Through Fabric: A Cross-world Thermal Covert Channel on TEE-enhanced FPGA-MPSoC Systems

Hassan Nassar*[1], Jeferson Gonzalez-Gomez*[1,2], Varun Manjunath, Lars Bauer, Jörg Henkel[1]
[1] *Karlsruhe Institute of Technology (KIT), Chair for Embedded Systems (CES)*
[1] {hassan.nassar, jeferson.gonzalez, henkel}@kit.edu
[2] *Instituto Tecnológico de Costa Rica (TEC)*

ABSTRACT

The ever-evolving computing landscape gets more complex in every moment and the need for heterogeneous compute systems becomes more relevant. As the usability of such systems grew, finding methods for securing them became more relevant. Commercial vendors already introduced Trusted Execution Environments (TEEs) for those systems. TEEs serve the need for isolation, where sensitive data are processed in a *secure world*, and non-trusted applications are executed in the *normal world*. In this paper, we introduce Through Fabric: a novel attack against TEE-enhanced FPGA-MPSoCs. We show that existing benign hardware accelerators can be manipulated from the secure world to implement a temperature-based covert channel. We successfully run this attack on a commercial FPGA-MPSoC within the OP-TEE environment without additional access rights. We use an open-source implementation of AES for the accelerator and we reach a transmission speed of 2 bits per second with bit error rate of 1.9% and packet error rate of 4.3%. We are the first to show that a TEE can be bypassed on FPGA-MPSoCs via temperature-based covert channel communication.

1 INTRODUCTION

Computers and smart devices are a cornerstone of the modern era. From payments, to entertainment systems, and even to medical services, they all rely on some computing systems processing our sensitive data. Trusted Execution Environments (TEEs) aim to create isolation for applications handling sensitive data. It virtually separates the computing resources between a secure world and a normal world. Such a TEE can be implemented through hardware [1] and software [2]. However, attackers are continuously finding new methods to leak data even with TEEs employed.

One of the methods used by the attackers is covert channel communication between two attackers. Usually, one acts as a transmitter, and the other as a receiver with the transmitter usually sending sensitive data to the receiver. Several demonstrations of covert channel communications exist, e.g., thermal covert channels [3] or power covert channels [4].

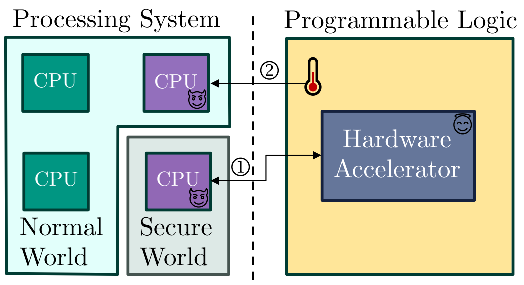


Figure 1: Overview of our new cross-device thermal covert channel on a TEE-enhanced FPGA-MPSoC

FPGAs are a prominent component of the computing landscape. With the ability to change the implemented hardware at run-time, they offer an interesting option to accelerate a variety of applications. Several systems now include FPGAs as part of the compute infrastructure alongside the multiprocessor system-on-a-chip (MPSoC), e.g., Zynq-MPSoCs [5].

Previous works tackled the establishment of covert channels on FPGAs [4, 6, 7]. These works, assume (i) an FPGA-MPSoC running bare metal without any TEE mechanism existing and (ii) that they are able to implement malicious hardware for the transmitter and receiver, e.g., ring oscillators. These two assumptions are, however, not very realistic, as TEE can be easily established on FPGA-MPSoCs, e.g., OP-TEE [2]. Furthermore, malicious hardware is easily detected and banned on FPGA-MPSoCs [8]. In contrast to them, our work establishes a thermal covert channel on FPGA-MPSoCs without these two assumptions.

In a simplified view, the system we target is depicted in Fig. 1. A colluding application (e.g., from a dishonest vendor) running in the secure world uses a benign AES accelerator from the FPGA, the so-called Programmable Logic (PL), to transmit messages by modulating the temperature of the PL ①. The malicious receiver running in the normal world reads the temperature sensor of the PL ②, which is accessible from the normal world, to decode the messages. In this way, we are able to communicate information between CPUs executing applications in a cross-world fashion, employing the temperature of the PL in the FPGA-MPSoC as the means for communication.

Our **contributions** in this work are the **following**:

- We propose a novel thermal covert channel (TCC) that leverages benign hardware accelerators to effectively transmit information stealthily between two CPUs through the FPGA.

