuzz testing, vulnerability scanning, and penetration testing [149], are essential to identify and address potential vulnerabilities in the system. Moreover, with the rising prevalence of wireless connectivity and internet access, embedded systems are becoming more susceptible to attacks. Thus, addressing security concerns

is crucial at multiple abstraction levels, encompassing both software and hardware security [153]. To effectively address these security concerns, designers must consider a diverse range of techniques at various levels of abstraction. At the hardware level, secure hardware modules like Trusted Platform Modules (TPMs), Hardware Security Modules (HSMs), and secure enclaves can provide a secure execution environment [154]. On the software front, techniques such as code obfuscation, code randomization, and software diversification can mitigate the impact of side-channel attacks and logical attacks. By adopting a holistic approach to embedded system security, we can fortify these systems against potential attacks and contribute to the continued improvement of our quality of life [16].

1.1.1 Performance and Power Consumption in Embedded System

Performance and power consumption are pivotal aspects of every embedded system and must be carefully considered during the design phase. These systems heavily rely on batteries for operation and are constrained by limited energy resources. Efficient power consumption is crucial for optimal embedded system functionality. Additionally, energy consumption significantly influences system performance [12]. Enhancing the performance of embedded systems has consistently presented a primary scientific challenge for researchers. Motivations for performance improvement include cost-effective hardware, reduced power consumption, an enhanced user experience, and personal satisfaction [11]. Conversely, energy consumption plays a vital role in the design of every electrical and digital system [15-17]. Hence, multi-core technology extends to embedded systems, offering the potential for achieving both high performance and low power consumption [14-13]. A brief comparison between single-core and multi/many-core systems underscore the benefits of the latter in meeting the high-performance requirements of complex embedded software. This is primarily due to the power consumption and radio frequency effects in single-core systems, whereas multi/many-core systems operate at lower frequencies. As technology advances and the demand for high performance increases, the utilization of multiple cores or different types of cores within the same chip area has experienced exponential growth [8].

1.2 Motivation

The escalating sophistication of electronic systems, particularly embedded systems, has paralleled an increase in their vulnerability to security threats. Side channel and covert channel attacks represent some of the most insidious risks, exploiting the unintended leakage of information through the physical operation of devices. These attacks are capable of extracting sensitive data, such as cryptographic keys, by analyzing power consumption, electromagnetic emissions, or even acoustic signals, underscoring a critical security flaw in the design of these systems. The omnipresence of embedded systems in critical sectors—automotive, healthcare, defense, to name a few—amplifies the urgency to fortify their security. The challenge is not trivial; it demands a multifaceted strategy that addresses not just the immediate vulnerabilities but also anticipates future threats. The goal is to develop resilient systems that maintain their integrity and functionality, even in the face of sophisticated attacks. This calls for innovative countermeasures that optimize security without compromising on performance or power efficiency, a balance that is crucial for the sustainability and reliability of embedded systems.

1.3 Research Objectives

The overarching aim of this research is to redefine the security paradigm of embedded systems. By focusing on mitigating the risks posed by side and covert channel attacks, this work seeks to establish a new benchmark for security, performance, and energy efficiency in multicore environments. To achieve this, the study is structured around several core objectives:

1. Design a methodology that can effectively counter covert channel attacks launched simultaneously from various systems or devices, ensuring robust protection across complex scenarios.
    - Aim: Design and implement a methodology to counteract covert channel attacks from multiple systems or devices effectively.
    - Quantifiable goal: Develop a countermeasure that reduces the success rate of covert channel attacks by at least 94%, as measured in controlled environment simulations.
2. Develop a methodology to defend against thermal covert channel attacks with significantly reduced power consumption and an exceptionally high bit error rate, making information extraction virtually impossible.

- Aim: Construct a defense mechanism against thermal covert channel attacks that significantly lowers power consumption.
- Quantifiable goal: Achieve a reduction in power consumption by 20% compared to current standards without compromising the defense mechanism's effectiveness, alongside increasing the BER to 95% or higher, making data extraction highly impractical.

3. Designing a comprehensive detection framework capable of identifying covert channel attacks while offering an approach to system security.
    - Aim: Design a comprehensive detection framework capable of accurately identifying covert channel attacks, integrating this framework into system security measures.
    - Quantifiable goal: Implement a detection system with at least 94% accuracy in identifying different types of covert channel attacks.

This study introduces novel countermeasures against thermal and timing covert channels, offering a more secure and efficient solution for the security of multi and many-core systems. The research contributes significantly to the advancement of embedded systems, addressing critical challenges related to security, power consumption, and performance.

1.3.1 Key Contributions

- Introducing novel countermeasures against thermal and timing covert channels, leading to high improvement in overall security posture compared to existing solutions.
- Achieving a low power consumption for thermal attack defence compared to baseline methods, demonstrating significant advancements in energy efficiency.
- Establishing a comprehensive detection framework with high accuracy, providing valuable insights for proactive system security management.

1.4 Thesis Structure

The structural layout adopted in the remainder of this thesis is as follows:

Chapter 2: Chapter 2 presents an overview of covert channel attacks and security of multi/many-core systems. This chapter presents the weakness of security and performance of these systems and the importance of them for any device or system. Furthermore, general trends and challenges in embedded systems have been discussed.

Chapter 3: Chapter 3 explores a new countermeasure against thermal covert channel attacks called selective noise-based power-efficient and effective countermeasures proposed to improve the security of the systems.

Chapter 4: In this chapter, a specific countermeasure fan speed control-based defence, is introduced on top of previous work (selective noise-based countermeasure) to improve the security of the system and decrease power consumption. We also presented the combination of the different countermeasures to check the behaviour of the system.

Chapter 5: In the continuum, another covert channel attack is presented (Timing covert channel attacks). We analyse the behaviour of the frequency of the system and detect and fight any possible timing attacks, find the relationships between these attacks and thermal covert channel attacks, and a muti-cover channel attack is introduced. Modifying defence methods suggested in Chapter 3 and Chapter 4, tried to fight this type of attack.

Chapter 6: This chapter concludes by summarising the contributions detailed in the thesis. Finally, the future of research in the field of covert channel attacks which this thesis will play is highlighted.

1.5 Publications

The contribution of each chapter has been published as following papers during the time duration of this doctoral thesis:

- **Chapter 2:** Rahimi, P., Singh, A.K., Wang, X. and Prakash, A., 2021, December. Trends and challenges in ensuring security for low-power and high-performance embedded socs. In 2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC) (pp. 226-233). IEEE.
- **Chapter 3:** Rahimi, P., Singh, A.K. and Wang, X., 2022. Selective noise based power-efficient and effective countermeasure against thermal covert channel attacks in multi-core systems. Journal of Low Power Electronics and Applications, 12(2), p.25.
- **Chapter 4:** Rahimi, P., Singh, A.K. and Wang, X., 2022, October. Fan Speed Control Based Defence for Thermal Covert Channel Attacks in Multi-Core Systems. In 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS) (pp. 1-4). IEEE.
- **Chapter 5:** Rahimi, P., Singh, A.K. and Wang, X. Detection and Defence of Thermal and Timing Covert Channel Attacks in Multi-core Systems. Under Review.

Chapter 2

Literature Review

2.1 Overview

Embedded systems are an integral part of modern technology and can be found in a wide range of applications, such as home appliances, medical devices, and automobiles. These systems combine hardware and software to perform specific functions with real-time constraints [1]. Figure 2.1 demonstrates that embedded systems can control many common devices. A System on Chip (SoC) is a single-chip design that includes a CPU, peripherals, and sometimes memory for a computer system. Unlike traditional designs, this configuration does not feature a general-purpose bus, thus limiting its ability to interface with additional random devices. Instead, all the pins on the device are dedicated to built-in peripherals, with the manufacturer typically setting the maximum number of devices that a single SoC can support at any given time. SoC solutions offer several advantages, including reduced power dissipation, minimized chip interconnects, and decreased device size [47].

Despite their importance and widespread use, modern embedded systems are also vulnerable to security threats. These threats can range from individual systems, such as smart cards, wireless handsets, and networked sensors, to more complex connected systems, such as firewalls, network routers, servers, and gateways [43]. Therefore, it is crucial to increase the number of embedded

systems equipped with high-security features to protect against potential adversaries. Various types of attacks can threaten embedded SoCs, including logical and software attacks, as well as physical and side-channel attacks. In this study, we focus on examining covert-channel attacks more closely. Side-channel attacks exploit weaknesses in the physical implementation of a system, such as electromagnetic radiation or power consumption, rather than software vulnerabilities. These attacks are particularly concerning as they can compromise system security without being detected by traditional security measures. Therefore, developing countermeasures and strategies to mitigate the risk of side-channel attacks on embedded systems is essential [2].

Furthermore, embedded systems have become increasingly important in today's world due to their ability to control various common devices. They are designed to execute one or more assigned functions with real-time constraints, integrating hardware and software into a complete device or system. This integration of necessary chips and parts into a single chip facilitates the execution of applications or programs by loading them into chip memory or off-shelf components. Researchers have, therefore, focused on improving the performance and security of embedded systems while minimizing power consumption. In this chapter, we aim to examine all these issues more closely, specifically focusing on secure, low-power, and high-performance embedded systems. By doing so, we hope to provide valuable insights that can help advance the field of embedded systems and ensure the security and reliability of these critical components in modern technology [43].

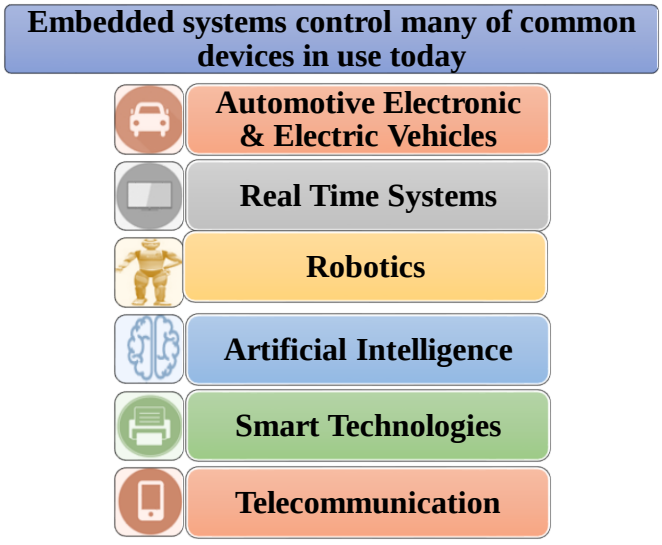

Figure 2.1 Applications domains for embedded systems [47].

The content of this chapter has been published in "Trends and Challenges in Ensuring Security for Low-Power and High-Performance Embedded SoCs", [107].

2.2 Security In Embedded SoCs

Embedded systems have become ubiquitous in our daily lives, and their importance is undeniable. However, with the increasing reliance on these systems, concern for security has also escalated. Embedded systems are utilized for storing, processing, and transmitting sensitive data in various computing systems, including mobile phones, airplanes, smart cards, medical equipment, network appliances, automobiles, and digital services [2]. The vulnerability of embedded systems to security threats has increased due to their wireless connectivity and internet connections. The need for security in embedded systems is not a new issue but has become more critical than ever. To ensure the security of embedded systems, it is crucial to consider security measures at multiple abstraction levels, including both software and hardware security. Simply focusing on one aspect of security may not be enough to provide complete protection in embedded systems. Therefore, it is necessary to develop secure embedded systems that can withstand various security threats. This involves ensuring that the hardware and software components are designed to provide a high level of security. Additionally, security measures should be implemented throughout the entire development process of an embedded system, including design, implementation, and testing. The security of embedded systems should be continuously monitored and updated to ensure protection against emerging threats [4].

Embedded systems have evolved into an integral part of modern life, permeating various aspects of our daily routines. Their importance is undeniable as they facilitate critical functions in numerous devices, such as mobile phones, airplanes, smart cards, medical equipment, network appliances, automobiles, and digital services [2]. As we become increasingly reliant on these systems, the issue of security has taken center stage, with escalating concerns over safeguarding sensitive data. The vulnerability of embedded systems to security threats has been exacerbated by their wireless connectivity and internet access. This integration into interconnected networks makes them susceptible to potential breaches, raising the urgency to address security concerns. While the need for security in embedded systems is not a novel problem, it has reached a tipping point in its significance [108]. Securing embedded systems requires a comprehensive approach, encompassing multiple abstraction levels and focusing on both software and hardware security. Merely emphasizing one aspect of abstraction may not be sufficient to guarantee complete protection. Hence, the challenge lies in developing embedded systems that can effectively

withstand a myriad of security threats. Achieving robust security involves designing both hardware and software components with a high level of protection in mind. A holistic approach must be taken throughout the entire development process, encompassing design, implementation, and testing phases. Incorporating security measures early on ensures potential vulnerabilities are addressed before they become critical issues. Moreover, the journey towards secure embedded systems does not end with development; it is an ongoing process that demands continuous monitoring and updates to counter emerging threats [108]. Regularly assessing the security posture of these systems helps identify and rectify any potential weaknesses that might arise over time.

2.2.1 Weaknesses in Embedded Systems

Weaknesses in the embedded systems occur because of the following reasons:

- **Network connectivity:** The rise of network connectivity in embedded systems has increased the risk of vulnerability to network attacks. As more embedded systems become equipped with wireless network connectivity, the potential for network attacks to exploit vulnerabilities in these systems also increases [10]. Network security has been a long-standing issue, as computer networks offer a wide range of coverage and use, thus providing an attractive target for attackers. One challenge in securing network-connected embedded systems is that the network attack surface is often much larger than the local access network [11]. This means that attackers have more potential entry points to exploit vulnerabilities in the system. Furthermore, since many embedded systems are designed to be small and lightweight, they may have limited resources available for implementing strong security measures, such as encryption or access control. To address these challenges, it is important to carefully design and implement security measures that can effectively mitigate the risks associated with network connectivity. This includes strong authentication and encryption mechanisms, limiting access to sensitive data, and implementing intrusion detection and response systems to detect and respond to attacks [12]. Therefore, the rise of network connectivity in embedded systems has increased the potential for vulnerability to network attacks. Careful design and implementation of security measures are necessary to minimize these risks and ensure the security of these systems [18].

- **Dynamic and configurable environment:** Embedded systems are often deployed in dynamic and configurable environments, presenting unique security challenges. In these

environments, the operating conditions of the system can change frequently and unpredictably, potentially impacting the effectiveness of security measures. For instance, an embedded system in an industrial setting may need to adapt to variations in temperature, humidity, and other environmental factors that could affect its performance. Similarly, an embedded system in a smart home or office might need to adjust to changes in the number and types of devices connected to the network [10]. To secure these systems effectively in such dynamic environments, it is crucial to implement security measures that are both flexible and adaptable. This could involve using machine learning algorithms to dynamically adjust security settings based on changing conditions, or employing virtualization technologies to isolate parts of the system, thereby preventing the spread of malware or other security threats [11]. Moreover, regularly testing and updating security measures is essential to maintain their effectiveness amidst changing conditions. This might include conducting vulnerability assessments, penetration testing, and other security audits to identify and mitigate potential weaknesses in the system [9]. Despite the challenges posed by their dynamic and configurable nature, embedded systems can maintain data integrity and confidentiality through careful design and implementation of security measures [12].

- **Software induced vulnerability:** Software-induced vulnerabilities represent a common concern in embedded systems, especially those that utilize third-party software. Downloading software from the internet can inadvertently introduce vulnerabilities into the system, which attackers could exploit [8]. A typical scenario involves the use of unverified third-party software, particularly risky if sourced from untrusted websites. In some instances, attackers may distribute fake versions of popular software containing malware or malicious code. Once installed, this software can facilitate attacks or enable the theft of sensitive data [9]. To mitigate the risks associated with software-induced vulnerabilities, implementing security measures that verify the authenticity and integrity of third-party software before installation is vital. Measures might include using digital signatures to confirm software authenticity and relying on secure, trusted software repositories. Additionally, regularly updating software and firmware to patch known vulnerabilities is crucial [11]. Addressing these vulnerabilities effectively reduces the risk of attacks exploiting system weaknesses. By ensuring the authenticity and integrity of

third-party software and keeping software and firmware up to date, the risk of attacks can be significantly mitigated, safeguarding the security of the system [9].

2.2.1.1 Weaknesses in the Performance

Weaknesses in the performance of embedded systems occur due to several reasons:

1. **Limited Resources:** Embedded systems are designed to operate with constrained resources, such as memory, processing power, and energy consumption. Consequently, performance issues may arise when the system is overloaded or when there is insufficient memory or processing power to handle complex tasks [157].
2. **Hardware Limitations:** The hardware components of embedded systems are tailored to meet specific requirements, including size, power consumption, and cost. These constraints may result in hardware that is less powerful than that found in traditional computer systems, potentially leading to performance issues [21].
3. **Software Complexity:** As the functionality and capabilities of embedded systems expand, so does the complexity of the software running on them. This increased complexity can complicate the optimization of performance and the identification and resolution of issues [159].
4. **Real-time Constraints:** Many embedded systems operate under strict real-time constraints, necessitating responses to events within specified time frames. Meeting these constraints can be challenging and may result in performance issues if not properly managed [20].
5. **Environmental Factors:** Embedded systems often operate in harsh and challenging conditions, such as extreme temperatures, vibrations, and electromagnetic interference. These environmental factors can adversely affect system performance and lead to failures [22].

To address these performance challenges, developers must carefully optimize their code, utilize efficient algorithms, and manage memory effectively. Employing hardware accelerators or co-processors to offload specific tasks from the main processor can enhance performance. Moreover, selecting appropriate hardware components during the design phase is crucial. Balancing the need for power efficiency with adequate processing power and memory is a delicate task, as each embedded system has unique requirements [23]. Despite these challenges, embedded systems have achieved remarkable success in various applications due to their specialized functionality.

Engineers and developers continue to innovate, finding creative solutions to optimize performance within the constraints of limited resources [159].

2.3 Embedded System Attacks

Security attacks on embedded systems can be classified by the means used to carry out the attacks. Typically, attacks on embedded systems fall into two main categories: physical and side-channel attacks (SCAs)/lateral attacks, and logical attacks, as illustrated in Figure 2.3 [4]. Physical and SCAs attacks exploit vulnerabilities in the physical implementation of the system. These attacks can be conducted through the analysis of power consumption, electromagnetic radiation, or even sound emissions from a device. On the other hand, logical attacks target vulnerabilities in the software or the logical design of the system [154].

In this section, we provide some examples of attacks against embedded systems and devices, closely examining the potential of these attacks, as well as their capabilities and implications. While this overview is not comprehensive, it is, from our perspective, quite representative and capable of covering a wide range of application domains, including hardware and software attacks.

2.3.1 Logical Attacks

Logical attacks typically involve sending messages to the device and then observing its behavior. The adversary's goal is often to trick the device into revealing its keys or to run malicious software. Logical attacks can be classified into two categories: 1) software attacks, and 2) attacks exploiting cryptographic and protocol weaknesses. These vulnerabilities arise from the design and algorithms of the embedded systems. The objective of these attacks is to access sensitive data, such as passwords, credit card numbers, or entire identities [5-6]. For instance:

1. **Code Injection:** Attackers inject malicious code into the embedded system, exploiting vulnerabilities in input validation or insecure data handling. This enables them to execute unauthorized commands or gain control over the system [160].
2. **Command Injection:** This attack involves injecting malicious commands into the system’s input, exploiting weak input validation to execute unauthorized commands [161].

To protect against logical attacks in embedded systems, developers must adhere to secure coding practices, conduct regular security audits, and employ strong input validation mechanisms. Implementing access control and privilege separation can also limit the potential impact of

successful attacks. Additionally, timely software updates and patches are crucial to address known vulnerabilities and enhance the overall security posture of embedded systems [160].

2.3.2 Software Attacks

The primary goal of security design in any system is to ensure confidentiality, integrity, and availability collectively known as the CIA triad. Confidentiality aims to prevent unauthorized access to sensitive information, integrity ensures that data remains unaltered and accurate, and availability guarantees that the system is accessible and operational when needed. These three pillars are fundamental in securing any system [5]. However, in today's technology-driven world, software attacks are becoming increasingly prevalent. A software attack uses software agents like trojans, viruses, or worms to compromise a system. The goal of such an attack is to exploit vulnerabilities resulting from poor design or flawed algorithms in the embedded system [155]. Software attacks are often inexpensive to execute and do not require substantial infrastructure, unlike physical attacks. Consequently, they have become a favored method for attackers seeking unauthorized access to sensitive data or to disrupt system operations [156]. These attacks can lead to the theft of valuable intellectual property, exposure of confidential customer information, or loss of critical business operations. To prevent software attacks, it is crucial to implement robust security measures, such as access control, authentication, encryption, and intrusion detection systems. Regular security audits and vulnerability assessments are vital in identifying and addressing system weaknesses before they can be exploited. Keeping all software and systems updated with the latest security patches and updates is also critical [158]. Although confidentiality, integrity, and availability are the primary goals of any security design, the importance of protecting against software attacks has significantly increased in today's digital landscape. By adopting appropriate security measures and maintaining vigilance, businesses and organizations can defend against the harmful effects of software attacks [31].

- Below are examples of software attacks against embedded systems, each illustrating their potential, capabilities, and implications [5-6]:
- Malware Injection: Attackers compromise the security of an embedded system by injecting malicious software, such as viruses, worms, or Trojans, potentially gaining unauthorized control [166].
- Remote Code Execution (RCE): Vulnerabilities within the software permit attackers to

execute arbitrary code remotely, allowing them to seize control of the embedded system and manipulate its functions [162].

- Data Manipulation: Attackers alter data within the embedded system to manipulate its behavior or compromise the integrity of crucial information.
- Race Condition: Attackers exploit timing vulnerabilities in the software, leveraging concurrency issues to perform unauthorized actions [163].

2.3.3 Cryptographic and Protocol Weakness Attacks

With rapid advancements in communication technologies like the internet, users across various domains, including companies, home users, and government agencies—increasingly rely on digital information and data. However, the absence of adequate security measures exposes users to the risk of unauthorized access to their sensitive information, leading to fear and mistrust. Consequently, data security is paramount in the design of any system or device reliant on embedded systems. An effective strategy to mitigate this risk involves designing secure systems that prevent unauthorized access to information and safeguard the privacy of both the system and its users. Ensuring the authenticity of electronic documents and maintaining their security is also crucial. To achieve this, developing a robust security framework with cryptographic algorithms at its core is essential [167].

Cryptography, the science of secure communication, is vital for data security. Nonetheless, cryptographic and protocol weakness attacks can exploit vulnerabilities in cryptographic algorithms, including asymmetric ciphers, hashing algorithms, and symmetric ciphers, as well as in protocol stacks. The primary objectives of cryptographic attacks are to gain unauthorized access to sensitive information or to compromise a system. Thus, it is imperative to design and implement cryptographic algorithms that are resistant to attacks and capable of withstanding constantly evolving threats. These algorithms should be regularly updated to remediate any known vulnerabilities. Moreover, adopting secure protocols, such as Transport Layer Security (TLS), can further bolster data security [168].

The growing dependency on digital information and data underscores the need for stringent data security measures. Cryptography is crucial in securing data, and any system or device that relies on embedded systems must feature a design that precludes unauthorized information access, ensuring both privacy and the authenticity of electronic documents. Continuously evaluating and

updating security measures are essential practices to stay ahead of evolving threats [31].
To counter cryptographic and protocol weakness attacks, embedded systems should employ robust and correctly implemented cryptographic algorithms, secure key generation and management, and routinely update software and firmware to address known vulnerabilities. Additionally, utilizing secure communication protocols and sidestepping known vulnerabilities can significantly improve the security of embedded systems against these types of attacks [169].

2.3.4 Physical and Side-Channel Attacks

Physical and side-channel attacks are methodologies that exploit a system's performance or discern characteristics of its performance. Physical attacks, also known as invasive attacks, involve direct tampering with the system or device. Conversely, non-invasive attacks, commonly referred to as side-channel attacks, indirectly exploit the system's performance without physical interference. These attacks leverage the monitoring and analysis of the system's electromagnetic emissions, power consumption, or other external indicators to extract sensitive information [170]. Examples of non-invasive attacks include power analysis, electromagnetic analysis, and acoustic analysis.
Both physical and side-channel attacks pose significant risks to systems and devices dependent on embedded systems. These risks manifest as the theft of sensitive information, unauthorized access, or the compromise of the system's functionalities [171]. To counteract such threats, it is imperative to deploy comprehensive security measures. These measures should encompass physical security provisions, such as tamper-resistant packaging and secure boot protocols, alongside cryptographic techniques like encryption and hashing to safeguard sensitive data. In summary, physical and side-channel attacks exploit or recognize performance characteristics of systems, with invasive attacks involving physical interference and non-invasive attacks exploiting performance indirectly. Implementing a robust security framework, which includes physical safeguards, cryptographic protections, and regular security audits to uncover and rectify potential vulnerabilities, is crucial in preventing these types of attacks [32]. A depiction of the side-channel attack process can be found in Figure 2.2.

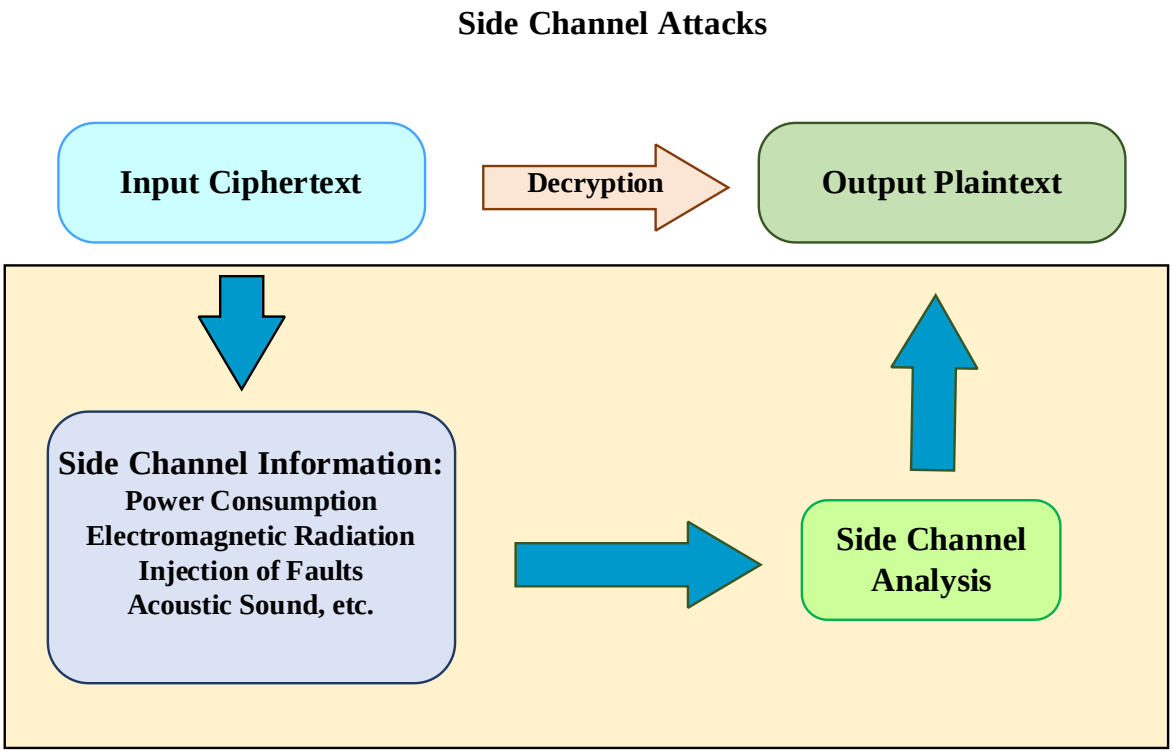

Figure 2.2 Diagram of side channel attacks [8].

- **Invasive attack*:*** Invasive attacks necessitate physical access to the device or system and involve directly tampering with the device to interfere with, operate, and observe its internal components. These attacks can have goals such as reading out the implementation of IP cores or extracting secret information, manipulating or damaging data or circuits, or accessing sensitive information [8]. Invasive attacks often result in irreversible modifications to the physical properties of the chip, employing standard reverse engineering techniques like optical or scanning electron microscopy (SEM). Certain invasive attacks may require special equipment and extensive time, potentially weeks, in specialized laboratories, with micro-probing and design reverse engineering being notable examples [32]. Despite their complexity and the high level of resources required, making them less common, the threat of invasive attacks remains significant. To counteract these threats, implementing physical security measures like tamper-resistant packaging and secure boot protocols is crucial. Regular security audits play a vital role in identifying and mitigating vulnerabilities that could be exploited via invasive attacks.

- **Non-invasive attacks:** Non-invasive attacks, also known as side-channel attacks, differ fundamentally in that they do not necessitate physically opening the device or system. These attacks exploit the system's emissions or operational behaviors, such as power consumption, electromagnetic emissions, timing information, or induced faults, during cryptographic processes to extract sensitive information. Non-invasive attacks, including power analysis, electromagnetic (EM) analysis, time analysis, and fault injection attacks, require an initial investment of time or creativity but are generally more affordable and scalable than their invasive counterparts [8]. To safeguard against non-invasive attacks, it's imperative to deploy strong cryptographic protocols and countermeasures like masking or blinding techniques, which obfuscate the information attackers seek. Physical security measures, including tamper-resistant packaging and secure boot protocols, further fortify defences against unauthorized access [8-10]. While invasive attacks involve direct physical tampering and are less common due to their complexity and resource requirements, non-invasive attacks pose a scalable and relatively low-cost threat by exploiting indirect system properties. Addressing both requires a comprehensive security strategy encompassing strong cryptographic defences, physical security measures, and ongoing vigilance through security audits.

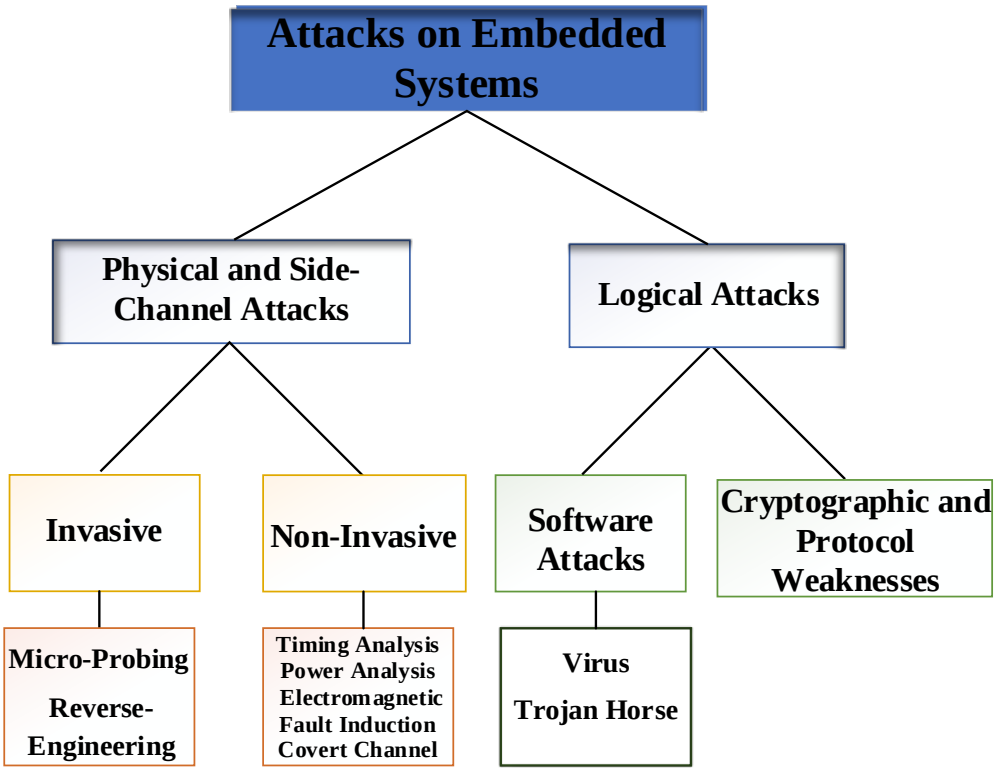

Figure 2.3 Attacks on embedded systems.

