Combating Stealthy Thermal Covert Channel Attack with its Thermal Signal Transmitted in Direct Sequence Spread Spectrum

Xiaohang Wang, *Member, IEEE,* Shengjie Wang, Yingtao Jiang, *Member, IEEE,* Amit Kumar Singh, *Member, IEEE*, Mei Yang, *Member, IEEE,* and Letian Huang, *Member, IEEE*

***Abstract*—Many-core systems are susceptible to attacks launched by thermal covert channel (TCC) attacks. Detection of TCC attacks often relies on the use of threshold-based approaches or variants, and a countermeasure to thwart the channel can be applied only after an attack is deemed to be present. In this paper, we describe a direct sequence spread spectrum (DSSS) based TCC, where its thermal data are modulated by a pseudo-random bit sequence. Unfortunately, such DSSS-based TCC has an extremely low signal strength that the signal is nearly indistinguishable from the noise and thus cannot be detected by any existing threshold-based detection methods. To combat this stealthy TCC, we propose a novel detection scheme that lets the received signal pass through a differential filter where irrelevant frequency components occupied mainly by the noise gets eliminated and the filtered signal is next compared against a threshold for successful detection. Experimental results show that the DSSS-based TCC can effectively survive detection by the existing detection methods with its BER as low as 4%. In contrast, with the proposed detection and countermeasure applied, the detection accuracy jumps to 89%, and the BER of the DSSS-based TCC soars to 50%, which indicates that the TCC is practically shut down.**

***Index Terms*—Defense against covert channel attack, many-core systems, thermal covert channel attack.**

Manuscript received April 07, 2022; revised June 11, 2022; accepted July 05, 2022. This article was presented at the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2022 and appeared as part of the ESWEEK-TCAD special issue. This work was supported in part by the National Natural Science Foundation of China under Grant 61971200, in part by Zhejiang Lab under Grants 2021LE0AB01 and 2021PC0AC01, in part by Major Scientific Research Project of Zhejiang Lab under Grant 2021LE0AC01, in part by the Open Research Grant of State Key Laboratory of Computer Architecture Institute of Computing Technology, Chinese Academy of Sciences under Grant CARCH201916, in part by the Key Technologies R&D Program of Jiangsu (Prospective and Key Technologies for Industry) under Grant BE2021003, in part by the Key Laboratory of Big Data and Intelligent Robot (South China University of Technology), Ministry of Education, and in part by the National Key Research and Development Program of China under Grant 2019QY0705.

X. Wang is with the School of Software Engineering, South China University of Technology, and also with Zhejiang Lab and with State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences. Xiaohang Wang is the corresponding author. E-mail: xiaohangwang@scut.edu.cn.

S. Wang is with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China.
E-mail: 201921043708@mail.scut.edu.cn.

Y. Jiang and M. Yang are with the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, USA.
E-mail: yingtao@egr.unlv.edu, mei.yang@unlv.edu.

A.K. Singh is with the School of Computer Science and Electronic Engineer, University of Essex, CO4 3SQ Colchester, United Kingdom.
E-mail: a.k.singh@essex.ac.uk

L. Huang is with the School of Communication and Information Engineering, Electronic Science and Technology of China, Chengdu 610054, China.
E-mail: huanglt@uestc.edu.cn.

I. Introduction

THERMAL covert channel (TCC) attacks in a many-core chip can sustain serious attacks targeting sensitive data/information [1]–[3], and data leaks can go unnoticed for a long period of time. In a nutshell, a thermal covert channel transforms a data stream into temperature variations, and the data are transmitted between two cores by means of heat transfer. Coming as a pair of transmitter and receiver [4], a thermal covert channel sees its transmitter core's temperatures going up or down by turning its circuity on or off. The receiver on the other end of the channel collects the thermal signals by reading its local thermal sensor(s) [5] and then extracts the data. Studies have shown that the transmission rate of a thermal covert channel can go as high as 8 bits per second and the channel's bit error rate (BER) can be brought down to below 10% [2].

To counter thermal covert channel attacks, recently, a couple of countermeasures [6], [7] have been proposed. In [6], strong noise whose spectrum matches that of the detected thermal covert channel is emitted to jam the data transmission. In [7], Dynamic Voltage and Frequency Scaling (DVFS) is applied to any core identified as the TCC transceiver to block the TCC traffic.

Any countermeasure can be applied only after a TCC is detected. The threshold-based detection method [6], illustrated in Fig. 1, remains the most popular one. If the signal amplitude is greater than the threshold, a hard decision on the presence of an attack will be made. Since this threshold-based detection is entirely dependent on the signal strength, it fails to detect TCCs whose amplitude is low and close to the noise floor. In this paper, we show that thermal signals can actually be embedded into noise with direct sequence spread spectrum (DSSS) modulation, leaving no differentiation between signal and noise to the detector as far as their amplitudes are concerned.

Detecting this improved stealthy DSSS-based TCC is challenging and demands a novel detection scheme due to the noise-like signal transmitted in the channel. In this regard, we propose a scheme to separate out only the useful signal components by removing low frequency noise and irrelevant signal components, after which a threshold-based method can be used to detect the existence of a potential thermal

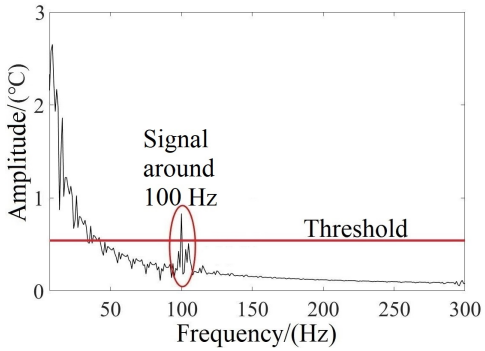


Fig. 1. A threshold-based detection method, like the one used in [6], can only detect a TCC whose signal strength is significantly higher than that of the noise.

covert channel. Once a DSSS-based TCC attack is detected, a countermeasure that relies on injecting strong noise to jam the TCC can be applied. The experimental results show that the proposed detection scheme can achieve a detection accuracy of as high as 89% for both the TCC without DSSS (herein referred as the baseline TCC) and the DSSS-based TCC. After applying this jamming method, the BERs of both the baseline TCC and the DSSS-based TCC skyrocket to 48.3%, which means any TCC attack is literally shut down for good.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces the design of the improved stealthy thermal covert channel attack by DSSS, and Section IV details a countermeasure method against such DSSS-based TCC. Section V reports the experimental results. Finally, Section VI concludes the paper.

II. Related Work

A. Construction of TCCs

Thermal covert channels have been demonstrated and studied in the context of cloud systems [8], operating system [9], and many-core chips [10], [11] *etc.* A thermal covert channel in a many-core system is able to transmit sensitive data among the cores in the way that a program running in the source core (*i.e.*, transmitter) affects the temperature that another core (*i.e.*, receiver) can observe. Of the proliferated thermal covert channel designs that have been proposed, they differ from each other in terms of line coding schemes adopted and their application areas.

Masti *et al.* [1] implemented a TCC attack with a data rate of 1.33bps and BER of 11% for a 1 hop channel on real machines. Bits '1' and '0' are respectively encoded by high and low temperature levels. A lower bit error rate can actually be achieved by changing the coding scheme of the thermal covert channel, as shown in [2], where bit '1' is encoded as a state transition from active to sleep, and bit '0' as a state transition from sleep to active. The TCC reaches 8 bps with a BER of approximately 0.1% on a real machine. Unlike [2], Long *et al.* [4] adopted return-to-zero (RZ) encoding in TCC. Another TCC encoding scheme was considered in [12], where sensitive data transmitted between the CPU and the DRAM in a package-on-package system are encoded as the number of bits flipped in the DRAM as a result of temperature variation.

With the proliferation of TCC coding schemes and target application areas, it is necessary to estimate the channel capacity of different TCC configurations [2]. In [13], a set of analytical models concerning the BER, SNR, and channel capacity of the TCC attacks were derived, which particularly help reveal how the transmission frequency and transmission power are related to TCC performance. For instance, the impact of noise on the TCC can be suppressed by letting the signal transmitted at a higher frequency.

B. TCC Detection Methods and Coutermeasures

The state-of-the-art countermeasures against TCC attacks proposed in [6], [7], [13] rely on the use of threshold-based detection approaches to detect the existence of a TCC. Specifically, the threshold-based detection requires scanning of the frequency spectrum, and a TCC attack is deemed to be present if the signal amplitude is higher than the threshold. Once a TCC is detected, a jamming noise whose transmission frequency matches the detected TCC is emitted to block the TCC transmission [6]. In [7], DVFS is applied to the cores that have a TCC transceiver to shut down any data transmission right from the source. Another countermeasure approach in [14] tries to separate the identified transmitter and receiver farther apart from each other through task migration. Since thermal signals decay very fast with respect to distance, the BER of TCC becomes unacceptably high when the distance between the cores running the transmitter and the receiver is beyond a few hops.