*Both authors contributed equally.

This work was partially funded by the "Helmholtz Pilot Program for Core Informatics (kikit)" at Karlsruhe Institute of Technology and by the German Federal Ministry of Education and Research (BMBF) through grant 01IS23066 as part of the Software Campus Projects "HE-Trust" and "VERITAS". This work was supported in part by the Deutsche Akademische Austauschdienst (DAAD).

ASPDAC '25, January 20–23, 2025, Tokyo, Japan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0635-6/25/01.
https://doi.org/10.1145/3658617.3697767

- We practically show how the isolation principle of a TEE-enhanced FPGA-MPSoC is broken by effectively communicating sensitive information from the secure world to the normal world via TCC.
- We propose countermeasures for this new attack while discussing the implications of such techniques both in the attack's effectiveness and the system's performance.

2 BACKGROUND AND RELATED WORK

2.1 Trusted Execution Environments (TEEs)

A TEE is a secure and tamper-resistant processing environment, which guarantees the authenticity and integrity of the system's dynamic state as well as code and data confidentiality [9]. ARM TrustZone [1] provides hardware support for several TEEs on ARM-based systems. It ensures an isolated execution context for security-critical applications by virtually separating the hardware and software resources into two spaces: the *secure world* or TEE and the *normal world* or Rich Execution Environment (REE) [10], e.g., Linux. This hardware-supported virtual separation allows computing and memory-shared resources to be private to each environment, providing the isolation mechanism needed for the TEE.

OP-TEE is a TrustZone implementation for Cortex-A processors, designed to isolate normal world applications Client Applications (CAs) and the operating system from secure world applications Trusted Applications (TAs) [2]. The isolation principle TEEs is effective for integrity and confidentiality against many attacks but does not address covert and side channels [11].

2.2 Thermal Convert Channels

Thermal covert channels (TCCs) have greatly evolved since their early implementation in multi-core systems [12]. More complex modulation and encoding schemes have been implemented to reduce detection, increase the transmission speed, and reduce error rates in recent years [13–15], achieving bandwidths of up to 300 bps with significantly low error rates between $1-11\%$ [16]. Recent approaches for TCCs have exploited other devices besides the main processor, such as the GPU [17] and memory subsystems [3], showing the feasibility and diversity of these attacks.

Thermal covert channels for FPGA-only systems have been explored. In [18], communication between electrically separated hardware modules was achieved through temperature variations using ring oscillators. A similar method was proposed in [19] for cloud-based FPGAs, modulating the temperature with custom hardware logic. Both methods target pure-FPGA systems using malicious hardware to generate heat. **In contrast**, our solution uses FPGAs for communication between CPUs in FPGA-MPSoCs, without needing malicious hardware, instead utilizing a benign hardware accelerator.

2.3 FPGA-based Covert Channels

Covert channels have been vastly demonstrated for FPGA-based systems over the recent years. The power distribution network has mainly been the exploited mechanism to implement the channels in multi-tenant FPGAs and FPGA-SoCs. In this context, the attacks have targeted the device's voltage [4, 7, 20] and frequency [6, 21] as the medium to modulate the power, obtaining fast transmission speeds and low error rates. However, these approaches rely on malicious custom hardware modules on the transmitter side (e.g., ring oscillators), which can be easily detected, restricted by some vendors or even stopped during run-time [8]. Moreover, the receiver modules in these approaches also rely on additional FPGA logic, where a time-to-digital coverter (TDC) is often required to measure the power in the system. Our solution, on the other hand, modulates the temperature of the FPGA logic. For this, it does not require malicious hardware on the transmitter side nor custom logic on the receiver side, as it leverages existing thermal sensors on FPGA-SoCs. Other approaches for covert channels use non-conventional means to modulate different resources on FPGA such as PCIe usage for cloud systems [22] or internal wiring [23] in multi-tenant FPGA systems.

3 THROUGH FABRIC: FPGA-MPSOC-TCC

3.1 Threat Model and Assumptions

The basic threat model for our FPGA-MPSoC-based attack follows the same principle as other covert channels [10], where a malware and a spy application communicate with each other in an illegitimate way.