It is important to understand that side-channel attacks are designed to exploit weaknesses in the design and implementation of a system. These attacks typically involve monitoring the physical properties of a system while cryptographic operations are being performed, in order to uncover sensitive information. Several types of side-channel attacks can be used to compromise a system. One such attack is a power analysis attack, which involves monitoring the power consumption of a device during cryptographic operations to extract secret keys or other sensitive information. Another type of attack is a time analysis attack, which involves measuring the time it takes for a device to perform cryptographic operations to extract sensitive information [9]. EM analysis attacks are another common type of side-channel attack. These attacks involve monitoring the electromagnetic radiation emitted by a device during cryptographic operations to extract sensitive information. Fault induction attacks are also used to exploit weaknesses in the design of a system. These attacks involve inducing faults or errors in a device to extract sensitive information. Overall, side-channel attacks are a significant threat to the security of cryptographic systems. They can be difficult to detect and can be carried out using relatively simple equipment, making them accessible to attackers with limited resources. As such, it is important to understand the different types of side-channel attacks and to take steps to mitigate the risk of their exploitation [10].

2.3.4.1 Different Types of Physical and Side-Channel Attacks

2.3.4.1.1 Power Analysis Attacks

Power analysis attacks refer to the analysis of the power consumption during the system's data analysis [11]. In electronic systems and devices, the instantaneous power is dependent on the information and data that is being processed in the system as well as the process performed by the system. Therefore, by analysing the power consumption of the system when it operates encryption or decryption, the key can be deduced [12].

Power Analysis attacks can be classified into three categories, Simple power analysis, Differential Power Analysis, and Correlation power analysis attacks [13].

2.3.4.1.2 Simple Power Analysis (SPA)

Simple power analysis (SPA) is a technique of side-channel attacks that monitors the power consumption in real time during the operation of cryptographic algorithms and processes in progress. Being possible to break the Advanced Encryption Standard (AES) in a few minutes, power analysis attacks have become a serious security issue for cryptographic devices such as smart cards. These attacks analyse the chip's present power consumption during a period of time. As it can be said each operation will have different power consumption profiles, that establish what type of function has been performed. For instance, there is a distinguishing multiplication function from an addition function based on the usage of the power between multiplications compared to addition. In addition, from the reading of the data from memory, the ratio of 1's vs. 0's will show in the power profile in this attack, attackers directly analyse and evaluate the power consumption. Of course, the power consumption during each operation is different and it also depends on the construction of the microprocessor that performs the execution [13]. The SPA attack is relatively simple to carry out and can be performed using inexpensive equipment. By analysing the power consumption of a device during cryptographic operations, an attacker can observe patterns that can be used to deduce sensitive information about the cryptographic algorithm being used. To prevent SPA attacks, countermeasures such as power analysis-resistant designs and algorithms can be employed. One approach is to use masking techniques to add random values to the data being processed, making it more difficult for attackers to extract sensitive information from the power consumption patterns. Another approach is to use power analysis-resistant algorithms that are designed to minimize the power consumption variations

during cryptographic operations [14]. Overall, Simple Power Analysis attacks are a significant threat to the security of cryptographic systems. However, by understanding the techniques used in these attacks and implementing appropriate countermeasures, it is possible to mitigate the risk of their exploitation [13].

2.3.4.1.3 Differential Power Analysis (DPA)

Differential Power Analysis (DPA) attacks are a statistical technique that can exploit the power consumption to identify data information to extract the secret keys while the cryptographic algorithm is running on the system. DPA attacks use the weakness that correlates between the electricity that is used in a chip of a smart card and the encryption key it contains. In this attack, the attacker analysis the power consumption information of the system during the cryptographic algorithm running on the device to break the key differential power attack runs in a different section of the chip and then applies statistical analysis to avoid the countermeasures, like added noise, which can be applied to hide each bit. Measuring the power that has been used can determine what kind of computational operations are carried out by a device. Therefore, this analysis will disclose a few bits of the crypto at each time; this process is continually carried out till eventually, the entire key is produced. Successful DPA attacks have been shown on a wide range of encryption algorithms. This attack is stronger than a simple power attack [13]. Furthermore, DPA attacks are more sophisticated than SPA attacks and require more advanced equipment and techniques. In a DPA attack, an attacker first captures power consumption traces of a device performing cryptographic operations with known inputs. The attacker then analyses these traces to extract information about the secret key being used in the cryptographic algorithm. To prevent DPA attacks, countermeasures such as masking and blinding can be employed. Masking involves adding random values to the data being processed, making it more difficult for attackers to extract sensitive information from the power consumption traces. Blinding involves modifying the cryptographic algorithm to prevent attackers from deducing sensitive information about the secret key. Overall, DPA attacks are a significant threat to the security of cryptographic systems. They require more advanced techniques than Simple Power Analysis attacks, but they can still be carried out using relatively simple equipment. By implementing appropriate countermeasures, it is possible to mitigate the risk of DPA attacks and protect sensitive information [15].

It’s important to note that DPA is a sophisticated and non-invasive attack that requires specialized equipment and expertise. To defend against DPA, countermeasures such as masking, shuffling, or other secure hardware and software implementations are employed [11].

2.3.4.1.4 Correlation Power Analysis (CPA)

Correlation power analysis (CPA) attack is a new method of the DPA. In these attacks, the attackers make a model of power consumption during the analysis step. CPA attack is a powerful analysis attack that is capable of breaking a cipher by using a combination of the power consumption of the device and the hamming weight of the key-dependent values of the algorithm. These attacks are statistical attacks and use the Pearson correlation modulus to correlate data or information [14]. In a CPA attack, an attacker first captures power consumption traces of a device performing cryptographic operations with known inputs. The attacker then uses statistical techniques to analyse the correlation between the power consumption and the data being processed. By analysing these correlations, an attacker can deduce sensitive information about the secret key being used in the cryptographic algorithm [16]. To prevent CPA attacks, countermeasures such as masking and blinding can be employed. Masking involves adding random values to the data being processed, making it more difficult for attackers to extract sensitive information from the power consumption traces. Blinding involves modifying the cryptographic algorithm to prevent attackers from deducing sensitive information about the secret key. Overall, CPA attacks are a significant threat to the security of cryptographic systems. They require some level of expertise and equipment, but they can still be carried out using relatively simple equipment. By implementing appropriate countermeasures, it is possible to mitigate the risk of CPA attacks and protect sensitive information [13]. A Brief Comparison of CPA attacks with other attack algorithms such as DPA attacks shows that they have a wide range of advantages for example they require a smaller number of power traces [15].

There are different steps to a CPA attack and these steps can be written as follows:

As the first step of the attack, the power traces can be used to gather the information of the plain text. For the next steps, the attackers assume a model of power consumption. Now the attackers can guess a key. After the guesses, the key can be used to calculate the intermediate value using the found guess key. In addition, during the cryptographic operations, the intermediate value will be generated. From this section variable data values and part of the key must be selected. This is

known as the selection function. For example, in the below equation, 'I' indicates the intermediate value 'd' indicates the plain text or the ciphertext and 'k' is a part of the key known as the subkey [15-16].

$$I = F(d, k) \tag{2.1}$$

Then power measurements should be done during the cryptographic operation running on the device [15]. CPA, like DPA, requires specialized equipment and expertise to conduct successfully. To defend against CPA, countermeasures such as noise addition, power shaping, or using hardware components with built-in resistance to side-channel attacks can be implemented. Regular security audits and evaluation of the embedded systems can also help identify and address potential vulnerabilities [13].

2.3.4.1.5 Countermeasures against Power Analysis Attacks

Countermeasures are implemented to prevent or make power analysis attacks more difficult to execute. However, it is important to note that no countermeasure can guarantee a hundred percent effectiveness against all types of power analysis attacks.

The effectiveness of a countermeasure can be measured by the level of difficulty an attacker would face in executing a successful power analysis attack. If the time and cost required to execute such an attack are significant, then the countermeasure can be considered effective. Effective countermeasures against power analysis attacks include techniques such as masking and blinding, which add random values to the data being processed and modify the cryptographic algorithm, respectively. These techniques make it more difficult for attackers to extract sensitive information from the power consumption traces. Other countermeasures include using power analysis-resistant designs and algorithms, reducing the power consumption variations during cryptographic operations, and physically isolating the cryptographic device from external influences. Therefore, while no countermeasure can guarantee absolute protection against power analysis attacks, implementing appropriate countermeasures can significantly increase the level of difficulty for attackers and reduce the risk of successful attacks [34].

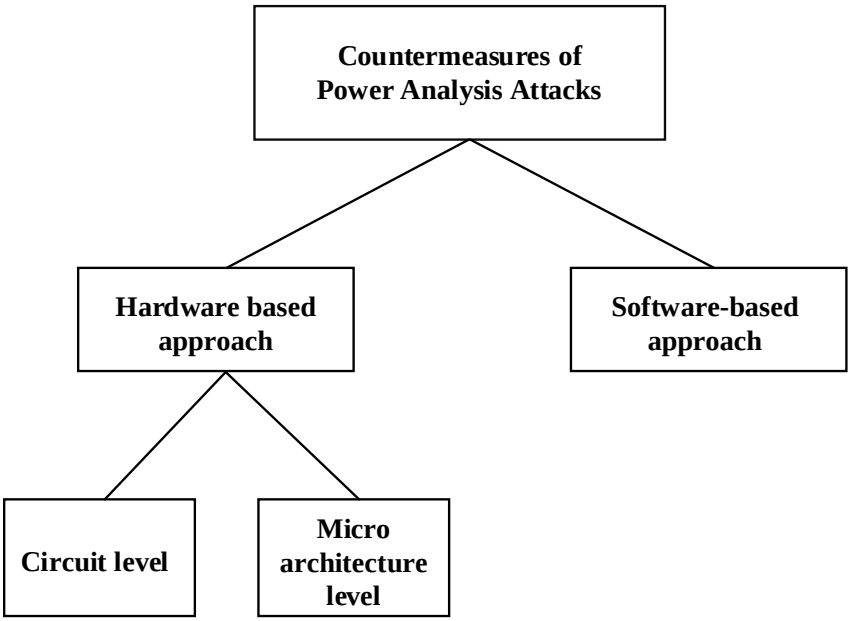

Figure 2.4 Classification of countermeasures against power analysis attacks.

As Figure 2.4 has been shown, the countermeasures against the power analysis attack are classified into two categories: hardware-based and software-based. Hardware-based countermeasures, as the name implies, involve adding new components or modifying existing components of a cryptosystem in order to reduce the leakage of sensitive information and data, such as power consumption. One example of a hardware-based countermeasure is the use of tamper-resistant hardware, which is designed to make it more difficult for attackers to extract sensitive information from a device. Tamper-resistant hardware typically includes features such as physical shielding, hard-to-read memory, and secure boot mechanisms, all of which help to prevent attackers from tampering with or extracting data from the device. Another example of a hardware-based countermeasure is the use of differential power analysis-resistant hardware, which is designed to minimize power consumption variations during cryptographic operations, making it more difficult for attackers to extract sensitive information from power consumption traces [34]. In addition, hardware-based countermeasures can also include the use of specialized processors, such as Field Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs), which are designed to perform cryptographic operations more efficiently and securely than traditional processors. Overall, hardware-based countermeasures can be effective in reducing the risk of power analysis attacks. However, they can also be expensive to implement and may not be feasible for all types of devices. It is important to carefully evaluate the costs and benefits of hardware-

based countermeasures before implementing them in a cryptosystem [35]. Hardware countermeasures can be categorized into two broad circuit-level and microarchitecture-level countermeasures. Circuit-level countermeasures include adding additional components to the circuits or making some changes to the circuits at an immense level. For instance, adding a source of noise or connecting a power line filter to the cryptographic device. Microarchitecture-level countermeasures include adding additional components or making some changes at the logic gate level. To illustrate the point, consider that to make a different power consumption, attackers add extra transistors to a CMOS gate [34-36].

Software-based countermeasures, as the name suggests, involve making modifications to the program or data code implemented on the microcontrollers of cryptographic systems. These modifications are intended to prevent attackers from extracting sensitive information from power consumption traces. One example of a software-based countermeasure is the use of random instruction insertion, where incorrect instructions are randomly inserted into the code being executed by the microcontroller. This makes it more difficult for attackers to determine which instructions are related to the cryptographic operation and which are not. Another example is the use of instruction permutation, where the order of instructions is randomized to make it more difficult for attackers to analyse the power consumption traces and extract sensitive information. Other software-based countermeasures include algorithmic modifications, such as adding noise to the data being processed or modifying the cryptographic algorithm to reduce power consumption variations during cryptographic operations. It is important to note that software-based countermeasures can be less effective than hardware-based countermeasures, as attackers may still be able to extract sensitive information from power consumption traces. However, they are often easier and less expensive to implement and can provide an additional layer of protection against power analysis attacks. Software-based countermeasures can be a valuable addition to a cryptosystem's security measures. It is important to carefully evaluate the effectiveness of any software-based countermeasure before implementing it and to consider the potential impact on system performance and complexity [35].

2.3.4.1.6 Security, Performance and Power Analysis of Power Analysis Attacks

Table 2.1 presents a clear depiction of the prevalence of power analysis attacks, indicating that a substantial number of researchers in the field monitor power consumption as a key metric during these attacks. The reason behind this approach lies in the direct correlation between power

consumption and the potential vulnerabilities that can be exploited. By closely observing the fluctuations in power consumption, researchers can effectively identify and analyse the attack patterns, facilitating the development of appropriate countermeasures. The significance of this power-based approach is that it allows researchers to detect subtle variations in power consumption caused by different attack scenarios. By understanding these variations, they can gain valuable insights into the weaknesses of a system and devise targeted defences against potential threats. The advantage of this methodology is that it is applicable across a wide range of devices and architectures, making it a versatile and widely adopted technique in the realm of security analysis. However, it is important to note that implementing countermeasures can sometimes come with trade-offs, particularly in terms of device performance. In certain cases, the integration of robust security measures may lead to a decrease in the overall performance of the devices. This challenge necessitates a careful balance between security and performance, where researchers must explore innovative ways to enhance device security without unduly compromising their efficiency and functionality. To address this concern, researchers often embark on a comprehensive analysis of the trade-offs between security and performance. They explore various mitigation strategies that offer a reasonable compromise between safeguarding against power analysis attacks and maintaining acceptable device performance [13]. This delicate balancing act ensures that the introduced countermeasures not only enhance security but also allow devices to deliver optimal functionality in real-world applications.

Table 2.1 Security, performance and power consideration of power analysis attacks in different research.

Reference	Attack	Defence Strategy	Performance Consideration	Power Consideration
[11]	✓	Boolean Masking	×	×
[12]	✓	Key-Dependent Control Flow Obfuscation	×	✓
[13]	✓	Energy Harvesting Countermeasures	✓	✓
[14]	✓	Logic Locking with Power-Concealing	×	✓

		Transformations		
[15]	✓	Chaotic Power Supply with Noise Injection	×	✓
[16]	✓	Time-Varying Leakage Current Balancing	×	✓
[34]	✓	Physical Unclonable Functions (PUFs) for Key Storage	×	✓
[35]	✓	Generative Adversarial Networks (GANs) for DPA Countermeasures	×	✓
[36]	✓	Privacy-Preserving Statistical Countermeasures	×	✓

2.3.4.2 Electromagnetic Attacks

Electromagnetic attacks represent a sophisticated category of side-channel attacks that leverage the analysis of electromagnetic radiation emitted by a target system [16]. These attacks have gained prominence due to their non-invasive and passive nature, allowing attackers to exploit the inherent electromagnetic emissions produced during the normal operation of a device without causing any discernible physical harm [17]. This covert characteristic makes EM attacks particularly insidious, as they can compromise sensitive information without triggering immediate suspicion. The fundamental principle underpinning these attacks revolves around the precise measurement and analysis of the electromagnetic radiation emanating from a targeted system. By doing so, attackers can effectively unveil hidden or confidential data processed within the device. Successful execution of an EM attack necessitates an intricate understanding of the intricate layout of the semiconductor chips within the system [16]. Consequently, the measurement of electromagnetic radiation should ideally be conducted within a controlled and isolated environment to ensure the accuracy and reliability of the obtained results [17]. One of the noteworthy advantages of EM attacks is their potential to circumvent traditional security measures that focus on digital

safeguards. Unlike conventional hacking methods that exploit software vulnerabilities, EM attacks venture into the realm of the physical characteristics of a system. This paradigm shift highlights the importance of not only fortifying software and network defences but also bolstering the physical security of hardware components. To mitigate the risks posed by EM attacks, organizations must adopt a multifaceted approach that encompasses both hardware and software countermeasures. Implementing robust encryption algorithms and access controls can certainly help safeguard sensitive data from potential breaches. Furthermore, the physical design of semiconductor chips should be carefully structured to minimize electromagnetic radiation leakage, thereby reducing the attack surface for potential adversaries [16]. To fight against electromagnetic attacks, developers and designers of embedded systems can employ various countermeasures, such as using EM-shielding techniques, incorporating secure hardware designs, implementing cryptographic algorithms resistant to side-channel attacks, and applying randomization techniques to reduce information leakage. Regular security assessments, including testing for vulnerabilities related to electromagnetic radiation, are also crucial to identify and mitigate potential risks [17].

2.3.4.2.1 Countermeasures Against Electromagnetic Attacks

The countermeasures against EM attacks can be categorized into two broads: hardware and software modifications.

2.3.4.2.1.1 Hardware Modifications of the Electromagnetic Attacks Countermeasures

A promising approach to mitigating the risks EM attacks involves the incorporation of an additional metal layer onto the chip's architecture, particularly within the area that emits radiation. This strategic placement of the extra metal layer serves a twofold purpose: it introduces deliberate noise into the EM field, and it contributes to the obfuscation of emanations [17]. By positioning this supplemental metal layer atop the chip, the unintended electromagnetic emissions generated during the device's operation are met with interference, resulting in a muddled and less coherent EM field. This noise injection plays a crucial role in thwarting the efforts of potential attackers aiming to exploit these emissions to glean sensitive information. Furthermore, another avenue for bolstering security against EM attacks involves embracing advancements in chip fabrication technology. The reduction of transistor size, also known as technology scaling, presents a compelling solution [17]. This involves implementing smaller transistors in the chip's design, a trend in line with the evolution of Moore's Law. Smaller transistors inherently generate less

electromagnetic radiation due to their reduced dimensions, making it significantly more challenging for attackers to detect and decipher any emanations that might leak sensitive data. This miniaturization process effectively minimizes the attack surface available to adversaries seeking to exploit EM emissions.

Additionally, electromagnetic attacks delve into the exploitation of inadvertent electromagnetic emissions emitted by electronic devices. This covert extraction of sensitive information demands a two-pronged approach: attackers can augment their equipment's hardware components to amplify the effectiveness of their attacks [18]. These hardware modifications could involve fine-tuning the capabilities of sensors, antennas, and receivers to capture even fainter emissions with greater accuracy. By enhancing the sensitivity and precision of their equipment, attackers can amplify the signal-to-noise ratio, thus improving their ability to discern valuable data from the background electromagnetic noise. Overall, fortifying systems against electromagnetic attacks necessitates a multifaceted strategy. The incorporation of an extra metal layer on the chip, coupled with the adoption of smaller transistor sizes, constitutes a formidable defence against the exploitation of electromagnetic emissions. Furthermore, recognizing that attackers can enhance their equipment to extract more data from unintended emissions underscores the importance of proactive defence measures. Organizations and designers must continuously innovate and adapt to these evolving threats by implementing countermeasures that address both the hardware and software aspects of security, thereby creating a robust shield against the complex landscape of electromagnetic attacks [17]. Here are some examples of this type of countermeasure:

1. **Antennas**: The attacker can modify the antenna to enhance its sensitivity or directivity. A more sensitive antenna can pick up weaker signals, while a directional antenna can focus the electromagnetic emissions from the target device, making it easier to capture the data.
2. **Amplifiers**: Amplifiers can be used to boost the strength of the electromagnetic emissions picked up by the attacker's equipment. This can improve the signal-to-noise ratio, making it easier to extract the data [164].
3. **Filters**: Filters can be used to remove unwanted frequencies from electromagnetic emissions, making it easier to isolate the signals of interest.
4. **Modulators**: Modulators can be used to change the frequency of the electromagnetic emissions, making them easier to detect and capture.
5. **Probes**: Probes can be used to make physical contact with the target device, allowing the

attacker to capture electromagnetic emissions more directly [165].

It's worth noting that hardware modifications may not always be necessary to carry out electromagnetic attacks, as sensitive information can often be extracted using off-the-shelf equipment. However, modifications can be used to increase the effectiveness of such attacks in certain circumstances [16].

2.3.4.2.1.2 Software Modifications of the Electromagnetic Attacks Countermeasure

One of the effective strategies to enhance the security of encryption procedures against power analysis attacks involves the implementation of nonlinear key updates. This technique introduces complexity to the process of updating encryption keys, rendering it difficult for attackers to correlate power traces with specific key changes. By incorporating these nonlinear key updates, the vulnerability of the encryption process to power analysis attacks is significantly diminished [17]. Another approach to bolstering security entails the incorporation of key-use counters. These counters serve as a protective measure by restricting attackers from amassing the substantial volume of samples they typically require executing an effective attack. This countermeasure serves to thwart attackers' efforts by curtailing their ability to gather the critical information needed to exploit potential vulnerabilities in the encryption process [18]. Cumulatively, the integration of these supplementary measures has yielded a considerable reduction in the susceptibility of the encryption process to information leakage through electromagnetic (EM) side-channel attacks. This reduction, in turn, substantially elevates the threshold for the number of samples needed to orchestrate a successful attack. Consequently, the practicality of executing such attacks is significantly undermined, making them increasingly implausible. However, it is important to acknowledge that, even in the face of these fortified countermeasures, an attacker armed with an infinite pool of samples would still retain the capacity to execute Electromagnetic Analysis (EMA) attacks on the attenuated signal. This underscores the importance of ongoing research and vigilance in the realm of encryption security [61]. Furthermore, the evolution of technology holds promise for an eventual shift away from traditional semiconductor-based methodologies. The adoption of alternatives to semiconductor technology presents a potential avenue to eliminate the leakage of sensitive information through EM side-channels. This emerging possibility offers a glimpse into a future where the inherent vulnerabilities associated with current semiconductor-based systems can be mitigated, thereby ushering in a new era of enhanced encryption security [60]. It is good to note that no single countermeasure can guarantee complete protection against

software electromagnetic attacks. Employing a combination of these countermeasures and following best practices in software and hardware security will provide a more robust defence against this type of threat [17].

2.3.4.2.2 Security, Performance and Power Analysis of Electromagnetic Attacks

A brief comparison between different studies shows that in most of them, the main concern of researchers is to apply a good countermeasure for electromagnetic attacks to decrease the possible risk (Table 2.2), however, some of these defence implementations may affect other aspects of the devices such as power consumption, performance or both of them.

Table 2.2 Security, performance and power consideration of electromagnetic attacks in different research.

Reference	Attack	Defence Strategy	Performance Consideration	Power Consideration
[16]	✓	Layered metal shields	×	×
[17]	✓	Cage structures	×	×
[18]	✓	Leakage-resilient algorithms	×	×
[60]	✓	Run-time monitoring	×	×
[61]	✓	Power gating	✓	✓

2.3.4.3 Timing Attack

Timing attacks are a type of side-channel attack that involves analyzing the time it takes for a system to perform cryptographic operations. These attacks exploit the fact that the time it takes for a system to operate can vary depending on the input data and the secret key used for the operation. By carefully measuring the time it takes for a system to perform a cryptographic operation under different conditions, an attacker may be able to deduce sensitive information about the system, such as the secret key used for encryption or decryption. Timing attacks can be particularly effective against systems that are vulnerable to variations in timing caused by conditional branching or memory access patterns. For example, an attacker may be able to exploit timing variations in an encryption algorithm to determine the value of a secret key or other sensitive information. Similarly, an attacker may be able to exploit timing variations in a password-checking algorithm to determine whether a given password is correct [19]. To prevent timing attacks, it is important to ensure that cryptographic operations take a constant amount of time regardless of the

input data or secret key. This can be achieved through the use of constant-time algorithms and careful coding practices that eliminate timing variations caused by conditional branching and memory access patterns. Additionally, it is important to use appropriate measures to protect sensitive information, such as key encapsulation and secure storage techniques. By taking these steps, it is possible to greatly reduce the risk of timing attacks and protect sensitive information from unauthorized access [11-19]. Timing analysis attacks can allow recovery of the devices or systems secret key. Timing analysis attacks are classified into keystroke and SSH timing analysis attacks [11].

Generally discovered sources of timing weakness include:

- Data dependent differences in instruction times
- Early exit
- Data dependent code branches
- Cache access times [19].

2.3.4.3.1 Countermeasure Against Timing Attacks

As Figure 2.5 shows, the countermeasures against the Timing attacks can be classified in three categories:

- Application-Level Countermeasures.
- Hardware Countermeasures.
- Operating System Countermeasures.

The main goal of functional countermeasures is to get a great balance between performance and security [37].

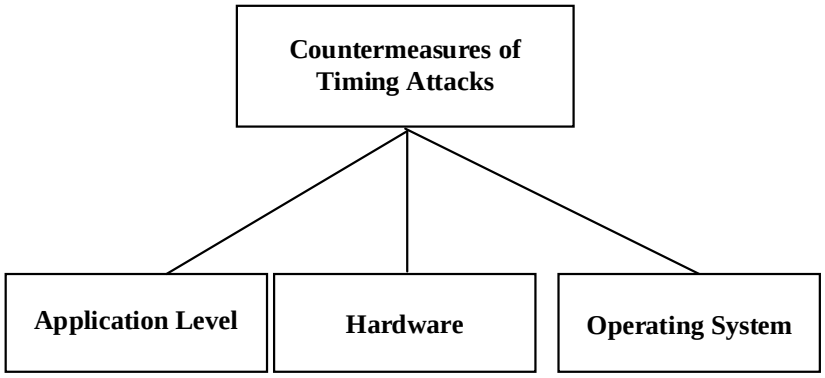

Figure 2.5 Classification of countermeasures against timing attacks.

2.3.4.3.1.1 Application-Level Countermeasures

To address timing attacks, countermeasures for cryptographic applications have been developed, showing great promise and ease of implementation while providing a satisfactory level of security. However, a closer examination reveals several shortcomings associated with these countermeasures [37]. One major drawback is that application-level countermeasures tend to be heavy and often fall short of providing practical effectiveness. They require substantial computational resources and may not be feasible for resource-constrained systems. Another challenge arises from the need for uniformity in countermeasure application [38]. To patch the applications against timing attacks, all of them must be modified with the same countermeasure. This approach can be cumbersome and time-consuming, particularly in large systems with multiple applications, as it demands the modification of each application individually. Unfortunately, the implementation of these bulky countermeasures significantly impacts the overall system performance and energy consumption. The additional computational overhead required to defend against timing attacks can lead to reduced system efficiency and increased power consumption, which are critical concerns for energy-efficient and high-performance systems. Nevertheless, despite these issues, the countermeasure method does offer a quick and immediate solution to halt and prevent timing attacks, as evidenced by research sources [58-59].

2.3.4.3.1.2 Hardware Countermeasures

With a slight modification in the hardware, it can make the attacks less successful and more difficult to happen again with a little extra effect on the system performance [37]. Hardware countermeasures are an effective way to mitigate timing attacks as they can provide strong protection against side-channel attacks at the hardware level. Several hardware countermeasures

can be implemented to prevent timing attacks, some of which include:

1. **Pipelining:** Pipelining is a technique that involves breaking down a computational task into smaller stages, which can be executed in parallel. By executing multiple stages in parallel, pipelining can reduce the time it takes to execute a cryptographic operation and can also make it more difficult for attackers to identify the exact timing of each stage.
2. **Differential Power Analysis Countermeasures:** DPA is a type of side-channel attack that can be used to extract the secret key of a cryptographic algorithm by analysing the power consumption of a device during operation. DPA countermeasures involve adding noise or randomness to the power consumption of a device, making it difficult for attackers to distinguish between the power consumed by the device when performing cryptographic operations and the power consumed during other operations.
3. **Shielding and Isolation:** Shielding and isolation can also be effective hardware countermeasures against timing attacks. Shielding can be used to protect sensitive components from electromagnetic interference, while isolation can be used to prevent attackers from accessing sensitive components directly [58-59].

By implementing these hardware countermeasures, system designers and developers can greatly reduce the risk of timing attacks and provide strong protection against side-channel attacks at the hardware level [58].

2.3.4.3.1.3 Operating System Countermeasures

Since hardware countermeasures can be expensive to implement, and application layer countermeasures can be extensive. However, the operating system can be an effective candidate for addressing security threats. By using different hardware resources for security domains in the operating system using the scheduler, the system can be protected from attacks that exploit the victim's execution behaviour through a spy process. To prevent fine-grained attacks that target scheduler policies, changes to the scheduling algorithm can be made. One effective approach is to introduce a minimum threshold behind the context switch that occurs between two processes. This approach can help to reduce the effectiveness of attacks that target the scheduler policies. Overall, the search for effective countermeasures against security threats is ongoing, and there is no one-size-fits-all solution. However, by implementing a combination of hardware and software-based countermeasures and continually improving and updating them, we can enhance the security of operating systems and protect against potential attacks [37-38].

2.3.4.3.2 Security, Performance and Power Analysis of Timing Attacks

Table 2.3 shows different aspects of devices in timing attacks and their countermeasures. As we can see in some research has been tried to consider the performance when defence strategies have been implemented [37-38-58] but in some cases no, furthermore in most of them, did not consider power consumption during an attack or its countermeasures.

Table 2.3 Security, performance and power consideration of timing attacks in different research.