These threshold-based detection methods, however, fails to detect the TCC whose signal amplitude drops to a level comparable to or even lower than the thresholds. This problem can be more pronounced in a DSSS-based TCC.

C. Direct Sequence Spread Spectrum (DSSS)

Direct sequence spread spectrum (DSSS) [15] is widely used to reduce signal interference in digital communications. In DSSS, transmitters use periodic pseudo-noise (PN) sequences before transmission to spread the spectrum of the signal and transmit the signal close to the noise level. The signal is difficult to be detected by a third party, should the PN sequence used in the transmitter be unknown to the eavesdropper.

Apparently, if the TCC signals can be modulated and transmitted in DSSS, detection of the channel can be extremely hard. The conventional DSSS cannot not be used directly in TCC, due to the drastic difference between wireless and thermal signals, and the high implementation cost which cannot be supported in on-chip systems as considered in this paper. For on-chip systems, thermal signals have the following features.

1) Signal amplitude and transmission frequency are coupled, since reducing amplitude can only be achieved by decreasing the heating time and thus reducing period.

2) Thermal signal variation is slow and cannot be accurately controlled, and the signal amplitude is very limited due to chip thermal power budget cap.

3) Signal length is correlated to the initial temperature of signal.
4) Thermal noise concentrates on the low-frequency domain.
5) The frequency band of TCC is much narrower than wireless signals.

Therefore, the modulation and coding schemes in TCC should be carefully designed. The purpose of the conventional DSSS is to reduce signal interference. Different from the conventional DSSS, the pseudo-random code is used in this paper to reduce the heating time and thus the amplitude of the TCC signal, so as to make the signal appear to look like noise and improve the stealthiness of the TCC. Conventional detection methods against DSSS as [16] cannot be directly applied to TCC due to the high implementation overhead and unique physical features and transmission characteristics of thermal signals. The rest of the paper is thus dedicated to the creation of a DSSS-based TCC and the countermeasure against it.

D. Comparison with Other On-chip Covert Channels

The cache covert channel transmits sensitive information by encoding the bits with the latencies of accessing last level cache (LLC) [17], which can achieve throughputs up to tens of Kbps. However, detection of cache covert channel is found to be easy with the use of hardware performance counters [18]. In addition, various defense mechanisms are proposed to mitigate cache timing channel attacks (*e.g.*, using prefetchers, based on architectural changes, cache allocation technology, *etc*. [18]). In a sharp contrast, TCC, even at a lower transmission rate, can be made much more stealthy and thus, poses a much greater security challenge to the system than cache overt channel.

Beyond caches, integer and floating-point units have been found to be sources of information leaks [19], which can achieve throughputs up to hundreds or even thousands of Kbps. In addition, covert channel attacks through branch prediction units [20] have been shown as well, which achieve up to 624 Kbps when the transmitter and receiver can have a well optimized and aligned clock signals. However, the above two types of covert channels can only be established between different threads of the same core (that is, within a core), which is different from the thermal covert channels that can be established between different cores.

III. Improved Stealthy Thermal Covert Channel Attack

In this section, we propose an improved stealthy TCC with signal transmitted in DSSS. Specifically, the baseband signal is modulated by multiplying it with a pseudo-random code such that the signal is spread over a wider spectrum and the signal amplitude is close to the noise level.

A. Threat Model

There are at least one transmitter, one receiver, and one possible defender in a TCC. The transmitter and the receiver of a TCC are specially designed programs, where the former runs on a processing core located in the secure zone, and the latter runs on a processing core outside the secure zone. Supported by ARM TrustZone [21], Intel software-guard extensions (SGX) [22] technologies, or alike, the secure zone in a many-core platform provides a secure environment to protect users' private code and data from unauthorized access. Unfortunately, it was demonstrated that both the ARM TrustZone and Intel SGX could be compromised by some means [23]–[26]. A TCC attack is set to bypass the security check/monitoring mechanism on data transfer between a program/core sitting inside the TEE enclave and another program/core outside the TEE enclave. There are two possible cases of security check/monitoring. (1) The data transfer between TEE enclaves or outside is performed under attestation or monitoring [27]. (2) Runtime monitoring units monitor the data transfer between cores/programs [28]. However, TCC can bypass the security check/monitoring and leak the protected data in TEE without being supervised or monitored. The transmitter obtains the sensitive data and sends it to the receiver via the thermal covert channel, which can be loaded into the security zone of a multi-core system by using the methods described in [24]–[26]. For instance, the program in support of TCC can be embedded into the self-signed application before a self-signed application is loaded into an 'Enclave' [24]. Even worse, this TCC-enabled data transmission cannot be detected by the remote attestation mechanism of SGX. The malicious program includes the TCC codes, which appear the same as normal programs, except only a small portion of the malicious code, and almost the entire program is legitimate. Therefore, it is not possible to detect the TCC codes by the existing approaches as the TCC codes are not faulty. Furthermore, there are various fault injection attacks and side channel attacks against TEEs, including CLKSCREW [24], Plundervolt [29], VoltJockey [25], [26], TPM-Fail [30], and Bluethunder [31], *etc*. which can obtain the key. Although these attacks need many attempts or a few to tens of minutes, automating the attack process (*e.g.*, using a script file) or combining them will greatly accelerate attacks. The receiver at the other end of the channel collects the temperature signal by reading its thermal sensors and recovers the original data from the temperature signal. The defender runs a program that can access all the cores' thermal sensors, and it is granted ROOT privilege that it can apply countermeasures to neutralize any detected TCC attacks. Note that the defender can be located at any core and the physical distance between the transmitter and the receiver has no direct implication on the effectiveness of the defender. In addition, the attacker cannot terminate the defender as the defender is very possible to have a higher privilege (*e.g.*, it can be part of a secure OS or hypervisor).

Herein it is assumed that each core has a thermal sensor [32], which is implied by modern many-core chips with fine-grain power budgeting [33], [34]. In addition, the number and resolution of thermal sensors have a significant effect on TCC attack, which are possible to be ever improved in the future [35]. The thermal sensors in a real machine have a resolution of 1°C [36], and the state-of-the-art thermal sensor has a resolution close to 0.1°C or better [37], [38].

Herein it is also assumed that the receiver, transmitter, and defender all have access to the thermal sensors or the

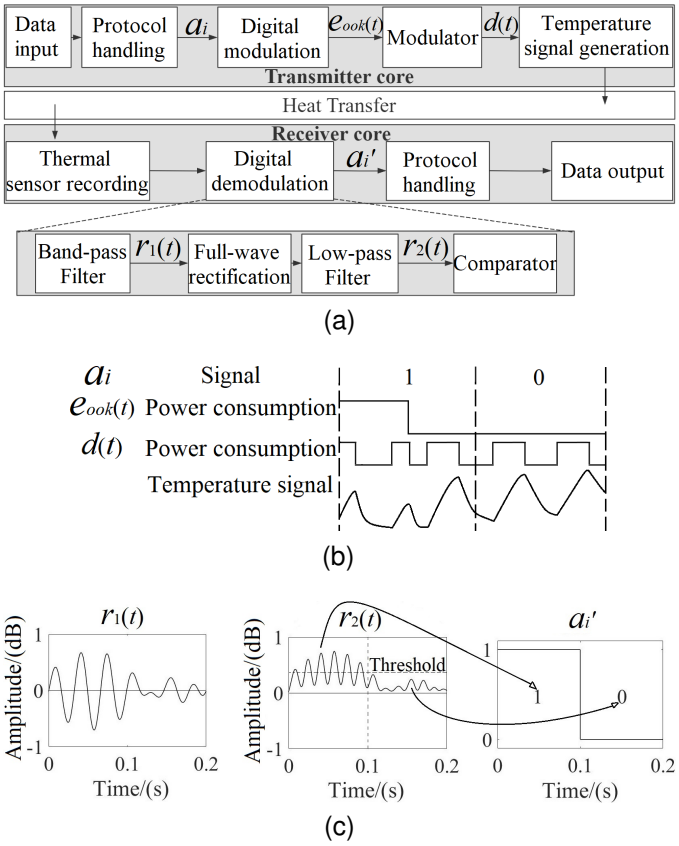


Fig. 2. (a) The flow of the transmission and receiving of the improved stealthy thermal covert channel signals. Signals of the modules in the (b) transmitter and (c) receiver.

same thermal data files through the system APIs or calling the rdmsr instruction from model-specific registers (MSR) in the processor's user space [36]. In the case that the receiver or the defender does not have the permission to read the thermal sensor data directly, it shall still be able to obtain the temperature information by measuring DRAM decay times, as indicated in [12].

