On Countermeasures against the Thermal Covert Channel Attacks targeting Many-core Systems

***Abstract*—Although it has been demonstrated in multiple studies that serious data leaks could occur to many-core systems thanks to the existence of the thermal covert channels (TCC), little has been done to produce effective countermeasures that are necessary to fight against such TCC attacks. In this paper, we propose a three-step countermeasure to address this critical defense issue. Specifically, the countermeasure includes detection based on signal frequency scanning, positioning affected cores, and blocking based on Dynamic Voltage Frequency Scaling (DVFS) technique. Our experiments have confirmed that on average 98% of the TCC attacks can be detected, and with the proposed defense, the bit error rate of a TCC attack can soar to 92%, literally shutting down the attack in practical terms. The performance penalty caused by the inclusion of the proposed countermeasures is only 3% for an 8×8 system.**

***Index Terms*—Manycore Systems, Thermal Covert Channel, Defense against Covert Channel Attack, Signal Detection.**

I. Introduction

Chip-level security can be compromised by covert channels as they can help leak sensitive information while leaving legitimate users in the dark. Of many communication media seen in covert channels (e.g., heat transfer, timing, or inaudible sound), thermal covert channels (TCC) [1], [2] by the means of heat transfer can be particularly dangerous.

Like any other communication systems, a thermal covert channel involves a pair of transmitter and receiver. In the eight-core chip example illustrated in Fig. 1, assume that there is a covert channel between core A and core B. Note that core A sits in a secured zone where sensitive information is prohibited from being shared with the cores outside this zone, and core B is in a non-secured zone. A transmitter program running on core A can create temperature signals from the sensitive data (e.g., user passwords) by manipulating the activities and correspondingly the power consumption of processor core. In this example, bit '1' is represented as a sequence of rise and fall of temperature, while bit '0' is represented as no change in temperature. The temperature signals can travel through the chip by heat transfer among processor cores, and eventually the thermal signals reach the destination, core B, where they can be picked up by the local thermal sensor of core B, and subsequently, the original data can be recovered and sent to the attacker through network.

Since a thermal covert channel is committed to transmit thermal signals over a chip, it is prone to be disrupted/degraded by the thermal noise and ambient temperature variations. Recently, there have been new designs of thermal covert

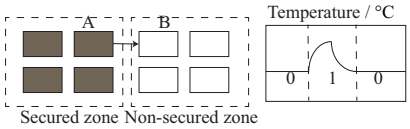

Fig. 1. TCC communication in an 8 multi-core system. The arrow from core A to core B indicates that the heat flow from core A to core B.

channels that show increase in both throughput and high immunity to noise. In a study [2], it was shown that use of a high transmission frequency, higher than the spectrum of the thermal noise, can deliver a throughput up to 20 bps and BER performance lower than 10% for an one-way transmission.

Apparently, the performance in thermal covert channels poses great risks and danger to data/information security in today's ubiquitous multi-core/many-core systems. Unfortunately, there is no effective countermeasure against this attack. In this paper, for the first time, we attempt to address this problem by proposing a countermeasure against thermal covert channel attacks based on an observation illustrated in Fig. 2.

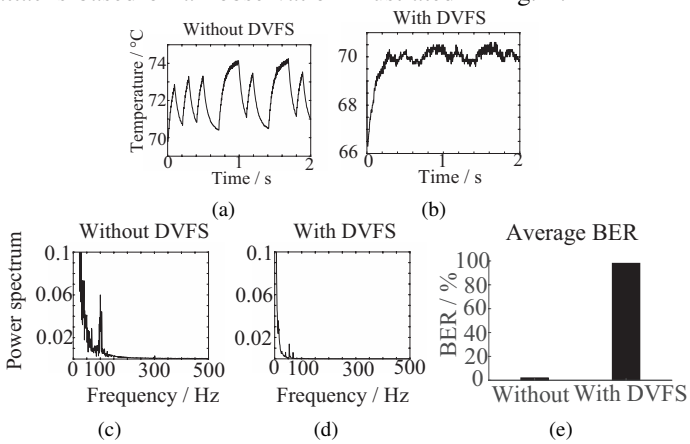

Fig. 2. Temperature timing diagrams and power spectrum diagrams of the receiver core without and with DVFS control on the transmitter core. (a) and (b) are temperature traces recorded by the receiver core. (c) and (d) are the corresponding power spectrums of thermal signals. In (a) and (c), the frequency level of the transmitter core is 2GHz. In (b) and (d), the frequency level of the transmitter core is scaled down to 500MHz in the first 2 seconds. (e) shows the comparison of the bit error rates of the receiver with and without DVFS applied on the transmitter core.