Reference	Attack	Defence Strategy	Performance Consideration	Power Consideration
[11]	✓	Time Warp	×	×
[19]	✓	Brute-force timing attacks	×	×
[37]	✓	Remote cache timing attacks	✓	×
[38]	✓	Formal verification	✓	×
[58]	✓	Probabilistic padding	✓	×
[59]	✓	Instruction scheduling	×	×

2.3.4.4 Fault Injection Attacks

Fault injection is a technique used to identify and test the robustness of hardware and software systems by simulating faults and errors that can occur in the real world. The technique involves introducing controlled faults or errors into a system to observe its behaviour and identify any vulnerabilities or weaknesses that can be exploited by attackers [11]. In a fault injection attack, various external parameters and environmental conditions are manipulated to induce errors in the system. These include supply voltage, clock speed, temperature, radiation, electromagnetic interference, and more. By manipulating these parameters, attackers can cause the hardware or software to behave unpredictably, leading to false activation, data corruption, or even physical damage. The outcome of a fault injection attack can be devastating, especially if it is targeted towards secure or sensitive systems. Attackers can use the technique to gain access to sensitive data, corrupt the code, or disrupt the normal operation of the system. Additionally, fault injection can be used as a precursor to other types of attacks, such as software attacks, to identify vulnerabilities that can be exploited [39]. To mitigate the risk of fault injection attacks, it is important to design systems with robustness and resilience in mind. This can include implementing

error-correcting code, incorporating redundancy, and designing hardware and software components that are resilient to external factors such as temperature, radiation, and electromagnetic interference. Additionally, it is important to regularly test and validate systems for fault injection vulnerabilities to identify and mitigate any potential weaknesses [54]. It's important to note that fault injection attacks are typically conducted in controlled environments or specialized testing setups, as they can have unpredictable consequences and cause damage to the targeted systems. However, performing such attacks helps developers identify vulnerabilities and implement robust countermeasures to enhance the security of embedded systems [11].

2.3.4.4.1 Countermeasure Against Fault Injection Attacks

The very simple countermeasure against fault injection attacks is to execute twice the same algorithm of the cryptographic for each input, and then check if the results match. While executing an algorithm twice and comparing the results may provide some level of protection against naturally occurring faults, it may not be effective against deliberate fault injection attacks by motivated attackers. Injecting two different weaknesses, one for each implementation of the algorithm can allow an attacker to bypass this countermeasure [39-54-55]. Furthermore, with the advancement of fault injection techniques, attackers can now inject faults with exact timing while the software is running, potentially affecting multiple implementations of the same algorithm. While it may be difficult to inject faults in a wide range of instructions, attackers can still potentially exploit vulnerabilities in the algorithm to compromise its security. To improve the effectiveness of countermeasures against fault injection attacks, it may be necessary to use more advanced techniques such as error correcting codes or redundancy to detect and correct errors introduced by fault injection attacks. It is also important to continually monitor and update cryptographic algorithms to ensure that they remain secure against evolving attack techniques [56-57].

2.3.4.4.2 Security, Performance and Power Analysis of Fault Injection Attacks

In fault injection attacks a few studies have been done regarding all parts of devices during an attack and their defence such as performance and power consumption. However, by implementing some countermeasures, researchers try to decline the possible effect of defence strategy on performance [39-54-56]. To illustrate the point, consider Table 2.4.

Table 2.4 Security, performance and power consideration of fault injection attacks in different research.

Reference	Attack	Defence Strategy	Performance Consideration	Power Consideration
[11]	✓	Resilient logic styles	×	×
[39]	✓	Threshold logic	✓	✓
[54]	✓	Masking and scrambling techniques	✓	✓
[55]	✓	Post-processing techniques	×	×
[56]	✓	Adversarial training	✓	×
[57]	✓	Error detection and correction codes	×	×

2.3.4.5 Covert Channel Attacks

A covert channel is a method of communication that is not intentionally built for transmitting information. Unlike traditional communication methods, such as writing a message or checking whether a file is protected, covert channels rely on an insider process to leak information to an outsider process that would not normally be allowed to access that information. One way that covert channels operate is by using a Trojan horse program, which is surreptitiously inserted into a computer system ahead of time. This program is designed to allow an insider process to bypass security measures and communicate with an outsider process without detection. Because the outsider process is unprivileged, it is not subject to the same level of security checks and can receive information that would normally be restricted [40]. Covert channels can be used for a variety of purposes, including espionage, theft of intellectual property, or other malicious activities. Because they are designed to be hidden from detection, they can be difficult to identify and prevent. However, by understanding how covert channels operate and being vigilant for signs of their use, it is possible to minimize the risk of unauthorized information disclosure and protect sensitive data [41]. The mechanism which the covert channel uses is not built for communication, like writing, and checking if a file is protected to convey 1 or 0. The covert channel uses an insider process that leaks sensitive information to an outsider process [40-41]. In 1973, Lampson et al. [79] analysed this issue and mentioned the possibility of using covert channels, i.e., detecting system properties not designed for communication, with the aim of leaking restricted data. These attacks pose a significant security threat against any aspect or part of our system, from cloud systems to operating systems and multi/many-core chips, etc. [63].

The covert channel can compromise chip-level security. A covert channel is a type of communication that is hidden from detection by normal security measures, allowing attackers to transmit information in a way that is not supposed to be possible or permitted. Covert channels can be created through various means, including exploiting weaknesses in a system's design, using unusual or unexpected inputs, or taking advantage of system resources in ways that were not intended. One example of a covert channel is a timing channel, which uses differences in response times to transmit information. For instance, an attacker might send a series of requests to a server, with the timing of the responses indicating the bits of a secret message. Another example is a storage channel, which uses data storage mechanisms to transmit information. For instance, an attacker might use the amount of available disk space to signal the bits of a message or use variations in the frequency of read or write operations to encode information. Covert channels can be difficult to detect and mitigate, as they often operate at a low level and are designed to be difficult to distinguish from normal system behaviour. Some common detection techniques include traffic analysis, which looks for patterns in network traffic that may indicate a covert channel, and behaviour analysis, which looks for unusual activity or resource utilization within a system [40]. Mitigation strategies for covert channels often involve a combination of design improvements and detection techniques. For instance, a system can be designed with fewer vulnerabilities or stricter access controls to reduce the potential for covert channels to be exploited. Additionally, monitoring and detection tools can be used to identify and mitigate covert channels when they are discovered. Overall, covert channels pose a significant risk to system security and can be challenging to address. However, with careful attention to design and vigilant monitoring and detection, it is possible to reduce the risk of covert channels and protect sensitive data and information [41].

2.3.4.5.1 Thermal Cover Channel

In many communication media the covert channel such as timing, heat, or indistinct sound. The heat transfer which is known as thermal covert channels (TCC) can be particularly dangerous [47]. Thermal covert channels can spatially be traced in multi-core systems. Such as any communication system, a thermal cover channel contains a receiver and transmitter [48]. In the transmitter program, the temperature signals are produced from sensitive data or information such as user passwords. It can move over the chip by heat transfer among processor cores, and finally, the thermal signals reach the destination (receiver) [47]. While thermal signals are transmitted with a thermal covert channel throughout a chip, it is highly likely to be degraded or disrupted by

environmental temperature variations or thermal noise [49]. An advanced countermeasure against the thermal covert channel needs to check and scan the signal frequency spectrum, to find positioning affected cores and blocking them [47]. In the case of the thermal covert channel, the mechanism used to transfer information is through the transfer of heat. This is done by manipulating the temperature signals produced from sensitive data or information, such as user passwords, and transferring them over the chip by heat transfer among processor cores. The thermal signals then reach the destination or receiver where they are reconstructed into the original information. However, the use of the thermal covert channel can compromise chip-level security and it is highly likely to be degraded or disrupted by environmental temperature variations or thermal noise. This can be particularly dangerous as it can lead to sensitive information being leaked to unauthorized parties. To prevent the use of thermal covert channels, an advanced countermeasure needs to be implemented. Before applying any defence strategy, the most important step is detection. There are different methods to detect attacks introduced. For instance, Kean et al. [86] experiments illustrated that one system can include different tags with several codes that can be all detected in a relatively short time, during the normal operation of the system. Experiments presented by Murdoch et al. [97] illustrated those changes in clock skews resulting from modest temperature changes can be remotely identified through network packet timestamps, even over tens of router hops [91]. Huang et al. [62-63] and Jiachen W et al [67] shown that for the detection step, each core can measure the spectrum of its CPU workload traces that are recorded through a few fixed time intervals, and then it uses a frequency scanning technique to detect if there exist any thermal covert channel attacks. This can involve scanning the signal frequency spectrum to identify affected cores and blocking them. It is also important to implement temperature monitoring and control mechanisms to prevent the production and transfer of temperature signals [49]. Overall, covert channels, including the thermal covert channel, are a serious security concern and should be addressed with appropriate countermeasures to prevent the unauthorized transfer of sensitive information [47]. In recent years, some new designs of thermal covert channels have been proposed. For instance, Masti et al. [65] introduced a new design of thermal covert channels that has improved noise immunity and throughput. They showed the feasibility of side channels and especially thermal channels on multi-core systems for the first time, e.g., created a 1-hop TCC channel (a channel of transmitter and receiver are 1-hop apart from each other) that could reach a throughput of 1.33 bps along with a BER of 11%. In this study, researchers

did not consider power consumption and high BER. In separate research [75], confirmed that a TCC channel has a throughput of more than 45 bps, and it also showed that a TCC channel with a throughput of more than 5 bps with a BER less than 1%. Recis et al. [78] researched thermal-related attacks where sensitive information (e.g., user password) is transmitted by reading the duration and speed of the fan. The angular speed of the fan changes based on the chip temperature to prevent it from overheating. A thermal covert channel over an x86-based scheme could transmit data with an average BER of 13.22% [65]. A different encoding platform based on return-to-zero was proposed in [62]. In this method, it was illustrated that a high transmission frequency could prevent thermal noise from participating with other active cores. Furthermore, Jiachen W et al. [67] suggested a countermeasure strategy based on scanning the frequency spectrum rapidly to detect any possible attack. When a thermal covert channel is detected, and its frequency of transmission is found, a strong noise source is applied to the transmission frequency band for targeted jamming that a thermal covert channel could be blocked with a packet error rate (PER) of 85%. In this strategy, by adding strong scour noise, the systems may face overheating during the transmission process or result in increased power consumption. Another method that can be used to fight the TCC attacks is Dynamic Voltage Frequency Scaling (DVFS) [62-63]. In this countermeasure, after detecting the TCC attack and finding its position, it is tried to block and fight possible attacks. In the scheme, Huang et al. [62-63] confirmed that this approach could cause TCC attacks to suffer extremely high BER (> 92%), by applying the DVFS strategy.

2.3.4.5.1.1 Factors Affecting the Thermal Covert Channel

The heat generated by a CPU is intricately linked to its power consumption. Manipulating the amount of power utilized by a core enables direct control over its temperature. Leveraging this principle, the source application can exploit fluctuations in power consumption to induce temperature variations within the CPU. This ingenious mechanism forms the basis of the thermal covert channel, where data transmission occurs. To facilitate this process, the source application strategically introduces changes in power usage, leading to corresponding temperature fluctuations. These variations in temperature can subsequently be utilized to encode and decode data symbols, achieved through distinct utilization patterns. The primary strategy to elevate power consumption involves generating increased utilization across the system's cores [39]. By carefully crafting utilization patterns, the source can effectively communicate symbols through resulting temperature changes. It is crucial to note that while the ideal scenario envisions complete control

over the thermal behaviour via the source, modern systems exhibit a more complex landscape. Factors such as power and thermal management strategies, as well as various performance-enhancing tools, exert their influence on the source's impact on the thermal channel. Moreover, the presence of other concurrent processes can introduce thermal noise through their CPU utilization. To advance the experiments beyond controlled laboratory settings and into real-world scenarios, it is imperative to account for all potential variables that can shape a system's thermal behaviour [38]. Bartolini et al. [73] meticulously outline these factors and their controllability within a laboratory environment. This approach serves to mitigate significant interferences that could otherwise disrupt the integrity of the thermal communication channel. By understanding and managing these influencing factors, researchers can bridge the gap between theoretical underpinnings and practical applicability, facilitating the translation of thermal covert channel research into real-world settings.

- **Thermal Noise:** Thermal noise originates from two primary factors: fluctuations in the ambient temperature surrounding a device and additional processes that impose a heavy load on the CPU. In the research conducted by Bartolini et al. [73], efforts were made to minimize the interference caused by other concurrent processes, allowing only the essential core operations of the operating system to execute. Consequently, temperature variations are predominantly attributed to the intrinsic source itself. In practical scenarios involving potential attacks, a similar state of low foundational utilization can be achieved when the system remains inactive, such as a smartphone charging overnight or a laptop idling in an office setting over the weekend.
- **Core Pinning:** In modern multi-core systems, the utilization of multiple logical cores is harnessed to enhance overall system performance. As a consequence, processes may be dynamically shifted among these logical cores to optimize resource allocation. In the study by Bartolini et al. [73], an approach was employed to anchor the source and sink processes to specific cores, effectively preventing such migrations. However, this core pinning technique is incompatible with the threat model, as it introduces modifications to the natural behaviour of the thermal channel, a deviation that is prohibited within the model's constraints. Consequently, the source process is compelled to distribute its load across various cores. In the context of this attack scenario, where the system is not actively engaged, only the core harbouring the sourcing process will experience significant

temperature fluctuations. He introduced the concept of the "all-core channel," which encompasses the thermal channel formed by aggregating temperature readings from all available cores. Within this channel, the amalgamation of temperature variations induced by the source transmission coalesces with the thermal noise engendered by other concurrent processes. In scenarios characterized by minimal thermal noise, the primary fluctuations observed within the all-core channel correspond to the transmitted signal itself.

- **Frequency Governor:** In modern devices, a strategy known as dynamic voltage and frequency scaling is employed to optimize power consumption. This technique dynamically adjusts CPU frequency during runtime, effectively reducing power consumption while striving to uphold peak performance. The underlying software within the operating system responsible for managing the CPU's operating frequency is referred to as the "governor". This governor employs the CPU utilization as a pivotal factor in determining the appropriate frequency to apply, based on its preset policy. Given that distinct frequency levels yield varying levels of heat production, the correlation between thermal behaviour and the utilization generated by running processes becomes more intricate in systems with dynamic frequency adjustments [39]. In the experimental setups detailed by Bartolini et al. [73], the CPU frequency was held at a fixed value, thus negating any interference that might arise from the governor's behaviour. However, within the context of the presented scenario, processes lack the authority to dictate frequency adjustments, necessitating the consideration of governor effects on the system. It's worth noting that various devices are equipped with distinct default governors, which can either impede or augment the thermal dynamics of the thermal channel. A notable facet is that frequency governors construct their frequency policy around CPU utilization a parameter that the source process can actively influence. This introduces a level of controllability over the governor's behaviour, allowing the source to indirectly steer the operating frequency and, in turn, modulate the heat generation profile of the CPU. While the absence of fixed frequency control poses challenges, the capacity to manipulate CPU utilization opens avenues for subtle influence on the governor's frequency decisions [62]. In navigating this intricacy, researchers must acknowledge the interplay between governor policies, CPU utilization, and their collective impact on the thermal behaviour of the system.

2.3.4.5.2 Timing Covert Channel

Among all forms of information leakage, the covert channel can provide a secret communication medium between two processors to leak sensitive information that violates the security policy of a system [116]. Timing covert channel attack is one of the basic side-channel attack methods introduced at CRYPTO'96 by Paul Kocher [117]. This technique allows the attacker to extract an algorithm's secret, such as a secret key. Timing attacks have mostly been used in cryptography; however, they can apply to other possible attack targets [118]. Execution of cryptographic algorithms, due to performance optimizations, often computations in non-constant time. If these operations include secret or sensitive information, these timing variations can result in the leak of some information that provided useful knowledge of this implementation, therefore, attackers can utilize this information to recover the secret keys [119]. Furthermore, timing characteristics depend on the encryption keys because different systems or devices take different amounts of time to process different inputs. Therefore, a timing attack looks at how long this process takes and utilizes statistical analysis to find the decryption key and gain access [120]. The basic idea behind timing attacks is that the time taken to perform a cryptographic operation depends on the values of the input data and the internal state of the system, including the secret key. In a timing attack, the attacker measures the execution time of the algorithm on different inputs and analyses the statistical variations in the timings to infer the values of the secret key [139]. The attacker can use various techniques to measure the execution time, such as hardware performance counters, software timers, or network latency. The attacker usually needs to perform many measurements to extract enough information to infer the secret key accurately. Timing attacks can be challenging to implement in practice, as they require a precise measurement of execution time and sophisticated statistical analysis to extract the key values accurately [140]. Koç and Acıiçmez et al. [126] summarised timing attacks between threads on a single core. This field is growing quickly, and many attacks have since been proposed, exploiting resources shared between cores and across packages. Biswas et al. [123] conducted a study on timing channels as well as focusing on hardware-based timing channels such as cache attacks (in-system timing channels), Zander et al. [124] surveyed timing covert channels (remote timing side channels). Ge et al. [129] surveyed timing-based covert channels and timing-based side channels in cloud settings. Xu et al. [138] surveyed timing attacks and defences on different scales, therefore, providing a comprehensive overview of this type of research landscape. Bernstein et al. [131] suggested a statistical timing

attack on the Advanced Encryption Standard (AES). They applied the first round of AES utilizes plaintext to directly index a look-up table, which results in changing execution timing. Hence, the attackers can exploit or recover the secret keys using the correlation between the time and the data from the victim. Bonneau et al. [122] reviewed cache collisions deployed during the operating process and recorded execution times to attack. There are also some similar techniques targeting multi/many-core systems. Spreitzer et al. [130] apply Bernstein's attack on mobile devices and try to improve the attack. Kadam et al. [137] suggest how to fight against this type of attack by randomizing the process. They illustrated how can provide a good level of obfuscation. Jiang et al. [133] introduced a method that applies the relationship between the timing information in each encryption process and the number of unique cache accesses to find all secret keys. Timing attacks are not limited to cryptography and can apply to other systems that exhibit variations in execution time based on the input data or internal state. Protecting against timing attacks requires designing algorithms and systems that execute in constant time, independent of the input data or internal state. Timing attacks are particularly effective against implementations of cryptographic algorithms that use conditional statements or loops that depend on the secret key. These conditional statements or loops can cause variations in execution time, which the attacker can use to infer the values of the secret key. For example, if a cryptographic algorithm uses a conditional statement that checks if a certain bit of the secret key is set, the execution time will vary depending on whether the bit is set or not. The attacker can measure the execution time for different input values and infer the bit value of the secret key by analyzing the statistical variations in the timings [141]. Preventing timing attacks requires designing cryptographic algorithms and systems that execute in constant time, independent of the input data or internal state. A constant-time algorithm ensures that the execution time is the same for all input values, including those that depend on the secret key. One way to achieve constant-time execution is to use bitwise operations and arithmetic operations that do not depend on the secret key. Another way is to use randomization techniques, such as blinding or masking, to ensure that the execution time is independent of the secret key [139]. Figure 2.6 shows the process of timing attacks.

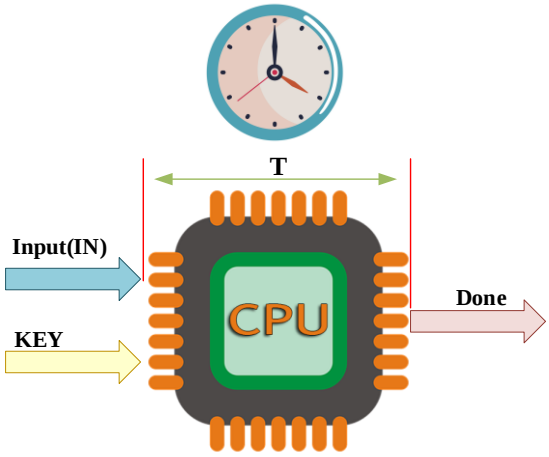

Figure 2.6 Process of timing covert channel attacks.

Figure 2.6 illustrates the key point of timing attacks is to identify the timing differences in cryptographic operations, instead of the actual operating time. Since timing attacks deduce secret information and keys only from runtime measurements of successive requests, they are a real threat to multi/many-core systems or devices. Timing covert channel attacks are particularly difficult to detect and prevent because they do not rely on the content of the communication itself but on the timing of events. Moreover, these attacks can be executed using minimal resources, making them particularly attractive to attackers [142]. To prevent timing covert channel attacks, several countermeasures can be employed, such as reducing the variability in the timing of events, adding noise to the timing signals, and monitoring for suspicious timing patterns. Additionally, security-conscious software design can help prevent such attacks by avoiding sensitive information exchange through shared resources that can be manipulated by attackers [143]. Timing covert channel attacks can be classified into two types: storage-based and network-based. In storage-based timing covert channel attacks, the attacker manipulates the timing of events by varying the time it takes for data to be written or read from a shared storage location. In network-based timing covert channel attacks, the attacker manipulates the timing of events by delaying or accelerating the transmission of packets over a network [142]. An example of a storage-based timing covert channel attack is a cache-timing attack, where an attacker monitors the time, it takes for a cache to retrieve data and infers the contents of the cache based on the timing variations. An example of a network-based timing covert channel attack is a TCP/IP timing attack, where an attacker varies the timing of packets transmitted over a network to convey hidden information [144]. To detect and prevent timing covert channel attacks, several techniques can be employed. One such technique is

statistical analysis, which involves monitoring the timing patterns of events and looking for suspicious variations that may indicate the presence of a covert channel. Another technique is to use timing-resistant cryptographic algorithms that are designed to be resistant to timing attacks. Overall, timing covert channel attacks are a serious threat to the security of computer systems and networks. It is essential for security professionals to be aware of these attacks and to take appropriate measures to detect and prevent them [145]. For instance, one approach to detecting a timing covert channel is to use statistical analysis to look for patterns in the timing of events. For example, if two seemingly unrelated events consistently occur close together in time, it may be a sign that a covert channel is in use. Statistical analysis tools such as entropy analysis, autocorrelation analysis, and chi-square analysis can be used to identify such patterns. Another approach is to use software tools that monitor the behaviours of the system and look for anomalies that may indicate the presence of a covert channel. For example, if an application is using more CPU resources than expected or accessing the network more frequently than usual, it may be a sign that a covert channel is in use [133]. In addition, it may be possible to prevent the use of timing covert channels by implementing security measures such as access control, encryption, and intrusion detection. By limiting access to shared resources and monitoring their usage, it may be possible to prevent unauthorized parties from using timing covert channels to transfer information. Different methodologies have been introduced to detect timing attacks, for instance implementation of a Change Point Detection (CPD) algorithm. M. Basseville et al. [133] and Deldari, Shohreh, et al. [134] used CPD to detect abrupt shifts in time series trends and identify the system behaviours or Fanjas, Clément, et al. [135] suggested frequency scanning to detect these kinds of attacks. Detection of a timing covert channel attack can be managed by frequency scanning. In this study, frequency scanning was processed to detect existing timing attacks. In this study, bit ‘0’ is represented as low frequency while bit ‘1’ is represented as a sequence of high frequency. In our experiments, has assumed that minimal interference from the external processes in terms of computing workloads can change CPU frequency, if such interference is observed, we note an attack happens. Countermeasures against timing covert channels involve implementing security measures to prevent the unauthorized transfer of information by manipulating the timing of events.

2.3.4.5.3 Traffic/Header-Based Covert Channels

Traffic/Header-Based Covert Channels represent a sophisticated method of clandestine communication that exploits the inherent structure and characteristics of network traffic headers to discreetly transmit information. By embedding hidden messages within what appear to be legitimate network packets, these covert channels pose a significant challenge to detect without employing specialized detection techniques. The stealthy nature of these channels relies on the complexity and variability of network protocols, making their detection and prevention a formidable task for network security systems [174].

2.3.4.5.3.1 Types of Traffic/Header-Based Covert Channels

The intricacies of these covert channels span across various network protocols, each presenting unique challenges in detection and prevention:

1. IP-Based Covert Channels:

- **Time to Live (TTL) Field:** Attackers manipulate TTL values to encode and transmit data secretly. By varying the TTL field, information can be passed in a manner that is difficult to discern from regular traffic.
- **Identification (ID) Field:** This technique involves using the fragmentation process and the ID field values to facilitate covert data transmission. By carefully managing packet fragmentation, secret messages can be communicated.
- **Type of Service (TOS) Field:** The TOS bits are employed for covert communication, with specific patterns of these bits encoding information unnoticed in regular traffic [175].

2. TCP-Based Covert Channels:

- **Sequence Number Field:** Altering sequence numbers within TCP packets offers a method to transmit information covertly, exploiting the way packets are reconstructed at the destination.
- **Acknowledgment Number Field:** Similar to sequence numbers, manipulating acknowledgment numbers allows for the exchange of information hidden within the normal acknowledgment process.
- **Window Size Field:** By adjusting the window size values communicated in TCP headers, data can be signalled in a covert manner, exploiting the flow control mechanism [176].

3. HTTP-Based Covert Channels:

- **Order of Header Fields:** Altering the order in which header fields are presented can convey data, utilizing the flexibility in header ordering to encode messages.
- **Unused Header Fields:** This strategy involves utilizing header fields that are rarely used under normal circumstances for hidden communication, thereby reducing the likelihood of detection.
- **HTTP Response Splitting:** This involves injecting malicious code or messages into HTTP responses to create a covert channel, exploiting vulnerabilities in web applications [177].

2.3.4.5.3.2 Challenges in Detection and Prevention

Detecting and preventing traffic/header-based covert channels involve overcoming several challenges:

1. **Subtlety:** These channels are designed to blend seamlessly with normal network traffic, making them nearly invisible without detailed analysis.
2. **Diversity:** The wide range of fields and protocols that can be exploited for covert communication means that attackers have numerous avenues to establish these channels, complicating detection efforts.
3. **Resource Constraints:** The comprehensive monitoring and analysis required to detect these covert channels can be resource-intensive, demanding significant computational and network resources to scrutinize all traffic effectively [178].

In general, traffic/header-based covert channels present a complex challenge to network security, requiring continuous advancements in detection methodologies and a deep understanding of network protocol nuances to effectively mitigate their risks.

2.3.4.5.4 Countermeasure against Covert Channel Attacks

As we mentioned earlier the term “covert channel” refers to a communication channel that allows information to be transmitted in a manner that was not intended by the designers of the system. Such channels can be created due to weaknesses in the system’s design or design oversights. If a covert channel is created due to oversights, it may be possible to fix the issue once it is discovered. Covert channel attacks have been studied for different purposes, such as operating systems [78], multi-core chips [72-74], and cloud systems [77]. Covert channels can be set up using a large number of communication media that span from inaudible sound [40], arrival timing of packets

[103], magnetic field [107-108], and inter-light [105-106] to voltage [102]. At the chip level, covert channels can use dynamic frequency scaling [101], exploit cache-timing [104-109], etc. for data and information transmissions. For example, Darwish et al. [110] introduced a new online streaming methodology for the mitigation of covert timing channels. This method eliminates covert timing channels while having an impact on the Quality of Service (QoS). To test the performance of the proposed model, the classification-based strategy was applied. They confirmed that the classification accuracy of this model's flow fell down to 50%. However, if the channel is an inherent part of the system's design, it may be impossible to eliminate without a complete redesign of the system. In cases where a covert channel cannot be eliminated, the next best option is to limit its use. This can be achieved by decreasing the channel's size or capacity. The amount of information that can be leaked through the channel can be tolerable depending on the sensitivity of the information being transmitted. For example, if the channel's capacity is very small, it may not be possible to leak critical information before it expires, making the capacity of the channel acceptable [48]. However, increasing the channel's capacity can introduce noise or slow down system mechanisms, which can limit the system's performance. Therefore, it is essential to strike a balance between the capacity of the channel and the performance of the system. Overall, covert channels can be a serious issue for system security, and efforts should be made to eliminate them or limit their use. Designers should take a proactive approach to identify and address potential covert channels during the system design phase, but if a channel is discovered after deployment, appropriate measures should be taken to mitigate its impact [49]. In general, the covert channel happens for two reasons: weaknesses in the system design and design oversights. When covert channels happen because of oversights, they may be fixed once discovered but those main to the system can never be eliminated without the system redesigning [50]. If a covert channel is not fixed in the design stage, the best next option is to remove its possible utilization. However, it may cause a very ineffective system, while covert channels are often removed entirely just by replacing automated methods with manual methods. If a covert channel cannot be removed, its size or capacity should be decreased. Those capacities can be tolerable depending on the amount of leak-sensitive information. For instance, if the channel is very small so the critical data and information cannot be leaked before it expires, the capacity of the channel is acceptable. Increasing the channel capacity can lead to some problems because it indicates introducing noise or slowing down system mechanisms, which both cause the limited system's performance [51-52].

2.3.4.5.5 Security, Performance and Power Analysis of Covert Channel Attacks

Table 2.5 indicates when a countermeasure is implemented to an attack to fix the problems, it may affect the performance so, in the different studies, researchers try to consider the performance of devices and decrease its effect regardless of power consumption.

Table 2.5 Security, performance and power consideration of covert channel attacks in different research.

Reference	Attack	Defence Strategy	Performance Consideration	Power Consideration
[40]	✓	Content inspection	×	×
[41]	✓	Formal verification of security protocols	×	×
[47]	✓	Model checking	✓	×
[48]	✓	Traffic flow analysis	✓	×
[49]	✓	Instruction set randomization	✓	×
[50]	✓	Code obfuscation	✓	×
[51]	✓	Time padding	✓	×
[52]	✓	Resource isolation	✓	×

2.4 General Trends and Challenges

Embedded SoCs are prevailing in many application domains such as automotive, robotics, and the Internet of Things (IoT), and are expected to have increasing penetration in several domains. Considering the widespread adoption of embedded SoCs and their security concerns related to physical and side-channel attacks, we anticipate the following upcoming trends and challenges.

2.4.1 Increasing Physical and Side-Channel Attacks and Defence Mechanisms

The development of defence mechanisms to guard the ecosystem of devices equipped with embedded SoCs against physical and side-channel attacks is a challenging task. In addition to the emergence of new attack types, attackers are also becoming increasingly sophisticated in their techniques, making it even more difficult to identify and defend against these attacks. One of the primary challenges in developing defence mechanisms is the need to consider several aspects of

information leakage. Physical and side-channel attacks can exploit various types of leakage, such as power consumption, electromagnetic radiation, and timing information. Therefore, defence mechanisms need to be developed that can counter different types of information leakage effectively. Another challenge in developing defence mechanisms is the need to balance security with performance and cost. Some defence mechanisms may introduce additional overhead that can negatively impact the performance of the system, while others may be too costly to implement on a large scale. Therefore, it is crucial to develop defence mechanisms that strike a balance between security, performance, and cost. Another critical factor to consider when developing defence mechanisms is the need to address the entire ecosystem of devices equipped with embedded SoCs. This includes devices used in various application domains, such as IoT, Industry 4.0, and automotive, each of which may have unique security requirements and constraints. Therefore, defence mechanisms need to be adaptable to different application domains and scalable to protect a large number of devices. Finally, the development of defence mechanisms should involve ongoing research and collaboration between academia, industry, and government agencies. This collaboration will help to ensure that defence mechanisms remain up-to-date and effective against emerging threats and vulnerabilities. The development of defence mechanisms to guard the ecosystem of devices equipped with embedded SoCs against physical and side-channel attacks is a complex and ongoing process that requires a multi-faceted approach. With the increasing penetration of embedded SoCs in various application domains, the need for robust defence mechanisms will only continue to grow in importance.