The spy or *receiver* application, runs in the normal world as a CA. On the other hand, we assume a malware *transmitter*, which is a malicious or colluding application that gets triggered by an unaware innocent application to perform a security-critical operation on its behalf, following the same model as other TEE-focused covert channels [10, 24]. In the context of TEEs, this colluding program is a Trusted Application (TA) that executes in isolation in the secure world through the OP-TEE framework, while the triggering program is a Client Application (CA) in the normal world. In a practical scenario, this colluding application could be provided by a dishonest vendor, or a vulnerable TA could be exploited by an attacker. The spy or *receiver* application runs in the normal world as a CA belonging to the same or even potentially a different user. Neither the *transmitter* nor the *receiver* requires any privileged permissions to perform any of the operations involved in the attack. A real example of this type of colluding application is the recent *xz utils* vulnerability [25], where through an elaborate supply chain attack, an adversary inserted a backdoor into *xz utils*, a widely used open-source library, and almost succeeded in merging the malicious updates into major Linux distributions before its detection.

Differently from other FPGA-based approaches in the state-of-the-art, we assume that the hardware accelerator used to modulate the temperature of the PL is completely benign. Moreover, neither the transmitter nor the receiver modules of the attacker require to implement any extra hardware on the FPGA logic.

3.2 System Overview

3.2.1 Software. A simplified overview of the software components of the system is depicted as a flow in Fig. 2. In the normal world, the innocent CA intends to perform a hardware-accelerated AES decryption on a ciphertext through the decryption TA, which resides in the secure world. To do so, the innocent CA uses the OP-TEE client API ① to invoke the AES decryption TA, unaware of its malicious nature. In turn, the OP-TEE client API routes the request to the OP-TEE driver ② in the Linuxkernel. On the Linux kernel side of the normal world, the OP-TEE driver then directs the request to

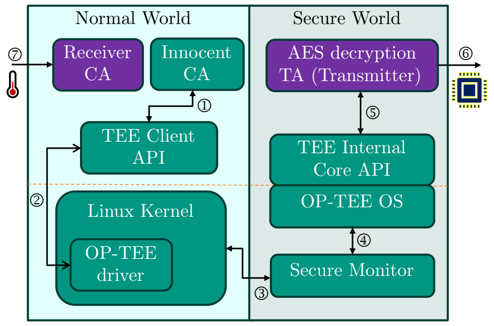


Figure 2: Overview of the software components of the system

the secure monitor ③ on the secure world, which itself handles the communication between worlds by routing the request to the OP-TEE Trusted Operating System ④. Through the internal API, the OP-TEE OS framework determines the malicious AES decryption TA as the one being invoked and passes the control to it to handle the request ⑤. Finally, the TA uses the hardware accelerator API ⑥ to perform the decryption. In a normal (benign) operation, at this point, the execution control would return in a reversed path back to the CA with the ciphertext being decrypted. However, because of its malicious nature, before returning control, the TA leverages the hardware accelerator **again** ⑥ to encode and leak secret data (e.g., the key or plaintext) by modulating the temperature of the programmable logic on the FPGA.

Notably, the TA is configured to keep the execution context (instance) after the sessions ends, using the `TA_FLAG_INSTANCE_KEEP_ALIVE` flag [10]. This allows the TA to store, and further leak, private data after the transaction has finished.

In the normal world, another malicious application (i.e., the receiver CA, potentially owned by a different user) continually reads the temperature sensor of the FPGA ⑦ to detect the beginning of the transmission and decode the secret being sent by the TA, hence establishing an illegitimate communication channel between the secure and the normal world.

3.2.2 *Hardware.*

To demonstrate our attack, we employ an existing hardware accelerator as the heating mechanism. As Fig. 1 shows, the processor residing in the trusted world is connected to one accelerator on the PL. The connection is done via an AXI crossbar. We use the benign AES 128 bit decryption engine from [26] as the accelerator, as it is available as open source and as it is highly parallelized. The AES engine receives the cipher text and decryption key as input from the processor and returns the plain text in one clock cycle.

Notably, while a custom and more power hungry hardware accelerator (e.g., ring oscillators) would benefit the transmission by heating faster, depending on the attacker to compromise the hardware or implement their own logic could be either be impractical or easy to detect. We decided to employ a completely benign module, with a realistic use case. The selected AES engine performs a security-related operation, which justifies its use from the secure world, while also being a tested device provided by an honest vendor.