B. Transmitting the Bit Streams

Like any TCCs, the improved stealthy TCC involves a transmitter and a receiver pair, as shown in Fig. 2(a). They run on the same or different cores and communicate with each other through the thermal covert channel. In order to support the bit stream transmission between the transmitter and the receiver, the communication protocol in [6] is adopted to ensure the synchronization between the transmitter and receiver.

The transmitter has the following five modules:

1) The *data input module* reads the data to be transmitted.

2) The *protocol handling module* adds preamble (*e.g.*, 101010) and control bits into the sensitive data bit stream.

3) The *digital modulation module* uses on-and-off keying (OOK) to modulate the data bit stream, and the binaries are encoded with the return-to-zero (RZ) code. So, bit '1' is represented as a rise and fall sequence in temperature level, and '0' as keeping low temperature, as illustrated in Fig. 3. That is,

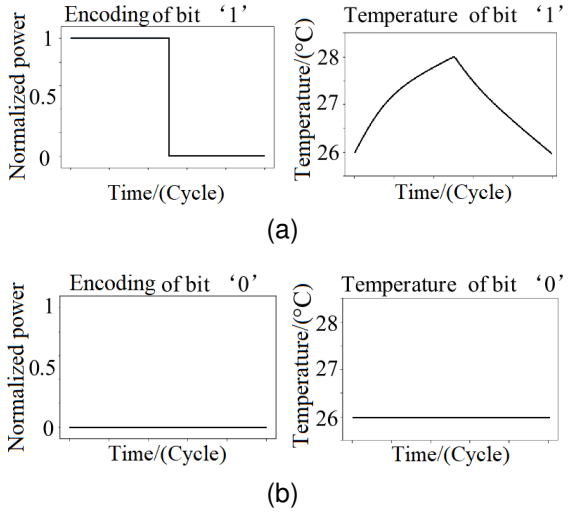


Fig. 3. Power and temperature profiles of (a) bit '1' and (b) bit '0'.

$$e_{OOK}(t) = s(t) \times c(t) \tag{1}$$

$$s(t) = \sum_i a_i g(t - iT_b) \tag{2}$$

where $s(t)$ is the signal to be transmitted, $c(t)$ is the carrier signal, a_i is the binary value ('1' or '0') of the i-th bit, T_b is the symbol width in baseband, and $g(t)$ is the baseband pulse with a duration of T_b.

4) The *modulator* generates a PN code. It is generally required to have nearly equal numbers of bit '1's and bit '0's in the PN code. According to [15], we must ensure

$$P_{num} \geq \lfloor \frac{m+1}{2} \rfloor \tag{3}$$

where P_{num} is the number of consecutive bit patterns in a sequence (*e.g.*, {0}, {1}, {00}, {11}, *etc.*), and m is the number of bits in a PN code. Generally, as opposed to the PN code of {1010101} with only two patterns, {1} and {0}, the PN code {0100011} is more preferable, since it has four patterns: two patterns with a length of 1, *i.e.*, {1} and {0}; one pattern with a length of 2, *i.e.*, {11}; and one pattern with a length of 3, *i.e.*, {000}. The PN code is generated by a program using a linear feedback shift register (LFSR) [15]. The modulated signal is multiplied by the PN code, $pn(t)$, to create the final signal, $d(t)$.

$$d(t) = e_{OOK}(t) \times pn(t) \tag{4}$$

In Fig. 2(b), the PN code is {1001}.

5) The *temperature signal generation module* generates temperature signals by controlling the power consumption of the cores according to the modulated data bit stream. That is, the transmitter runs computation-intensive codes to generate heat that contributes to the rising temperature, or keeps idle to cool down the core for a desired lower temperature [6].

The receiver has the following four modules:

1) The *thermal sensor recording module* picks up the temperature signal.

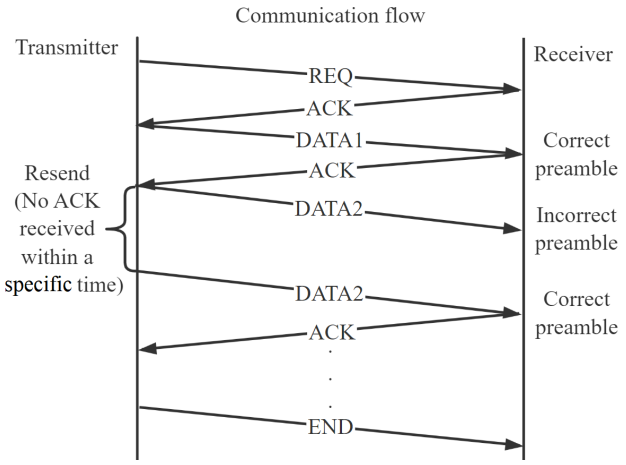


Fig. 4. Timing diagram of a communication session of the covert channel.

2) The *digital demodulation module* uses a band-pass filter whose center frequency f corresponds to the transmission frequency of the TCC to extract the thermal signal. The signal $r_1(t)$ after the band-pass filter is shown in Fig. 2(c). After filtering, the signal passes through a full-wave rectifier to convert the alternating current (AC) signal to direct current (DC). This DC signal then enters a low-pass filter so as to extract the signal envelope. Finally, a hard binary decision is made by having

$$a_i' = \begin{cases} 1 & A_i \geq A_{thre} \\ 0 & A_i < A_{thre} \end{cases} \tag{5}$$

where a_i' is the binary value of the i-th bit, A_i is the amplitude of the i-th bit after filtering, and A_{thre} is a preset threshold. Fig. 2(c) shows the signal $r_2(t)$ as it flows through the full-wave rectification, the low-pass filtering, and hard decision to recreate a_i' ($\{1, 0\}$ in this example).

3) The *protocol handling module* checks the integrity of the packet by examining its preamble. If the packet is found uncompromised, the receiver notifies the transmitter with an acknowledgment.

4) The *data output module* finally extracts the sensitive data from the bit stream.

The communication protocol is shown in Fig. 4. As shown in Fig. 4, the transmission starts with the transmitter sending an REQ packet to the receiver and keeps waiting. The receiver sends the ACK packet to the transmitter if it receives the REQ packet. Then the transmitter starts to send data packets to the receiver after it receives the ACK packet. In particular, the receiver sends an ACK packet to the transmitter when it receives a packet with the correct preamble and the transmitter continues to send data packets after receiving the ACK packet. On the contrary, if an error is detected in the preamble, the receiver does not send ACK to the transmitter, and the transmitter resends this packet after a specific time. Once all the data are transmitted, the transmitter sends an END packet to the receiver to terminate the transmission.

Fig. 5(a) shows typical RZ-encoded TCC signals, and Fig. 5(b) shows the signal in the improved stealthy TCC. It can be seen that there is a continuous temperature buildup in

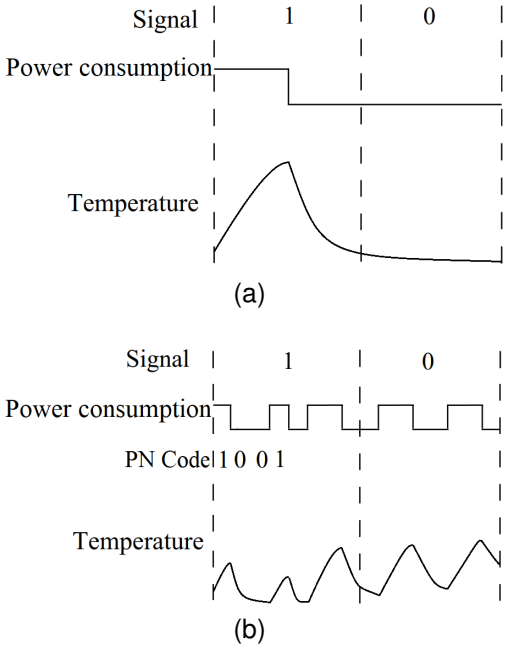


Fig. 5. Waveforms of (a) thermal signals encoded by the RZ scheme in [6], and of (b) the proposed improved stealthy TCC.

Fig. 5(a). Compared with Fig. 5(a), the time of heating up is greatly shortened in the improved stealthy coding scheme, as shown in Fig. 5(b), which substantially suppresses signal amplitude. Correspondingly, the threshold-based detection methods in [6] and [7] can no longer detect the improved stealthy TCCs. Fig. 6 shows the signal spectrum of [6] and the improved stealthy TCC with the transmission frequency in band B. The noise in band A is fairly strong, and thus, it will be much less desirable to have a TCC transmit data in this band. One can see from Fig. 6(b) that the signal amplitudes of the improved stealthy TCC are below the threshold, and as a result, any threshold-based detection method will not be able to detect this type of TCC. Simply decreasing the threshold will do no good in signal detection. Actually, a lower threshold tends to create a high false positive rate in signal detection and can cause confusions between actual signals and noise.

