Enhanced Detection of Thermal Covert Channel Attacks in Multicore Processors

Krithika Dhananjay
Stony Brook University
Stony Brook, NY, USA
krithika.yethiraj@stonybrook.edu

Vasilis F. Pavlidis
Aristotle University of Thessaloniki
Thessaloniki, Greece
vpavlid@ece.auth.gr

Ayse K. Coskun
Boston University
Boston, MA, USA
acoskun@bu.edu

Emre Salman
Stony Brook University
Stony Brook, NY, USA
emre.salman@stonybrook.edu

***Abstract*—Information leakage through temperature based covert channels is a growing threat in modern multicore processors. Thus, accurately detecting the presence of such thermal covert communication channels in real time is crucial for ensuring the security of confidential data. Existing detection techniques fail when covert channels are implemented with low power benchmarks, as shown in this paper. A novel detection technique is proposed by considering the transitions in the CPU workload as the primary metric. The proposed approach can detect covert channels established with low power programs with 100% detection accuracy and less than 2% false positive rate.**

I. Introduction

Confidential data such as bank passwords, personal contacts and medical records often need to be legitimately transferred between two computing hardware elements [1]. To prevent the disclosure of information, this type of communication is protected using strict security protocols. Nevertheless, malicious attackers find ways to bypass these security controls and exfiltrate the sensitive information. Covert channel communication [2]–[4] is one such attack, where a compromised sender process modifies properties of a computing hardware, such as execution time [5], shared processor cache states [6], sound produced by mobile devices [7] and processor core temperature [8] to transmit secret information to a compromised receiver process.

Recently, covert channel communication using heat as a carrier, referred to as thermal covert channel (TCC) communication, has gained considerable attention [9]–[11]. Masti *et al.* identified TCC as a dangerous leakage source between individual cores of a multicore processor [8]. Specifically, to send a bit '1' in a TCC, a program is executed in the transmitting core to raise its temperature and to transmit a bit '0', the execution of the program is stopped, thereby reducing its temperature, as shown in Fig. 1. The receiving core is a neighboring core that has similar changes in its temperature profile due to the thermal coupling between the two cores. In modern processors, temperature information from the thermal sensors is available to user applications without the need for special permissions [12]. Therefore, by processing its temperature sensor output, the transmitted bits can be decoded by the receiving core. Following Masti *et al.*, researchers have studied TCC modeling [9], [12], detection [11], [13] and mitigation techniques [10], [13], [14]. Furthermore, some studies identify techniques that increase communication throughput with minimal error rates [12], [15], [16].

Since TCC attacks do not need physically shared resources, run time detection techniques are needed to determine their presence before blocking the attack. Huang *et al.* recently proposed two techniques for TCC detection [11], [13]. The first technique analyses the temperature spectrum of each core to detect a TCC [13]. This technique is relatively inefficient because of the need to scan frequencies and filter out the noise components at each frequency step. Therefore, the instruction per cycle (IPC) based detection was proposed [11]. Both of these techniques assume that the covert channel is established by executing compute intensive programs to sufficiently increase the temperature of the source core. As shown in previous work [16], it is possible to establish a high bandwidth TCC by executing low power benchmarks in densely integrated multicore processors where there is strong thermal crosstalk. Since the existing detection techniques fail to consider such a scenario, in this paper, we propose a novel detection metric to detect a high bandwidth TCC established with low power programs. Additionally, the proposed metric eliminates the need to scan the frequency spectrum at multiple frequency steps. The primary contributions of this work are as follows:

- We propose a novel detection technique that leverages the transitions in the CPU workload (measured in terms of the differences in giga instructions per second, referred to as $\Delta GIPS$) to detect a TCC established by low power programs. The detection is achieved by scanning the power spectral density (PSD) of the $\Delta GIPS$ profile of each core.
- The threshold for detection is statistically determined based on exhaustive execution of typical SPLASH-2/PARSEC benchmark applications without a TCC attack.

Our results demonstrate that the proposed technique can detect 100% of the TCCs at the pre-determined threshold with negligible ($<2\%$) false positive rate for five different low power programs from SPLASH-2/PARSEC benchmark suites.

The organization of the rest of the paper is as follows. A brief background on TCC attack model, framework, and communication protocol is provided in Section II. Existing TCC detection techniques and their drawbacks are described in Section III. The proposed TCC detection technique is detailed in Section IV. The results are presented in Section V. Finally, the paper is concluded in Section VI.