Assume the 4-core system has a thermal covert channel with a transmission frequency of 100Hz. The frequency level (CPU working frequency) of the transmitter core is 2GHz, which is a normal frequency level of modern many core systems. If no countermeasure is applied, the temperature signal and its power spectrum are shown in Figs. 2(a) and 2(c), respectively. If the frequency level of the transmitter core is decreased to a low level (e.g., 500MHz), the thermal signal has a significantly different profile, as shown in Figs. 2(b) and 2(d). In this case,

no signal around the 100Hz can be detected, as the bit error rate (BER) of the TCC reaches 98% (Fig. 2(e)).

Since a high BER is preferred to thwart a TCC attack, the high BER in Fig. 2(e) indicates that DVFS control can be exploited to defend against TCC attack. However, blindly applying this technique may lead to significant performance loss. In the worst case, when all the suspect cores run at their lowest frequency levels (e.g., 500MHz), the performance can be unacceptably low. This issue is thus addressed in this paper also. Essentially, the proposed DVFS-based countermeasure includes the following steps.

1. First, a detection scheme is applied to check if any thermal covert channel attack exists in a many-core system.

2. Second, the exact locations of the transmitter and receiver cores in the TCC are located.

3. Third, a countermeasure based on DVFS is applied to suppress the TCC signal transmission.

The rest of this paper is organized as follows. Section II presents the preliminaries and previous works that are related to thermal covert channel attacks. Section III presents the details regarding the defense against TCC. In section IV, the proposed countermeasure is evaluated. Finally, section V concludes this paper.

II. PRELIMINARIES AND RELATED WORK

A. Threat Model

Temperature sensors are essential for dynamic thermal management [4] and are widely deployed in chips. The number and accuracy of temperature sensors are possible to be ever improved in the future [5]. The attacker (being the thermal covert channel transceiver) can read the local thermal sensor data by calling the software interfaces (e.g., Intel's processor Model Specific Register, MSR) with a typical resolution of 1°C [6]. The accuracy of some latest digital thermal sensors can be up to 0.1°C. The defender (being the global manager core) can also read all cores' temperatures and distribute the detection and blocking programs to each core.

B. Communication Flow

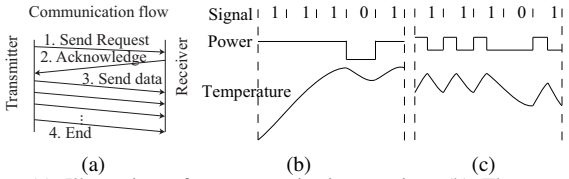

Fig. 3. (a) Illustration of a communication session. (b) The waveform of temperature variations without the NRZ scheme. (c) The waveform of temperature variations with the NRZ scheme.

The transmitter and the receiver first need to reach an agreement on the transmission frequency for the thermal covert channel. Flow of a communication session in [2] is shown in Fig. 3(a). First, to initiate a session, the transmitter sends a request packet to the receiver and waits for a reply. Second, when the receiver identifies that the received packet is a request packet, it replies with an acknowledge packet to the transmitter. Third, if the transmitter doesn't receive an acknowledge packet within a given amount of time, it resends another request packet. Otherwise, it transmits data packets to the receiver and sends a termination packet to mark the end of this current session. Finally, the receiver keeps receiving data packets sent by the transmitter until it gets a termination packet that terminates a session.

C. Transmitter and Receiver

To finish a communication session, the transmitter reads the sensitive data and packetizes them with a specific preamble filed(e.g., 101010), which marks the beginning of a packet, and an Error Correcting Code (ECC) field. Basically, the packet to be transmitted dictates the transmitter core's program to generate temperature signals by letting the core run computation-intensive code to boost the temperature or insert idle CPU cycles to cool down the chip.