2.4.2 Joint Consideration of Security, Power and Performance for Defence Mechanisms

A robust defence mechanism can be heavy and power hungry. Therefore, exploration of light-weight mechanisms is expected, as the devices may often have resource and battery constraints. Based on the observations in this chapter, those researchers have now started considering security, power, and performance aspects jointly due to the importance of the three metrics, it is expected that such a joint consideration will be widely done in the future. Since there is always a trade-off amongst these metrics and the design space can be huge, it is challenging to find the best design point that can satisfy the security, power, and performance requirements.

2.4.3 Hardware- and Software-based Security

As started earlier in Section 2.2, security solutions for physical and side-channel attacks are mainly based on monitoring SoC properties and then analyzing them to perform an attack. The analysis is typically implemented in the form of software tools. With the increasing use of embedded SoCs in various application domains, the enhancement on the software tooling side is expected to continue. In addition, it is expected to have hardware-based solutions as well, though these are costly. These observations indicate that the development of hardware- and software-based defence mechanisms are going to continue hand-in-hand. Further, based on the anticipated increase in attacks in future and current advances in defence solutions, we anticipate that security solutions will jointly consider software and hardware in order to achieve the best protection mechanism at minimal cost. This requires addressing the challenges of identifying the defence components to be run on the hardware and software. It may also bring the notion of adaptation of the components over the hardware and software depending upon different scenarios. This will require addressing the challenges of identifying adaptation instances and performing them efficiently.

2.4.4 Attacks and Defences for AI-based Solutions

The widespread adoption of AI algorithms will enable their deployment in embedded SoCs. Therefore, it is expected to have new and/or enhanced physical and side-channel attacks focused on AI algorithms [56]. With the increase in attack mechanisms, the development of novel defence mechanisms is expected while addressing the involved unforeseen challenges.

In this chapter, we have examined the different security techniques in embedded systems. These techniques based on different approaches such as performance, and power consumption of embedded SoCs were explored. It is obvious that most of the existing works have tried to implement a good countermeasure to fix the possible risk of attacks regardless of the performance or power consumption of embedded SoCs. Many researchers have considered only performance, or power. Several recent research activities consider both power and performance.

2.5 Summary

In Chapter 2, we explore security techniques for embedded systems, focusing on approaches that consider performance, power consumption, and side-channel analysis. A notable observation from existing research is the emphasis on implementing countermeasures against potential attacks, often at the expense of performance and power efficiency. Although some studies have addressed performance or power consumption individually, there has been a growing effort in recent years to tackle both aspects concurrently. However, it is important to acknowledge that achieving a balance between security, performance, and power consumption presents a significant challenge. Optimizing for these three objectives simultaneously is a complex endeavor, with few researchers managing to find optimal solutions in this multi-objective optimization scenario. Furthermore, this chapter provides an updated examination of side-channel attacks, emphasizing emerging trends and challenges. We pay particular attention to thermal and timing covert channels, which are alarming due to the minimal equipment or specialized environments needed for exploitation. Such vulnerabilities pose a substantial risk, as attackers could potentially extract sensitive data or information with relative ease. To bridge this critical gap, forthcoming chapters will delve deeper into the challenges posed by these covert channels. We will propose effective defense strategies to counteract them, thereby contributing to the broader research landscape presented in this dissertation. This discussion aims not only to highlight the complexities of integrating security measures into embedded systems but also to push the frontier of current research by offering novel solutions to these pervasive challenges.

Chapter 3

Selective Noise Based Countermeasure Against Thermal Covert Channel Attacks

3.1 Overview

As we mentioned earlier, covert channels are a type of communication channel used to transmit information in a manner that is not intended or authorized by the system security policy. In chip-level security, covert channels refer to attacks that exploit vulnerabilities in a system to create a means of communication between processes that are not allowed to communicate. These covert channels can allow two parties to exchange information and data over a shared network without detection, which can be very damaging if used for malicious purposes. The difficulty in detecting covert channels lies in the fact that they can often use seemingly innocuous communication media, such as timing, heat, or indistinct sounds. In a multi-core system, many potential side channels can be exploited as covert channels, but those that use heat as a means of transmitting information can be particularly dangerous. This is because heat is a byproduct of the computing process and is often difficult to detect or monitor. As a result, attackers can use covert heat channels to communicate sensitive information without detection. Overall, covert channels pose a significant

threat to system security and can be difficult to detect and prevent. As technology continues to advance, it will be important to stay vigilant in identifying and mitigating potential covert channel vulnerabilities. The heat transfers are known as thermal covert channels (TCC) [74]. Thermal covert channels can be traced in multi-core systems. As in any communication system, a thermal covert channel includes a pair of a transmitter and a receiver [65]. On the transmitter side, the temperature signals are generated from sensitive data such as user passwords by manipulating other activities like power consumption. While the receiver's site, which is on the other end of data transmission, reads its thermal sensor and recovers the original sensitive information or data transmitted [62]. Figure 3.1 illustrates an eight-core chip example, which assumes that there is a covert channel between core A and core B. It should be noted that core A is placed in a secured zone, where sensitive information does not have permission to be shared with the other cores outside this zone, and core B is located in a non-secured zone [66]. The temperature signal can move around the chip by heat transfer among processor cores until it reaches the destination (receiver) [63-64]. In such cases, the bit '1' shows a range of rising and full of the temperature signal, while a bit '0' shows no changes. While thermal signals are committed to transferring temperature signals over a chip, it is highly likely to be degraded or disrupted by environmental temperature variations or thermal noise [63]. An advanced countermeasure against the thermal covert channel needs to position of affected cores and fight TCC attacks.

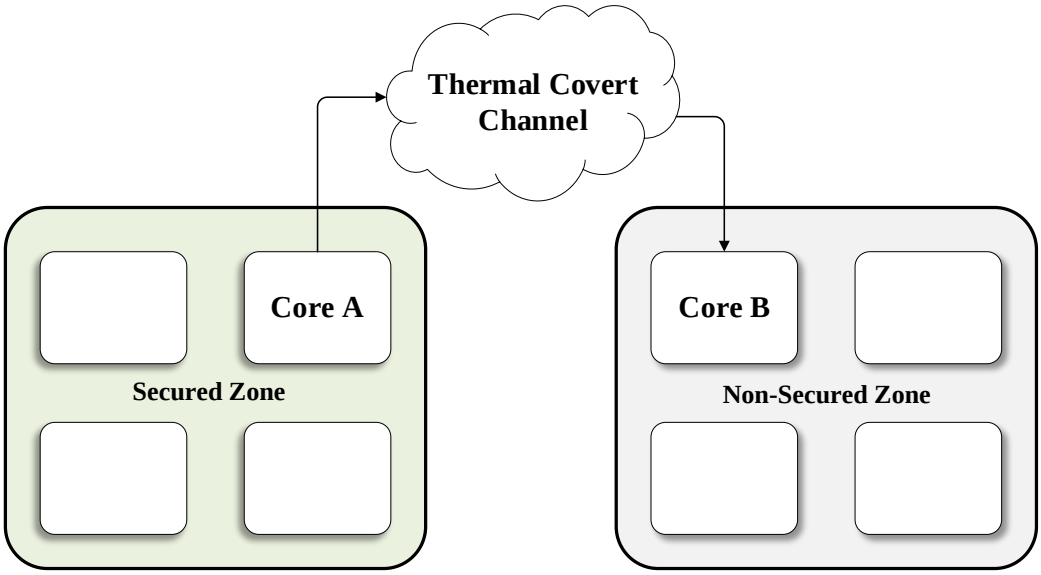

Figure 3.1 TCC communication in a multi-core system. The arrow from core A to core B shows that the heat flow from core A to core B.

In this chapter by considering the previous work gaps, the following novel contributions are proposed:

- A defence strategy for defending against thermal covert channel attacks by changing thermal signal patterns.
- Improve the suggested strategy by considering the power consumption of the system by monitoring the temperature of each bit of transmission.
- Extensive evaluation and comparison against state-of-the-art, demonstrating advantages of our strategy that can protect the integrity and confidentiality of our data as covert channel attacks become more pervasive and insidious.

This chapter has been written upon findings presented and published in "Selective noise-based power-efficient and effective countermeasure against thermal covert channel attacks in multi-core systems" [113].

3.2 Selective Noise Based Countermeasure Methodology

The increase in the operating temperature of an embedded system results in bad experience and reliability problems. Reliability issues through system design are being extensively examined. It seems that in today's digital world, TCC attacks are posing an increasingly big security threat to multi-core/many-core systems, however, there has been little work done to improve the defence strategy or countermeasures against them.

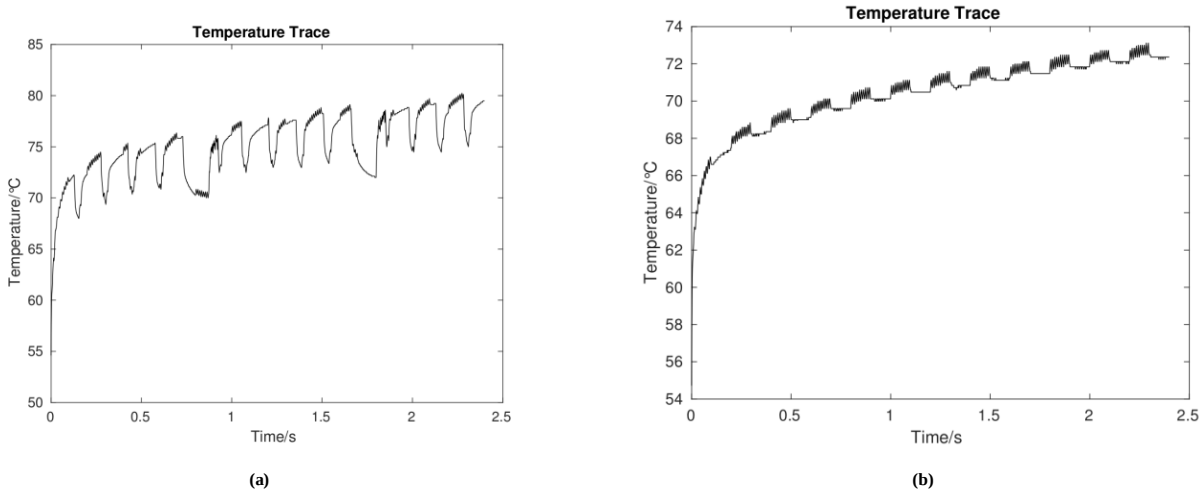

Figure 3.2 Figure (a) and (b) are temperature traces recorded by the receiver core without and with DVFS control, respectively.

In this chapter, we focus on thermal covert channel attacks and propose a novel selective noise-based countermeasure. We have aimed to consider the weakness of previous works and propose a new countermeasure in a way that is undetectable by the attackers. For instance, Huang et al. [62-63] introduced a detection based on signal frequency scanning, positioning affected cores, and blocking to thwart possible TCC attacks, as shown in Figure 3.2. All experiments are performed using a simulator, Sniper-v7.2 [82], and more details of the simulator are provided in Section 5. However, it seems that the temperature signal is still at risk because the attackers can come up with a new method and find the thermal signal's pattern. Jiachen W et al. [67] suggested a countermeasure strategy based on scanning the frequency spectrum rapidly to detect any possible attack. When a thermal covert channel is detected and its frequency of transmission is found, a strong noise source is applied to the transmission frequency band for targeted jamming. These examples [62-63-67] have motivated us to propose a new countermeasure scheme based on recording temperature and adding extra noise (extra thread), in fact, selective noise, that changes the pattern of the thermal signal. The purpose of adding a selective noise is to generate more temperature when the bit is '1' which results in increasing the duration of bit '1'. For instance, if the bit was '1' just for 2ms, by the proposed method we increased it to 5ms and modified the behaviour of the temperature signal. To address this problem, we present a new method that can improve the Bit Error Rate and performance as well. In our approaches, we have changed the pattern of the temperature signals by adding extra noise during the transmission process. In this

method, the attackers cannot identify or guess the temperature signal pattern, so the sensitive information cannot leak easily (Figure 3.3).

In this work, the main focus is on the temperature, power consumption, and BER of the thermal covert channel in the multi-core system. The term covert channel is explored when the source and the sink transfer the data or information actively, as against the term side-channel, utilized, where attackers recognize an unaware system to deduce sensitive data and information e.g., user password or a cryptographic key and change them into the temperature signals [75]. Researching the security issues related to isolation and separation in computing systems is a well-defined area of study.

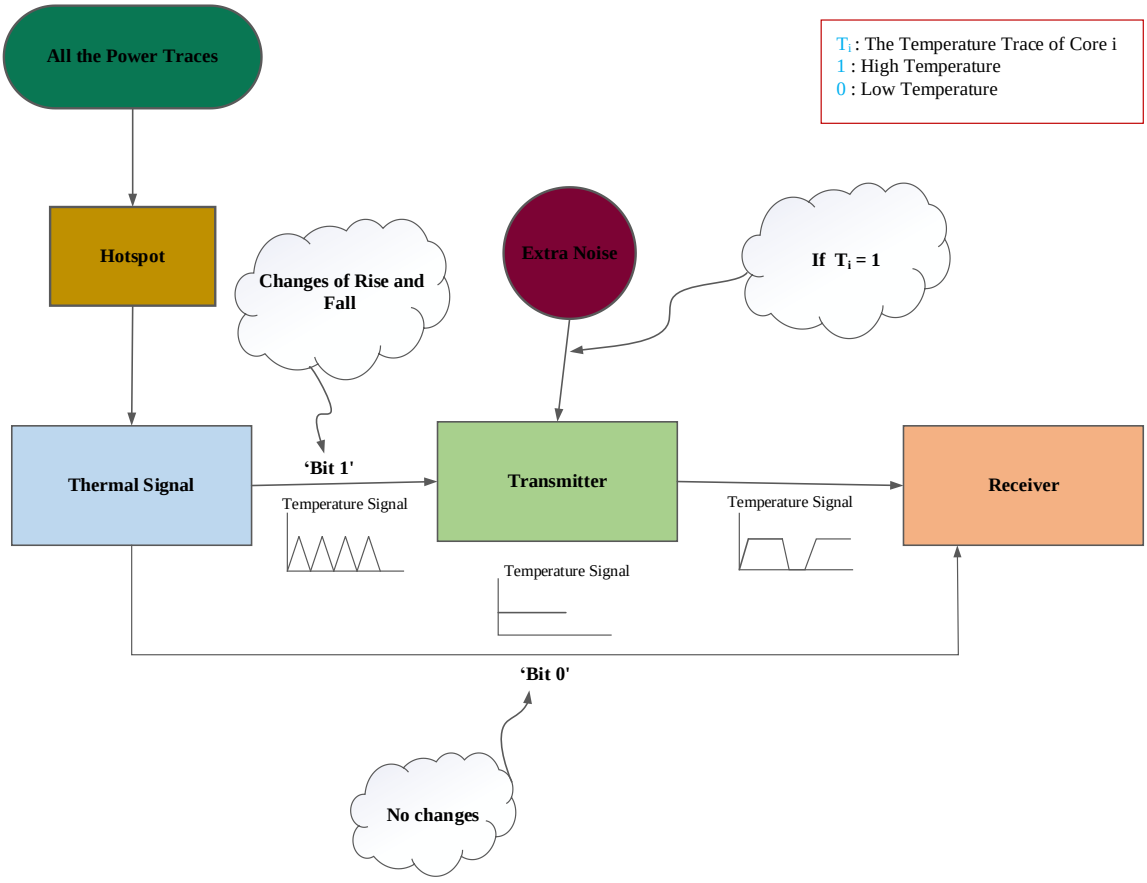

Figure 3.3 Process of extra noise countermeasure.

As with any countermeasure strategy, the TCC countermeasures include two main steps, detection, and defence, which are summarized below:

- ***Detection:*** Detecting a thermal covert channel can be challenging, as it often involves monitoring small temperature changes that may be difficult to detect. One approach to detecting a thermal covert channel is to use a thermal imaging camera to monitor the temperature of the shared resource over time. This can help identify any patterns or fluctuations in temperature that may indicate the presence of a covert channel. However, this approach can be expensive and may not be practical in all situations [67].

Another approach is to use software tools that monitor the behaviour of the shared resource, such as the CPU or hard drive, and look for anomalies that may indicate the presence of a covert channel. For example, if an application is using more CPU resources than expected or accessing the hard drive more frequently than usual, it may be a sign that a covert channel is in use. In addition, it may be possible to prevent the use of thermal covert channels by implementing security measures such as isolation and access control. By limiting access to shared resources and monitoring their usage, it may be possible to prevent unauthorized parties from using thermal covert channels to transfer information [63]. In any countermeasure strategy, the first and most important step is detection.

- ***Defence*****:** Defending against thermal covert channels involves taking steps to prevent or detect their use. Overall, the thermal covert is not similar to many types of covert channels. It does not rely on any shared sources such as catch or memory, which supports it easily circumventing the system's defence. TCC using a simple on-off keying line coding strategy to encode bit '0' and bit '1' is presented in [65]. There are different techniques suggested to fight this type of attack. However, in all of these strategies, the sensitive data is still to be in danger, for instance in [65] BER is very low, in [6] devices may face overheating and high-power consumptions, and in [62-63] thieves can have access to the thermal signal's pattern. By considering these issues, we proposed a novel countermeasure to address them.

3.3 System and Threat Model

In this section, we present the thermal covert channel architecture and explain the challenges linked with TCC attacks. Then, we show our threat model considered for all the implemented attacks. Finally, we describe the methodology used in the TCC attack.

3.3.1 TCC Communication Architecture

Only limited works have focused on the thermal covert channel due to the complexity of the attacks (e.g., requiring expert knowledge and skills) and noisy traces [85]. A thermal covert channel relies on a pair of a transmitter and a receiver, which is shown in Figure 3.4. To start the transmission process, first, the receiver and the transmitter are required to achieve agreement on the transmission frequency for the thermal covert channel. Then, the transmitter reads the sensitive information and data and packetizes them with a so-called Error Correcting Code (ECC) [64]. When the packet is created, it requires to be changed to the temperature signal for transmission. Basically, the transmitter sends the thermal signal by controlling the power consumption of the transmitter core. It means that the transmitter runs computation-intensive code with a particular purpose to generate a high temperature, and it inserts idle CPU cycles to cool down the chip and generate a low temperature [62-63]. For instance, a program is used to change the temperature based on the bit, e.g., if ‘0’ then no change (inserting idle cycles to cool down the chip), and if ‘1’ then the temperature made high followed by a low to keep a low average temperature. On the receiver side, it gets the thermal signal from its local temperature sensor and transforms it into binary bitstreams. Then the receiver checks the ECC code for the integrity of the packet. If the received packet is recognized and contains the particular preamble field (e.g., 101010) throughout the transmission process, the receiver extracts the data fields from the packet; otherwise, the receiver rejects the binary bitstreams [63].

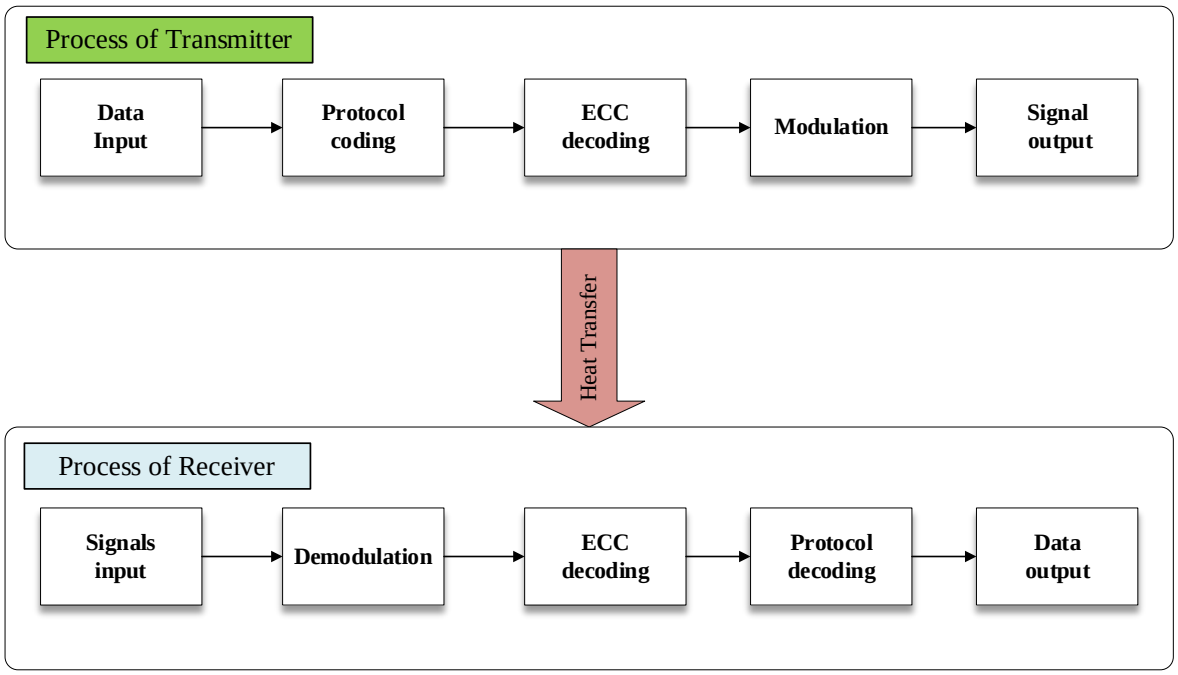

Figure 3.4 The components and signal flow of a communication system through a thermal covert channel: from the transmitter to the receiver [48].

3.3.2 Challenges of Thermal Covert Channel Attacks

The first thermal leakage issue is that they behave integrative, for instance, temperature accumulates when an operation is executed (e.g., Rivest–Shamir–Adleman (RSA) decryption). From a previous operation, the contribution of thermal leakage remains in its next operation. Therefore, it is probable that at a different time, a moment of the same operation (e.g., a multiplication) will have a different starting temperature. Therefore, it is highly likely that the same process (e.g., multiplication) at different time moments shows a different starting temperature [85]. To address this issue, there are two possible approaches. In the first method, pauses can be implemented between the operations. This can be performed by regularly stopping the clock or by implementing pauses after each process, for instance, regularly forcing a delay in the processor. The second solution for this issue is to periodically cool the processor after each operation. This approach can be implemented by adding an external cooling system, such as a high-speed fan [86]. Another issue that thermal traces face is the temperature difference offset with time. While the power is controlled by a voltage regulator, the temperature of the processor and the environment are not directly controlled. So, the offset changes multiple times during

implementation. Today, most systems and devices use dynamic clocks and voltage scaling to maintain the temperature in a specific safe range. Hence, the thermal leakage also shows drifts in the offset during the implementation processes. So, TCC analysis needs a mechanism to filter out such drifts, as they highly likely cannot work on traces with different offsets [87].

The last issue is one method of modeling the physical performance of the thermal leakage by analyzing the system as an RC network [94]. This network acts similar to a low-pass filter with a cut-off frequency somewhere in the kilohertz range. This frequency reaction can pose an issue since a wide range of systems and computers run in the Gigahertz range. With a low-pass filter, it seems it is very hard and difficult to measure any useful information and data when the systems are running at 800 MHz, even if it is the first order. Luckily, it is not needed to record every clock cycle for side-channel analyses can be differentiated. If these processes take long enough, differences can be visible in the thermal traces [85].

3.3.3 Threat Model

In thermal attacks, a malicious operation leaks data and secret information by modulating the devices' temperature [92]. Expect very few studies as reported in [94] which estimated the decay rate of the DRAM cells to determine the temperature of a TCC, most of the thermal covert channels need to have access to on-chip digital thermal sensors (DTS) to get the temperature information of the chip [93]. For dynamic thermal management, temperature sensors are essential and are deployed in chips in a wide range [98]. The accuracy and number of temperature sensors can be improved in the future [99]. The attackers (being the TCC transceiver) are able to read or reach the local thermal sensor data by calling the software interfaces such as MSR or Intel's processor Model Specific Register, with a normal resolution of 1 °C [100]. The accuracy of some latest digital thermal sensors can be up to 0.1 °C. The defender is able to read the temperatures of all cores and distribute the detection and blocking programs to each core [62].

Our threat model for TCC as the following assumptions:

- The attackers have direct access to the target device or systems in order to record and archive thermal traces of the executed decryption. Many key recovery attacks use a common set of methods or techniques for analyzing thermal traces. Simple thermal analysis (STA) includes directly inspecting traces to deduce sensitive data or information when measurements can be recorded to certain data properties [63].

The attackers can acquire the ciphertext. In the transmission process, the ciphertext data will be employed as default storage parameters for the specific transaction. So, during this process, it is most likely the attackers can access the ciphertext [68].

- The attackers have access to a similar system or device. For example, using off-the-shelf parts such as AVR, ARM, etc.
- The attackers have the capability to slow and cool down the target operating systems. This assumption can be reached by manipulating the external crystal, using some external fan, or by forcing the target system to compute a wide range of tasks in parallel [62-85].
- Since the signal on the transmitter side is already distorted by adding noise, it wouldn't be possible to detect it correctly at the receiver side by increasing the frequency of thermal sensors because the information received would be different from the original one. Further, for proper synchronization of data, both transmitter and receiver thermal sensors should operate at the same frequency. The transmitter and the receiver also need to reach an agreement on the frequency of transmission for the thermal covert channel [62].

3.4 Countermeasures Against Thermal Covert Channel Attacks

Our proposed work has different steps to detect and fight against such thermal covert channel attacks. For the detection step, we used frequency scanning-based detection, which was presented by Huang et al. [62-63].

- **Step 1 (Detection):** The detection algorithm uses the module of bandpass filtering to filter out the thermal signals that are out of every band of interest. The detection algorithm is also designed in a way that it can manage TCC whose receiver and transmitter through two different channels. While designing the countermeasure method, we consider that the receiver and the transmitter have access to the same thermal data files, and also the transmitter and the receiver threads are both sited in the same physical core and share the same thermal sensors, known as Intra-core channel [62-62]. The frequency scanning process is used to detect any existing TCC channels. In addition, each core specifies the signal's maximum amplitude from the frequency scanning process and leads it into a module known as the decision-making module. The decision-making module compares the

signal's amplitude against a threshold (the pre-set threshold for the signal amplitude in a detection cycle.) and decides whether a signal (e.g., 1 0 1 0) is from a covert channel or not. If the amplitude of the signal is higher than, an attack is present; otherwise, there is no attack happening. Statistically, the threshold is selected experimentally to be 0.02 dB [63]. Based on frequency scanning, if the defender, denoted as, the global manager, identified the TCC channel exists, the proposed countermeasure is applied.

- **Step 2 (Classify bits):** For applying the proposed countermeasure, we need to classify bits into '1' and '0' according to the thermal signal changes. In our work, the bit '1' represents changes in the thermal signal, while a bit '0' illustrates no changes.
- **Step 3 (Check bits):** For this step, the system checks if the bit is '1' or not by considering the changes in the thermal signal. Then if the bit is '1' the operation will start.
- **Step 4 (Adding extra noise)**: The last step is adding an extra noise to the temperature signal, which results in generating more heat to the system where the bit is '1', therefore, there is no bit change for a long duration, as Figure 3.5 shows.

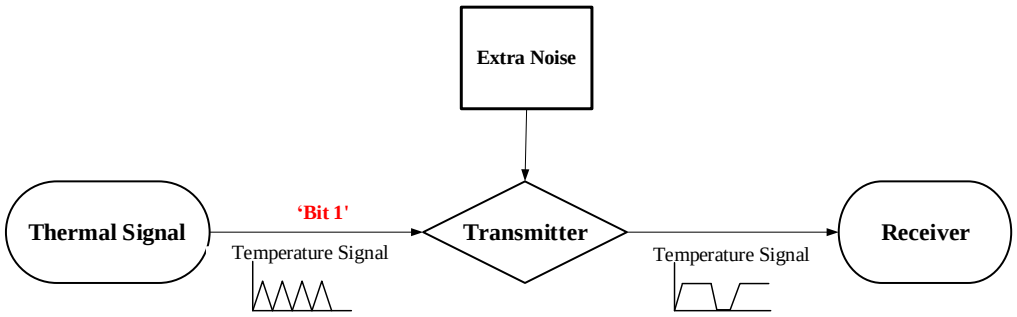

Figure 3.5 Proposed countermeasure process.

In the proposed method, once a core is determined to be associated with a thermal covert channel, a countermeasure shall apply to fight any future thermal covert channel transmission. Here we assume that a thermal covert channel thread is running on a dedicated logical core all over a full communication session.

We add extra noise to the thermal signal to generate more temperature when the thermal signal is at high temperature (*Th*), which results in increased duration of *Th* and changing the pattern of the thermal signal and making it secure.

Note that the global manager cannot immediately apply countermeasures to control the suspicious core. First, the global manager scales down the temperature of the suspicious core for a few

seconds (e.g., 0.5s) to decrease the risk, and then a countermeasure is implemented to block any future TCC transmission [62-63]. As for the simulation results, from Figure 3.6, one can see that the average temperature of the thermal signal without countermeasure is around 72 °C (Figure 3.6 (a)), and when the countermeasure is applied the average temperature increases to 76 °C (Figure 3.6 (b)).

3.4.1 Extra Noise Generation by Random Numbers

There are different methods introduced to generate random numbers. Some of them are very simple and some of them are very complicated. However, our purpose in using this operation is to pose some extra temperature in a very simple method that does not require special types of equipment or lab to implement, and all researchers and designers can have access to it very easily [95-96]. Therefore, for implementing our idea, we used C/C++ programming and the int "*rolldie"* (int rand number) function to create some random numbers. Algorithm 1 illustrates the process of generating random numbers. In the first step, the system gets two different random numbers (known as X1 and X2) and then sums them to achieve the final results.

Algorithm 1 Psudo Random Number Generator

Input: Two random numbers X1 and X2

Output: Psudo random sequence period of X1+X2

```
1: for n from 1 to 100 do
2:     X1 = rollDie(); // Simulates rolling a die to get a random number for X1
3:     X2 = rollDie(); // Simulates rolling a die to get another random number for X2
4:     m = X1 + X2;    // Calculates the sum of X1 and X2
5:     print m;        // Outputs the current sum to the pseudo-random sequence6:
6:     return
7: end for
```

We started our experiment with 20 random numbers, however, it did not have any effect on our system, as the duration of this process was too short and did not generate significant temperature. Hence, we increased these numbers until achieved the best results. The best result was reached when 100 random numbers were generated, and we got high BER and low power consumption. We considered this operation (generating random numbers) as one extra noise and then applied it

to the transmission process. Results are shown in Table 3.1.

Table 3.1 One extra noise with different random numbers.

Countermeasures	Generate Random Numbers	Bit Error Rate	Avg Power Consumption(w)
One Extra Noise	20	22%	22.50
	50	22%	22.80
	100	94%	23.74
	150	94%	23.96

During our work, we considered two important metrics, bit error rate, and power consumption. In our proposed countermeasure, it can be difficult for the attackers to identify or guess the temperature signal pattern, so the sensitive information cannot leak easily. Consider a system that has a thermal covert channel with a transmission frequency of 100Hz. Figure 3.6 shows the temperature signal without and with the proposed countermeasure. As Figure 3.6 (b) illustrates, when the proposed countermeasure is applied, it shows a significantly different thermal profile. In our proposed countermeasure, the BER of the TCC reaches 94%, and the power consumption did not increase significantly. A high BER indicates that our proposed countermeasure can be utilized to fight TCC attacks. On the other hand, system overheating leads to increased power consumption throughout the transmission process. Consequently, this issue is addressed in this chapter.