The defender does not know the signal transmission frequency, and thus it cannot detect the improved stealthy TCC. However, the receiver knows the signal transmission frequency and can correctly decode it as the filter center frequency equals to the signal transmission frequency, and only the frequency components within the filter band are left. After the filtering, the receiver uses a threshold comparison to correctly decode the signal since all the noise out of the band is filtered out, and the signal amplitude is higher than noise in the filter band.

IV. Countermeasure Using Differential Filter

To countermeasure the above thermal covert channel attack, an intuitive method is to use full-band jamming, where strong noise covering the entire band of TCC is injected to the channel with a sole purpose to jam the channel. However, this approach causes excessive power consumption and degrades system performance. In order to detect the DSSS-based thermal covert channel, the key is to detect signals of reduced

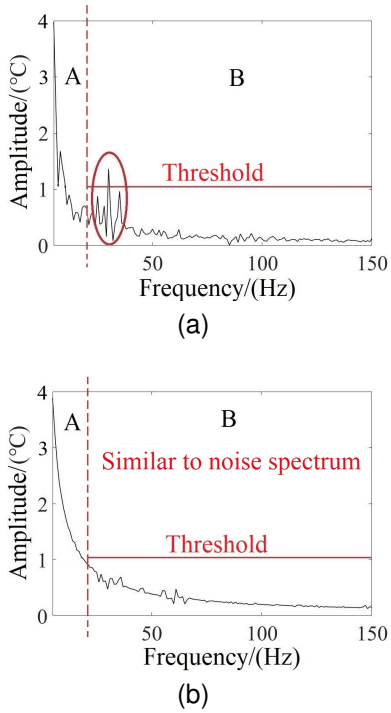


Fig. 6. The signal spectrum of (a) the existing TCC [6] and of (b) the improved stealthy TCC with the transmission frequency in band B.

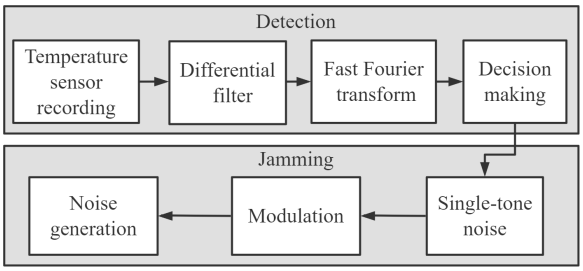


Fig. 7. Workflow of the proposed countermeasure against the DSSS-based TCC.

amplitude. In this section, we describe our two-step scheme to detect and fight against the proposed thermal covert channel attacks, as shown in Fig. 7. For detection, a differential filter is applied, and it analyzes the spectrum and filters out the DC and irrelevant components of the signal. Thereafter, a threshold-based method can be used to detect the existence of a potential thermal covert channel. In the jamming step, a narrow-band noise with the same frequency of the detected thermal covert channel is emitted to block the channel.

A. Differential Filter-based Detection Scheme

The differential filter is written as

$$y(n) = x(n) - x(n-L) \tag{6}$$

where $x(n)$ is the input signal, L is the difference step, $y(n)$ is the output signal after applying the filter.

The magnitude response function of the differential filter is [15]

$$\left|H(e^{j\omega})\right| = 2\left|\sin\tfrac{L\omega}{2}\right| \tag{7}$$

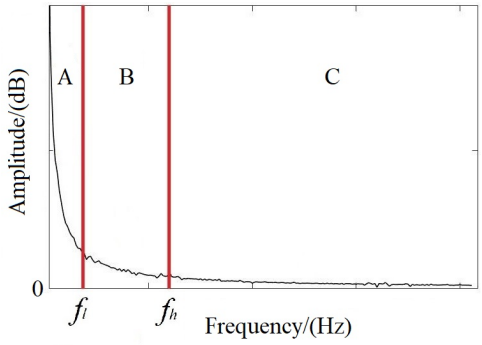


Fig. 8. The noise spectrum.

where ω is the angular velocity and $\omega = 2\pi f/f_s$, and f_s is the sampling rate.

The differential filter can remove narrow-band noise at frequency f_i and its multiples,

$$f_i = \frac{i}{L}f_s, i = 0, 1, ..., \frac{L}{2} - 1 \tag{8}$$

From the noise spectrum shown in Fig. 8, it can be seen that the low frequency components of the noise have high amplitudes. The TCC signal is severely jammed by noise if the transmission frequency falls in band A, which makes TCC transmission impossible, as shown in Fig. 8. In contrast, since noise in band C is extremely low, TCC signals falling into this band tend to be easily detected. However, signals in band B can escape from being detected, as the noise is moderate. For example, in our experimental setup in Table II, band A is below 20 Hz, band B is from 20 to 50 Hz, and band C is above 50 Hz.

Fig. 9(a) shows the magnitude response of the differential filter. The sampling rate f_s in this example is 1000. The frequency of the improved stealthy TCC is in the range of f_l to f_h, as shown in band B in Fig. 8. To avoid the differential filter mistakenly filtering out the signal, f_i cannot be in band B (*i.e.*, $f_i < f_l$ or $f_i > f_h$). According to Eqn. 8, $f_0 < f_l$, $f_1 = f_s/L > f_h$ (f_s=1000, $f_h \leq$100), and thus L is set to be 1 such that f_1 is in band C. One can see from Fig. 9(a) that the differential filter can filter out the thermal noise at low frequencies as expected. Fig. 9(b) shows the signal in the time domain. It can be seen that the differential filter converts the trace of temperature signals into a trace of the temperature differences (*e.g.*, Δ1, Δ2, *etc.*), as in Eqn. 6. In TCC, information is encoded by temperature variations/differences, the differential filter keeps only the useful signal components and removes the low frequency noise and irrelevant components of the signal to improve the signal-to-noise ratio (SNR). Doing so would allow a threshold-based method to be used over the filtered signal to detect whether there is a potential thermal covert channel or not, as shown in Fig. 9(c). The threshold ρ varies for different system configurations. Herein we propose a three-step scheme for calculating ρ. The threshold value ρ can be affected by a number of external factors (*e.g.*, cooling configuration, CPU frequency, *etc.*), and these factors are already incorporated in the method. In step 1, the average noise amplitude ρ_l is calculated from the sampled noise after the differential filter. In step 2, the maximum signal amplitude ρ_h is obtained from

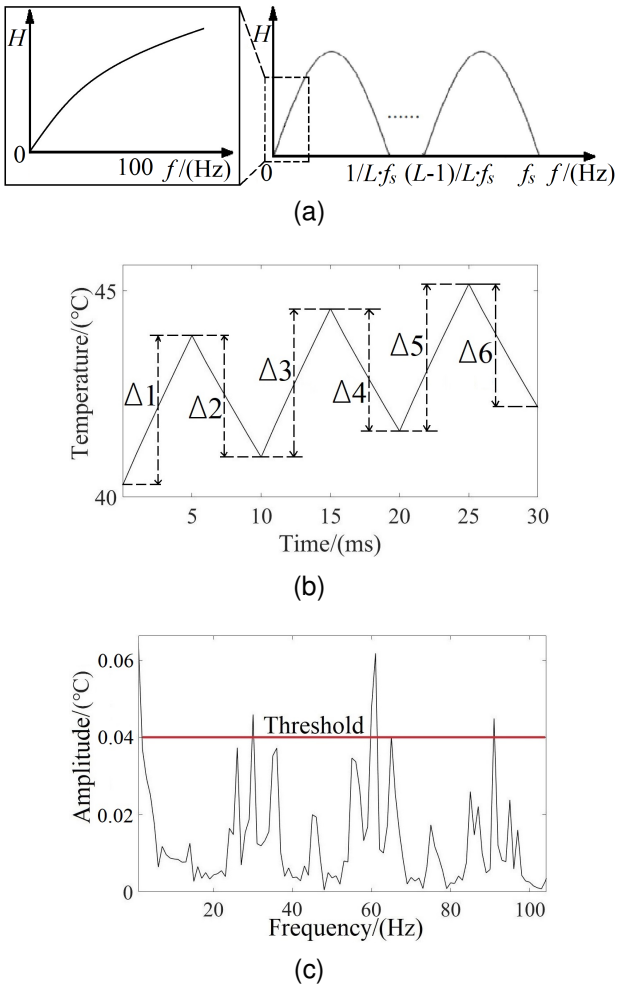


Fig. 9. Detection of the DSSS signal. (a) The magnitude response of the differential filter. (b) The signal in the time domain. (c) The signal spectrum after the differential filter.

the sampled improved stealthy TCC signal after the differential filter. In step 3, the proposed detection is applied to get the detection accuracy with a threshold ρ ranging from ρ_l to ρ_h. The threshold ρ is assigned with a value that leads to the highest detection accuracy.