3.3 Transmitter

The transmitter module of the attack is implemented within the malicious decryption TA. Algorithm 1 shows the implementation logic for the transmission. Upon being invoked (⑤ in Fig. 2), the TA performs the normal (benign) decryption of the ciphertext. However, as a malicious application, the TA proceeds to leak the newly decrypted plaintext by modulating the temperature of the FPGA. To do so, it first computes the number of decryptions needed to encode a bit of '1', and the time it requires to wait to encode a bit of '0', using the desired bit rate and the latency accelerator. Then for each bit in the secret, the TA performs the extra decryptions for each bit value that is a '1', or waits for the corresponding time for each bit of '0'. Finally, the application returns the plaintext normally to the calling CA.

In the case of a long secret, in order to avoid suspiciously long decryption delays, the transmitter module can leverage the `TA_FLAG_INSTANCE_KEEP_ALIVE` flag, as described in Section 3.2.1, to save it, while only leaking a few bytes at a time per call, especially with short ciphertext decryptions. On further calls, the attacker can obfuscate the transmission of the rest of the message by leveraging the decryption of longer ciphertexts.

Algorithm 1: TA transmitter for the TCC

```
Input: ciphertext: encrypted text from innocent CA
Result: plaintext: decrypted text
1  plaintext ← HwAESDecrypt(ciphertext);    /* Performs
     normal decryption */
2  N ← $1/(bit\_rate * latency\_acc)$ ;     /* Calculate the
     required number of decryptions to heat up
     enough to encode a '1' */
3  tdown ← $1/(2 * bit\_rate)$;
4  for bit in secret do
5      if bit is 1 then
6          for i = 0 to N-1 do
7              HwAESDecrypt(randominput) ;    /* Perform
                 extra decryptions to increase
                 temperature */
8          end
9      else
10         TEE_Wait(tdown) ; /* Sleeps to cool down the
             hardware */
11     end
12 end
13 return plaintext;
```

3.4 Receiver

The receiver is implemented as an application running in the normal world. It performs three simple steps which are shown as the three loops in Alg. 2. The first step (line 2 to line 6) is to continuously collect the data from the PL temperature sensor and store it in an array of size *total_samples*. This step as involves only reading data from a register. Once it collected the samples, it filters out the data to the desired frequency of communication (line 7 to line 11).

Algorithm 2: Normal world receiver for the TCC

```
Result: $msg$: demodulated message
1  $s \leftarrow 0$;
2  while $s < total\_samples$ do
3      $temps[s] \leftarrow readTemp()$ ; /* Get new temperature reading */
4      $sleep(sampling\_time)$;
5      i++;
6  end
7  $N \leftarrow total\_samples - win\_size$ ; /* Compute number of moving windows to decode */
8  for i in N do
9      $X \leftarrow FFT(temps[i : i + win\_size])$; /* Computes FFT of the moving window */
10     $filtered[i] \leftarrow X[\omega_k]$ ; /* Filter the data at the frequency index $\omega_k$ */
11 end
12 for j in N do
13     $bit_w \leftarrow filtered[(j * win\_size : j * win\_size + win\_size]$ ; /* get the demodulated bit window */
14     if $max(bit_w) > \rho_1$ then
15         $msg[j] \leftarrow 1$ ; /* if high absolute change then bit is '1' */
16     else
17         if $max(bit_w) < \rho_2$ then
18             $msg[j] \leftarrow 0$ ; /* if low absolute change then bit is '0' */
19         else
20             if $|slope(bit_w)| > \delta_H$ then
21                 $msg[j] \leftarrow not(msg[j-1])$ ; /* if high absolute change in slope then bit flipped */
22             else
23                 $msg[j] \leftarrow msg[j-1]$ ; /* if low absolute change in slope then bit stayed the same */
24             end
25         end
26     end
27 end
28 return $msg$;
```

The receiver performs this by calculating the Fast Fourier Transform (FFT) over a moving window and keeping only the bins that correspond to the frequency of communication.

The final step is to decode the filtered data into the corresponding sent bits (line 12 to line 27). To achieve this, the receiver applies two criteria: an absolute value and a gradient. During the moving window corresponding to each bit, if a value higher than a pre-computed high threshold (ρ_1) is achieved then a bit value '1' is interpreted. Similarly, if the maximum value is lower than the pre-computed low threshold (ρ_2) then a bit value '0' is interpreted. However, when multiple bits of the same value are sent sequentially, then the temperature saturates to a value in between. To solve this, we compute the gradient between two consecutive samples as a moving slope. If the slope of the readings from the moving window is greater than the pre-compute threshold for a high slope δ_H, it means that a rapid change of temperature occurred. Consequently, a bit with value opposite to the previously received bit is transmitted, so we toggle the value based on the last received bit. Otherwise, if the slope is low, it means that the same value is received and no toggling occurs.