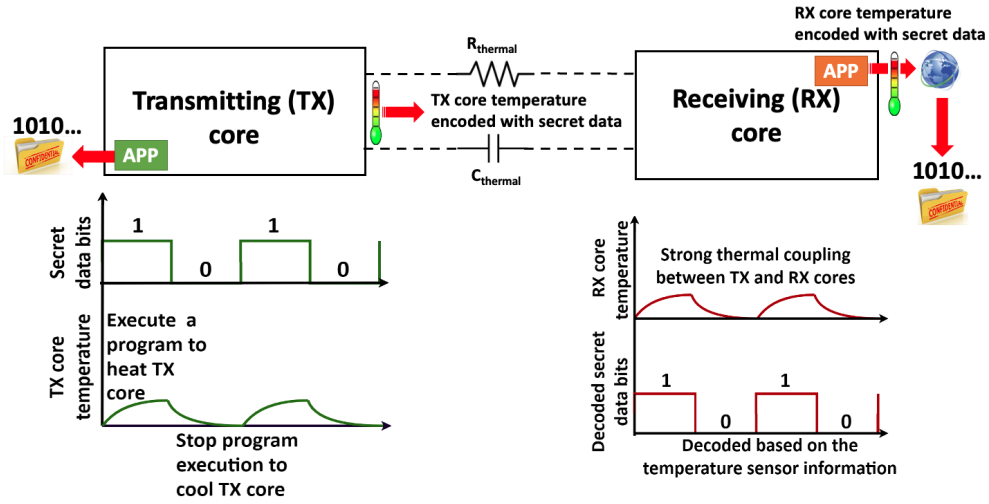


Fig. 1. Thermal covert channel communication in multicore processors.

II. Background

In this section, we briefly discuss the threat model of the TCC attack, followed by the communication architecture (including communication protocol), encoding process within the transmitting core, and offline decoding technique.

A. Threat model

In our threat model, the transmitting core is in a secure zone with access to confidential information whereas the receiving core is in a low-security zone and is looking to exfiltrate the confidential information. Technologies such as Intel SGX [17] and Arm Trustzone [18] provide special security enclaves to software applications (apps) that handle sensitive information. We assume that the app on the transmitting core is protected by such technologies. However, research has shown that these enclaves can also be compromised by malwares [11]. In order to secretly transmit the sensitive data to the receiver core sitting in the low-security zone, the malware program is executed by the attacker to raise the power consumption (and therefore the temperature of the transmitting core) to send a bit '1'. Similarly to send a bit '0', the program execution is stopped and hence the transmitting core is cooled down, as shown in Fig. 1. Because of the thermal coupling between the cores of a processor, the temperature of the adjacent cores also has a similar variation. Therefore, the attacker app in the receiving core can decode the secret data by reading its temperature sensor information and processing it, as illustrated in the figure.

B. Communication Architecture

The TCC communication protocol, communication encoding and decoding processes are briefly discussed in this subsection.

1) Communication protocol: The malware program in the transmitting core reads the sensitive data and breaks it into several blocks or frames with the same number of data bits. Therefore, even if a part of the transmitted data bits are lost in communication, the fragmentation of the entire secret data enables successful communication for the remaining blocks. The program appends a stream of preamble bits to each block of data to enable synchronization between the transmitting and the receiving cores. In this work, we use a series of alternating '1's and '0's as preamble bits.

The temperature profile of the transmitting core is encoded with the secret information by controlling the execution of the malware program, as described in Section II-A. The app on the receiving core continuously monitors the temperature profile information by reading the temperature sensor. Because of the thermal coupling between the transmitting and the receiving cores, the secret data bits can be decoded offline by processing the temperature profile of the receiving core.

2) Encoding process: The secret data bits are encoded on the temperature profile of the transmitting core by varying its power consumption. A simple amplitude shift keying (ASK) modulation technique is used to create the modulated power profile [19]. Typically, a compute intensive program is executed to sufficiently increase the temperature of the transmitting core and therefore reduce the error rates of communication [8], [11], [12], [15]. If the secret data bits are represented by the vector $v[n]$, where $0 < n < N$ for N bits to be transmitted per block, and $p(t)$ is the power consumption of the transmitting core when the malware program is executed for bit-width (T_b) amount of time, the modulated power consumption of the transmitting core, $p_m(t)$, encoded with the secret data bits is given as,

$$p_m(t) = \sum_{n=1}^{N} v[n] \times p(t - nT_b). \quad (1)$$

Since the temperature of the transmitting core is proportional to its power consumption, the secret data bits are also encoded on the temperature profile of the transmitting core.