Fig. 10(a) shows the magnitude response of a high-pass filter with a cut-off frequency of 20 Hz (lower than the transmission frequency of TCC to ensure that no useful signal component gets filtered out). Fig. 10(b) shows the signal spectrum of the improved stealthy TCC with a transmission frequency of 30 Hz after applying the high-pass filter. It can be seen that the threshold-based methods [6], [7] cannot detect the improved stealthy TCC in 30 Hz with the high-pass filter. The reason is that the high-pass filter only removes low frequency noise, but it cannot extract the signal components needed for information encoding.

To enable the detection, the temperature data are collected from the local thermal sensors. Each core's temperature signal is recorded for a duration of 1.5 seconds (experimentally determined). The recorded signal goes through the differential filter, and the frequency spectrum of the filtered signal is then determined by applying Fourier analysis. Threshold-based

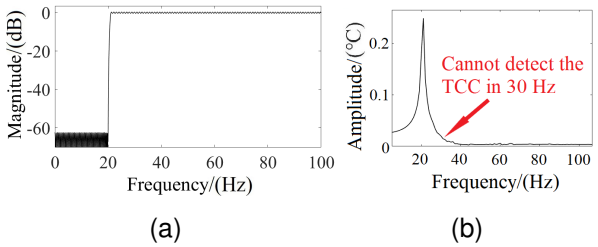


Fig. 10. (a) The magnitude response of the high-pass filter. (b) The signal spectrum after passing the high-pass filter.

decision making is performed to check whether there is a potential thermal covert channel attack or not. If a thermal covert channel attack has been detected, the jamming process is performed to block it.

B. The Jamming Scheme

The single-tone noise is adopted in the jamming process in the defender. If the thermal covert channel is detected, a consecutive bit sequence of bit '1's is generated as jamming noise by the single-tone noise module.

The bit sequence generated by the single-tone noise module is modulated by the same frequency as the transmission frequency of the detected TCC. The output of the modulation module is

$$N_{jam}(t) = n_{jam}(t) \times c_{jam}(t) \tag{9}$$

where $n_{jam}(t)$ is the jamming noise generated by the single-tone noise module, and $c_{jam}(t)$ is the carrier signal.

Finally, the narrow-band noise is generated by passing through the same temperature signal generation module as described in Section III.B.

V. Experimental Results

A. Experimental Setup

Experiments have been performed on two different platforms, both to evaluate the improved stealthy TCC attack and the proposed countermeasure.

The first is a many-core simulator, Sniper [39], with McPAT [40] integrated as the power model and Hotspot version 6.0 [41] as the temperature simulator. We simulate both 2-D and 3-D many-core chips with different levels of power consumption pertaining to the processor cores. Table I lists the simulator configuration, and the benchmarks are selected from PARSEC [42], SPLASH-2 [43], and AES [44]. In the experiments, the cores other than the ones designated as the transmitters and receivers are running multiple threads of these benchmarks; actually, each of the benchmarks is expanded to include 4, 8, or 16 threads. These threads shall be treated as the sources of thermal noises to the TCC. The floorplan of the processor cores follows the same one used in [45].

The second sets of experiments were performed in a real computer with its configurations listed in Table II. The **modprobe msr** instruction is run to load the msr module for thermal sensor reading and set the CPU frequency governor to the performance mode with the **cpufreq-set -g performance**

TABLE I
SIMULATION CONFIGURATIONS

Instruction set architecture	x86-64
CPU frequency	2000MHz
Number of cores (2D)	3×3 / 4×4
Number of cores (3D)	3×3×3 / 4×4×3
2D network topology	Mesh
3D network topology	3-D mesh, with each layer has its own 2D mesh and the layers are connected vertically through Through-Silicon-Vias (TSV) [46]
Fetch/decode/commit size	4/4/4
L1 D cache	16KB, 2-way, 32B line, 2 cycles, 2 ports, dual tags
L1 I cache	32KB, 2-way, 64B line, 2 cycles
L2 cache	64KB, 64B line, 6 cycles, 2 ports
Main memory size	2GB
Benchmarks	
PARSEC	Blackscholes, Canneal, Fluidanimate, Streamcluster, Swaptions, X-264, Dedup, Freqmine
SPLASH-2	Barnes, Raytrace
AES	Encoder, Decoder
Hotspot configuration	
Chip thickness	$0.00015m$
Specific heat capacity	$1.75 \times 10^6 J/(m^3 \cdot K)$
Silicon thermal conductivity	$100W/(m \cdot K)$
Temperature threshold for DTM	$373.15K$
Heat sink side	$0.06m$
Heat sink thickness	$0.0069m$
Heat sink thermal conductivity	$400W/(m \cdot K)$
Specific heat capacity of heat sink	$3.55 \times 10^6 J/(m^3 \cdot K)$
Thermal sensor resolution	0.1^oC

TABLE II
REAL MACHINE CONFIGURATIONS

Processor	Intel i7-8700 @4.6 GHz
Memory	16 Gbytes
DRAM frequency	2666MHz
Physical cores	6
Logical cores	12
Fan speed	1200rpm
Operate system	Ubuntu 16.04.5 LTS
Dynamic fan speed	OFF

instruction. Across these experiments, the CPU frequency is fixed at 4.6 GHz, two hardware threads of a selected core are dedicated to build a thermal covert channel, while the rest of the cores are loaded with various benchmarks summarized in Table I. Again, all these benchmarks running at different cores pose as the sources of thermal noise to the thermal covert channel under test.

B. Evaluating the Improved Stealthy Attack

The bit rate is set to vary between 1 and 120 bps in the 0 hop channel and between 1 and 25 bps in the 1 hop channel in the simulator simulating the 2D many-core system, and the BER vs. bit rate results are shown in Fig. 11.

Fig. 12 compares the BER of the proposed thermal covert channel against that of the baseline TCC [6] in systems of different sizes; here the countermeasure in [6] is adopted in these experiments. In Fig. 12(a), the distance between the transmitter and receiver is 0 and denoted as 0 hop (*i.e.*, the transmitter and receiver are running on the same core), and the transmission frequency is 50 Hz. From Fig. 12(a), one can see that the BER of the proposed thermal covert channel is 89.7% lower than that of the baseline TCC on average. In Fig. 12(b), the transmitter and the receiver are 1 hop away from each other (*i.e.*, the transmitter and receiver are placed on two adjacent cores), and the transmission frequency is 30 Hz. From Fig. 12(b), the BER of the proposed thermal covert channel on average is 63.9% lower than that of the baseline TCC. The reason is that the countermeasure in [6] cannot detect the transmission frequency of the proposed thermal covert channel, and thus no countermeasure, like jamming, can be applied.

Fig. 13 compares the BER of the proposed thermal covert channel against that of the baseline TCC [6] on the real machine of different distances between the transmitter and receiver; here the countermeasure in [6] is adopted in these experiments. From Fig. 13, one can see that, when the distance

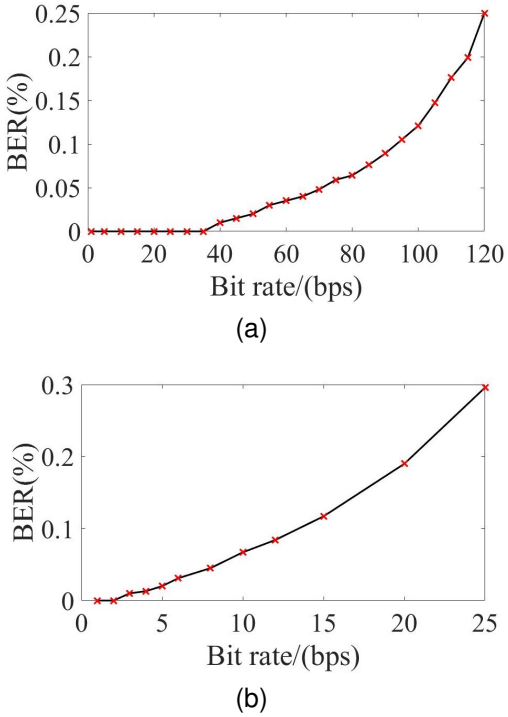


Fig. 11. BERs of (a) 0 hop and (b) 1 hop improved stealthy TCCs in the simulator simulating the 2D many-core system.