To set the thresholds ρ_1, ρ_2, and δ_H we use a subset of the sent data and analyze it. Each bit is sent over a period T and an interval of temperature samples t is recorded. We collect the temperature recorded for the '1' bits in dataset t_1. Similarly, the temperatures for '0' bits are collected in dataset t_0. The high threshold (ρ_1) is set as $\rho_1 = \mu(max(t_1)) - \sigma(max(t_1))$ with μ being average and σ being standard deviation. For the low threshold (ρ_2), we set it as $\rho_2 = \mu(max(t_0)) + \sigma(max(t_0))$. Finally, for the slope threshold δ_H, we collect t_g which is the dataset of any bits that are of opposite value than the previous bit. We then calculate δ_H as $\delta_H = \mu(max(t_g) - min(t_g))/T$.

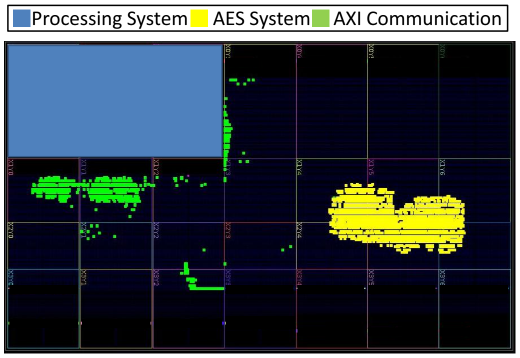


Figure 3: Floor plan of the implemented system on the ZCU102 UltraScale+ FPGA evaluation board

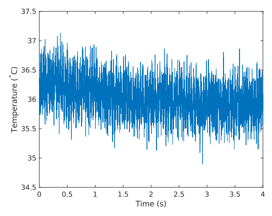
4 EVALUATION

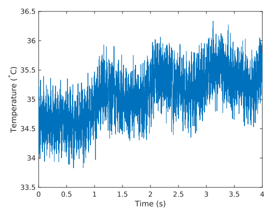
4.1 Evaluation Platform

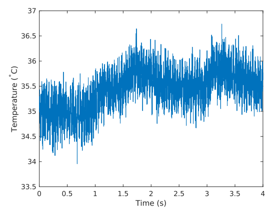
We run our experiments on a ZCU102 FPGA-MPSoC [5] which is supported by OP-TEE. The design is implemented using Vivado 2018. The MPSoC contains a quad-core ARM CORTEX A-53 and a PL with 200k LUTs. The AES accelerator is open source from [26]. It uses 11k LUTs and the PS to PL AXI interface uses 3.1k LUTs, at a clock frequency of 100MHz. The normal world operating system is a custom Linux distribution built using the PetaLinux SDK from Xilinx. Figure 3 shows the implemented system on the FPGA.

4.2 Transmitter & Receiver

By modulating the usage of the AES accelerator, the transmitter creates changes in the temperature profile of the PL. This is apparent from Fig. 4, as with sending each different packet of 8-bits the PL temperature profile is different. However, the temperature is not only affected by the usage of the accelerator. For example, the ambient temperature and different noise sources cause the temperature to fluctuate. Therefore, even with the different temperature profiles, it is necessary that the receiver apply further filtering.


(a) Unfiltered "0000_0000"


(b) Unfiltered "1010_1010"


(c) Unfiltered "1100_1000"

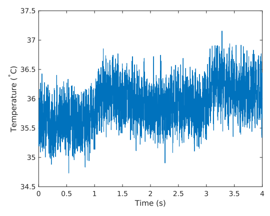
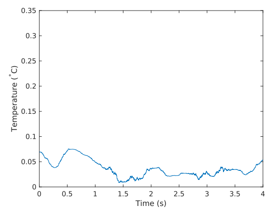
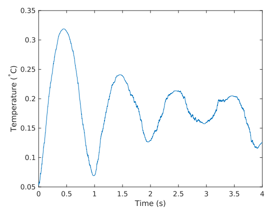
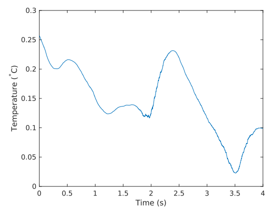

(d) Unfiltered "1000_1000"

Figure 4: Temperature of the PL based on sending the different 8-bit messages. Each message has a slightly different profile.