3) Offline decoding process: Due to the thermal coupling between the transmitting and the receiving cores, the secret data bits can be decoded by analyzing the temperature profile of the receiving core. The attacker app in the receiving core continuously records the information from the temperature sensor,

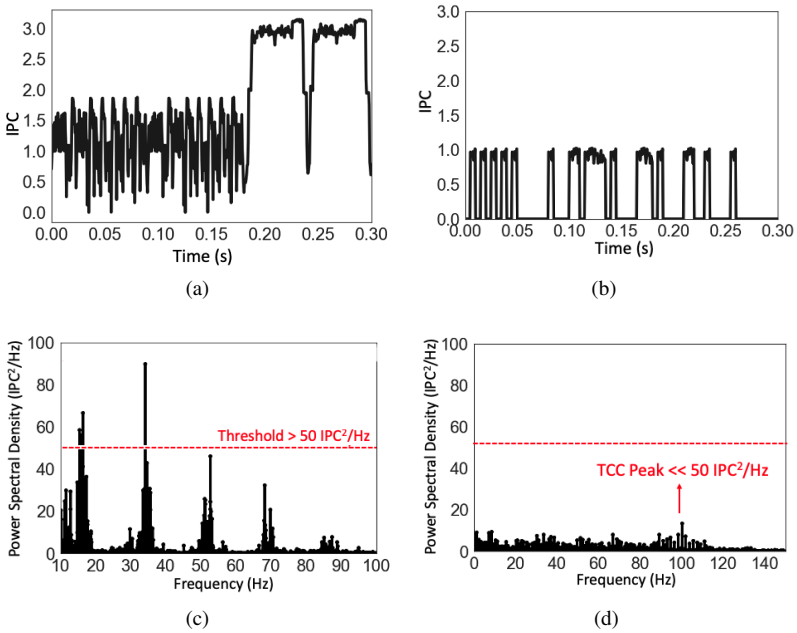


Fig. 2. IPC profiles showing (a) time domain IPC data of a core sequentially executing random applications from SPLASH-2 and PARSEC suites, (b) time domain IPC data of a core executing *raytrace* program encoded with the secret data, (c) power spectral density of IPC data for a core sequentially executing random applications from SPLASH-2 and PARSEC suites and (d) power spectral density of IPC data for a core executing *raytrace* program encoded with the secret data.

which is sampled at every T_b. The sensor resolution and refresh rates must be considered while decoding the temperature sensor output. Temperature sensors in modern processors often have a resolution (referred to as S) of approximately 1°C [20], [21], as assumed in this work. The secret data bits are decoded by comparing the cumulative rise and fall of the temperature with S [16].

III. Limitations of Existing TCC Detection Techniques

Huang *et al.* recently proposed two techniques for TCC detection [11], [13]. The first technique analyzes the temperature profile of each core in the frequency domain [13]. Specifically, power spectrum of temperature profile of each core is scanned at high frequencies and compared against a fixed threshold to detect the presence of a TCC. However, this frequency scanning technique requires the use of band pass filter at several frequency steps, thereby increasing the overhead in each detection cycle. More importantly, relying on temperature as the metric causes the detection to be very slow since temperature is a low frequency signal. Thus, once the covert channel is detected, it is possible that some of the secret information is already transmitted.