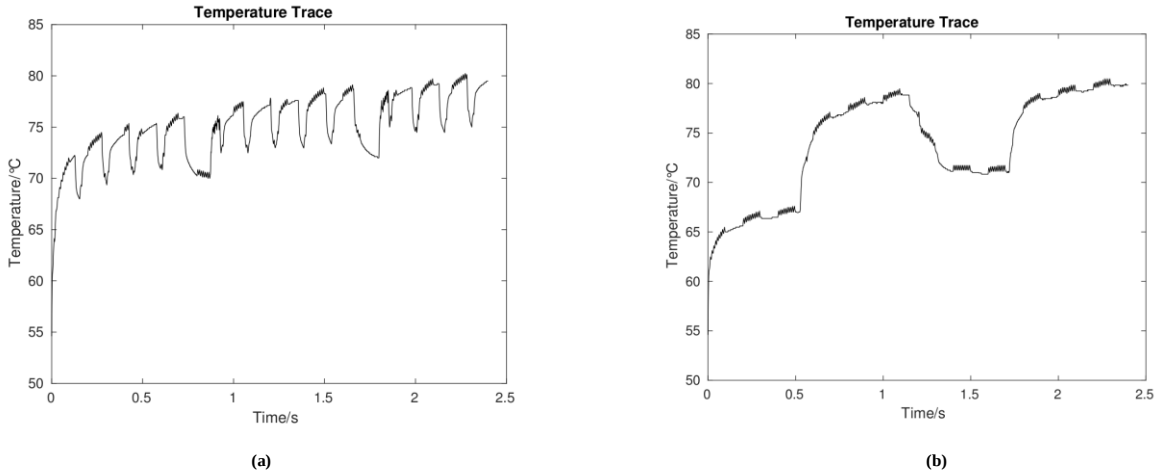

Figure 3.6 Temperature timing diagrams of the receiver core without (a) and with proposed countermeasure on the transmitter core (b).

3.4.2 Selective Noise based Countermeasure for TCC

To fight TCC attacks and address the system's overheating issue, we proposed a new countermeasure, which essentially consists of selective noise. In other words, this operation is designed based on the recorded current temperature of bit '1'. If the current temperature shows any changes or fluctuation, then the system decides whether to add extra noise or not, which is known as selective noise. We will discuss it in more detail in the next section. By adding selective noise, we could first change the pattern of the thermal signal, and then avoid increasing the unnecessary temperature, which results in high power consumption. The methodology is illustrated in Figure 3.7 and has the following main steps:

- **Step 1 (Record temperature):** This step is monitoring and recording temperature. After the extra noise is added when the bit is '1'. Then the current temperature is recorded by the temperature sensor and used for the next step.
- **Step 2 (Temperature monitoring of processors and adding selective noise):** In the last step, the next bit's temperature is compared with the recorded temperature (*TR*) from the previous step. If the temperature of the current bit is less than *TR* or starting to decrease, the selective noise is added to keep the temperature at the same level, otherwise we move to the next bit. This progress continues until the end of the transmission process.

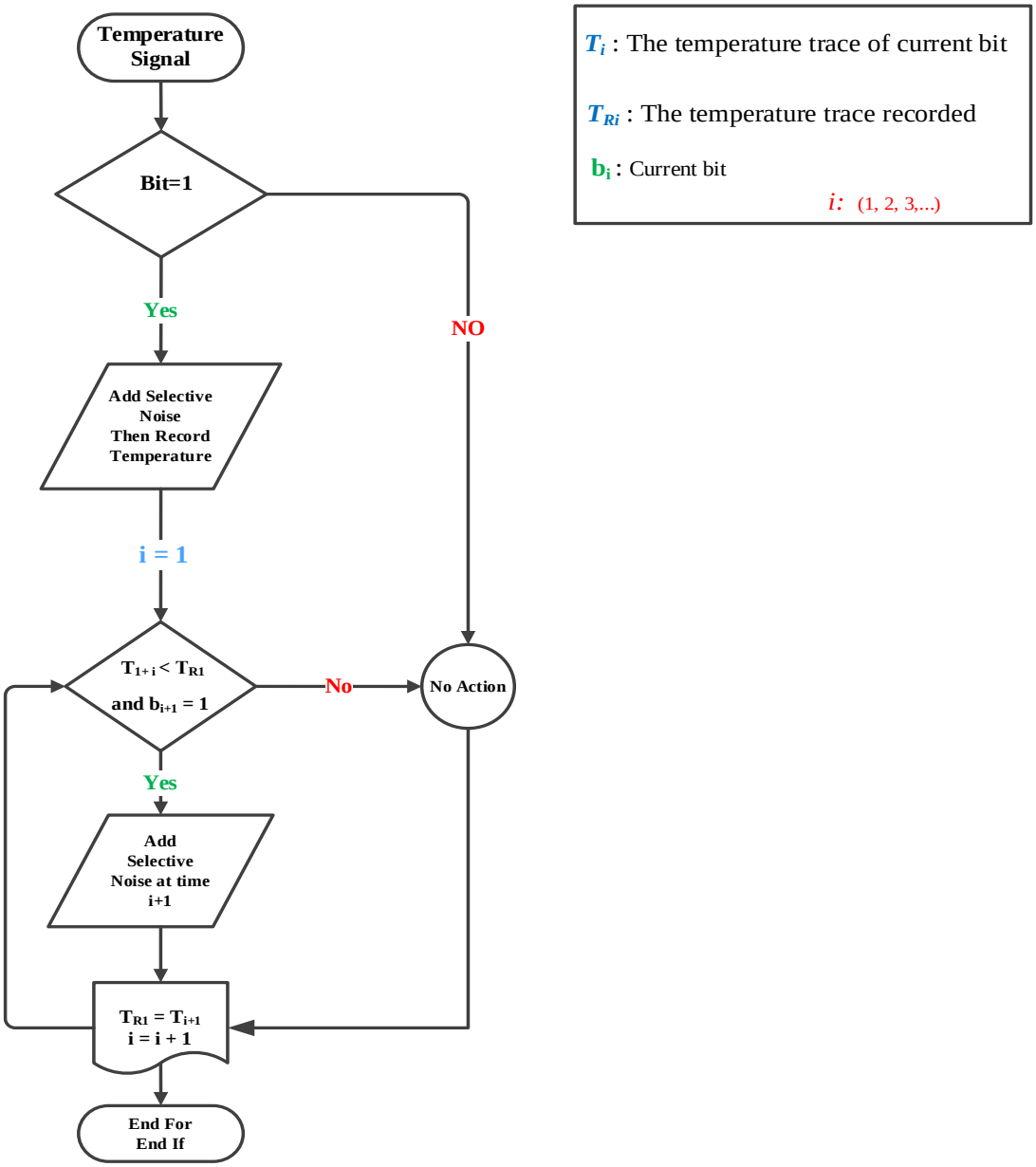

Figure 3.7 The process of adding selective noise.

Here, we adopt different countermeasure strategies that take into account both security requirements and power consumption. Specifically, adding selective noise effectively can change the pattern of the temperature signal. Compared with our previous method, the BER of the receiver does not show significant changes (94%) but the power consumption will not be extremely high. As Figure 3.8 shows by adding selective noise, the thermal signal shows different behaviour. The temperature fluctuation decreases, and we could keep it at the same level, which results in, avoiding system overheating and low power consumption. When the transmission process is

complete, at the receiver side, the receiver reads the temperature sensor and records the thermal signal then decodes the signal to recover the original data. Next, if the received packet is specified uncompromised and includes a special preamble field during the transmission, the receiver extracts the data fields from the packet, otherwise, the receiver drops the binary bitstreams and waits for retransmission [63].

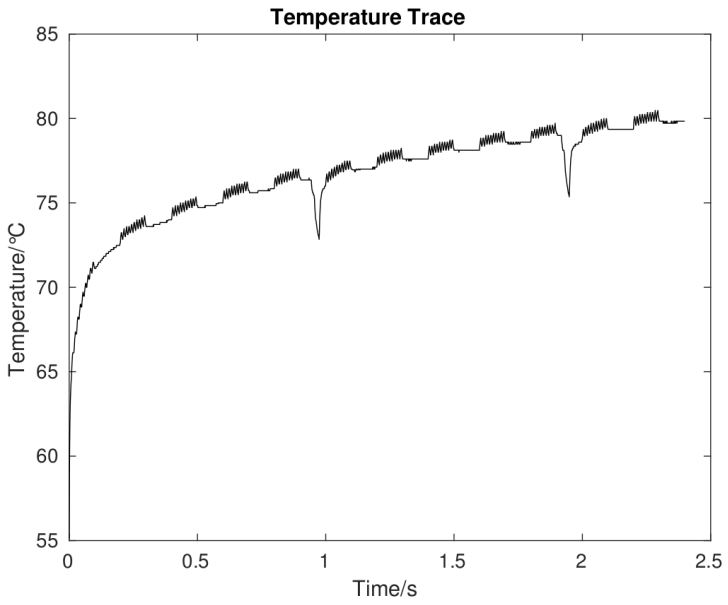

Figure 3.8 Outcome of the proposed selective noise-based countermeasure.

3.5 Experimental Evaluation

Two sets of experiments have been performed, first, adding extra noise to every bit ‘1’ and adding selective noise. Mostly, these experiments are implemented by using a multi-core simulator, Sniper-v7.2 [82] as a fast and reliable simulator for multi-core [83]. McPAT-v1.0 integrated as the power model. McPAT is an integrated power, area, and timing modeling framework for multithreaded, and many-core/multi-core architectures [84]. This analysis has been done in order to thwart the covert thermal channel attack of our target device. To covertly transmit sensitive information from a secured zone to a non-secured one, the thermal covert channel programs require generating and measuring temperature signals by using the HotSpot simulator.

We chose Hotspot-v6.0 thermal model to dynamically produce temperatures for all the cores. We adopt a few benchmarks from PARSEC and SPLASH-2 and their temperature signals will be treated as the thermal noise to a thermal covert channel. The experimental setup uses Sniper and Hotspot simulation tools and TCC programs. Their configuration details are now tabulated in Table 3.2. The floorplan of the processor cores follows the one reported in [62-63].

Table 3.2 Configuration Details

Sniper Configuration	
Instruction set architecture	x86-64.
Operate System	Ubuntu 16.04.5 LTS
Number of cores	4×4, 8×8
Number of SMT threads per core	2
Frequency of CPUs (MHz)	2000
Benckmarks of PARSEC	Blackscholes, Canneal, Fluidanimate, Streamcluster, Swaptions, X-264, Dedup, Freqmine.
Benckmarks of SPLASH-2	Raytrace, Barnes.
Chip thickness	0.15mm.
Silicon thermal conductivity	100W/(m · K)
Silicon specific heat capacity	1.75 × 106J/(m3 · K)
Heat sink side	0.06m.
Heat sink thickness	6.9mm.
Heat sinks thermal conductivity	400W/(m · K)
Specific heat capacity of heat sink	3.55 × 106J/(m3 · K)
Transmission frequency	10Hz, 20Hz, 50Hz, 80Hz, 100 Hz, 150 Hz, etc.
Preamble of a packet	1010101.
Packet size in bits	64.
ECC method	Hamming code.

Table 3.2 shows the transmission frequency of the TCC dynamically changes with a range of 10Hz, 50Hz, 150Hz, and 200Hz, etc., and the working CPU frequency of the transmitter/receiver core is

2000MHz. In addition, it is identified whether there is a TCC attack or not. If an attack is found, the position of the transmitters/receivers associated with the thermal covert channel is required to be found. This positioning accuracy is referred to as Pacc.

$$P_{acc} = \begin{cases} 1 & P_{detected} = P_{transmitter/receiver} \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

Where:

$P_{detected}$: is the position (core id) of the detected thermal covert channel transmitter/receiver cores.

$P_{transmitter}$: is the actual position of the transmitter/receiver cores.

Table 3.3 shows the average accuracy of positioning transceiver in different multi-core systems, which is around 97%. By using this detection strategy, we can almost always identify a TCC attack.

Table 3.3 Detection accuracy of Pacc.

System Sizes	Pacc
4×4	0.975
8×8	0.97

Table 3.4 illustrates the proposed countermeasure strategy in different systems, as one can see this method efficiently fights TCC attacks. The average BER of the TCC attacks across 8×8 systems can be higher than 94%. An interesting fact is that the average power consumption when the selective noise is in effect does not increase significantly. The formula which is used for computing the average power consumption is the following [22]:

$$P_{avg} = \frac{\sum_{c=1}^{N_c}\left(\sum_{u=1}^{N_c} 2_{cu}\right)}{N_c}[W] \quad (3.2)$$

Where:

Nc: The total number of cores in the multi-core processor.

c: Index representing an individual core (ranges from 1 to *Nc*).

u: Index representing utilization (ranges from 1 to *Nc*).

2cu: A term representing the power consumption associated with the utilization of core u by core c.

Table 3.4 Comparison of our experimental result with different related research.

Countermeasures	Bit Error Rate	Avg Power Consumption(w)
TCC with periodically scanning the frequency and add a strong noise source [67].	85%	22.27
TCC with DVFS [62-63].	92%	22.31
TCC with One Extra Noise	94%	23.74
TCC with Selective Noise	94%	22.81

Figure 3.9 shows a bar chart of BER of the proposed countermeasure method compared to the DVFS in different systems. One can see that the average BER's of our proposed countermeasure is always higher than that of the DVFS countermeasure for different system sizes.

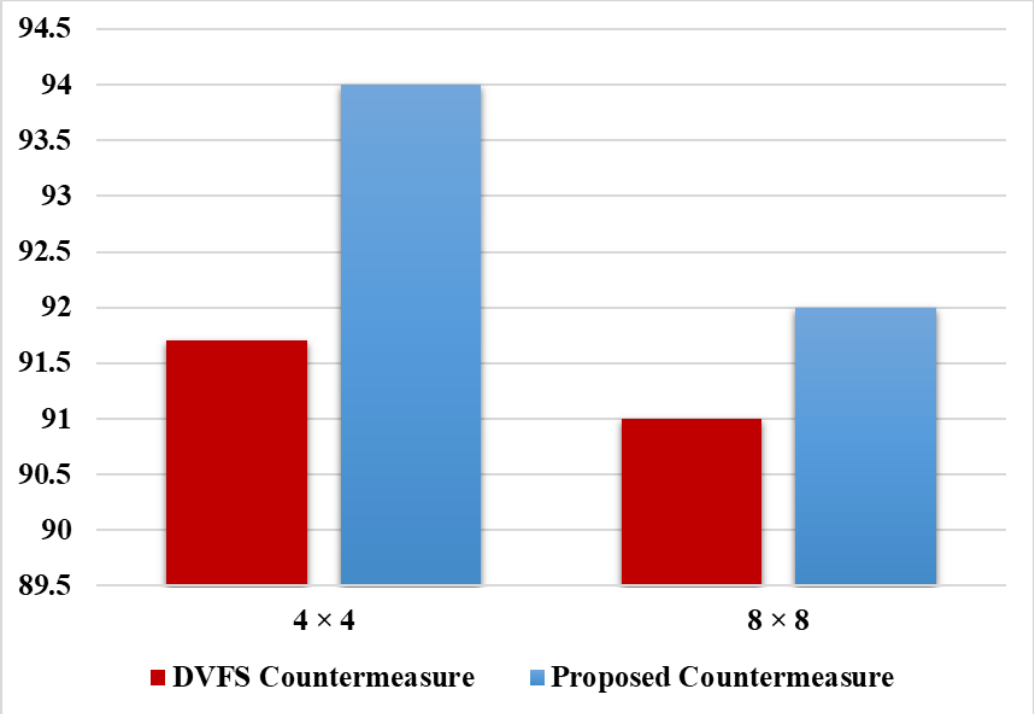

Figure 3.9 BER with the proposed method and DVFS in different systems.

We also determined that the average power consumption increased, along with the number of extra noises. The average and maximum temperatures also follow this trend. As shown in Table 3.5, by increasing the number of noises we are able to reach a high BER, but it may cause a rise in the temperature average power consumption of the systems, which is not good for the systems.

Table 3.5 Experimental results with different numbers of extra noise.

Countermeasures	Bit Error Rate	Avg Power Consumption(w)
Two Extra Noise	95%	25.77204
Three Extra Noise	98%	28.19446
More than three Extra Noise (Five Extra Noise)	98%	34.20743

Table 3.6 indicates different aspects of considerations in TCC attacks. In thermal covert channel attacks, most of the existing methodologies did not consider high BER, low power consumption, and performance along with countermeasures. However, in the proposed work, we examine the thermal covert channel attacks, in multi-core systems with the goals of optimizing two metrics: bit error rate and power consumption.

Table 3.6 Consideration of various aspects in thermal covert channel attacks by existing and our approaches.

References	Countermeasure	High BER	Performance	Low Power Consumption
[1-2]	✓	✓	✓	×
[4]	✓	×	×	×
[6]	✓	×	×	✓
[8]	✓	×	×	×
[10]	✓	×	×	×
[11]	✓	×	×	×
[13]	✓	×	×	×
[19]	✓	×	×	×
Proposed work	✓	✓	×	✓

Experimental results have confirmed that TCC attacks are possible in case of the leakage of sensitive information and data over several seconds or milliseconds. So, it seems that an application must repeatedly check the data and information that thwart such attacks, which may be costly or result in overheating the devices.

This work proposes a novel countermeasure against thermal covert channel attacks, offering several advantages over existing methods:

- **High Bit Error Rate:** By introducing selective noise-based countermeasure, the bit error rate increases significantly, and making successful TCC attacks highly improbable. This surpasses other methods that focus on masking or hiding information, which may still be vulnerable to skilled attackers.
- **Low Power Consumption**: The countermeasure requires minimal additional hardware and operates efficiently, minimizing its impact on power consumption, a crucial aspect for embedded systems. This is in contrast to approaches requiring specialized equipment or complex algorithms, which can lead to increased power usage.
- **Easy Implementation:** Last but not least is easy to implement, unlike some countermeasures that require specific hardware or modifications such as [85-86], and [90], the proposed method can be readily implemented in various devices and systems with minimal adjustments. This broad applicability makes it a valuable tool for enhancing security across diverse platforms.

3.5.1 Limitation and Future Work

While offering significant advantages, the proposed approach has limitations to consider:

- **Pattern Dependence:** The effectiveness of the countermeasure relies on the specific pattern of the thermal signal. When the pattern deviates from the expected format. e.g., 1 1 0 0 1 1 1 1 instead of 1 0 1 0 1 0 1, this strategy cannot work very well. Because, when the temperature of the current bit (e.g., 1 1 1) is higher than the previous bit (e.g., 1 1), the system might struggle to distinguish between legitimate temperature changes and manipulated ones, potentially leaving it vulnerable to attacks.
- **Future Enhancements:** Addressing the pattern dependence and exploring alternative countermeasure strategies for more robust protection against diverse TCC attack scenarios are crucial areas for future research.

3.6 Summary

To the best of our knowledge, this is the first work to have used the pattern of the thermal signal to reduce the risks and dangers that threaten sensitive information and data of the many core/multi-core systems. Our experimental results have confirmed that the novel proposed countermeasure could be practically against TCC attacks with a high BER of 94%. With its low overhead, power consumption, and complexity, the proposed countermeasure is a suitable scheme that can be used by many-core/multi-core systems to fight against such attacks. This study shows that thermal attacks are inevitable, however appropriate countermeasures should be designed to fight them by considering other aspects of the system such as power consumption as an important metric in multi-core systems.

In this chapter a new countermeasure was introduced for thermal covert channel attacks, however, it has some weaknesses. So based on these weaknesses which are mentioned in the previous section, we suggest a new approach which is based on monitoring temperature and fan speed (**Chapter 4**).

Chapter 4

Fan Speed Control Based Defence for Thermal Covert Channel Attacks

4.1 Overview

As we discussed in previous chapters multi-core systems are ubiquitous, driving advancements in both general-purpose computing and embedded applications. Their powerful CPUs and GPUs generate significant heat, which can damage components and negatively impact performance [111]. Proper cooling is crucial for optimal performance and longevity. Active (heat sinks with fans) and passive (heat sinks only) cooling solutions exist, each with advantages depending on factors like system design, usage, and environment [112]. Beyond performance and hardware health, uncontrolled heat management has broader implications:

- Reduced Power Efficiency: Research shows uncontrolled heat affects power consumption, impacting usability and performance [114]. In systems prioritizing low power, excessive temperatures can significantly hinder availability [115].

- Security Threats: Multi-core systems are vulnerable to thermal covert channels, where temperature fluctuations are used to secretly transmit data [112]. This is particularly concerning in systems demanding both low power and high performance, as temperature increases can compromise their availability and reliability.

Our previous proposed countermeasure, "Selective Noise Based Power-Efficient and Effective Countermeasure against Thermal Covert Channel Attacks in Multi-Core Systems", introduces controlled noise (additional heat) to disrupt the thermal signal pattern. This approach achieved a 94% BER while consuming minimal power (22.81 W) [113]. Despite its effectiveness, our approach had limitations. It may not work well if the main thermal signal pattern is different from the assumed pattern (e.g., consecutive high temperatures, Figure 4.1). This is because the countermeasure relies on comparing current and previous temperatures to decide when to apply noise, and system cannot decide to apply the selective-noise countermeasure. Hence, the multi-core system will be at risk. Therefore, we investigated the potential security threat posed by temperature-based covert channels in multi-core systems and proposed a new defence strategy called "Fan Speed Control Based Countermeasure". Our experiment involves monitoring system temperatures and dynamically adjusting fan speeds to maintain appropriate levels, mitigating covert channel attacks. We found that thermal effects are more pronounced at higher operational power and CPU frequencies, highlighting the importance of dynamic cooling strategies.

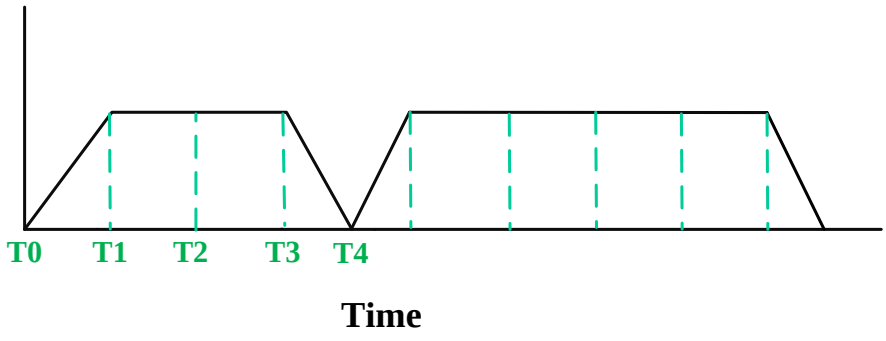

Figure 4.1 Thermal signal with new pattern.

As Figure 4.2 Shows system without any TCC the fan does not work, and the temperature does not show any fluctuations, however, when we add TCC the thermal signal starts to fluctuate followed by the fan starting to work. Therefore, by this strategy and monitoring the behaviour of the temperature of the multi-core system, we will be able to detect any possible TCC attacks. In this chapter, we will discuss this work and try to address it.

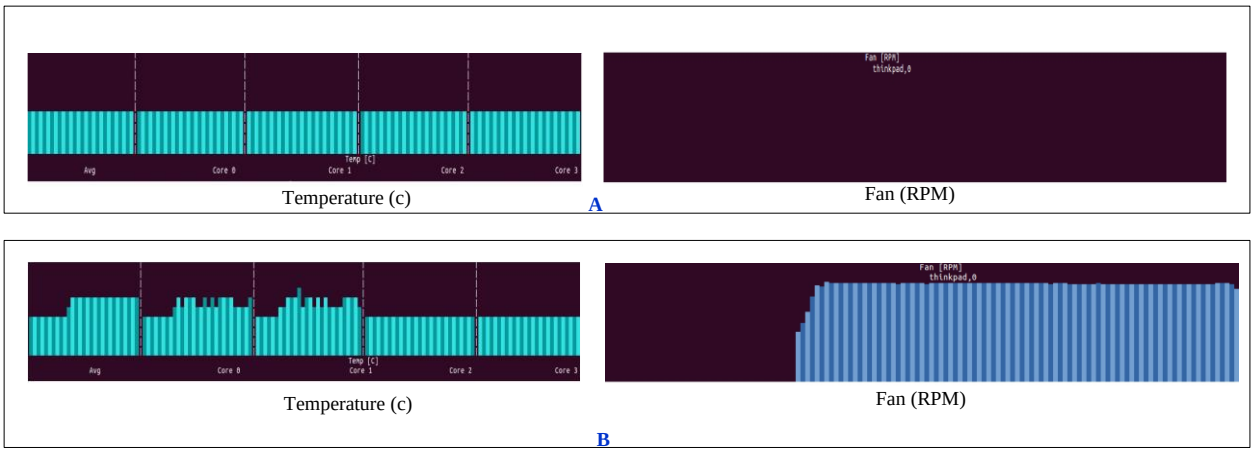

Figure 4.2 System behaviour: (A) System without TCC, (B) System with TCC.

In this chapter by considering the above weakness from previous research, tried to address and make the following novel contributions:

- A defence technique for defending against thermal covert channel attacks.
- A combination of different defence strategies to wipe out thermal covert channel attacks improves BER and increases the power consumption of the system.
- Conduct an extensive evaluation and comparison against state-of-the-art, demonstrating the advantages of our methodology that can protect the integrity and confidentiality of our data since covert channel attacks become more permeative and insidious.

The findings and content of this chapter have been published in "Fan speed control based defence for thermal covert channel attacks" [172].

4.2 Fan Speed Controlling Based Defence Methodology

In this study, we have tried to propose a new methodology to address previous issues. In this technique, we have controlled the fan's speed by monitoring the behaviour of the system's temperature. Figure 4.4 illustrates, the approach has different steps:

- **Step 1 (Set up the Fan speed):** In this step, the fan's speed is set up at the appropriate level. The fan speed is chosen experimentally to be 1,200 RPM when the bit is '1'.
- **Step 2 (Record temperature):** The receiver records the temperature signal by reading its thermal sensor. The sensor recording module reads the temperature information through the system. This step is monitoring and recording temperature after setting up the fan's

speed. Then the current temperature is recorded and used for the next step.

- **Step 3 (Temperature monitoring of processors and adopting an appropriate level for fan speed):** In this step, the next bit's temperature is compared with the recorded temperature (*TR*) from the previous step. If the temperature of the current (*TC*) bit is greater than TR, the speed of the fan begins to increase to 2,000 RPM. To consider this speed, first, we monitored the behaviour of the fan by S-tui which is a free and open-source terminal UI for monitoring a system. *S-tui* allows to monitor of processors' temperature, frequency, power, and utilization in a graphical way from the terminal. It is written in Python and needs root permission to use the *S-tui* (Figure 4.3). Furthermore, we should consider the power consumption of the fan which may affect the total energy consumption. Therefore, in this range of fan speed the power of the fan is 3.6 (w) [111]. Otherwise, if the *TC* is less than *TR* the speed of the fan decreases to 1,000 RPM. If the Tc is equal to *TR* the process will come back to the recording temperature step. This progress continues until the end of the transmission process.

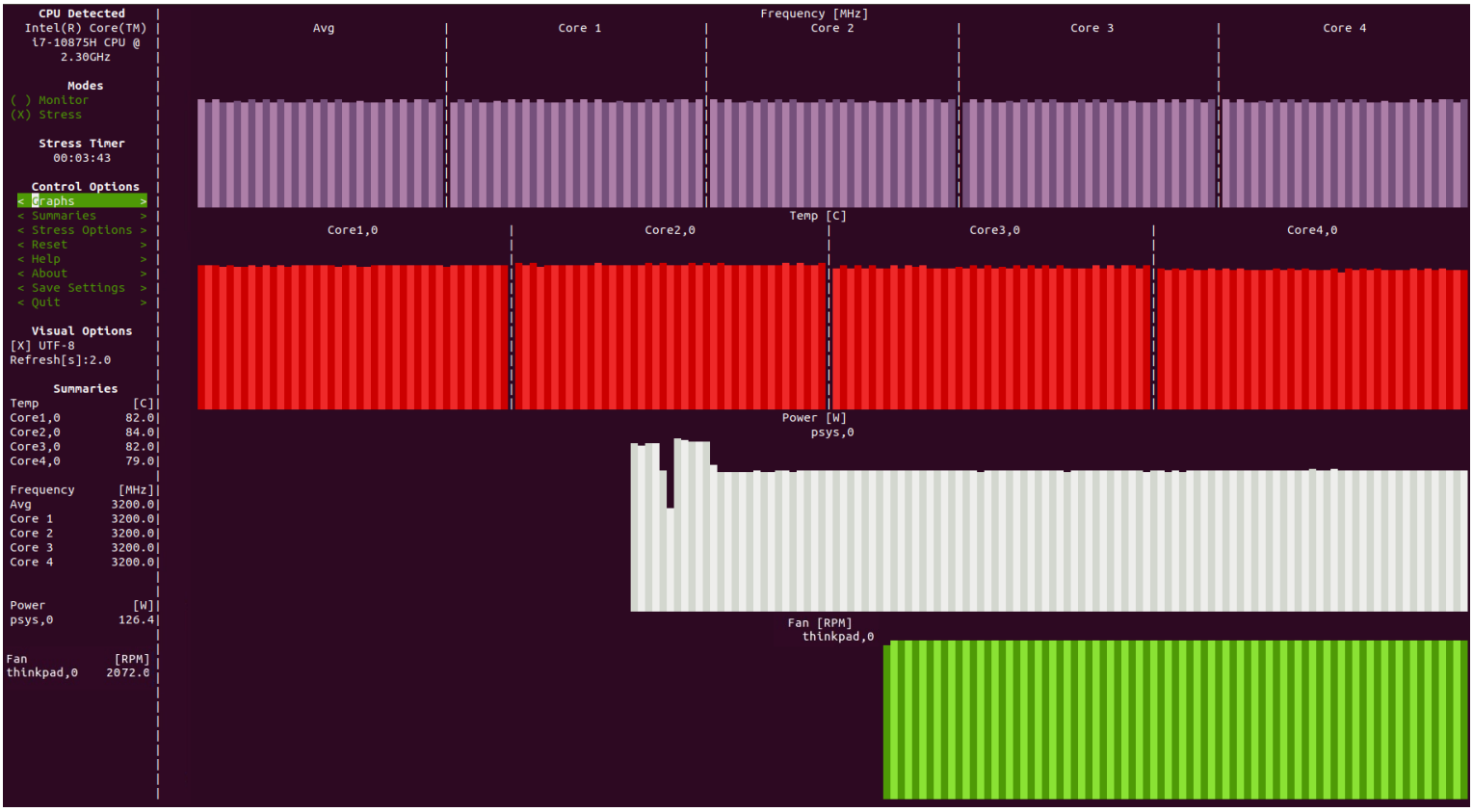

Figure 4.3 Monitoring system behaviour.