(a) Filtered "0000_0000"


(b) Filtered "1010_1010"


(c) Filtered "1100_1000"

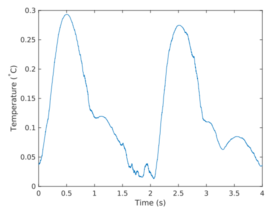

(d) Filtered "1000_1000"

Figure 5: Data received by the receiver after applying filtering from Alg. 2. It can be seen that '0' bits and '1' bits are differentiable.

On the receiver side, after applying Alg. 2, the receiver is able to filter out all the effects from other sources. Figure 5 shows the same four packets after applying the filtering at the transmit frequency. For sending all zeroes, the temperature stays at a very low level. For alternating between '0' and '1', the peaks are very distinct. When mixing zeroes and ones in a series they are still differentiable using the latter part of Alg. 2.

4.3 Channel metrics

In order to evaluate the effectiveness of the thermal covert channel (TCC), we run the compromised Trusted Application (TA), which performs several consecutive descriptions using the AES accelerator, in order to send 8,000 bits encoded in 8-bit packets on the FPGA-MPSoC board. Table 1 shows the result metrics for our new cross-world thermal covert channel from this experiment including bit error rate (BER), packet error rate (PER), and transmission rate. As it can be seen, the channel is effective under the tested scenario, achieving a transmission rate of 2 bps, which is on par with similar TCCs on non-CPU devices [17, 18]. Moreover, our attack was able to produce very low error rates, in a similar range to other state-of-the-art approaches for thermal covert channels (1-11%) [16]. Notably, since the attack is performed within the OP-TEE environment, its effectiveness shows how the isolation and data confidentially principles of the TEE have been effectively broken by our attack.

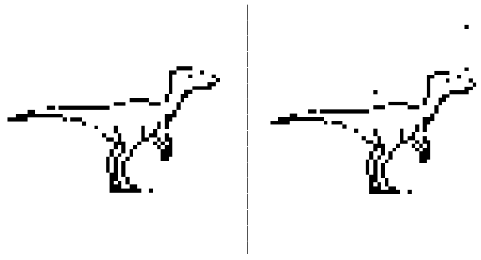


Figure 6: Representation of the channel performance when transmitting a binary image. The image on the left was sent, the image on the right was received.

Table 1: Thermal covert channel evaluation metrics

Bits	Packets	Transmission rate (bps)	BER (%)	PER (%)
8000	1000	2	1.9	4.3

To visually represent the performance of the covert communication, we sent a 64x64 pixel binary image with our thermal cover channel. Figure 6 shows the sent and received images from this test. In this experiment, the obtained BER was less than 2%, which further shows the applicability of the channel.

4.4 Comparison to state of the art

As mentioned in Section 2, other works exploited covert channels on FPGAs before. However, as Table 2 shows, our work is distinct from them in several ways. First, our work is the first to exploit a temperature-based covert channel between CPUs using the FPGA. The second distinction is that our work neither requires special malicious hardware for the transmitter nor for the receiver. Finally, none of the related works showed that they were able to break TEE on FPGAs.

5 COUNTERMEASURES

Countermeasures for thermal covert channels have often aimed to control the thermal response of the affected component (e.g., CPU and GPU), employing mechanisms such as Dynamic Voltage and Frequency Scaling (DVFS) [15, 17]. Although such solutions might be implemented in FPGA-based designs, their scope is more limited in this domain, since the frequency/voltage of the PL is not necessarily modifiable at run-time, especially without affecting the functionality of other hardware modules implemented in the FPGA. Hence, we discuss further techniques, while highlighting their possible drawbacks in the system.

Table 2: Comparison to related works. Our work does not need any malicious hardware on the transmitter or receiver side.

Work	Requires Mal. (HW) Transmitter	Requires Mal. (HW) Receiver	Covert Channel	Break TEE
Our work	✗	✗	temperature	✓
Ref. [4]	✓	✓	voltage	✗
Ref. [6]	✓	✓	frequency	✗
Ref. [7]	✗	✓	voltage	✗
Ref. [20]	✓	✓	voltage	✗
Ref. [21]	✗	✓	frequency	✗
Ref. [22]	✗	✗	PCIe	✗
Ref. [23]	✓	✓	inter. wiring	✗

5.1 Noise

A naive solution for covert channel mitigation is creating noise, specifically aimed at the channel transmission frequency. A direct way of creating the noise would be by leveraging the accelerator the same way the attacker does, by starting a periodic application to modulate the temperature of the PL. Noise-based solutions have been proposed for TCC before [15, 17]. An issue with this technique is that it normally comes with high overhead in terms of energy and performance for the system.