The second approach analyzes the frequency spectrum of the CPU workload of each logical core to detect a TCC [11]. The CPU workload is measured in terms of the instructions per cycle (IPC), which varies much faster than temperature. In this technique, the power spectral density of IPC for each core is obtained and the maximum amplitude of the spectrum is compared against a set threshold to determine if a TCC is present. Note that the power consumption of benchmark applications from SPLASH-2 or PARSEC suites occupy a low frequency band of approximately 0-10 Hz. Therefore, a TCC attack is typically established at greater frequencies to avoid interference [15]. Thus, the above detection technique focuses on a frequency range from 10 Hz up to 500 Hz. Unfortunately, this technique fails to detect TCC attacks established with low power applications. This limitation is illustrated in Fig. 2. Figs. 2(a) and (c) show, respectively, the time domain and frequency domain IPC spectrum of an Intel Haswell processor core when there is no covert channel. The core executes random benchmark applications from the SPLASH-2 and PARSEC suite sequentially. Figs. 2(b) and (d) depict, respectively, the time domain and frequency domain IPC spectrum of the same core executing a *raytrace* based TCC program. Thus, in Figs. 2(b) and (d), an attacker establishes a covert channel attack by using *raytrace* program, which is one of the lowest power apps in the benchmark suite. From the time domain IPC data, it is observed that the peak IPC of the processor core with *raytrace* based TCC [Fig. 2(b)] is lower than the peak IPC of the core without TCC [Fig. 2(a)]. Similarly, in the frequency domain, it is observed that the peak power spectral density (PSD) of IPC without TCC [Fig. 2(c)] is greater than the peak PSD of the IPC with TCC [Fig. 2(d)], even in the range of 10 Hz to 100 Hz. In [11], the threshold for detection is set to be greater than the average IPC without TCC. For example, according to Fig. 2(c), the threshold should be greater than the average peaks of 50 IPC^2/Hz. However, the PSD of the IPC spectrum with TCC in Fig. 2(c) is much lower than 50 IPC^2/Hz. Therefore, this type of TCC cannot be detected by analyzing only the IPC metric of each core. The following section describes an enhanced detection technique that overcomes this drawback.

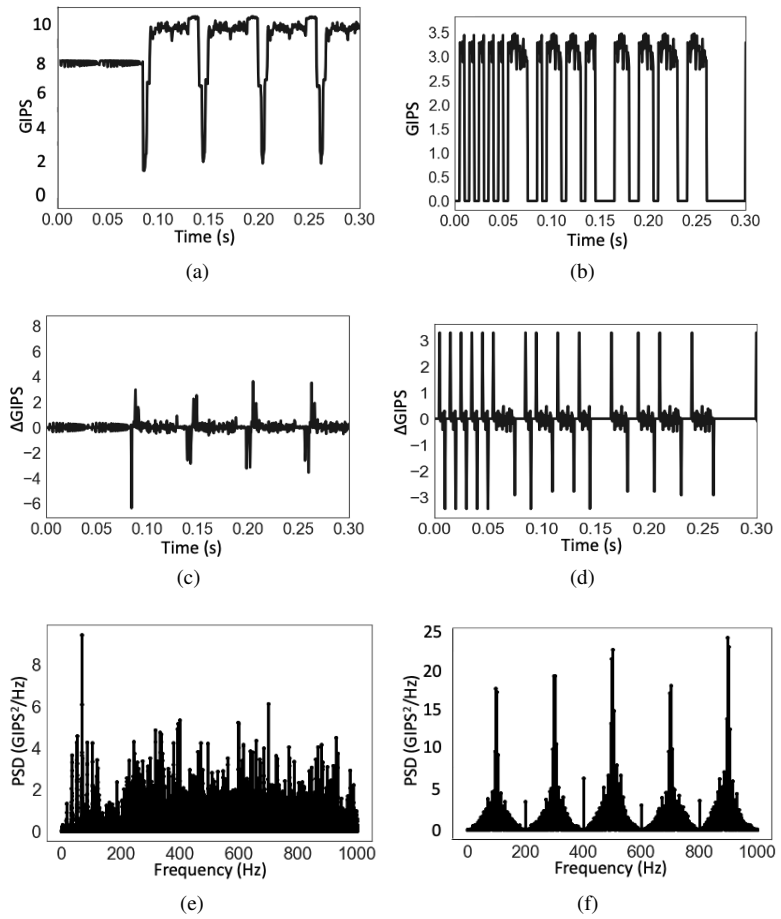


Fig. 3. $GIPS$ profiles showing (a) time domain $GIPS$ data of a core sequentially executing random applications from SPLASH-2 and PARSEC suites, (b) time domain $GIPS$ data of a core executing *raytrace* program encoded with the secret data, (c) time domain $\Delta GIPS$ data of a core sequentially executing random applications from SPLASH-2 and PARSEC suites, (d) time domain $\Delta GIPS$ data of a core executing *raytrace* program encoded with the secret data, (e) power spectral density of $\Delta GIPS$ of a core sequentially executing random applications from SPLASH-2 and PARSEC suites, and (f) power spectral density of $\Delta GIPS$ of a core executing *raytrace* program encoded with the secret data.