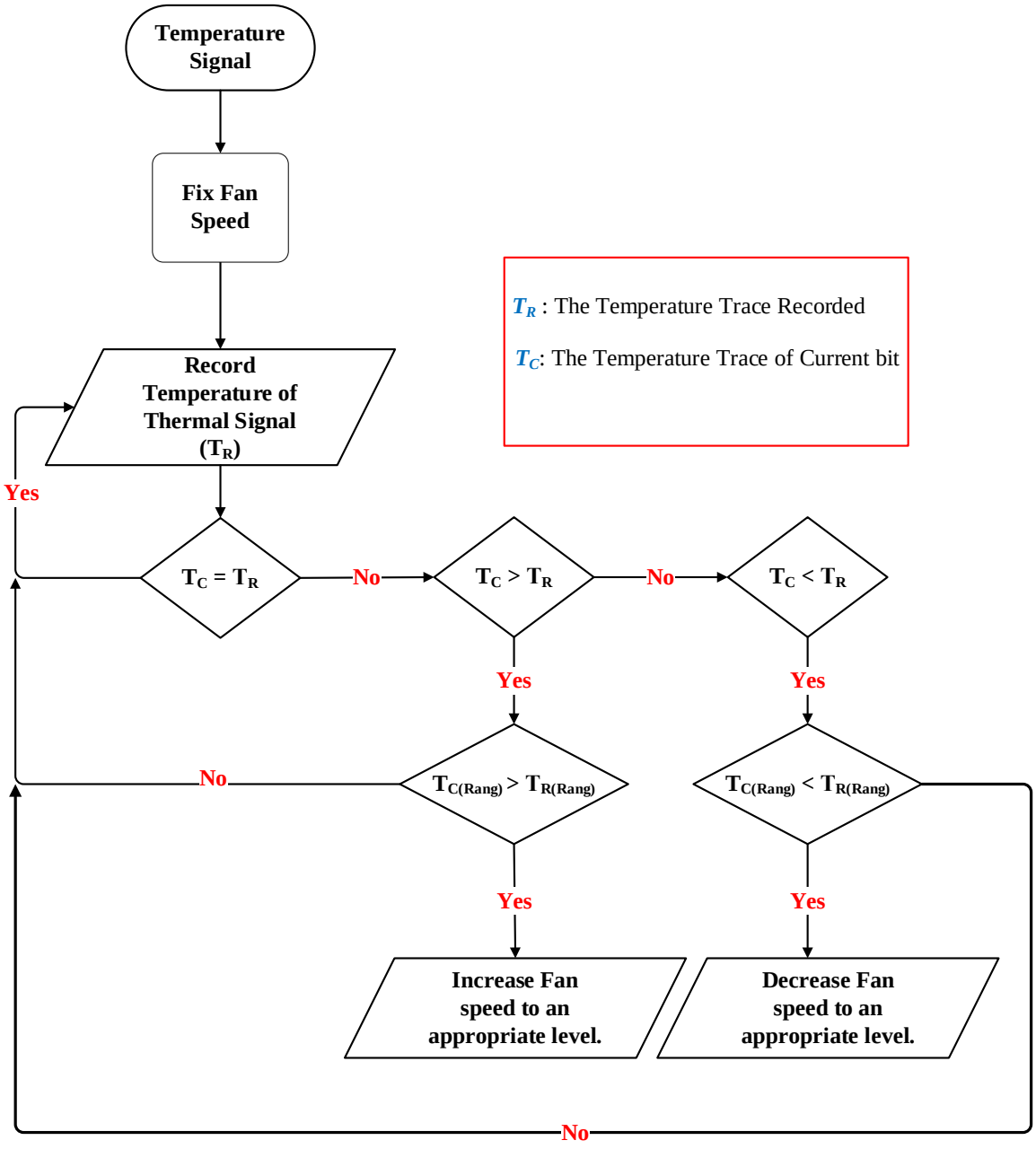

Figure 4.4 The process of proposed work.

Table 4.1 Different states of fan speed (5°C).

Range of Temperature (°C)	Fan's Speed (RPM)
Tc <65	1000
65<Tc<70	1500
70<Tc<75	2000
75<Tc<80	2500
Tc >80	Max

Here, we adopt different countermeasure strategies along with different temperature ranges that take into account both security requirements and power consumption. In this approach to avoid rising power consumption due to the speed of the fan, we consider different levels for different ranges of temperatures (Table 4.1).

We have also considered more range temperature (e. g. Table 4.2) for controlling fan speed to check the behaviour of the system. As Table 4.3 indicates, it results in decreasing the temperature, however average power consumption will increase because the fan needs to work more which affects energy consumption. Increasing the fan speed to maintain a lower temperature will cause the fan to work more, leading to higher energy consumption. This increase in energy consumption can be significant, especially if the system is running continuously or for extended periods. It is therefore essential to strike a balance between the desired temperature range and the energy consumption required to achieve it. One possible approach is to employ energy-efficient fans or optimize the system's cooling design to reduce the energy consumption of the fan. Additionally, it may be useful to evaluate the trade-offs between energy consumption and system performance when considering a wider temperature range. For example, if a lower temperature is critical for system stability or longevity, it may be worth accepting the higher energy consumption necessary to maintain that temperature. In summary, when considering a wider range of temperatures for controlling fan speed, it is crucial to consider the impact on energy consumption, system performance, and longevity. Striking a balance between these factors can help optimize the system's behaviours and ensure its reliable operation while minimizing energy consumption.

Table 4.2 Different states of fan speed (2°C).

Range of Temperature (°C)	Fan's Speed (RPM)
Tc <65	1000
65<Tc<67	1400
67<Tc<69	1600
69<Tc<71	1800
73<Tc<75	2000
75<Tc<77	2200
77<Tc<79	2400
79<Tc<81	2600
Tc <81	Max

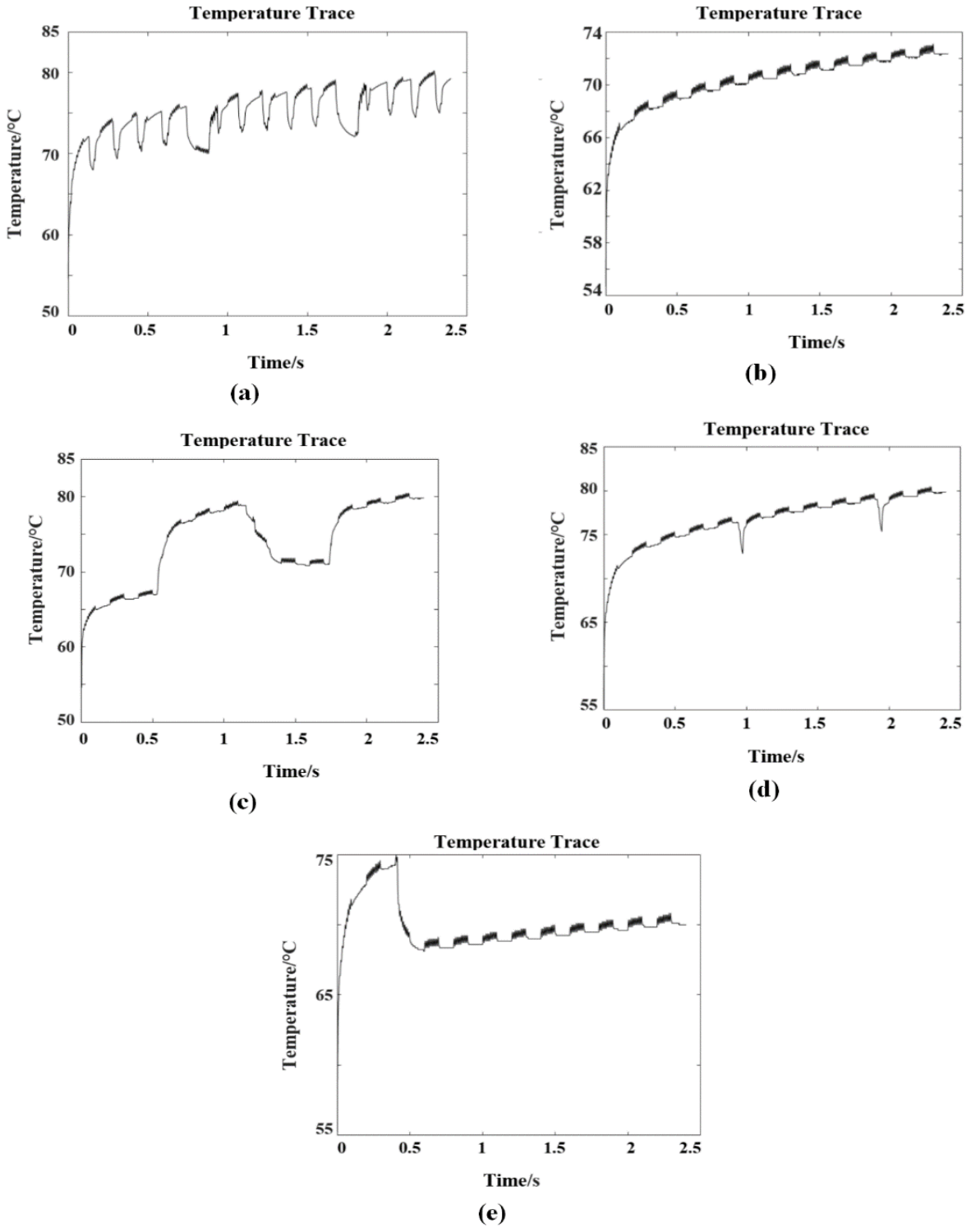

Figure 4.5 Different research outcomes: (a) Thermal signal without countermeasure, (b) Thermal signal with DVFS based countermeasure, (c) Thermal signal with one extra noise countermeasure, (d) Thermal signal with selective noise-based countermeasure, and (e) Thermal signal with fan speed-based countermeasure.

Figure 4.5 illustrates the temperature of the system with different methodologies. As this graph shows, with this new technique, we can decrease the temperature of the system as well as protect it from any potential attractions, which results in a decrease in power consumption.
Table 4.3 shows the experimental results of the different research. As this table shows the average BER of the thermal covert channel attacks across 8 × 8 systems can be higher than 95%.

4.2.1 Combination Methodologies

In this section, we try to combine different mythologies in regard to TCC attacks to achieve the best result (Low temperature, low power consumption as well as high bit error rate).

- **Thermal Covert Channel Attacks with DVFS and Fan Speed Controlling based Countermeasure.**

Applying DVFS and fan speed control-based countermeasures to the CPU core where the detected threads run to block the transmission. Scaling down the frequency level can effectively change the temperature signal drastically and by adding the fan speed control-based countermeasure, the temperature of the system decreases significantly, which results in decreased power consumption. So that the receiver will experience extremely high BER (Table 4.3). However, by increasing the workload of the system due to attacks, the temperature of the system starts to increase and fan speed-based control helps to control directly impact the cooling of the CPU core. Therefore, setting specific temperature thresholds to trigger fan speed adjustments can help us avoid system overheating and high-power consumption (Figure 4.6).

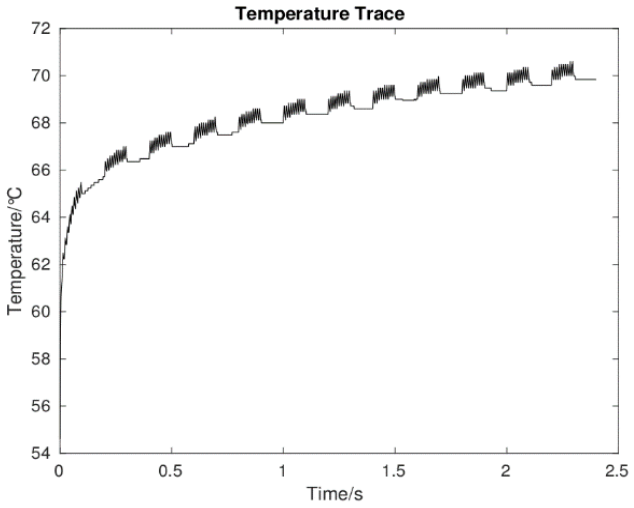

Figure 4.6 DVFS countermeasure along with the fan speed control-based countermeasure.

- **Thermal Covert Channel Attacks with Selective Noise and Fan Speed Controlling based Countermeasure.**

Figure 4.7 illustrates that when the fan speed countermeasure is implemented with selective noise, the BER rises to 97% (Table 4.3). This is mainly because when we added selective noise-based countermeasure to the system, it is helped the thermal signal to change its pattern however due to the extra workload that we applied to the CPU core system may face high temperatures so when we implemented the fan speed control-based control countermeasure on the top of the selective noise based countermeasure not only helps to improve the security on the system, it also helps to the reduction of average temperature the system, which result for the average power consumption decreases.

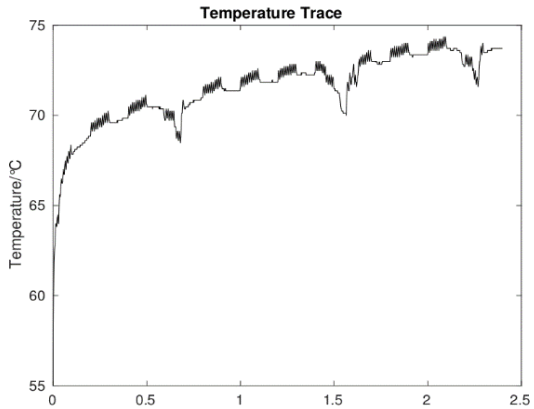

Figure 4.7 TCC with the selective noise and fan speed control-based countermeasure.

- **TCC with DVFS, Selective Noise and Fan Speed Controlling Countermeasure**

As Figure 4.8 and Table 4.3 illustrate, when all countermeasures are applied to the system, we can achieve the best result by considering BER, average power consumption and temperature.

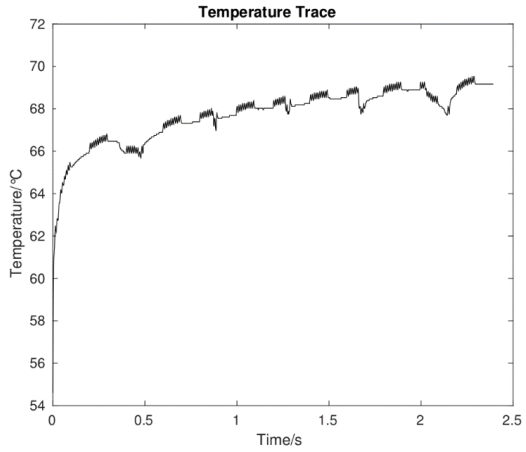

Figure 4.8 TCC with DVFS, selective noise and fan speed control-based countermeasure.

As Table 4.3 shows, one can see that the average BERs of our new proposed countermeasure are higher than that of the other countermeasure. We also determined that the average power consumption decreased in this study. For this result, we consider very 5 °C, this is mainly because if we change the speed fan in a very short time (e. g. 1 °C) the BER does not show any changes, but the power consumption of the system increases, resulting from the power consumption of the fan. Furthermore, if we consider the large duration for controlling the fan, the system will be at risk.

4.3 Experimental Results

All experiments are performed using a multi-core simulator, Sniper-v7.2 [82] as a multi-core simulator that is quick and dependable. As the power model, McPAT-v1.0 is included. McPAT is a multithreaded and many-core/multi-core architecture power, area, and timing modeling framework [83]. This analysis was performed in order to prevent a covert thermal channel attack on our targeted device. Thermal covert channel programming requires generating and measuring temperature signals using the Hotspot simulator to covertly transmit sensitive information from a secured zone to a non-secure one. To generate temperatures dynamically for all of the cores, we used the Hotspot-v6.0 thermal model. We use a few PARSECS and SPLASH-2 benchmarks, and their temperature signals are treated as thermal noise to a thermal covert channel. Sniper and

Hotspot are used in the experimental setup [84]. Different works have been done to fight TCC attacks. Figure 4.5 shows the different outcomes of different experiments. As Figure 4.5 (e) indicates when the controlling fan speed is implemented into the system, the temperature of the thermal signal decreases significantly which results in avoiding low power consumption. Furthermore, the bit error rate increases as well, which means that potential attacks can hardly happen to the system.

Table 4.3 Experimental results.

Countermeasures	BER	Avg Temperature (°C)	Avg Power Consumption(w)
TCC with DVFS	92%	70.35	22.31
TCC with One Extra Noise	94%	75.875	23.74
TCC with Selective Noise	94%	74.66	22.81
TCC with Controlling Fan speed. (Range of temperature 5°C)	95%	68.50	20.09
TCC with Controlling Fan speed with more range of temperature (4°C)	95%	68.39	20.29
TCC with Controlling Fan speed with more range of temperature (3°C)	95%	68.30	20.51
TCC with Controlling Fan speed with more range of temperature (2°C)	95%	68.25	20.79
TCC with Controlling Fan speed with more range of temperature (1°C)	95%	67.98	21.15
TCC with DVFS and Fan Speed	95%	67.56	19.65
TCC with Selective Noise and Fan Speed	97%	70.11	21.43
TCC with DVFS, Selective Noise and Fan Speed	98%	65.95	18.82

4.4 Summary

Our experimental results have confirmed that the new proposed countermeasure could be practically against TCC attacks with a high BER of 95% with its low temperature and low power consumption. The proposed countermeasure is a suitable scheme that can be used by multi-core systems to fight against these kinds of attacks. This study shows that thermal attacks are inevitable, however, an appropriate countermeasure should be designed to fight attacks by considering other aspects of the system such as power consumption as an essential metric in multi-core systems. Furthermore, by combining the proposed methodology with existing work (DVFS technique) and our previous experiment (selective noise countermeasure), we can achieve the best results (low power, low temperature, and high BER).

In this chapter, a new defence strategy for thermal covert channel attacks is presented. In our previous approach, we just considered one covert channel attack (Thermal covert channel attack), while by significant advance in different attacks in the modern systems, it is highly likely the attackers take advantage of different attacks that have similar behaviour, such as thermal and timing covert channel attack, to achieve secret data and information. In the next chapter these multi covert channel attacks are taken into account and by applying the suggested countermeasures in **Chapter 3** and **Chapter 4** we tried to fight this type of attack.

Chapter 5

Detection and Defence of Thermal and Timing Covert Channel Attacks

5.1 Overview

Among the most notable covert channel attacks, thermal and timing covert channels are more destructive, due to the nature of them. Thermal and timing covert channels are exploiting temperature variations and timing discrepancies to transmit sensitive information, respectively. Timing covert channel attacks refer to a class of attacks that exploit the variations in the timing of events to transmit hidden information between two communicating parties. In a timing covert channel attack, the attacker manipulates the timing of events, such as the time it takes for a process to execute or the time it takes for a packet to be transmitted, to convey information to a receiving party [140]. Time has always been known to contain information, but it was not till 1977 that Schaefer et al. proposed a methodology showing how time could be applied to carry information [119]. They suggested how a timing channel could be set up between two programs in a device or system to communicate covertly [125]. Data or information can be transmitted from one core to

another by the amount of time a sender holds the CPU. In 1996, Kocher introduced the first timing attack countermeasure in cryptographic.

He applied the variance in the execution time of a cryptographic operation to predict the sensitive information [120-121].

In recent times, attackers have become more cunning; based on their behaviour, timing, and thermal covert channels, they can combine, therefore combining both thermal and timing covert channels to craft more sophisticated and potent attacks. This novel approach allows them to control timing and manipulate thermal patterns, leading to greater secrecy and difficulty in detection. As these multi (thermal and timing) covert channel attacks continue to evolve, they pose numerous challenges for detection and prevention. For example, e.g., a thermal covert channel presents a unique challenge as it doesn't rely on shared sources like cache or memory, making it harder to detect. The amalgamation of timing and thermal covert channels leads to an expanded capacity for covert communication, facilitating the transmission of a larger volume of data surreptitiously. Consequently, there is a need to develop new strategies to cope with this heightened threat. To the best of our knowledge, this is the first time that a multi-covert channel attack has been considered. Similar to any other communication system, a multi-covert channel includes a pair of transmitter and receiver. In essence, the transmitter creates signals from sensitive data or information such as a password, and on the receiver side, the receiver reads the information from its sensor. Since a multi-covert channel is committed to transmitting signals over a chip, sensitive information is prone to leaking or being disrupted/degraded by attackers.

In response to this pressing issue, this chapter addresses the critical gaps in previous research and makes the following novel contributions:

- A detection strategy for multi-covert channel attacks.
- A defence strategy for defending against multi-covert channel attacks.
- Extensive evaluation and comparison against state-of-the-art, demonstrating advantages of our strategies that can protect the integrity and confidentiality of our data as covert channel attacks become more pervasive and insidious.

By considering these approaches, we can ensure efficiency and secure our system and devices, for protecting sensitive data and reliable performance.

The findings of Chapter 5 will be published in "Detection and defense of thermal and timing covert channel attacks in multi-core systems."

5.2 Timing Covert Channel Attacks

In this section, we have demonstrated a simple example of a timing attack. In this example, a microcontroller executes an algorithm (Algorithm 1) based on a 4-bit value.

Algorithm 1: Microprocessor Execution

Input: Input key, secret key.

Output: MES = IN ⊕ KEY**.** **//** Exclusive-OR the input with the secret Key

1: **for** EACH b BIT in MES **do** // For each bit of the resulting value (MES)

2: **if** b = 1 **then**

3: routine // Perform a certain operation (computing)

4: **else**

5: return

6: **end if**

7: **end for**

In any process, each bit of MES will check, if the bit is '1' an operation will be performed otherwise the operation will not execute. Figure 5.1 shows this process and based on this algorithm the user provided input (IN) Exclusive-OR (XOR) the secret key (KEY) and the store in a manufacturing execution system (MES). The attacker knows the input (IN) as well as the algorithm but does not know the KEY.

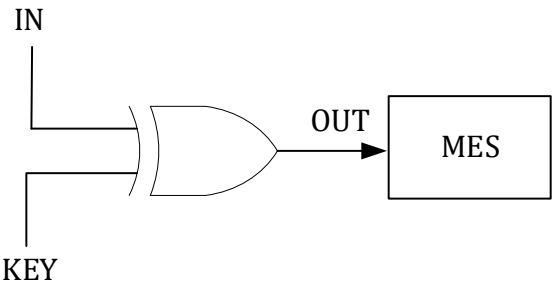

Figure 5.1 Operation process.

In the first step, the attacker measures the execution time for each IN (given input). If each routine execution takes 1 ms and the KEY '1101', Table 5.1 indicates this measurement.

Table 5.1 Execution time for 4 bits different input.

IN (User input)	IN ⊕ KEY	Execution Time
0001	1100	2 ms
0010	1111	4 ms
0011	1110	3 ms
0100	1001	2 ms

To access the KEY attacker follows bellow steps:

1. Take a random KEY value (KEY_i, i in [0:15])
2. Compute IN ⊕ KEY_i
3. Measure execution time of the operation with unknown key.
4. Compare time.

Figures 5.2 and Figures 5.3 illustrate the process of timing attacks.

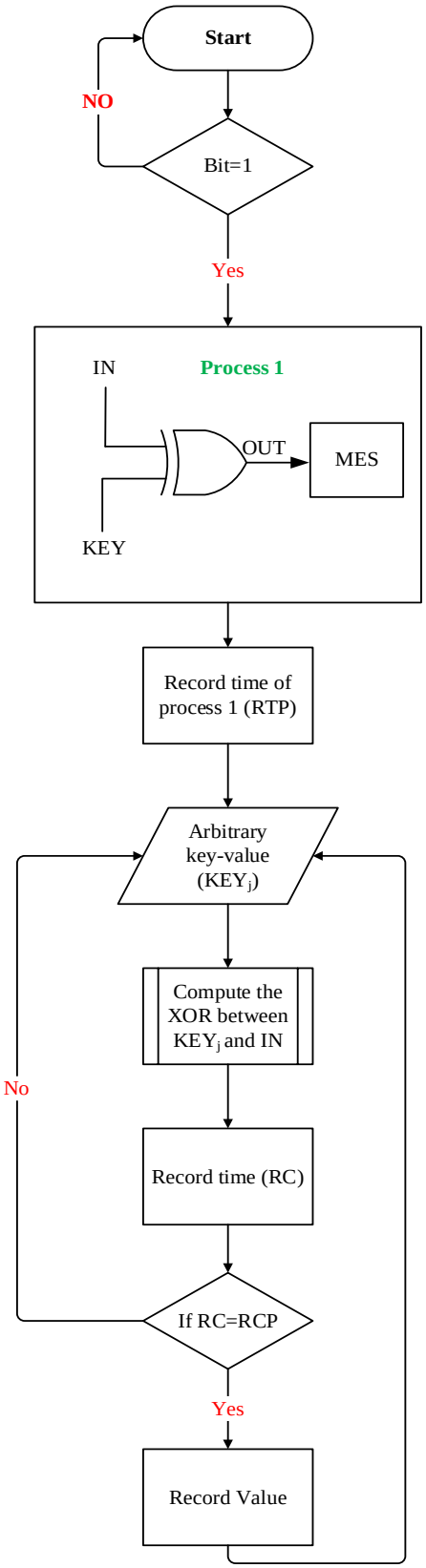

Figure 5.2 Process of timing attacks.

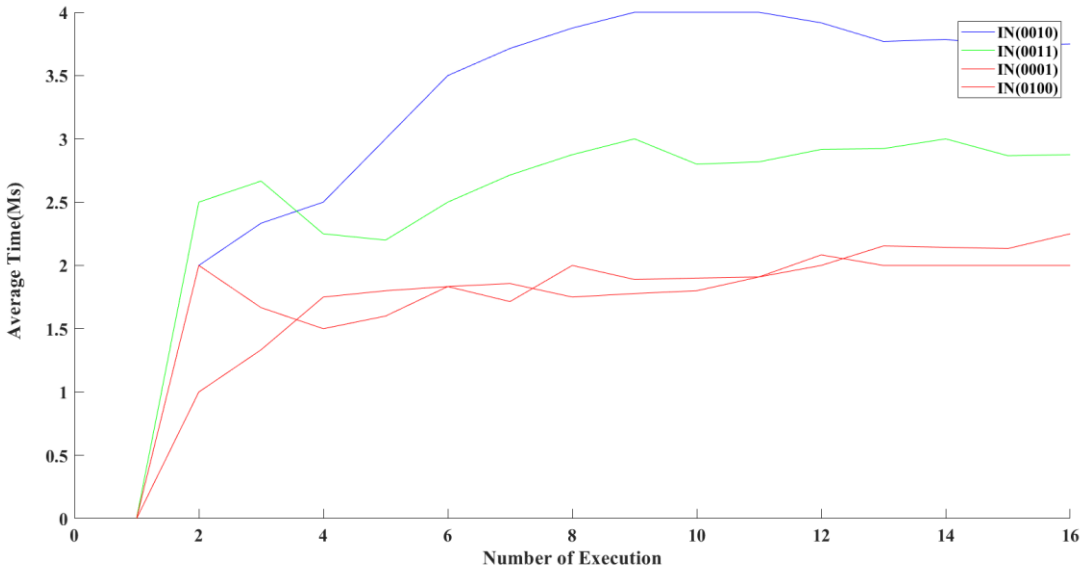

Figure 5.3 The process of cryptographic operation

Table 5.2 shows the execution time of different prediction operations. In this example, attackers after the 14th operations can access the secret keys.

Table 5.2 Predict execution time based on the random KEY.

IN (User input)	IN ⊕ KEY	Time prediction for KEY_0=0000	IN ⊕ KEY	Time prediction for KEY_1=0001	Time prediction for KEY_i=XXXX	IN ⊕ KEY	Time prediction for KEY_{13}=1101	Execution Time
0001	0001	1	0000	0	…	1100	2	2 ms
0010	0010	1	0011	2	…	1111	4	4 ms
0011	0011	2	0010	1	…	1110	3	3 ms
0100	0100	1	0101	2	…	1001	2	2 ms

5.3 Methodology for Multi-Covert Channel Attacks

- **Thread Model**

The attackers can access the local sensor data by using software interfaces such as Intel's processor Model Specific Register (MSR) and manipulate the data [14]. The defender called the global manager processing unit can also access all processing units' fundamental quantity and execute the detection and barricading programs to each processing unit [2].

- **Communication Process**

The initiation of a transmission session requires the transmitter and the receiver to first agree on the transmission frequency for the multi-covert channel. During communication, first, the transmitter sends a request packet to the receiver and waits for a response. Upon the receiver determines that the received message is a request message, a response is sent with an acceptance message to the transmitter. If the transmitter does not get an acknowledge message within a predefined timeout period, another request message is sent. If an acknowledgement message is received, data messages are transmitted to the receiver and a termination message is sent to indicate the end of this ongoing transmission process. The receiver then proceeds to receive data messages sent by the transmitter until it receives a termination message that terminates a process [173].

5.3.1 Finding Suitable Band for Covert Channels

This study analyses the power spectrum of each core within a multi-core system during the execution of a typical application like Blackscholes from the PARSEC benchmark suite (Figure 5.4). The analysis divides the power spectrum into three distinct frequency bands:

Band A (DC - 50 Hz): This Band highly susceptible to disruption by signals generated by typical applications. This means that any data transmission within this band is prone to interference and errors. Furthermore, this range of frequency likely captures slow-varying signals related to core utilization, cache activity, and other low-frequency system events [63].

Band B (50 - 400 Hz): This range of frequency is a suitable band for transmitting data through covert channels, as Figure 5.6 illustrates.

Band C (> 400 Hz): Considered the cut-off frequency for reliable data transmission due to high potential for interference. This region is deemed unsuitable for data exchange within the multi-core system. This band likely encompasses high-frequency signals related to internal

clocking, signal integrity issues, or noise sources [173].

It is important to note that timing and thermal signals are particularly vulnerable to influence from signals generated by normal applications [64]. This vulnerability highlights the need for careful selection of transmission frequencies to minimize interference and ensure reliable data transmission within the multi-core system.

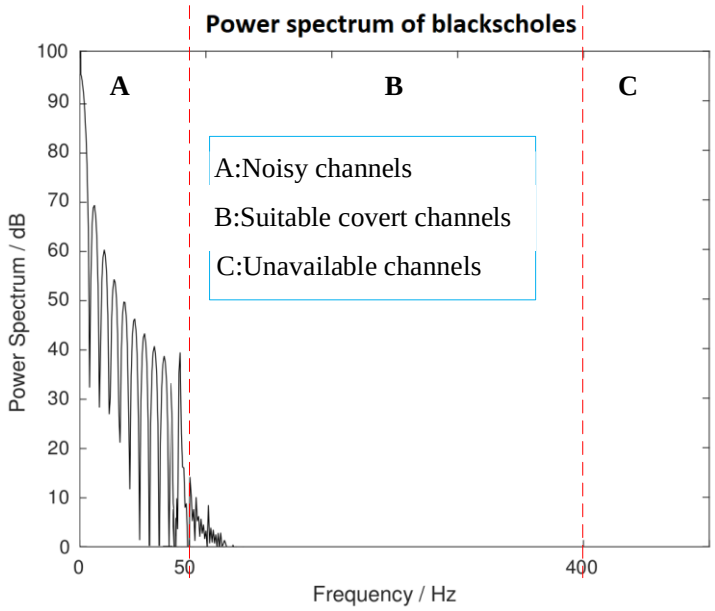

Figure 5.4 The power spectrum of a core's temperature signal obtained when one typical application runs.