5.2 Application-Specific Delay

Similar to the classical use for memory contention on spinning processors [27], an increasing delay penalty can be applied to a TA using the accelerator with an abnormal frequency of requests. In this case, after the application has invoked the API method to use the accelerator, before queuing the request, the method checks the number of times this particular application has requested the accelerator in a period of time. If the number of times is bigger than the threshold for normal use, then a delay is added to the request. Since the transmitter requires to use the accelerator multiple times to encode the bits as temperature variations, each use would last increasingly longer, disturbing the timings of the channel. A simple yet flexible function to obtain the delay for a number of requests per second (n) can be described as follows:

$$\text{delay}(n) = \min\left(\alpha \cdot (n - \text{threshold})^{\beta},\ \text{max\_delay}\right)$$

where α is a scaling factor that determines the severity of the penalty, β determines how sharply the penalty increases, and the threshold is the number of requests considered as normal use. We employ a *max_delay* limit as an upper boundary for the dealy. Figure 7 shows the delay for different configurations of α and β, with an threshold of 5 requests per second and a *max_delay* of 30 seconds. As it can seen, this function can be used to enforce different degrees of delay penalty depending on the frequency of the requests to the particular accelerator.

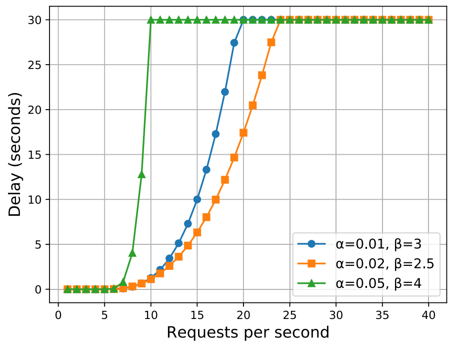


Figure 7: Delay added to the acceleration request for different number of requests per second from the same application.

6 DISCUSSION

We show and validate how a thermal covert channel can be implemented to communicate sensitive information from the secure world to the normal world, using benign accelerators. Although the principle of the attack is simple, we believe the relevance of this approach relies on how the attack effectively breaks the isolation and data privacy principles of a TEE to covertly leak data. Moreover, from an attacker's perspective, its implementation simplicity makes it an attractive medium for extracting victim's information from the secure enclave.

One issue that should be addressed is the delay that it could produce to decryption requests by the unaware user. Certainly, if this time is high enough, it would raise suspicions. However, as briefly mentioned in Section 3.3, it should be noted that a decryption is rarely done over a single 128 bit value. Usually, a ciphertext is large and requires several sequential descriptions. For example, an image can easily reach tens of MBs in size. Decrypting it, including fetching it from the memory would take a relatively long time that can be leveraged to obfuscate the transmission and reduce delays. Moreover, although not technically complex, without the disclosure of this new attack to the public, there is currently no reason to monitor the timing patterns of the accelerator request or the FPGA's temperature with fine granularity Hence, the attack would not be noticeable in current systems.

Although the proposed countermeasures might seem simple at first glance, mitigating the attack, while minimizing the performance, power, and energy impact becomes a challenge. Moreover, without accurate attack detectors, any countermeasure renders impractical as no commercial vendor would penalize its own system without reason. Due to the recent surge in covert channels, we believe more attention should be put on effective yet efficient detection and countermeasure mechanisms for such threats.

7 CONCLUSION

In this paper, we proposed a **new thermal covert channel** that leverages the usage of a **benign hardware accelerator** in the programmable logic of an FPGA-MPSoC-based system to establish illegitimate communication **between two CPUs**. Our new temperature-based channel has proven to be **effective**, achieving a transmission rate of up to 2 bps, with very low bit and packet error rates. Moreover, we **implemented the attack in the OP-TEE framework** as the Trusted Execution Environment (TEE), showing how the **isolation and data confidentiality of TEE is broken** by our approach. Finally, we **proposed and discussed countermeasures** against accelerator-based covert channels.