IV. Proposed Technique for Detecting Low-power and High Bandwidth TCC

A. Enhanced detection metric

In this section, we propose two enhancements to the existing techniques to detect a TCC attack established by a low power program. First, we use the metric giga instructions per second (GIPS), calculated as $GIPS = IPC \times f$, where f is the frequency of the processor. The reason behind adopting the GIPS is that IPC of a processor is independent of the processor frequency. In a scenario where an attacker chooses to increase the frequency of the transmitting core [22] to raise its temperature to transmit a bit '1', the IPC metric alone falls short of detecting this situation. Therefore, we rely on the metric GIPS. However, GIPS is only a scaled version of IPC and the peak power spectral density of GIPS without TCC is still greater than the peak power spectral density with TCC for a *raytrace* based attack, as discussed in Section III. Therefore, in this work, we propose to leverage the *transitions* in the GIPS profile to detect the TCC. Specifically, a GIPS spectrum with TCC has significant variations in its magnitude because of the frequent turning ON (for encoding a bit '1') and OFF (for encoding a bit '0') of the application. Therefore, we propose to calculate the difference between the consecutive samples of GIPS, referred to as $\Delta GIPS$, to quantify these variations. Thus, the proposed metric for a particular cycle c is defined as,

$$(\Delta GIPS)_c = GIPS\Big|_{c=m}^{c=m+1}, \tag{2}$$

where the GIPS of two consecutive cycles are subtracted. This metric is then quantified as a function of the number of cycles. For example, Figs. 3(a) and (b) depict the GIPS data of the processor core without TCC and with TCC at 100 Hz, respectively. The corresponding $\Delta GIPS$ data is calculated according to (2) and plotted in Fig. 3(c) for the core without TCC. Similarly, $\Delta GIPS$ plot for the core with TCC is shown in Fig. 3(d). Please note that even though the absolute magnitude of GIPS is larger in Fig. 3(a) [and thus resulting in higher peaks in the $\Delta GIPS$ plot shown in Fig. 3(c)], the occurrence of the peaks is random and less frequent compared to Fig. 3(d) where the occurrence of the peaks is consistent and more frequent. Specifically, Fig. 3(d) represents a Dirac comb function [23] and the frequency spectrum of this type of Dirac comb function is also a Dirac comb with peaks at odd

multiples of the fundamental frequency (100 Hz), as illustrated in Fig. 3(f). Thus, these peaks can be used to detect a TCC attack. To capture this effect, we propose to calculate the sum of these peaks to determine if the resultant magnitude is greater than the peak value in the power spectrum without TCC in Fig. 3(e). This algorithm is described in the following section.

B. Detection algorithm

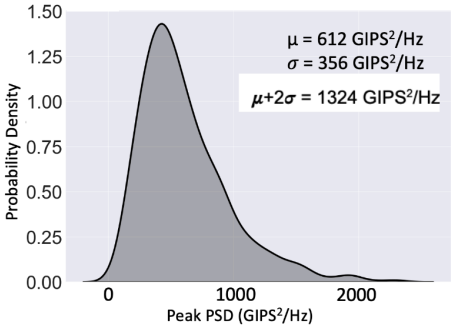
The TCC detection algorithm is shown in Alg. 1. We assume that TCC detection program can be configured for the secure cores. Based on the configuration, this algorithm is executed as an asynchronous thread on these cores that have access to secure information. The detection cycle is assumed to execute 10 times in every second. In each detection cycle, each secure core extracts the GIPS data and calculates the frequency spectrum of $\Delta GIPS$ for that core, as shown in line 4 of Alg. 1. If the frequency response represents a Dirac comb function $[C(f)]$, as discussed in the previous section, the sum of all the peaks in F is stored in $P_{\Delta GIPS_i}$ (see lines 5 to 9). If $P_{\Delta GIPS_i}$ crosses a pre-determined threshold, a $detect$ flag is set, as shown in lines 10 to 12. The threshold is statistically calculated based on exhaustive set of simulations of common applications from SPLASH-2/PARSEC suite, as described in the following subsection.