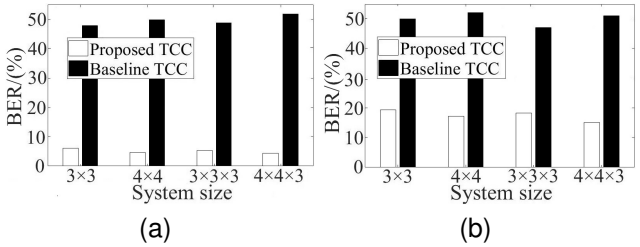


Fig. 12. Comparison of BERs of (a) 0 hop and (b) 1 hop of the improved stealthy TCC and the baseline TCC [6] with the countermeasure in [6] in the simulator simulating both the 2D and 3D many-core systems.

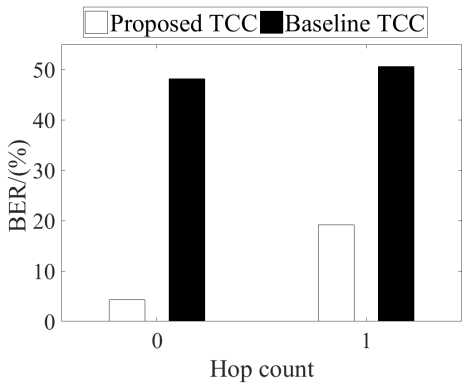


Fig. 13. Comparison of BERs of the improved stealthy TCC and the baseline TCC [6] on the real machine with the countermeasure in [6].

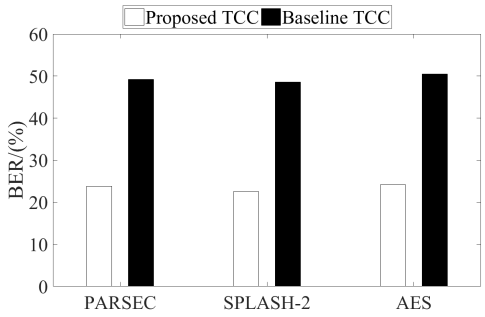


Fig. 14. Comparison of BERs of the improved stealthy TCC and the baseline TCC [6] with the countermeasure in [6] in the simulator simulating the 2D many-core system.

is 0 hop, the BER of the proposed thermal covert channel is 91.1% lower than that of the baseline TCC. On average, the BER of the proposed thermal covert channel is 76.6% lower than that of the baseline TCC. The reason is that the countermeasure in [6] cannot detect the transmission frequency of the proposed thermal covert channel, and thus no countermeasure, like jamming, can be applied.

Fig. 14 compares the BER of the proposed thermal covert channel against that of the baseline TCC [6] in the simulator simulating the 2D many-core system with different channel noises (by running benchmarks from PARSEC, SPLASH-2, and AES); here the countermeasure in [6] is adopted in these experiments. The system is set to have a total of 4×4 cores, and the distance between the transmitter and receiver is 1 hop. From Fig. 14, one can see that, the BERs with different channel noises are close. It can also be seen that when the benchmarks from SPLASH-2 run as channel noise, the BER of the proposed thermal covert channel is 53.7% lower than that of the baseline TCC. On average, the BER of the proposed thermal covert channel is 52.5% lower than that of the baseline TCC, with the same reason as in Figs. 12 and 13.

Fig. 15 compares the BER of the proposed thermal covert channel against that of the baseline TCC [6] on the real machine with different channel noises (by running benchmarks from PARSEC, SPLASH-2, and AES); here the countermeasure in [6] is adopted in these experiments. The distance between the transmitter and receiver is 0 hop. From Fig. 15,

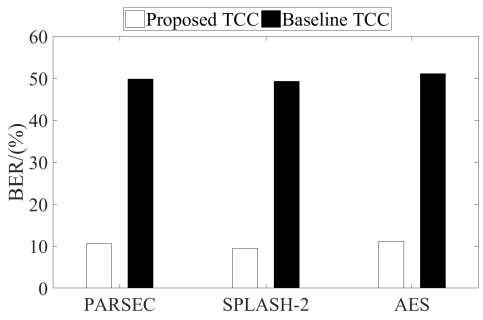


Fig. 15. Comparison of BERs of the improved stealthy TCC and the baseline TCC [6] on the real machine with the countermeasure proposed in [6].

one can see that, the BERs with different channel noises are close. It can also be seen that when the benchmarks from SPLASH-2 run as channel noise, the BER of the proposed thermal covert channel is 80.7% lower than that of the baseline TCC. On average, the BER of the proposed thermal covert channel is 79.2% lower than that of the baseline TCC, with the same reason as in Figs. 12 and 13.

C. Evaluating the Countermeasure

1) Evaluating the Proposed Detection Scheme:

Fig. 16 shows the performance of the proposed detection scheme described in Section IV.A with different thresholds. The detection error is measured by

$$error = \left| f_{trans} - f_{detect} \right| \tag{10}$$

where f_{trans} is the TCC transmission frequency, and f_{detect} is the detected TCC frequency. From Fig. 16, one can see that, the detection error is high when the detection threshold is set to be too high or too low; a low threshold leads to high false positive (treating normal applications as TCCs) rate and a high threshold leads to high false negative (failing to detect the TCC), as shown in Fig. 17. Specifically, when the threshold is lower than 0.07°C, the true positive (TP) rate decreases with the increase of threshold, but the true negative (TN) rate increases with the increase of threshold. When the threshold is 0.03°C, the total detection accuracy reaches the maximum of 88%, with the true positive rate and true negative rate being 46% and 42%, respectively. In the above experiments, half of them generate TCC signals, and the remaining half generate noise. Therefore, the ideal case of the true positive rate, the true negative rate, and the total accuracy are supposed to be 50%, 50%, and 100%, respectively. In addition, the number of samples (*a.k.a.*, the number of data points collected from the sensor to perform fast Fourier transform) affects the detection error. Fig. 18 shows that the performance of the proposed detection scheme described in Section IV.A with different sampling numbers. From Fig. 18, one can see that higher sample numbers lead to lower detection errors. It can also be seen that when the sample size is greater than 1500, the detection error is lower than 2.7 Hz. The reason is that large sample numbers lead to high accuracy and thus improved detection accuracy, as the case when the number of samples is over 1500.

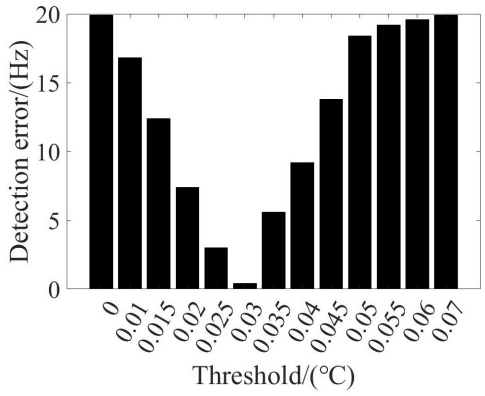


Fig. 16. Detection errors with different thresholds.

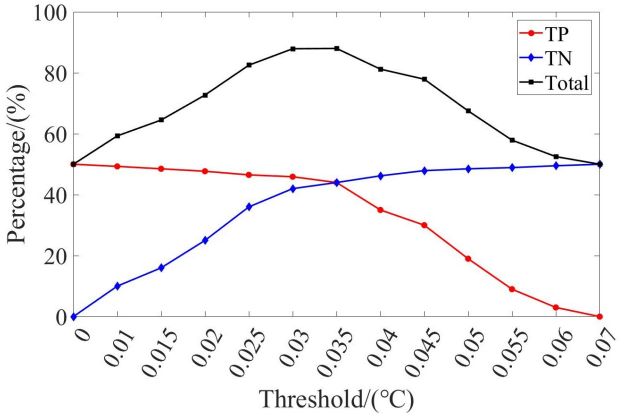


Fig. 17. The accuracies of detecting the improved stealthy TCC under different detection thresholds. 'TP', 'TN' are the percentage of true positive cases and true negative cases over all the cases, respectively, and "total" gives the sum of 'TP' and 'TN'.

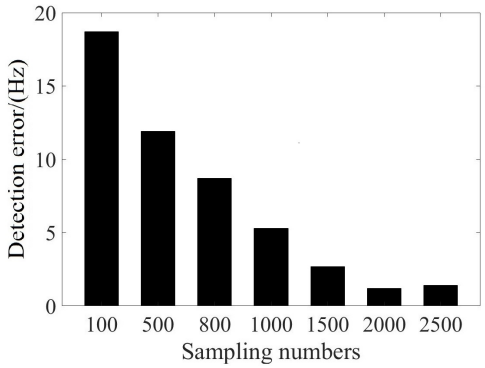


Fig. 18. Detection errors with different sampling numbers.