5.3.2 Strategy for Separate Covert Channels

This section details the approach, including detection and countermeasure steps illustrated in Figure 5.4. Leveraging the experimentally identified frequency band B from the previous section, it proposes dividing Band B into two sub-bands for data transmission through thermal and timing covert channels.

- **Channel Selection:**

We experimentally select below range of frequencies for transmitting data through thermal and timing covert channel.

- **Thermal Covert Channel:** Based on its effectiveness in heat transfer and lower susceptibility to noise, a frequency range of 50Hz to 200Hz is chosen for thermal channel transmission [46].

- **Timing Covert Channel:** Considering system specifics, data rate, and noise levels, frequencies between 200Hz and 400Hz are deemed suitable, offering a balance between achievable data rates and minimal interference [179].

Detection Mechanism: Each core independently employs bandpass filters to scan Band B, allowing only designated frequencies to pass through and blocking others. This comprehensive scan aims to identify any potential signals within the designated range. Following the scan, each core analyses the signal's maximum amplitude (strength/intensity) at each frequency. Whenever the amplitude exceeds a predefined threshold (ρ) for either timing or thermal channels, an attack is detected. Experimentally chosen thresholds for double covert channel strategy are:

- $20db < \rho < 40db$: Timing covert channel attack
- $40db < \rho1 < 50db$: Thermal covert channel attack

Countermeasure*:* Based on our access to the system root, an appropriate countermeasure will be applied, as shown in Figures 5.5.

If we have access to the root, we can apply a DVFS-based countermeasure.

- **DVFS-based countermeasure:** Modern CPUs often possess dynamic voltage and frequency scaling capabilities, enabling dynamic adjustments to voltage and frequency based on workload [18]. This thesis proposes a DVFS-based countermeasure to mitigate covert channels by manipulating core frequencies. To apply this technique, each core is individually managed using DVFS to disrupt signal transmission.

  Timing channels: Rapid frequency changes (e.g., frequency hopping) disrupt precise time intervals essential for data encoding.

  Thermal channels: Reduced voltage and frequency lead to lower heat generation, making temperature-based signal detection challenging.

Figure 5.6 and Table 5.4 demonstrate a significant increase in bit error rate, rendering the transmitted signal unusable, effectively thwarting covert communication.

- **Challenges and Considerations:**

Performance Overhead: Frequent DVFS adjustments can introduce performance overheads that require careful management.

Adaptive Attackers: Attackers may develop techniques to counter DVFS measures, necessitating continuous adaptation of defensive strategies.

- **Fan speed control-based countermeasure:** The next countermeasure that we can add to

disrupt multi-covert channels attacks is the fan speed control-based countermeasure, which is based on controlling the fan speed at an appropriate level. In multi-core systems, factors like execution time, frequency, and voltage contribute significantly to heat generation [173]. By strategically modulating temperature and frequency fluctuations through precise fan speed control across individual cores, this approach effectively disrupts covert channels and hinders attack effectiveness.

Methodology:

This strategy includes below steps:

1. Fan speeds are dynamically adjusted based on real-time temperature readings, disrupting the communication channels attackers rely upon for data exfiltration.
2. Table 5.3 outlines pre-defined fan speed settings for different temperature ranges, aiming to maximize covert channel disruption without compromising system performance.
3. This approach targets timing covert channels, where attackers exploit heat signatures generated during sensitive data transmission. By actively controlling fan speeds, these thermal variations are intentionally blurred, making signal access significantly more challenging for attackers.

Table 5.3 Different states of fan speed.

Range of Temperature (°C)	Fan's Speed (RPM)
Tc <65	1000
65<Tc<68	1500
68<Tc<71	1700
71<Tc<74	1900
74<Tc<77	2100
77<Tc<80	2300
80<Tc<83	2600
Tc≥83	Max

Through this strategy's ability to disrupt thermal patterns and conceal sensitive information, fan speed control-based countermeasures offer an effective approach to mitigating the threat of multi-covert channel attacks in multi-core systems.

Advantages:

Offers a unique approach to combating multi-covert channels by manipulating thermal patterns.

Disrupts communication channels without impacting performance significantly.

If we do not have access to the root, we can add either selective noise-based countermeasures or fan speed control-based countermeasures.

- **Selective noise-based countermeasure:** This section expands on previous work regarding single-channel thermal covert channel mitigation by proposing a modified approach for multi-channel scenarios. Unlike the prior system, which solely relied on signal temperature, this new strategy focuses on detecting and disrupting bit '1' transmissions, typically representing high-frequency/high-temperature states. This methodology includes below steps:
    1. Bit Monitoring: The system continuously monitors the transmitted signal and records each bit value.
    2. Noise Injection at Bit '1': When a bit '1' is detected, pre-determined noise is injected to disrupt the established pattern and potentially mask any covert information encoded.
    3. Temperature/Frequency Tracking: Following noise injection, the system records the current signal frequency/temperature as a baseline for comparison.
    4. Adaptive Noise Modulation: During the next bit transmission, the system compares the current frequency/temperature with the baseline. If the current value is lower (indicating a potential transition to bit '0'), additional noise is injected to maintain the high state, further disorienting covert channels and introducing intentional delays.
    5. Iterative Cycle: The monitoring, noise injection, and comparison steps continue for each subsequent bit until the entire transmission is processed.

This multi-faceted approach offers several advantages over the previous single-channel method:

- Targeted Defence: Focuses on bit '1', specifically disrupting channels relying on high-frequency/high-temperature encoding.
- Adaptive Noise Modulation: Dynamically adapts noise injection based on signal analysis, countering covert attempts to change encoding patterns.
- Delay Introduction: Intentional delays disrupt covert channel timing-sensitivity, making information extraction challenging.

Our proposed selective noise-based countermeasure presents a robust and adaptable solution for mitigating multi-channel thermal covert channel threats, offering enhanced protection for sensitive systems and data. This expanded version provides a more comprehensive explanation of the process, highlights the key benefits of the improved approach, and emphasizes its effectiveness against multi-channel covert channels.

As Figure 5.6 and Table 5.4 indicate, by using two different covert channel frequencies for transmitting data, the system will face overheating as well as high power consumption. This is mainly because running two covert channels simultaneously requires additional processing resources, including CPU and memory usage. This increased load can cause the system's components to work hard and generate more heat and power consumption to cool the system down.

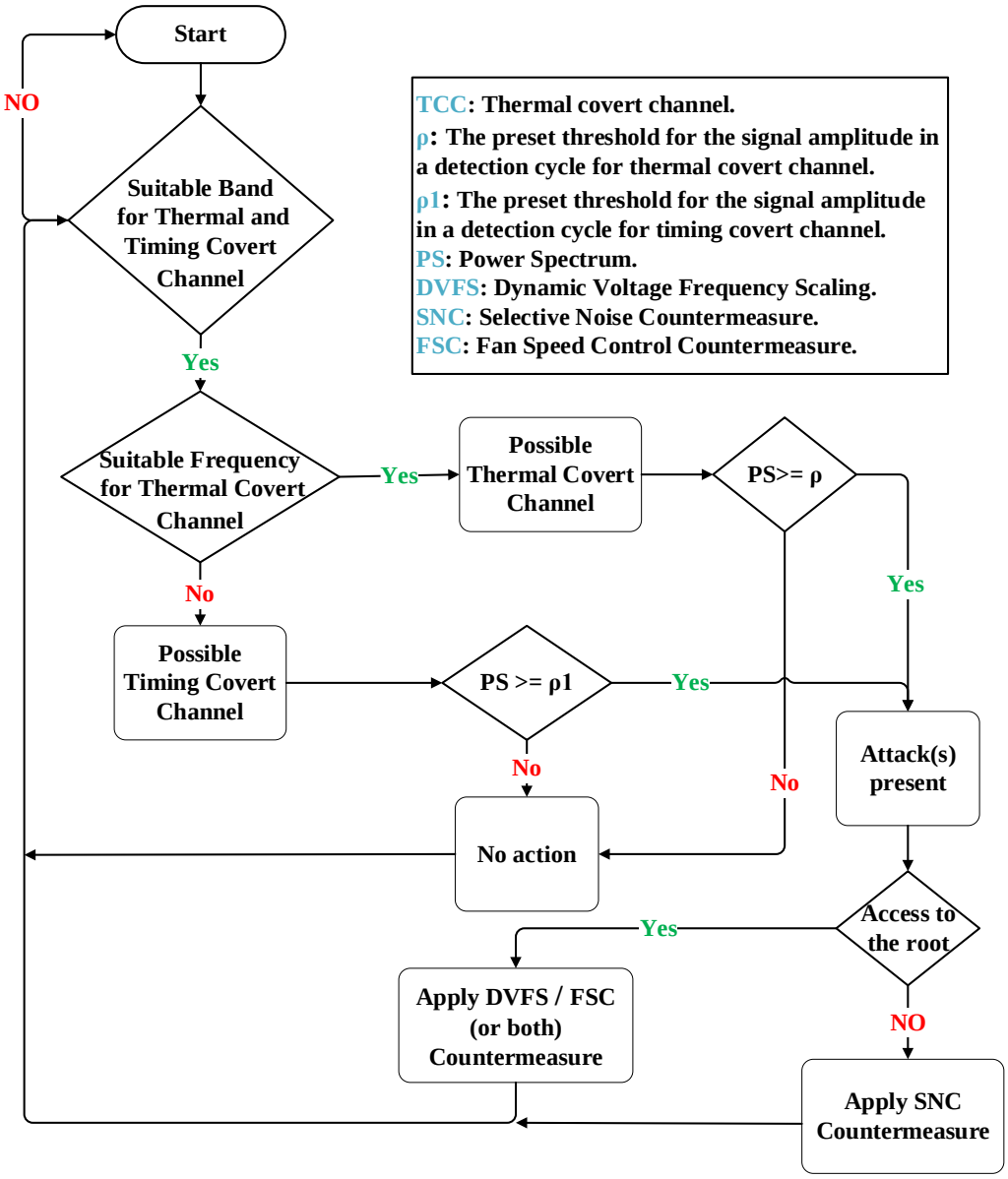

Figure 5.5 Diagram of process of detection and defence of Thermal and Timing Covert Channel Attacks in double covert channels (Process1).

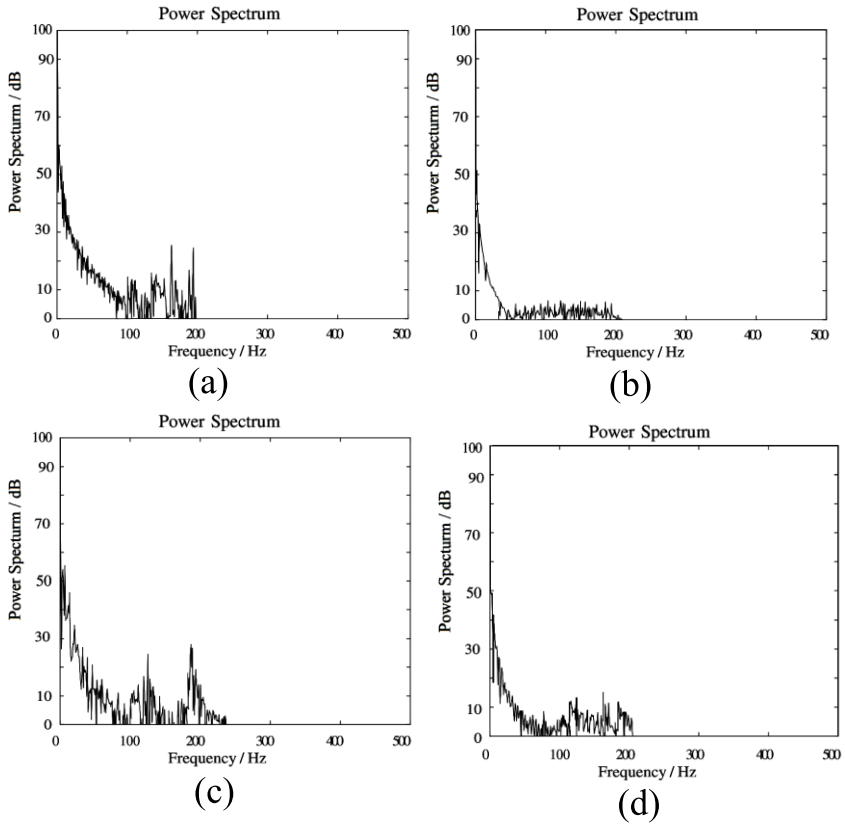

Figure 5.6 System behaviour with double covert channels, (a) system without countermeasure, (b) system with DVFS countermeasure, (c) system with Selective Noise countermeasure and (d) system with Fan Speed Control countermeasure.

5.3.3 Strategy for Single Covert Channel for Transmitting both Timing and Thermal Data

To address the performance concerns associated with dual covert channels (overheating, power consumption), this thesis proposes a novel, synergistic approach depicted in Figure 5.7. This approach leverages a single covert channel for transmitting both timing and thermal data, achieving an efficient and robust solution. Instead of employing separate channels, we ingeniously intertwine timing and thermal parameters to encode and transmit information simultaneously. This technique modulates thermal variations with timing information, essentially embedding the timing data within the thermal signal.

- **Detection and Countermeasures**

To detect and apply proper countermeasure to the system, we need to follow below steps:

- **Pre-shared Parameters:** Sender and receiver agree on channel parameters like frequency and packet transmission intervals before use, facilitating detection through analysis of these parameters.

- **Packet Rate Thresholds:** Deviations from pre-defined acceptable packet transmission rates can trigger alerts for potential attacks.
- **Adaptive Thresholds:** For this approach, considering $\rho > 50$ dB allows for setting adaptive thresholds based on specific attack scenarios, further enhancing detection accuracy.

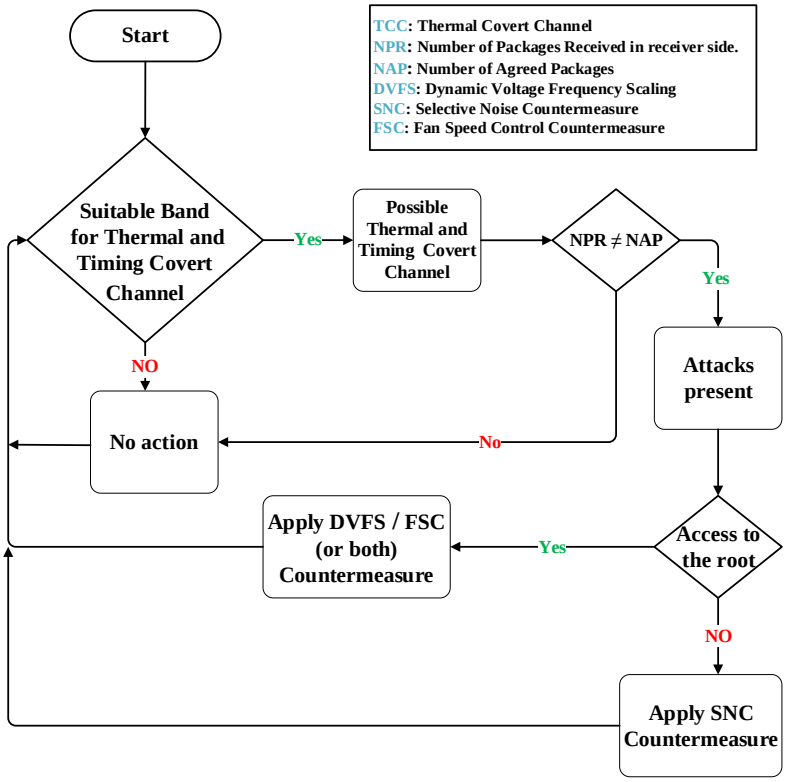

Figure 5.7 Diagram of the process of detection and defence of Thermal and Timing Covert Channel Attacks in single covert channels.

In Figures 5.8, we can see different outcomes of a timing and thermal covert channel, which uses a single channel to transmit data from a transmitter to a receiver.

This single-channel approach has several benefits:

- **Reduced power consumption:** By eliminating the need for a dedicated timing channel, we significantly reduce the overall power footprint of the system. This is crucial for applications where energy efficiency is paramount, such as mobile devices and battery-powered sensors.
- **Minimized overheating risk:** With one less channel generating heat, the overall thermal load on the system is reduced. This translates to less stress on the cooling mechanisms and a lower risk of overheating, leading to improved system stability and reliability.

- **Enhanced data transmission efficiency:** By cleverly multiplexing both data streams onto a single channel, we can potentially achieve higher overall data throughput compared to using separate channels. This is because the modulation technique can pack more information into the same bandwidth.
- **Increased robustness:** Simplifying the communication architecture by unifying the channels leads to a more robust system. This is because there are fewer components and potential points of failure, making the system less susceptible to errors and disruptions.

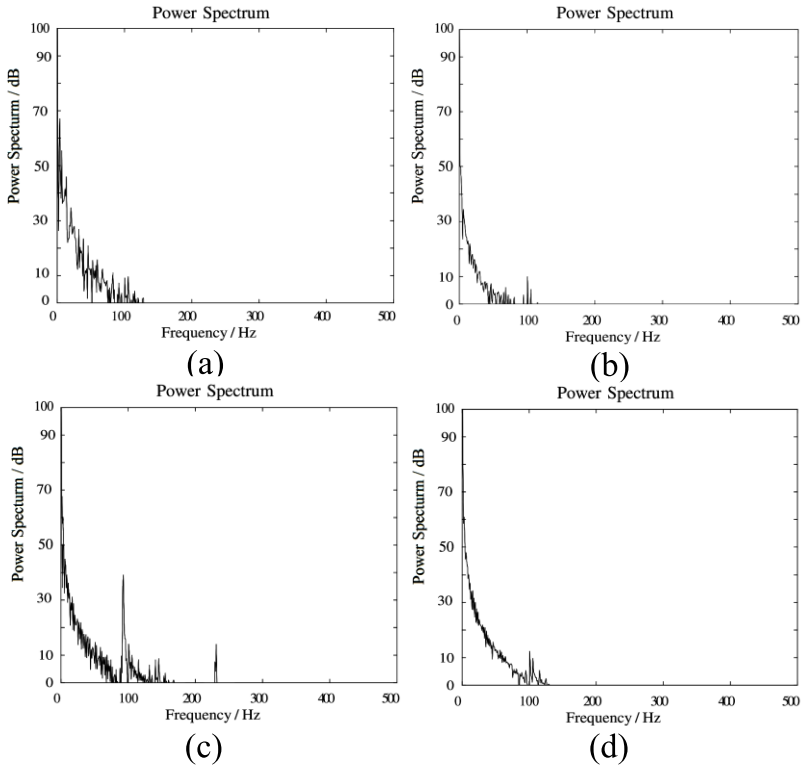

Figure 5.8 Thermal and Timing covert channel attacks are using one channel. (a) Multi attacks without countermeasure, (b) Multi attacks with DVFS countermeasure, (c) Multi attacks with selective noise countermeasure, (d) Multi attacks with Fan speed control counter.

5.3.4 Combination Methodology (Multi covert channel with DVFS and Fan speed controlling based countermeasure)

If we have access into the root, we can combine different countermeasures. This approach involved synergistically combining various methodologies to achieve optimal power efficiency and maximize bit error rate. As shown in Tables 5.4 and 5.5, integrating a dynamic voltage and frequency scaling countermeasure with a fan speed control-based one dramatically boosts BER to

98% while significantly reducing power consumption. This remarkable improvement stems from the combined effect of optimizing fan operation and dynamically adjusting processor voltage and frequency. By dynamically tailoring fan speed to the system's actual cooling demands (Figure 5.9), we eliminate unnecessary fan usage, leading to substantial power savings. Additionally, DVFS allows us to dynamically adjust processor voltage and frequency based on real-time workload requirements, further minimizing power consumption without compromising performance. This synergistic combination of countermeasures effectively thwarts multi-covert channel attacks while ensuring optimal power efficiency for our system. Furthermore, dynamically scaling processor voltage and frequency further amplifies the power reduction. Adjusting these parameters in the transmission process based on workload demands allows the system to operate at the minimum power required for its current task, effectively squeezing out every ounce of efficiency. And also, by dynamically adjusting fan speed based on the system's cooling needs, we can avoid unnecessary full-speed operation, leading to substantial power savings. This targeted approach ensures adequate cooling while minimizing energy expenditure. In general, the combined effect of these two countermeasures creates a powerful synergy, delivering a dramatic boost in BER while simultaneously slashing power consumption. This technique underlines the potential of employing a multifaceted approach to combating covert channel attacks while keeping power efficiency in mind.

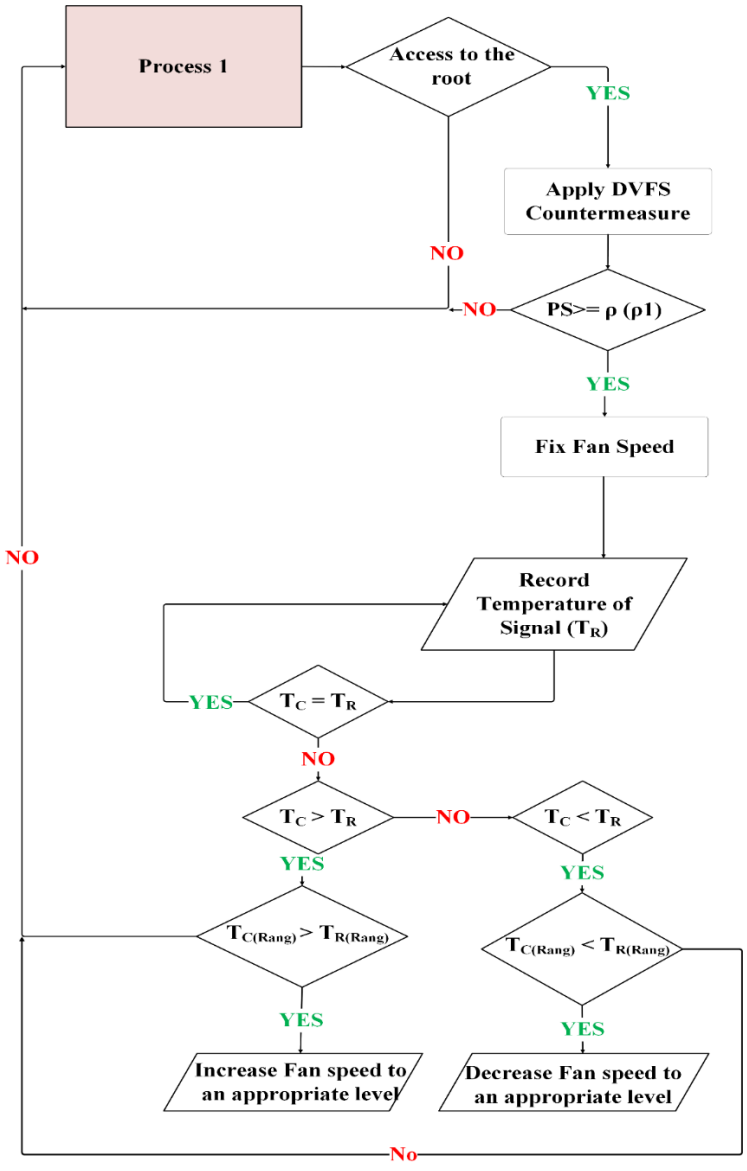

Figure 5.9 Diagram of process combine defence strategy in multi covert channels.

Where:

Process 1: Represent Process of double multi covert channel or single multi covert channel.

ρ: The preset threshold for the signal amplitude in a detection cycle for thermal covert channel.

ρ1: The preset threshold for the signal amplitude in a detection cycle for timing covert channel.

PS: Power Spectrum.

FSC: Fan Speed Control Countermeasure.

T_C: The temperature trace of current bit.

T_R: The temperature trace recorded.

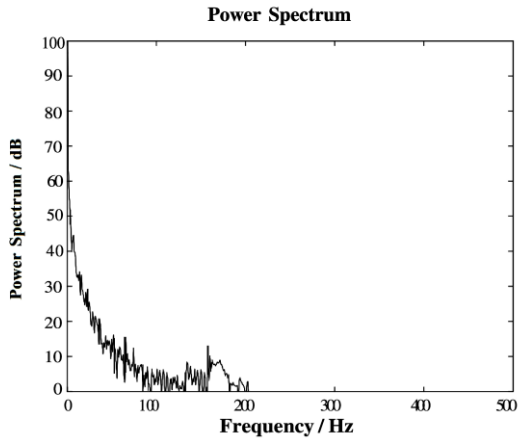

Figure 5.10 Double covert channel with Selective Noise and fan speed countermeasure.

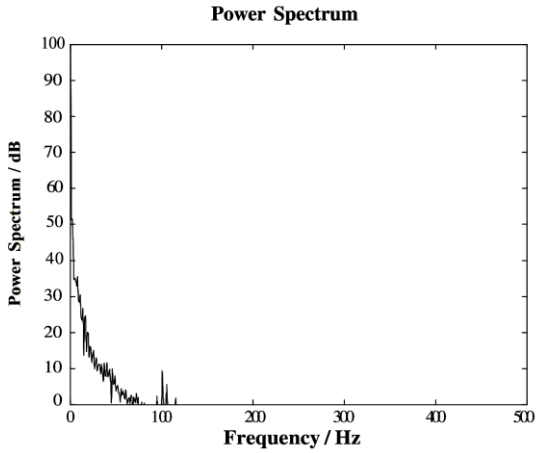

Figure 5.11 Single covert channel with Selective Noise and fan speed countermeasure.

As Figures 5.10, 5.11 and Tables 5.4 and 5.5 indicate, combine methodology was effective in reducing overheating and power consumption. However, further consideration is needed to ensure that the system does not overheat or consume too much power under all conditions. This increased load can cause the system's components to work harder and generate more heat and power consumption to cool the system down.

5.4 Experimental Results

Two sets of experiments have been performed, first, considering two different covert channels for transmitting data, and then applying one covert channel for both thermal as well as timing covert channel. Mostly, these experiments are implemented by using MATLAB as well as a multi-core

simulator, Sniper-v7.2 a fast and reliable simulator for multi-core. McPAT-v1.0 integrated as the power model. McPAT is an integrated power, area, and timing modeling framework for multithreaded, and many-core/multi-core architectures [82]. Figures 5.5 and 5.6 demonstrate that the introduction of various countermeasure systems yields distinct behaviours. For instance, when employing DVFS with the primary objective of obstructing communications within a thermal and timing covert channel. Thermal and timing attacks can be blocked by applying DVFS to the cores participating in the thermal and timing covert channel, or when implementing the selective noise-based countermeasure, the power spectrum experiences increased fluctuation due to the additional noise introduced into the system. This added noise has the potential to distort the signal, consequently generating supplementary peaks or valleys in the power spectrum [112]. Alternatively, the utilization of a fan speed-based countermeasure leads to a reduction in the system's temperature, subsequently causing a decline in the system's frequency. Overall, all of these countermeasures are applied to the cores where the respective threads are executed, effectively neutralizing any potential transmission of timing, thermal, or multi-covert channels. Experimental results have confirmed that the proposed countermeasures could practically shut down thermal and timing attacks by forcing their transmissions to be extremely high. Table 5.4 indicates these results.

Table 5.4 Experimental results of double covert channel for timing and thermal.

Countermeasures	BER (Thermal Covert Channel)	BER (Timing Covert Channel)	Avg Power Consumption(w)
DVFS	92%	96%	27.77
Selective Noise	94%	95%	33.50
Fan Speed Controlling	95%	93%	25.45
Double covert channel with DVFS and Fan Speed	97%	98%	26.86

As tabulated in Table 5.4, the results of considering two different covert channels for transmitting data and the impact they can have on the temperature of the communication system. The data suggests that the transmission process can lead to significant increases in temperature, which can

pose a risk to the system. Additionally, it may be necessary to consider alternative approaches to address the issue of overheating. This could involve implementing additional cooling systems or using more efficient transmission protocols to reduce the amount of heat generated during the transmission process. To address this issue, a new approach is proposed that involves using a single covert channel for both timing and thermal covert channel transmissions. This approach aims to reduce the overall amount of data being transmitted, which can help reduce the amount of heat generated during the transmission process as tabulated in Table 5.5. By using a single covert channel, the proposed approach can also simplify the overall design of the communication system. This can make it easier to manage and maintain, as well as potentially reducing the overall cost of the system. Furthermore, the proposed approach can also improve the overall security of the system by making it more difficult for attackers to use covert channel techniques to transmit sensitive information. This is because using a single covert channel for both timing and thermal covert channel transmissions can make it more difficult for attackers to conceal their transmissions and avoid detection.

Table 5.5 Experimental results of single covert channel for timing and thermal.

Countermeasures	BER	Avg Power Consumption(w)
DVFS	96%	24.16
Selective Noise	93%	29.53
Fan Speed Controlling	95%	23.38
Single covert channel with DVFS and Fan Speed	98%	21.45

As indicated in Table 5.5, while using a single covert channel for both timing and thermal covert channel transmissions can help However, second approach may result in other issues, such as a loss of packages during transmission. This is because using a single channel for both types of covert channel transmissions can increase the likelihood of interference or signal overlap, which can result in lost or corrupted data packages. In addition, the use of a single channel may also limit the amount of data that can be transmitted, which can result in a slower overall transmission rate. To address these issues, it may be necessary to optimize the communication system's design and configuration. This could involve adjusting the transmission rate, optimizing the signal processing

algorithms, or implementing error correction techniques to reduce the risk of lost or corrupted data packages. The proposed work has offered several advantages over a single-channel approach, which are summaries below:

- **Increased Redundancy:** If one channel becomes unreliable or noisy, the other channel can still be used to transmit data.
- **Improved Security:** Combining two different covert channels makes it more difficult for an attacker to detect and disrupt the covert communication.
- **Potential for Higher Bandwidth:** By utilizing both timing and thermal channels, the overall bandwidth of the covert communication can be increased.

Despite all benefit this work has, it also introduces additional complexity and synchronization challenges:

- **Increased Complexity:** Designing and implementing a covert channel that combines multiple channels is more complex than a single-channel approach.
- **Synchronization Requirements:** Both the sender and receiver need to be synchronized to effectively utilize both channels for data transmission.
- **Potential Interference:** The timing and thermal channels may interfere with each other, making it difficult to distinguish between the two channels.

To address these issues, it may be necessary to optimize the communication system's design and configuration, as well as consider alternative approaches to address the issue of overheating.

5.5 Summary

The chapter suggested different methods used to block transmission data through thermal and timing covert channels while the system is under attack. It detailed the steps involved in creating and executing these methods and explained how we identified and separated sensitive data from other data transmitted through these channels. Furthermore, the chapter evaluated the effectiveness of our detection techniques against anticipated attempts by attackers to circumvent them. The results revealed that using two channels for transmitting data can increase the accuracy of the data transmission, but it also comes at the cost of increased power consumption of the system, which is not ideal. On the other hand, if we use only one channel for both methods, the power consumption decreases, but we may experience data loss during the transmission process, which ultimately decreases the accuracy rate. This trade-off presents a challenge in selecting the

appropriate transmission method and highlights the need to carefully consider factors such as accuracy and power consumption when designing covert channels. Overall, this chapter sheds light on the complexities and challenges of covert channel transmission and highlights the need for further research in this area. In addition to evaluating the effectiveness of our detection techniques, the chapter also examined the potential risks associated with covert channel transmission. It highlighted the need for security measures to protect against such risks and emphasized the importance of designing secure transmission methods.