Algorithm 1 $\Delta GIPS$ based TCC detection

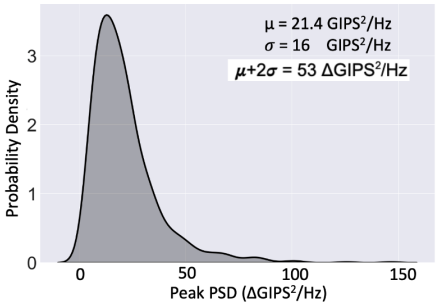
1: **Inputs:** $\Delta GIPS_i, T_{det}, N_{cores}$
2: **Initialization:** $f = 10 : 10 : 1000Hz$
3: **for** $1 \leq i \leq N_{cores}$ **do**
4: $\mathcal{F} \leftarrow$ Fast Fourier Transform (FFT) spectrum of $\Delta GIPS_i$
5: **if** $F \in$ C(f) **then**
6: $P_{\Delta GIPS_i} \leftarrow$ Sum of all peaks in $\mathcal{F}$
7: **else**
8: $P_{\Delta GIPS_i} \leftarrow$ Peak in $\mathcal{F}$
9: **end if**
10: **if** $P_{\Delta GIPS_i} > T_{det}$ **then**
11: $detect_i \leftarrow 1$
12: **end if**
13: **end for**

C. Threshold determination

In existing IPC based detection technique [11], the threshold for detection is estimated based on the average of the IPC power spectrum for SPLASH-2/PARSEC applications when there is no covert channel. The variations in the power spectrum for different applications and its impact on threshold are not considered extensively. Thus, in this work, we sequentially execute random applications from SPLASH-2/PARSEC suite in a core with dispersed instants of idle time to mimic a processor core when there is no covert channel. In order to statistically model the threshold for detection, the peak power spectral density of $GIPS$ and $\Delta GIPS$ is recorded for 1000 such simulations. These amplitudes follow a Gaussian distribution, as shown in Fig. 4(a) for $GIPS$ and Fig. 4(b) for $\Delta GIPS$.



(a)



(b)

Fig. 4. Probability distribution functions of (a) peak power spectral density of $GIPS$ and (b) peak power spectral density of $\Delta GIPS$. This data is used to statistically determine the detection threshold.

In order to consider the 95% of the variation in the amplitudes, the maximum threshold for detection is calculated based on the 4σ value (mean plus 2σ), which corresponds to 1324 $GIPS^2/Hz$ for $GIPS$ and 53 $\Delta GIPS^2/Hz$ for $\Delta GIPS$. Thus, if $P_{\Delta GIPS_i}$ in Alg. 1 is greater than the threshold calculated above, a TCC is said to be detected.

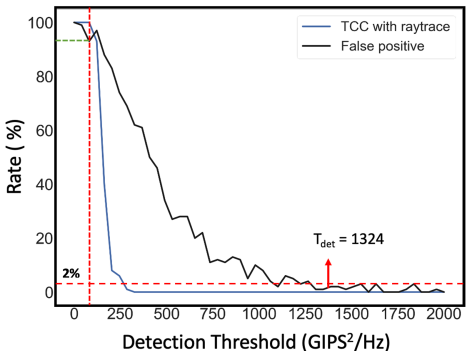
V. Simulation Results

To simulate a thermal covert channel attack, 10 blocks of secret data are encoded in the power profile based on ASK by executing different programs from SPLASH-2/PARSEC suites on a 4-core Intel Haswell architecture, operating at a frequency of 3.5 GHz [16], [24]. Each secret data block comprises of 10 bits of preamble and 100 bits of data. The different TCC workloads are simulated for a 4-core CPU based on an Intel Haswell architecture [24]. The architecture-level simulations to obtain the transient IPC and GIPS traces are performed using the SNIPER multicore simulator [25]. To evaluate the effectiveness of the proposed detection technique, we simulate 100 such communications and determine detection accuracy or TCC detection rate R_{det},

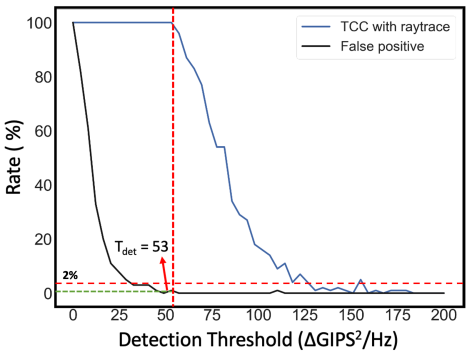
$$R_{det}(in\%) = \frac{N_{detected}}{N_{TCC}}, \tag{3}$$

where, $N_{detected}$ is the number of TCCs that are detected (when the sum of the peaks of $\Delta GIPS_i$ spectrum crosses the threshold, as shown in lines 10 to 12 in Alg. 1) and N_{TCC} is the total number of TCCs established.