REFERENCES

[1] ARM, "Trustzone for Cortex-A", 2023, 04.09.23. [Online]. Available: https://www.arm.com/technologies/trustzone-for-cortex-a

[2] TrustedFirmware, "About OP-TEE", 04.09.23. [Online]. Available: https://optee.readthedocs.io/en/latest/general/about.html

[3] S. Chen, W. Xiong, Y. Xu, B. Li, and J. Szefer, "Thermal covert channels leveraging package-on-package DRAM", in *IEEE (TrustCom/BigDataSE)*, 2019.

[4] D. R. E. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels using FPGAs", *ACM TODAES*, 2021.

[5] *ZCU102 Evaluation Board User Guide*, Xilinx, 2023.

[6] L. Bossuet and C. A. Lara-Nino, "Advanced covert-channels in modern SoCs", in *HOST*, 2023.

[7] M. Gross, R. Kunzelmann, and G. Sigl, "CPU to FPGA power covert channel in FPGA-SoCs", Cryptology ePrint Archive, Paper 2023/429, 2023.

[8] H. Nassar, H. AlZughbi, D. Gnad, L. Bauer, M. Tahoori, and J. Henkel, "LoopBreaker: Disabling interconnects to mitigate voltage-based attacks in multi-tenant FPGAs", in *ICCAD*, 2021.

[9] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: What it is, and what it is not", in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015.

[10] N. Mishra, A. Chakraborty, U. Chatterjee, and D. Mukhopadhyay, "Time's a thief of memory", in *Smart Card Research and Advanced Applications*, I. Buhan and T. Schneider, Eds., 2023.

[11] X. Lou, T. Zhang, J. Jiang, and Y. Zhang, "A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography", *ACM Computing Surveys*, 2021.

[12] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms", in *USENIX Security*, 2015.

[13] Z. Long, X. Wang, Y. Jiang, G. Cui, L. Zhang, and T. Mak, "Improving the efficiency of thermal covert channels in multi-/many-core systems", in *DATE*, 2018.

[14] D. B. Bartolini, P. Miedl, and L. Thiele, "On the capacity of thermal covert channels in multicores", in *EuroSys*, 2016.

[15] H. Huang, X. Wang, Y. Jiang, A. K. Singh, M. Yang, and L. Huang, "On countermeasures against the thermal covert channel attacks targeting many-core systems", in *DAC*, 2020.

[16] I. Miketic, K. Dhananjay, and E. Salman, "Covert channel communication as an emerging security threat in 2.5d/3d integrated systems", *Sensors*, 2023.

[17] J. González-Gómez, K. Cordero-Zuñiga, L. Bauer, and J. Henkel, "The first concept and real-world deployment of a GPU-based thermal covert channel: Attack and countermeasures", in *DATE*, 2023.

[18] T. Iakymchuk, M. Nikodem, and K. Kępa, "Temperature-based covert channel in FPGA systems", in *ReCoSoC*, 2011.

[19] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs", in *FPGA*, 2019.

[20] I. Giechaskiel, K. Rasmussen, and J. Szefer, "Reading between the dies: Cross-SLR covert channels on multi-tenant cloud FPGAs", in *ICCD*, 2019.

[21] A. Fellah-Touta, L. Bossuet, and C. A. Lara-Nino, "Combined internal attacks on SoC-FPGAs: Breaking AES with remote power analysis and frequency-based covert channels", in *EuroS&PW*, 2023.

[22] I. Giechaskiel, S. Tian, and J. Szefer, "Cross-VM covert- and side-channel attacks in cloud FPGAs", *ACM TRETS*, 2022.

[23] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, "Leakier wires: Exploiting FPGA long wires for covert- and side-channel attacks", *ACM TRETS*, 2019.

[24] H. Cho, P. Zhang, D. Kim, J. Park, C.-H. Lee, Z. Zhao, A. Doupé, and G.-J. Ahn, "Prime+count: Novel cross-world covert channels on ARM TrustZone", in *Annual Computer Security Applications Conference*, 2018.

[25] NIST, "CVE-2024-3094 detail", Mar 2024. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2024-3094

[26] P. Maene and I. Verbauwhede, "Single-cycle implementations of block ciphers", in *Lightweight Cryptography for Security and Privacy*, 2016.

[27] T. E. Anderson, "The performance of spin lock alternatives for shared-memory multiprocessors." *IEEE TPDS*, 1990.