Fig. 19(a) shows the average accuracy of the experiments on the cases that the transmitter and the receiver are 0 hop and 1 hop apart in the simulator simulating the 2D many-core system. The experiments of Fig. 19(b) are performed on the real machine, and the distance between the transmitter and receiver is 0 hop. Accuracy is defined as follows. A total of 1000 experiments were performed and there are D_0 cases that the detection method successfully detects the TCC attacks. The accuracy α is defined as

$$\alpha = \frac{D_0}{1000} \times 100\% \tag{11}$$

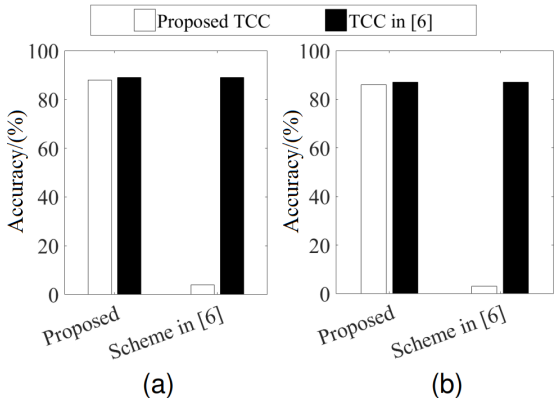


Fig. 19. The detection accuracy of the proposed detection vs. the detection in [6] against the improved stealthy TCC and TCC in [6] in (a) the simulator simulating the 2D many-core system and (b) the real machine, respectively.

As can be seen from Fig. 19, for both simulations and the real machine, the detection accuracies of the proposed detection against both the improved stealthy TCC and TCC in [6] are over 85%. The detection accuracy against the improved stealthy TCC in the simulations even reaches 89%. The detection accuracy of the detection in [6] is low in both simulations and the real machine against the improved stealthy TCC, *i.e.*, lower than 5%. The reason is that the signal amplitude in the improved stealthy TCC is close to that of noise, and thus the detection in [6] fails to detect the improved stealthy TCC.

2) Evaluating the Countermeasure Scheme:

The BER performance of the proposed countermeasure with both detection and against the improved stealthy TCC attack is compared against that of the countermeasure proposed in [6] in the simulator simulating the 2D many-core system, and the results are shown in Fig. 20. The transmitter and the receiver are 1 hop away from each other, and the transmission frequency is set to be 30 Hz. One can see that with the countermeasure in [6], the BER of TCC is low, *i.e.*, 18.9%. The reason is that the threshold-based detection method in [6] cannot detect the improved stealthy TCC, and thus, this attack cannot be blocked. On the contrary, the proposed countermeasure can accurately locate the transmission frequency of the improved stealthy TCC with a detection accuracy of 89%. Jamming noise can precisely block the TCC transmission. Therefore, with the proposed countermeasure, the BER of TCC is higher than that with the countermeasure in [6] applied, which is close to 50%. A TCC experiencing such a high BER is literally shut down for meaningful data transmission.

D. Runtime and Power Overheads of the Countermeasure

The proposed countermeasure runs periodically (every two seconds) where 1500 thermal sensor data are recorded (with a sampling frequency of 1000Hz). Fast Fourier transform (FFT) [47] and the differential filter are performed with these thermal sensor data which take less than 8ms. It is negligible compared to running normal applications (benchmarks from PARSEC) whose execution times are usually tens of seconds or even more.

The average power consumption of the proposed countermeasure is 14.19W and is mainly contributed by the noise

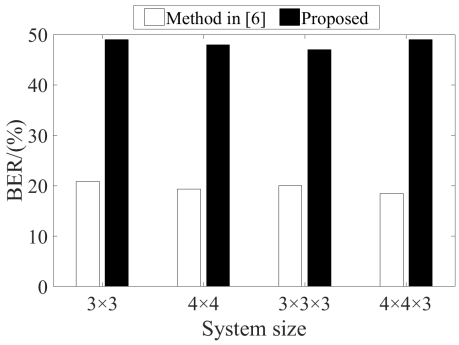


Fig. 20. BERs of the countermeasure method proposed in [6] and the proposed countermeasure method under different system configurations and sizes of the 2D and 3D many-core systems.

jamming module, which is only performed once a TCC is detected. In a 4×4 many-core system, the energy consumption overhead of the proposed countermeasure is lower than 0.32% of the total energy consumption of the whole system on average when running normal applications (benchmarks from PARSEC).

VI. Conclusion

This paper described an improved stealthy thermal covert channel attack that is based on DSSS to reduce the amplitude of the signal and make the signal appear to look like noise. To detect this type of stealthy thermal covert channel, the received signal is first filtered with a differential filter to remove the thermal noise at low frequencies and extract encoded information, after which a threshold-based decision is made regarding whether there is a potential thermal covert channel attack or not. The experimental results confirmed that the DSSS-based covert channel can defeat traditional threshold-based detection schemes. However, when the proposed detection and countermeasure is applied, the TCC's BER jumps to 50%, effectively shutting down the attacks.

References

[1] R. J. Masti, D. Rai, A. Ranganathan, C. Muller, L. Thiele, and S. Capkun, "Thermal covert channels on multi-core platforms," in Proc. USENIX Security Symp., pp. 865–880, 2015.

[2] D. B. Bartolini, P. Miedl, and L. Thiele, "On the capacity of thermal covert channels in multicores," in Proc. ACM Conf. Computer Systems, pp. 24–39, 2016.

[3] S. Chen, W. Xiong, Y. Xu, B. Li, and J. Szefer, "Thermal covert channels leveraging package-on-package DRAM," in Proc. IEEE Int'l Conf. Trust Security Privacy Comput. Commun., pp. 319–326, 2019.

[4] Z. Long, X. Wang, Y. Jiang, G. Cui, L. Zhang, T. Mak. "Improving the efficiency of thermal covert channels in multi-/many-core systems," in Proc. Design, Automation and Test in Europe Conf. and Exhibition, pp. 1459-1464, 2018.

[5] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in Proc. IEEE Int'l Symp. High-Performance Computer Architecture, pp. 171–182, 2001.

[6] 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 Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3276-3287, 2020.

[7] 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 Proc. Design Automation Conf., pp. 1-6, 2020.

[8] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: high-speed covert channel attacks in the cloud," in Proc. Symp. Usenix Security, pp. 9-23, 2013.

[9] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen, and R. Schlichting, "An exploration of L2 cache covert channels in virtualized environments," in Proc. ACM Cloud Computing Security Workshop, pp. 29–40, 2011.

[10] Y. Wang and G. E. Suh, "Efficient timing channel protection for on-chip networks," in Proc. IEEE/ACM Int'l Symp. Networks on Chip, pp. 142–151, 2012.

[11] G. Venkataramani, J. Chen, and M. Doroslovacki, "Detecting hardware covert timing channels," IEEE Micro, vol. 36, no. 5, pp. 17–27, 2016.

[12] S. Chen, W. Xiong, Y. Xu, B. Li, and J. Szefer, "Thermal covert channels leveraging package-on-package DRAM," in Proc. IEEE Int'l Conf. Trust Security Privacy Comput. Commun., pp. 319–326, 2019.

[13] 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," accepted for publication in IEEE Trans. Computers, doi: 10.1109/TC.2022.3160356.

[14] Q. Wu, X. Wang and J. Chen, "Defending against thermal covert channel attacks by task migration in many-core system," in Proc. IEEE Int'l Conf. Circuits and Systems, pp. 111-120, 2021.

[15] S. Haykin, Communication systems. John Wiley, 1978.

[16] L. Chang, F. Wang, and Z. Wang, "Detection of DSSS signal in non-cooperative communications," in Proc. IEEE Int'l Conf. Communication Technology, pp. 1-4, 2006.

[17] G. Saileshwar, C. W. Fletcher, M. Qureshi, "Streamline: a fast, flushless cache covert-channel attack by enabling asynchronous collusion," in Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems, pp. 1077-1090, 2021.

[18] J. Kaur, S. Das, "A survey on cache timing channel attacks for multicore processors," Journal of Hardware and Systems Security, vol. 5, no. 2, pp. 1-21, 2021.

[19] O. Aciicmez, J. Seifert, "Cheap hardware parallelism implies cheap security," Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 80-91, 2007.

[20] C. Hunger, M. Kazdagli, A. Rawat, A. Dimakis, S. Vishwanath and M. Tiwari, "Understanding contention-based channels and using them for defense," IEEE Int'l Symp. High Performance Computer Architecture, pp. 639-650, 2015.

[21] ARM, "Building a secure system using TrustZone technology", https://developer.arm.com/documentation/genc009492/c/.