Furthermore, to evaluate this detection against cores executing nominal SPLASH-2/PARSEC applications without any

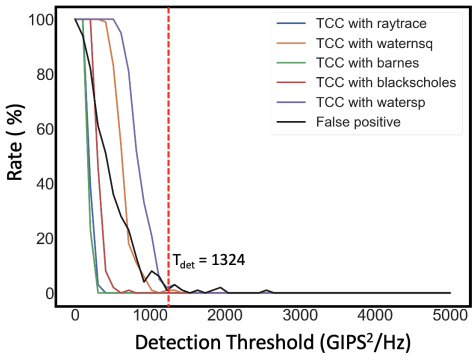


(a)

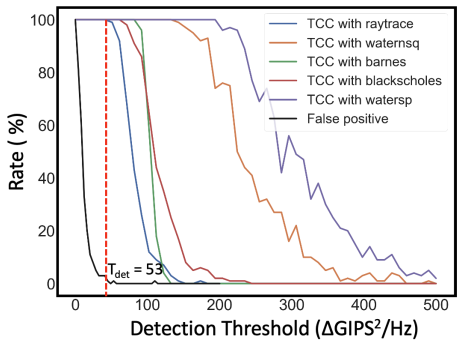


(b)

Fig. 5. TCC detection rate and false positive rate vs. detection threshold for (a) traditional $GIPS$ (or IPC) based detection and (b) proposed $\Delta GIPS$ based detection.



(a)



(b)

Fig. 6. TCC detection rate and false positive rate vs. detection threshold for (a) traditional $GIPS$ (or IPC) based detection and (b) proposed $\Delta GIPS$ based detection.

covert channel attacks, we execute 100 simulations of a sequence of such applications and define a metric referred to as false positive rate R_{fp},

$$R_{fp}(in\%) = \frac{N_{fp}}{N}, \tag{4}$$

where, N_{fp} is the total number of typical simulations without TCC that get detected as a TCC (*i.e.* false positives) and N is the total number of such simulations.

Fig. 5(a) depicts the variation of R_{det} (blue plot) and R_{fp} (black plot) as a function of detection threshold for the existing GIPS (or IPC) based detection. The same data is shown for the proposed $\Delta GIPS$ based detection in Fig. 5(b). In this example, *raytrace* (one of the lowest power applications in the suite, making it among the most difficult to detect) is used to establish a TCC attack. The detection threshold (T_{det}) determined in the previous section with the 4σ variation is marked in both plots. First, it can be observed that the false positive rate is less than 2% when the threshold is increased beyond T_{det} in both of the plots, thereby verifying the proposed procedure to calculate the detection threshold (T_{det}). Importantly, as observed from Fig. 5(a), at the marked T_{det}, the TCC detection rate for the existing approach is 0%. Thus, the existing detection technique based on GIPS (or IPC) metric fails to detect any of the TCCs encoded by executing the low power *raytrace* application. Alternatively, the detection rate for the same *raytrace* based TCC with the proposed $\Delta GIPS$ metric at T_{det} is 100%, as illustrated in Fig. 5(b). Furthermore, the red and green lines in the plots show that for the thresholds where R_{det} is 100%, R_{fp} is 95% in Fig. 5(a) whereas R_{fp} is less than 2% in Fig. 5(b).

The proposed algorithm is evaluated for TCC encoded by executing four more low power applications, *water.nsq*, *barnes*, *blackscholes* and *water.spatial*. Similar plots for the variation of R_{det} and R_{fp} as a function of detection threshold are shown in Fig. 6(a) (for the conventional $GIPS$ metric) and Fig. 6(b) (for the proposed $\Delta GIPS$ metric). As observed from the plots, R_{det} and R_{fp} are approximately 0% for all of the applications at T_{det} in Fig. 6(a), which relies on the existing detection method. If the detection threshold is reduced to increase detection rate, the false positive rate also quickly increases. Alternatively, with the proposed algorithm, the TCC detection rate with all of the applications is 100% at $T_{det} = 53$ whereas the false positive rate is less than 2%, as shown in Fig. 6(b). Thus, the proposed detection algorithm overcomes the limitation of the existing detection technique, thereby successfully detecting a TCC attack established by executing low power benchmarks.