Chapter 6

Conclusion

The optimization of security, power consumption, and performance remains a crucial balancing act in today's embedded systems landscape. However, potential risks like thermal and timing covert channels must also be factored into the equation. Sensitive information leakage is a growing concern within embedded systems. As technology advances, the risk of covert channels hidden communication channels between processors has escalated, posing a significant security threat. Among the most notable are thermal and timing covert channels [1], which exploit different techniques such as temperature variations and timing discrepancies to transmit sensitive information, respectively. These attacks are particularly insidious because they can be executed on multi-core systems without specialized equipment, making them highly accessible to potential attackers. Therefore, it is essential to consider not just the traditional metrics of security, power consumption, and performance, but also the potential for covert channel exploitation during system design and implementation. This thesis delves into the research challenges presented by thermal and timing covert channel attacks, focusing on two main categories: "Detection and Mitigation."

- **A novel selective noise-based countermeasure for thermal covert channels (Chapter 3):** Building upon existing research and addressing their limitations, Chapter 2 introduces a novel selective noise-based countermeasure for thermal covert channels, overcoming the limitations of previous methods by achieving higher detection accuracy and reduced power consumption.
- **A unique fan-speed control-based countermeasure (Chapter 4):** Further enhances security and power efficiency by proposing a unique fan-speed control-based

countermeasure. This defence strategy complements the selective noise approach and offers significant advantages over existing fan control methods, as evidenced by the presented experimental results.

- **Addressing multi-covert channel attacks (Chapter 5):** Recognizing the evolving nature of timing and thermal covert channel threats, this chapter explores the concept of multi-covert channel attacks, which combine both thermal and timing channels. Such combined attacks pose a significant risk due to their expanded communication capacity and the unique challenges they present for detection and prevention. Thermal covert channels, for instance, are notoriously difficult to detect as they don't rely on shared resources like cache or memory. The amalgamation of timing and thermal covert channels leads to a concerning increase in covert communication capabilities, enabling the transmission of larger volumes of data undetected. This necessitates the development of new strategies to counter this heightened threat. **Chapter 5** addresses these critical gaps in previous research, proposing novel approaches for detection and mitigation. The proposed methods demonstrate measurable effectiveness in countering covert channels, providing evidence of exceeding the state-of-the-art through comprehensive evaluations and comparisons.

This thesis contributes significantly to the field of embedded systems security by providing practical solutions for countering thermal and timing covert channels, including multi-channel attacks. The proposed countermeasures offer improved security, power efficiency, and address the limitations of existing approaches. Future research can explore the application of these techniques to different embedded system architectures and investigate the integration with other security mechanisms.

6.1 Future Work

After conducting extensive research during my Ph.D. study, a question still remains and arises in the author's mind is, 'What is the next step(s)?'. The primary aim of posing this question is to highlight the research deficiencies and concepts. Towards it, the following future works can be carried out.

- **Advanced detection and mitigation methods for other covert channel attacks**: Focus on developing more advanced detection and mitigation techniques for different covert channel attacks. This could involve the creation of more sophisticated algorithms that can

identify and neutralize covert channels in real time, as well as the development of hardware and software solutions that can prevent these channels from being exploited in the first place.

- **Exploration of a security strategy at the hardware level:** Exploration of new approaches to designing embedded systems that minimize the potential risk of covert channels. This could involve the integration of security measures at the hardware level, the use of specialized software tools for detecting and mitigating covert channels, and the implementation of new design paradigms that prioritize security and efficiency.
- **Develop comprehensive frameworks for other covert channel attacks**: Focus on the development of more comprehensive frameworks for evaluating the security, power consumption, and performance of embedded systems. These frameworks could incorporate multiple metrics and provide a more nuanced and holistic view of the overall effectiveness of embedded systems in real-world scenarios.

References

[1] M. Gupta, "Security challenges and cryptography in embedded systems," International Journal of Advance Research in Science and Engineering, vol. 4, no. 1, 2015.

[2] M. Brown, "Five Steps to Improving Security in Embedded Systems," Contemporary Technologies in Automation, vol. 3, 2013.

[3] P. Wang and L. Liu, "Strategy of library information resource construction based on FPGA and embedded system," Microprocessors and Microsystems, vol. 83, 2021.

[4] R. Zurawski, Embedded Systems Handbook: Embedded systems design and verification. CRC Press, 2018.

[5] P. Marwedel, Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things. Springer Nature, 2021.

[6] D. Papp, Z. Ma, and L. Buttyan, "Embedded systems security: Threats, vulnerabilities, and attack taxonomy," in 2015 13th Annual Conference on Privacy, Security and Trust (PST). IEEE, 2015.

[7] A. Abbasi et al., "Challenges in designing exploit mitigations for deeply embedded systems," in 2019 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2019.

[8] A. Vega, P. Bose, and A. Buyuktosunoglu, Rugged Embedded Systems: Computing in Harsh Environments. Morgan Kaufmann, 2016.

[9] H. Huang, C. Hu, and J. He, "A security embedded system based on TCM and FPGA," in 2009 2nd IEEE international conference on computer science and information technology. IEEE, 2009.

[10] J. A. Ambrose et al., "Side channel attacks in embedded systems: A tale of hostilities and deterrence," in Sixteenth International Symposium on Quality Electronic Design. IEEE, 2015.

[11] H. Gamaarachchi and H. Ganegoda, "Power analysis-based side channel attack," arXiv preprint arXiv:1801.00932, 2018.

[12] S. D. Putra, A. S. Ahmad, and S. Sutikno, "Power analysis attack on implementation of DES," International Conference on Information Technology Systems and Innovation (ICITSI). IEEE, 2016.

[13] O. Lo, W. J. Buchanan, and D. Carson, "Power analysis attacks on the AES-128 S-box using

differential power analysis (DPA) and correlation power analysis (CPA)," Journal of Cyber Security Technology, vol. 1, no. 2, 2017.

[14] F. Chai and K. Kang, "A New Power Analysis Attack and a Countermeasure in Embedded Systems," in 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). IEEE, 2018.

[15] V. K. Rai et al., "Correlation power analysis and effective Defence approach on light encryption device block cipher," Security and Privacy, vol. 2, no. 5, 2019.

[16] H. A. Khan et al., "Malware detection in embedded systems using neural network model for electromagnetic side-channel signals," Journal of Hardware and Systems Security, vol. 3, 2019.

[17] J. Selvaraj et al., "Electromagnetic induction attacks against embedded systems," in Proceedings of the 2018 on Asia Conference on Computer and Communications Security. 2018.

[18] A. Bogdanov et al., "Differential cache-collision timing attacks on AES with applications to embedded CPUs," in Topics in Cryptology-CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings. Springer Berlin Heidelberg, 2010.

[19] T. Hanawa et al., "Low-power and high-performance communication mechanism for dependable embedded systems," in 2008 International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems. IEEE, 2008.

[20] S. Singha et al., "A study on power optimization techniques in psoc," in 2019 International Conference on Communication and Signal Processing (ICCSP). IEEE, 2019.

[21] A. Minaie, J. H. Gosling, and R. Sanati-Mehrizy, "Availability and Structure of Security in Embedded Systems," in 2017 ASEE Annual Conference & Exposition. 2017.

[22] M. Brown, "Five Steps to Improving Security in Embedded Systems," Contemporary Technologies in Automation, vol. 3, 2013.

[23] X. Yang and J. H. Andrian, "A low-cost and high-performance embedded system architecture and an evaluation methodology," in 2014 IEEE Computer Society Annual Symposium on VLSI. IEEE, 2014.

[24] A. K. Singh et al., "Learning-based run-time power and energy management of multi/many-core systems: Current and future trends," Journal of Low Power Electronics, vol. 13, no. 3, 2017.

[25] A. K. Singh et al., "Mapping on multi/many-core systems: Survey of current and emerging trends," in Proceedings of the 50th Annual Design Automation Conference, 2013.

[26] M. Gupta, "Security challenges and cryptography in embedded systems," International Journal of Advance Research in Science and Engineering, vol. 4, no. 1, 2015.

[27] J. Chen, H. Guo, and W. Hu, "Research on improving network security of embedded system," in

2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom). IEEE, 2019.

[28] G. Meng, "Discussion on internet security strategy of embedded system," in 2010 2nd International Conference on Computer Engineering and Technology. Vol. 3. IEEE, 2010.

[29] M. Barr and N. Jones, "High Performance Embedded Systems," in Barr group, 2016.

[30] M. Gupta, "Security challenges and cryptography in embedded systems," International Journal of Advance Research in Science and Engineering, vol. 4, no. 1, 2015.

[31] S. Jajodia, P. Samarati, and M. Yung, "Encyclopedia of Cryptography, Security and Privacy," 2019.

[32] M. Karlsson and O. Zaja, "Improving Security in Embedded Systems with IEEE 802.1 X.", 2021.

[33] I. Polian and M. Stöttinger, Eds., Constructive Side-Channel Analysis and Secure Design: 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3–5, 2019, Proceedings, vol. 11421. Springer, 2019.

[34] J. Bucek and M. Novotný, "Differential power analysis under constrained budget: Low-cost education of hackers," in 2013 Euromicro Conference on Digital System Design. IEEE, 2013.

[35] A. Dubey, R. Cammarota, and A. Aysu, "Maskednet: The first hardware inference engine aiming power side-channel protection," in 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 2020.

[36] Q. Ge et al., "A survey of microarchitectural timing attacks and countermeasures on contemporary hardware," Journal of Cryptographic Engineering, vol. 8, 2018.

[37] J. Huang and X. Li, "Cache-collision side-channel analysis and attacks against AES-GCM," International Journal of Big Data Intelligence, vol. 7, no. 4, 2020.

[38] F. E. Potestad-Ordóñez et al., "Hardware countermeasures benchmarking against fault attacks," Applied Sciences, vol. 12, no. 5, 2022.

[39] R. J. Masti et al., "Thermal covert channels on multi-core platforms," in 24th USENIX security symposium (USENIX security 15), 2015.

[40] X. Lou et al., "A survey of microarchitectural side-channel vulnerabilities, attacks, and Defences in cryptography," ACM Computing Surveys (CSUR), vol. 54, no. 6, 2021.

[41] F. Abid et al., "Towards an open embedded system on chip for network applications," in Proceedings of the 4th international conference on Circuits, systems and signals, 2010.

[42] N. Kumar and M. Rattan, "Implementation of embedded RISC processor with dynamic power management for low-power embedded system on SOC," in 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS). IEEE, 2015.

[43] V. Kartha and V. Barwatkar, "A Review Paper on Low Power Embedded System Design,"

Research Gate., 2017.

[44] M. Duranton, "The challenges for high performance embedded systems," in 9th EUROMICRO Conference on Digital System Design (DSD'06). IEEE, 2006.

[45] M. Malewski, D. M. J. Cowell, and S. Freear, "Review of battery powered embedded systems design for mission-critical low-power applications," International Journal of Electronics, vol. 105, no. 6, 2018.

[46] H. Huang et al., "On countermeasures against the thermal covert channel attacks targeting many-core systems," in 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020.

[47] M. Alagappan et al., "DFS covert channels on multi-core platforms," in 2017 IFIP/IEEE International Conference on Very Large-Scale Integration (VLSI-SoC). IEEE, 2017.

[48] J. Wang et al., "Combating enhanced thermal covert channel in multi-/many-core systems with channel-aware jamming," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, 2020.

[49] H. Okhravi, S. Bak, and S. T. King, "Design, implementation and evaluation of covert channel attacks," in 2010 IEEE international conference on technologies for homeland security (HST). IEEE, 2010.

[50] S. Zander, P. Branch, and G. Armitage, "Capacity of temperature-based covert channels," IEEE communications letters, vol. 15, no. 1, 2010.

[51] T. Vateva-Gurova and N. Suri, "On the Detection of Side-Channel Attacks," in 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC), Taipei, Taiwan, 2018

[52] M. A. Elsadig and Y. A. Fadlalla, "Packet length covert channel: A detection scheme," in 2018 1st International Conference on Computer Applications & Information Security (ICCAIS). IEEE, 2018.

[53] A. Barenghi et al., "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," Proceedings of the IEEE, vol. 100, no. 11, 2012.

[54] R. Kumar et al., "Parametric trojans for fault-injection attacks on cryptographic hardware," in 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography. IEEE, 2014.

[55] K. Murdock et al., "Plundervolt: Software-based fault injection attacks against Intel SGX," in 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020.

[56] Y. Liu et al., "Fault injection attack on deep neural network," in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2017.

[57] M. Wu et al., "Eliminating timing side-channel leaks using program repair," in Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2018.

[58] W. Schindler, "Exclusive exponent blinding may not suffice to prevent timing attacks on RSA,"

in Cryptographic Hardware and Embedded Systems--CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings. Springer Berlin Heidelberg, 2015.

[59] P. Rohatgi, "Electromagnetic attacks and countermeasures," in Cryptographic Engineering, 2009.

[60] C. Wang et al., "Electromagnetic equalizer: An active countermeasure against EM side-channel attack," in 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE, 2018.

[61] H. Huang et al., "Detection of and countermeasure against thermal covert channel in many-core systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 2, 2021.

[62] X. Lou et al., "A survey of microarchitectural side-channel vulnerabilities, attacks, and Defences in cryptography," ACM Computing Surveys (CSUR), vol. 54, no. 6, 2021.

[63] R. J. Masti et al., "Thermal Covert Channels on Multi-core Platforms," arXiv e-prints, 2015.

[64] M. Alagappan et al., "DFS covert channels on multi-core platforms," in 2017 IFIP/IEEE International Conference on Very Large-Scale Integration (VLSI-SoC). IEEE, 2017.

[65] R. Paccagnella et al., "Lord of the ring (s): Side channel attacks on the CPU On-Chip ring interconnect are practical," in 30th USENIX Security Symposium (USENIX Security 21), 2021.

[66] J. Wang et al., "Combating enhanced thermal covert channel in multi-/many-core systems with channel-aware jamming," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, 2020.

[67] R. G. Gebremedhin and T. L. Marzetta, "Thermal Conduction as a Wireless Communication Channel," in GLOBECOM 2022-2022 IEEE Global Communications Conference. IEEE, 2022.

[68] S. Chen et al., "Detecting privileged side-channel attacks in shielded execution with Déjá Vu," in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017.

[69] Z. Long et al., "Improving the efficiency of thermal covert channels in multi-/many-core systems," in 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2018.

[70] N. Tuptuk and S. Hailes, "Covert channel attacks in pervasive computing," in 2015 IEEE international conference on pervasive computing and communications (PerCom). IEEE, 2015.

[71] Z. Long et al., "Improving the efficiency of thermal covert channels in multi-/many-core systems," in 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2018.

[72] D. B. Bartolini et al., "On the capacity of thermal covert channels in multicores," in Proceedings of the Eleventh European Conference on Computer Systems, 2016.

[73] K. Cao et al., "A survey of optimization techniques for thermal-aware 3D processors," Journal of Systems Architecture, vol. 97, 2019.

[74] Z. Wu et al., "Whispers in the hyper-space: high-bandwidth and reliable covert channel attacks

inside the cloud," IEEE/ACM Transactions on Networking, vol. 23, no. 2, 2014.

[75] G. Costa et al., "Covert channel attack to federated learning systems," arXiv preprint arXiv:2104.10561, 2021.

[76] B. W. Lampson, "A Note on the Confinement Problem," in Communications of the ACM, Oct 10, 1973.

[77] C. Reis et al., "Browser Security: Lessons from Google Chrome: Google Chrome developers focused on three key problems to shield the browser from attacks," Queue, vol. 7, no. 5, 2009.

[78] T. E. Carlson et al., "An evaluation of high-level mechanistic core models," ACM Transactions on Architecture and Code Optimization (TACO), vol. 11, no. 3, 2014.

[79] A. Pathania and J. Henkel, "HotSniper: Sniper-based toolchain for many-core thermal simulations in open systems," IEEE Embedded Systems Letters, vol. 11, no. 2, 2018.

[80] R. Chiş et al., "Multi-objective optimization for an enhanced multi-core SNIPER simulator," in Proc. Rom. Acad.-Ser. A., vol. 19, 2018.

[81] A. Aljuffri et al., "Applying thermal side-channel attacks on asymmetric cryptography," IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 29, no. 11, 2021.

[82] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature aware DVFS," IEEE Transactions on Computers, vol. 59, no. 1, 2010.

[83] A. Amouri, J. Hepp, and M. Tahoori, "Built-in self-heating thermal testing of FPGAs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 9, 2015.

[84] F. Gao et al., "Achieving a covert channel over an open blockchain network," IEEE network, vol. 34, no. 2, 2020.

[85] J. Chen and B. C. Schafer, "Thermal fingerprinting of FPGA designs through high-level synthesis," in Proceedings of the 2019 on Great Lakes Symposium on VLSI, 2019.

[86] C. Ramesh et al., "FPGA side channel attacks without physical access," in 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, 2018.

[87] T. Claeys et al., "Thermal covert channel in Bluetooth low energy networks," in Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, 2019.

[88] M. Hutter and J.-M. Schmidt, "The temperature side channel and heating fault attacks," in Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers 12. Springer International Publishing, 2014.

[89] S. Chen et al., "Thermal covert channels leveraging package-on-package DRAM," in 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2019.

[90] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations," Future Generation Computer Systems, vol. 96, 2019.

[91] T. Kim and Y. Shin, "ThermalBleed: A practical thermal side-channel attack," IEEE Access, vol. 10, 2022.

[92] F. Raffaelli et al., "A homodyne detector integrated onto a photonic chip for measuring quantum states and generating random numbers," Quantum Science and Technology, vol. 3, no. 2, 2018.

[93] V. Caprara Vivoli et al., "Comparing different approaches for generating random numbers device-independently using a photon pair source," New Journal of Physics, vol. 17, no. 2, 2015.

[94] T. Schmidbauer and S. Wendzel, "Sok: A survey of indirect network-level covert channels," in Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, 2022.

[95] H. Jiang et al., "A 2-in-1 temperature and humidity sensor with a single FLL wheatstone-bridge front-end," IEEE Journal of Solid-State Circuits, vol. 55, no. 8, 2020.

[96] P. K. Chundi et al., "Hotspot monitoring and temperature estimation with miniature on-chip temperature sensors," in 2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED). IEEE, 2017.

[97] K. Murdock et al., "Plundervolt: Software-based fault injection attacks against Intel SGX," in 2020 IEEE Symposium on Security and Privacy (SP). IEEE, 2020.

[98] L. Deshotels, "Inaudible sound as a covert channel in mobile devices," in 8th USENIX Workshop on Offensive Technologies (WOOT 14). 2014.

[99] D. R. E. Gnad et al., "Voltage-based covert channels using FPGAs," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 26, no. 6, 2021.

[100] F. Yao, G. Venkataramani, and M. Doroslovački, "Covert timing channels exploiting non-uniform memory access-based architectures," in Proceedings of the on Great Lakes Symposium on VLSI 2017. 2017.

[101] X. Zhang et al., "A covert channel over volte via adjusting silence periods," IEEE Access,

vol. 6, 2018.

[102] M. Guri et al., "An optical covert-channel to leak data through an airgap," in 2016 14th annual conference on privacy, security and trust (pst). IEEE, 2016.

[103] W. Liu et al., "Modeling of visible light channel based on matrix reconstruction," in Fourth International Conference on Wireless and Optical Communications. SPIE, 2016.

[104] M. Guri, "Magneto: Covert channel between air-gapped systems and nearby smartphones via CPU-generated magnetic fields," Future Generation Computer Systems, vol. 115, 2021.

[105] S. Pan and K. A. Makinwa, "A 0.25 mm 2-Resistor-Based Temperature Sensor with an Inaccuracy of 0.12° C (3σ) From− 55° C to 125° C," IEEE Journal of Solid-State Circuits, vol. 53, no. 12, 2018.

[106] M. Guri and D. Bykhovsky, "Air-jumper: Covert air-gap exfiltration/infiltration via security cameras & infrared (IR)," Computers & Security, vol. 82, 2019.

[107] P. Rahimi et al., "Trends and challenges in ensuring security for low-power and high-performance embedded SoCs," in 2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). IEEE, 2021.

[108] C. Su and Q. Zeng, "Survey of CPU cache-based side-channel attacks: systematic analysis, security models, and countermeasures," Security and Communication Networks, 2021.

[109] L. Bossuet, "Dvfs as a security failure of trustzone-enabled heterogeneous SoC," in 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2018.

[110] O. Darwish et al., "Towards a streaming approach to the mitigation of covert timing channels," in 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC). IEEE, 2018.

[111] Z. Shao, M. A. Islam, and S. Ren, "Heat behind the meter: A hidden threat of thermal attacks in edge colocation data centers," in 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2021.

[112] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs," in Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. 2019.

[113] P. Rahimi et al., "Selective noise based power-efficient and effective countermeasure against thermal covert channel attacks in multi-core systems," Journal of Low Power Electronics and Applications, vol. 12, no. 2, 2022.

[114] R. CHIŞ et al., "Multi-objective optimization for an enhanced multi-core SNIPER

simulator," Proc. Rom. Acad.-Ser. A., vol. 19, 2018.

[115] I. Miketic, K. Dhananjay, and E. Salman, "Covert Channel Communication as an Emerging Security Threat in 2.5 D/3D Integrated Systems," Sensors, vol. 23, no. 4, 2023.

[116] R. Spreitzer et al., "Systematic classification of side-channel attacks: A case study for mobile devices," IEEE Communications Surveys & Tutorials, vol. 20, no. 1, 2017.

[117] T. Meng et al., "A secure and cost-efficient offloading policy for mobile cloud computing against timing attacks," Pervasive and Mobile Computing, vol. 45, 2018.

[118] J.-F. Dhem, F. Koeune, and P. A. Leroux, "A Practical Implementation of the Timing Attack [C]," Springer. 1998.

[119] C. Rebeiro et al., "An introduction to timing attacks," in Timing Channels in Cryptography: A Micro-Architectural Perspective, 2015.

[120] N. Shrivastava and S. R. Sarangi, "Towards an optimal countermeasure for cache side-channel attacks," IEEE Embedded Systems Letters, 2022.

[121] X. Lou et al., "A survey of microarchitectural side-channel vulnerabilities, attacks, and Defences in cryptography," ACM Computing Surveys (CSUR), vol. 54, no. 6, 2021.

[122] A. K. Biswas et al., "A survey of timing channels and countermeasures," ACM Computing Surveys (CSUR), vol. 50, no. 1, 2017.

[123] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and Defences," Journal of Hardware and Systems Security, vol. 3, no. 3, 2019.

[124] Z. Chen et al., "Blockchain meets covert communication: A survey," IEEE Communications Surveys & Tutorials, 2022.

[125] J. Betz et al., "Survey on covert channels in virtual machines and cloud computing," Transactions on Emerging Telecommunications Technologies, vol. 28, no. 6, 2017.

[126] L. Caviglione, "Trends and challenges in network covert channels countermeasures," Applied Sciences, vol. 11, no. 4, 2021.

[127] Y. Tan et al., "Covert timing channels for IoT over mobile networks," IEEE Wireless Communications, vol. 25, no. 6, 2018.

[128] E. Karimi et al., "A timing side-channel attack on a mobile GPU," in 2018 IEEE 36th International Conference on Computer Design (ICCD). IEEE, 2018.

[129] A. Muñoz et al., "A survey on the (in) security of trusted execution environments," Computers & Security, vol. 129, 2023.

[130] C. Reinbrecht et al., "Cache timing attacks on NoC-based MPSoCs," Microprocessors and Microsystems, vol. 66, 2019.

[131] D. J. Bernstein, "Cache-timing attacks on AES," 2005.

[132] Z. H. Jiang et al., "A complete key recovery timing attack on a GPU," in 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2016.

[133] R. Hund et al., "Practical timing side channel attacks against kernel space ASLR," in 2013 IEEE Symposium on Security and Privacy. IEEE, 2013.

[134] S. Deldari et al., "Time series change point detection with self-supervised contrastive predictive coding," in Proceedings of the Web Conference 2021. 2021.

[135] C. Fanjas et al., "Real-Time Frequency Detection to Synchronize Fault Injection on System-on-Chip," Cryptology ePrint Archive, 2022.

[136] G. Kadam et al., "Rcoal: mitigating GPU timing attack via subwarp-based randomized coalescing techniques," in 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2018.

[137] D. G. Mahmoud et al., "Electrical-level attacks on CPUs, FPGAs, and GPUs: Survey and implications in the heterogeneous era," ACM Computing Surveys, vol. 55, no. 3, 2022.

[138] M. Xu et al., "Toward engineering a secure Android ecosystem: A survey of existing techniques," ACM Computing Surveys (CSUR), vol. 49, no. 2, 2016.

[139] D. Cotroneo et al., "Timing covert channel analysis of the VxWorks MILS embedded hypervisor under the common criteria security certification," Computers & Security, vol. 106, 2021.

[140] X. Liang and Y. Kim, "A survey on security attacks and solutions in the IoT network," in 2021 IEEE 11th annual computing and communication workshop and conference (CCWC). IEEE, 2021.

[141] D. Gruss et al., "Cache template attacks: Automating attacks on inclusive Last-Level caches," in 24th USENIX Security Symposium (USENIX Security 15). 2015.

[142] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and Defences," Journal of Hardware and Systems Security, vol. 3, no. 3, 2019.

[143] J. Tian et al., "A survey of key technologies for constructing network covert channel," Security and Communication Networks, 2020.

[144] C. Alcaraz et al., "Covert channels-based stealth attacks in industry 4.0," IEEE Systems Journal, vol. 13, no. 4, 2019.

[145] A. Purnal and I. Verbauwhede, "Advanced profiling for probabilistic Prime+ Probe attacks and covert channels in ScatterCache," arXiv preprint arXiv:1908.03383, 2019.

[146] K. Dhananjay et al., "High Bandwidth Thermal Covert Channel in 3-D-Integrated Multicore Processors," IEEE Transactions on Very Large Scale Integration Systems, vol. 30, no. 11, 2022.

[147] Z. Li et al., "SpiralSpy: Exploring a stealthy and practical covert channel to attack air-gapped

computing devices via mmWave sensing," in Proc. NDSS, 2022.

[148] I. Giechaskiel et al., "Cross-vm covert-and side-channel attacks in cloud fpgas," ACM Transactions on Reconfigurable Technology and Systems, vol. 16, no. 1, 2022.

[149] Q. Hao et al., "A hardware security-monitoring architecture based on data integrity and control flow integrity for embedded systems," Applied Sciences, vol. 12, no. 15, 2022.

[150] F. Boutekkouk, "A literature review on security-aware design space exploration approaches for embedded systems," International Journal of Security and Networks, vol. 17, no. 4, 2022.

[151] N. Moghadasi et al., "Research and development priorities for security of embedded hardware devices," IEEE Transactions on Engineering Management, 2022.

[152] A. Sghaier et al., "Fast Constant-Time Modular Inversion over F p Resistant to Simple Power Analysis Attacks for IoT Applications," Sensors, vol. 22, no. 7, 2022.

[153] L. Parrilla et al., "Time-and Amplitude-Controlled Power Noise Generator against SPA Attacks for FPGA-Based IoT Devices," Journal of Low Power Electronics and Applications, vol. 12, no. 3, 2022.

[154] J. Zhu et al., "Adversarial Examples Detection of Electromagnetic Signal Based on GAN," in 8th International Conference on Big Data Computing and Communications (BigCom). IEEE, 2022.

[155] Y. Zhang and K. Rasmussen, "Detection of Electromagnetic Signal Injection Attacks on Actuator Systems," in Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defences, 2022.

[156] G. Haas and A. Aysu, "Apple vs. EMA: electromagnetic side channel attacks on apple CoreCrypto," in Proceedings of the 59th ACM/IEEE Design Automation Conference, 2022.

[157] A. Berthelier et al., "Deep model compression and architecture optimization for embedded systems: A survey," Journal of Signal Processing Systems, vol. 93, 2021.

[158] A. Erbsen et al., "Integration verification across software and hardware for a simple embedded system," in Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, 2021.

[159] C. E. S. Santos et al., "Multi-objective adaptive differential evolution for SVM/SVR hyperparameters selection," Pattern Recognition, vol. 110, 2021.

[160] P.-H. Thevenon et al., "iMRC: Integrated Monitoring & Recovery Component, a Solution to Guarantee the Security of Embedded Systems," J. Internet Serv. Inf. Secur., vol. 12, no. 2, 2022.

[161] J. Martins et al., "Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems," in Workshop on next generation real-time embedded systems (NG-RES 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[162] Z. A. Al-Sharif et al., "Live forensics of software attacks on cyber–physical systems," Future

Generation Computer Systems, vol. 108, 2020.

[163] H. Polat et al., "Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models," Sustainability, vol. 12, no. 3, 2020.

[164] K. Papagiannopoulos et al., "The side-channel metrics cheat sheet," ACM Computing Surveys, vol. 55, no. 10, 2023.

[165] J. Park et al., "Anomaly detection in embedded systems using power and memory side channels," in 2020 IEEE European Test Symposium (ETS). IEEE, 2020.

[166] U. Weinrib et al., "Countermeasure against fault injection attacks," U.S. Patent Application No. 17/853,612, 2023.

[167] P. Kiaei et al., "Rewrite to reinforce: Rewriting the binary to apply countermeasures against fault injection," in 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE, 2021.

[168] J. Richter-Brockmann et al., "Fiver–robust verification of countermeasures against fault injections," IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021.

[169] A. Gangolli et al., "A systematic review of fault injection attacks on IoT systems," Electronics, vol. 11, no. 13, 2023.

[170] S. Delarea and Y. Oren, "Practical, low-cost fault injection attacks on personal smart devices," Applied Sciences, vol. 12, no. 1, 2022.

[171] B. Selmke et al., "On the application of two-photon absorption for laser fault injection attacks: pushing the physical boundaries for laser-based fault injection," IACR Transactions on Cryptographic Hardware and Embedded Systems, 2022.

[172] P. Rahimi et al., "Fan Speed Control Based Defence for Thermal Covert Channel Attacks in Multi-Core Systems," in 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2022.

[173] J. Tian et al., "A survey of key technologies for constructing network covert channel," Security and Communication Networks, 2020.

[174] H. Li and D. Chasaki, "Network-based machine learning detection of covert channel attacks on cyber-physical systems," in 2022 IEEE 20th International Conference on Industrial Informatics (INDIN). IEEE, 2022.

[175] S. Soderi and R. De Nicola, "CONNECTION: COvert chaNnel NEtwork attaCk Through bIt-rate mOdulatioN," in International Symposium on Emerging Information Security and Applications. Singapore: Springer Nature Singapore, 2023.

[176] C. Meadows, "Predicting Asymptotic Behaviour of Network Covert Channels: Experimental Results," in Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, 2022.

[177] C. Meadows, "Predicting Asymptotic Behaviour of Network Covert Channels: Experimental Results," in Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, 2022.

[178] S. Al-Eidi et al., "Covert timing channel analysis either as cyber attacks or confidential applications," Sensors, vol. 20, no. 8, pp. 2417, 2020..