[22] Victor Costan and Srinivas Devadas, "Intel SGX explained", http://eprint.iacr.org/2016/086.

[23] B. Lapid and A. Wool, "Cache-attacks on the ARM TrustZone implementations of AES-256 and AES-256-GCM via GPU-based analysis", https://eprint.iacr.org/2018/621.pdf?source=post_page.

[24] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "CLKSCREW: exposing the perils of security-oblivious energy management," in Proc. Usenix Secur. Symp., pp. 1057–1074, 2017.

[25] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "VoltJockey: breaching TrustZone by software-controlled voltage manipulation over multi-core frequencies," in Proc. ACM SIGSAC Conf., pp. 195-209, 2019.

[26] P. Qiu, D. Wang, Y. Lyu, and G. Qu, "VoltJockey: breaking SGX by software-controlled voltage-induced hardware faults," in Proc. Asian Hardware Oriented Security and Trust Symp., pp. 1-6, 2019.

[27] https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html.

[28] K. Wang, H. Zheng, and A. Louri, "TSA-NoC: learning-based threat detection and mitigation for secure network-on-chip architecture," IEEE Micro, vol. 40, no. 5, pp. 56-63, 2020.

[29] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss and F. Piessens, "Plundervolt: software-based fault injection attacks against Intel SGX," in Proc. IEEE Symp. Security and Privacy, pp. 1466-1482, 2020.

[30] D. Moghimi, B. Sunar, T. Eisenbarth and N. Heninger, "TPM-FAIL: TPM meets timing and lattice attacks," in Proc. USENIX Security Symp., pp. 2057-2073, 2020.

[31] T. Huo, X. Meng, W. Wang, C. Hao, P. Zhao, J. Zhai and M. Li, "Bluethunder: a 2-level directional predictor based side-channel attack against SGX," IACR Trans. Cryptographic Hardware Embedded Systems, vol. 2020, no. 1, pp. 321-347, 2020.

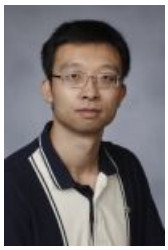
[32] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, S. Kottapalli, and S. Vora, "A 45 nm 8-core enterprise Xeon processor," IEEE J. Solid-State Circuits, vol. 45, no. 1, pp. 7–14, 2010.

[33] X. Wang, B. Zhao, T. Mak, M. Yang, Y. Jiang, and M. Daneshtalab, "On fine-grained runtime power budgeting for networks-on-chip systems," IEEE Trans. Computers, vol. 65, no. 9, pp. 2780–2793, 2016.

[34] J. Long, S. O. Memik, G. Memik, and R. Mukherjee, "Thermal monitoring mechanisms for chip multiprocessors," ACM Trans. Architecture and Code Optimization, vol. 5, no. 2, pp. 9–41, 2008.
[35] S. Paek, W. Shin, J. Lee, H.-E. Kim, J.-S. Park, and L.-S. Kim, "All-digital hybrid temperature sensor network for dense thermal monitoring," in Proc. Int. Solid-State Circuits Conf., pp. 260–261, 2013.
[36] "8th gen Intel® Core™ processor family datasheet." https://www.intel.com/content/www/us/en/products/docs/processors/core/8th-gen-core-datasheet-vol-1.html.
[37] S. Pan and K. A. 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 J. Solid-State Circuits, vol. 53, no. 12, pp. 3347–3355, 2018.
[38] S. Pan, C. Gurleyuk, M. F. Pimenta, and K. A. A. Makinwa, "A 0.12mm^2 wien-bridge temperature sensor with 0.1 °C (3σ) inaccuracy from -40 °C to 180 °C," in Proc. IEEE Int. Solid-State Circuits Conf., pp. 184–186, 2019.
[39] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," ACM Trans. Architect. Code Optim., vol. 11, no. 11, pp. 1–25, 2014.
[40] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in Proc. IEEE/ACM Int'l Symp. Microarchitecture, pp. 469–480, 2009.
[41] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in Proc. IEEE Int'l Symp. Computer Architecture, pp. 2– 13, 2003.
[42] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in Proc. IEEE Int'l Conf. Parallel Architectures and Compilation Techniques, pp. 72– 81, 2008.
[43] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in Proc. IEEE Int'l Symp. Computer Architecture, pp. 24–36, 1995.
[44] Y. S. Yang, J. H. Bahn, S. E. Lee, and N. Bagherzadeh, "Parallel and pipeline processing for block ciphe algorithms on a network-on-chip," in Proc. IEEE Int'l Conf. Information Technology: New Generations, pp. 849–854, 2009.
[45] C. Chao, K. Jheng, H. Wang, J. Wu and A. Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems," in Proc. ACM/IEEE Int. Symp. Networks-on-Chip, pp. 223-230, 2010.
[46] B. Li, X. Wang, A. K. Singh, and T. S. T. Mak, "On runtime communication- and thermal-aware application mapping in 3D NoC," in Proc. IEEE/ACM Int. Symp. Networks-on-Chip, pp. 1–8, 2017.
[47] C. S. Burrus and T. W. Parks, DFT/FFT and convolution algorithms: theory and implementation. John Wiley and Sons, New York, 1985.

Xiaohang Wang received the B. Eng. and Ph. D. degree in communication and electronic engineering from Zhejiang University, in 2006 and 2011, respectively. He is currently a Professor at South China University of Technology. He was the receipt of PDP 2015 and VLSI-SoC 2014 Best Paper Awards. He was the special session organizer of NoCS 2018, steering committee member of NoCArc 2014-2018, and TPC chair of ICCS 2021. He also served as the guest editor of the Mathematics, Integration, the VLSI Journal, Microelectronics Journal, and Computers and Electrical Engineering. His research interests include many-core architecture, power efficient architectures, optimal control, and NoC-based systems.

Shengjie Wang received the bachelor's degree in Software Engineering from South China University of Technology, Guangzhou, China. He is currently pursuing the master's degree in the School of Software Engineering. His research interests include hardware security and covert channel attacks.

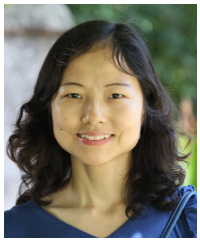


Yingtao Jiang received his Ph. D. in Computer Science from the University of Texas at Dallas in 2001, and joined the Department of Electrical and Computer Engineering (ECE), University of Nevada, Las Vegas (UNLV) as an assistant professor in the same year. He was promoted to the rank of full professor at the same department in 2013. He served as the ECE department chair between 2015 and 2018, and he is currently associate dean of UNLVs college of engineering. Besides STEM education, his research interests span a wide array of areas, including VLSI integrated circuit design, computer architectures, wireless networks, machine learning, cloud computing, biomedical engineering, and nanotechnologies.

Amit Kumar Singh is an Associate Professor at University of Essex, UK. He received the B.Tech. degree in Electronics Engineering from Indian Institute of Technology (Indian School of Mines), Dhanbad, India, in 2006, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore, in 2013. He was with HCL Technologies, India for a year and half until 2008. He has a post-doctoral research experience for over five years at several reputed universities. His current research interests are design and optimisation of multi-core-based computing systems with focus on performance, energy, temperature, reliability and security. He has published over 110 papers in reputed journals/conferences, and received several best paper awards, e.g., IEEE TC February 2018 Featured Paper, ICCES 2017, ISORC 2016, PDP 2015, HiPEAC 2013 and GLSVLSI 2014 runner up.

He is associate editor of IEEE Embedded Systems Letters, Design Automation for Embedded Systems, Journal of Low Power Electronics and Applications, and Frontiers in Neuroscience. He also edited a book for JLPEA journal and currently editing a special issue for JLPEA. He served as the publication chair of ESWeek-2021, Publicity co-chair of CF-2021, Local Co-chair of NASA/ESA Conference on Adaptive Hardware and Systems 2019, organized a special session at ESWeek-2021 and a tutorial at ESWeek-2018. He has served on the TPC of IEEE/ACM conferences like DAC, DATE, ICCAD, CASES and CODES+ISSS.

Letian Huang received the MS and Ph. D. degrees in communication and information system from the University of Electronic Science and Technology of China (UESTC), Chengdu, China in 2009 and 2016, respectively. He is an associate professor with UESTC. His scientific work contains more than 40 publications including book chapters, journal articles and conference papers. His research interests include heterogeneous multi-core system-on-chips, network-on-chips, and mixed signal IC design.



Mei Yang received her Ph. D. in Computer Science from the University of Texas at Dallas in Aug. 2003. In Aug. 2004, she joined in the Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, where she was promoted to full professor in 2016. Her research interests include computer architectures, interconnection networks, machine learning, and embedded systems.