VI. Conclusion and Future Work

Thermal covert communication attacks in multicore processors can enable the exfiltration of confidential information by exploiting thermal coupling between the physical cores. Previous works have shown that a high bandwidth TCC can be established by executing low power benchmark programs. However, the existing techniques fail to detect such TCC attacks, thus posing a danger to data security. Therefore, we

have proposed a novel detection technique in this work. The proposed approach can detect any TCC attack established by low power programs with 100% detection accuracy and less than 2% false positive rate. Furthermore, the detection threshold is statistically determined based on extensive simulations that mimic a practical processor without a TCC attack. In future work, this method can be extended to detect other types of covert channels such as power and EM. The potential of machine learning based detection approaches can also be explored.

REFERENCES

[1] P. Stavroulakis and M. Stamp, *Handbook of information and communication security*. Springer Science & Business Media, 2010.

[2] S. Cabuk, C. E. Brodley, and C. Shields, "Ip covert channel detection," *ACM Transactions on Information and System Security (TISSEC)*, vol. 12, no. 4, pp. 1–29, 2009.

[3] 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, pp. 481–487.

[4] 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, p. 2081, 2023.

[5] J. C. Wray, "An analysis of covert timing channels," *Journal of Computer Security*, vol. 1, no. 3-4, pp. 219–232, 1992.

[6] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: high-bandwidth and reliable covert channel attacks inside the cloud," *IEEE Transactions on Networking*, vol. 23, no. 2, pp. 603–615, 2014.

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

[8] R. J. Masti, D. Rai, A. Ranganathan, C. Müller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in *24th USENIX security symposium (USENIX security 15)*, 2015, pp. 865–880.

[9] S. Wang, X. Wang, Y. Jiang, A. Singh, L. Huang, and M. Yang, "Modeling and analysis of thermal covert channel attacks in many-core systems," *IEEE Transactions on Computers*, 2022.

[10] Q. Wu, X. Wang, and J. Chen, "Defending against thermal covert channel attacks by task migration in many-core system," in *IEEE International Conference on Circuits and Systems*, 2021, pp. 111–120.

[11] H. Huang, X. Wang, Y. Jiang, A. K. Singh, M. Yang, and L. Huang, "Detection of and countermeasure against thermal covert channel in many-core systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[12] D. B. Bartolini, P. Miedl, and L. Thiele, "On the capacity of thermal covert channels in multicores," in *Proceedings of the Eleventh European Conference on Computer Systems*, 2016, pp. 1–16.

[13] 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 *ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[14] J. Wang, X. Wang, Y. Jiang, A. K. Singh, L. Huang, and M. Yang, "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, pp. 3276–3287, 2020.

[15] 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 *Design, Automation & Test in Europe*, 2018, pp. 1459–1464.

[16] K. Dhananjay, V. F. Pavlidis, A. K. Coskun, and E. Salman, "High bandwidth thermal covert channel in 3-d-integrated multicore processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.

[17] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution." *Hasp@ isca*, vol. 10, no. 1, 2013.

[18] "Arm trustzone," https://developer.arm.com/ip-products/security-ip/trustzone.

[19] I. Miketic, K. Yethiraj, and E. Salman, "Information-theoretic perspective to thermal covert channels," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2023, pp. 1–5.

[20] J.-J. Horng, S.-L. Liu, A. Kundu, C.-H. Chang, C.-H. Chen, H. Chiang, and Y.-C. Peng, "A 0.7v resistive sensor with temperature/voltage detection function in 16nm finfet technologies," in *2014 Symposium on VLSI Circuits Digest of Technical Papers*, 2014, pp. 1–2.

[21] T. Oshita, J. Shor, D. E. Duarte, A. Kornfeld, and D. Zilberman, "Compact bjt-based thermal sensor for processor applications in a 14 nm tri-gate cmos process," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 3, pp. 799–807, 2015.

[22] M. Alagappan, J. Rajendran, M. Doroslovački, and G. Venkataramani, "Dfs covert channels on multi-core platforms," in *IEEE Int. Conf. on Very Large Scale Integration*, 2017, pp. 1–6.

[23] A. Córdoba, "Dirac combs," *letters in mathematical physics*, vol. 17, no. 3, pp. 191–196, 1989.

[24] N. Kurd, M. Chowdhury, E. Burton, T. P. Thomas, C. Mozak, B. Boswell, P. Mosalikanti, M. Neidengard, A. Deval, A. Khanna *et al.*, "Haswell: A family of ia 22 nm processors," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 49–58, 2014.

[25] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization*, vol. 11, no. 3, pp. 1–25, 2014.