To make the temperature variations periodical, the non-return-to-zero (NRZ) scheme is employed to avoid the excessive temperature ramp that otherwise may lead to overheat caused by transmitting continuous '1's, as shown in Fig. 3(b). That is, a '1' is represented as a rise in temperature followed by a fall, as shown in Fig. 3(c). The duration to transmit a single bit '1' consists of two times, t_h and t_l:

$$t_b = t_h + t_l, t_h \quad t_l \tag{1}$$

where t_h is the duration of the thermal signal in a high temperature, and t_l is the duration of that in a low temperature. Bit '0' is represented as a low temperature with a duration of t_b.

At the receiver side, it picks up the temperature signal from its local temperature sensor, and converts the thermal signal into binary bit streams. Next, the receiver examines the ECC code for packet integrity. If the received packet is determined uncompromised and contains a specific preamble during the transmission, the receiver extracts the data fields from the packet; otherwise, the receiver discards the binary bit streams. If the current packet doesn't contain a preamble, it leads to a bit error rate of 100%.

D. Related Work

Thermal covert channel attacks impose a major security threat against computer systems. A thermal covert channel doesn't rely on any shared resources such as cache and memory, which helps it circumvent modern system's defense. Masti et al. [1] showed a thermal covert channel could transmit data with a bit error rate of 11% and a throughput of 1.33 bps. Bartolini et al. [3] improved the efficiency of thermal covert channel with a bit error rate less than 1% and a throughput of more than 5bps over a 1-hop channel. Furthermore, the work in [2] showed that a properly high transmission frequency can significantly avoid the noise from the heat from other active cores and result in an even higher transmission speed.

Although thermal covert channel attacks are perceived to impose an increasingly big security threat to today's multi-core system, yet there has little work done to develop a defense or countermeasure against it. In the next section, a frequency-scanning-based detection and DVFS-based countermeasure is thus introduced to close this gap.

TABLE I
NOTATIONS USED IN ALGORITHMS 1 AND 2

T_i	The temperature trace of core i recorded in a detection cycle.
ρ	The preset threshold for the signal amplitude in a detection cycle.
F	The set of available frequencies of the thermal covert channels to be scanned in a detection cycle.
$4f$	Frequency increments.
n_c	Number of cores in the SoC.
mp_i	A variable recording the largest signal amplitude within Band B of core i in a detection cycle and it is initialized to be 0.
mf_i	A variable recording the signal frequency of core i that matches mp_i and it is initialized to be 0.
L	An initially empty list. It adds the position of a core that is considered as the most suspicious one in every positioning process.
SF_i	An initially empty list. It adds a thermal signal frequency, which matches the maximum amplitude within Band B of a core i in every positioning process.
δ_t	Error of transmission frequency.

III. COUNTERMEASURES AGAINST THERMAL COVERT CHANNEL ATTACKS

To fight against the thermal covert channel attack, the proposed countermeasure essentially consists of three major steps: detection, positioning and blocking. A detection unit time in the detection step, is hereinafter referred as a detection cycle, and all three steps are initiated by a core called global manager as shown in Fig. 4(a). The notations used in Algorithms 1 and 2 are listed in Table I.

A. Detection

Detection of a thermal covert channel attack is managed and coordinated by a core referred as the global manager, as shown in Fig. 4(a). Essentially, the global manager assigns the detection jobs to each core, and each core performs the detection individually and reports to the global manager whether there is a possible attack or not. In each detection

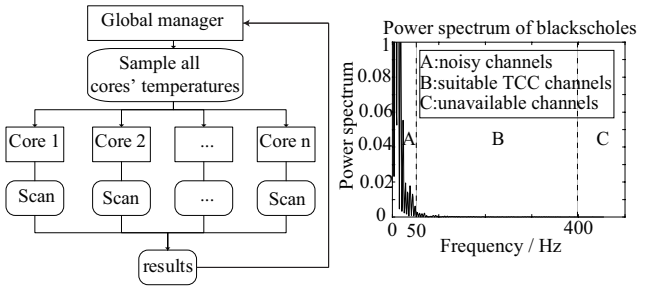

Fig. 4. (a) Detection scheme. (b) The power spectrum of a core's temperature signal obtained when one typical application runs.

cycle, the global manager samples and records each core's temperature values for one second with a given sampling frequency, followed by commanding each core to use its own temperature trace to execute a frequency scanning process to detect any existing TCC channels. After receiving the command, if a core is running a task, the core will pause the current task and run the frequency scanning based detection. After the core finishes the detection and reports the results to the global manager, it continues its original task.

1) Spectrum of TCC Signals: Fig. 4(b) shows the power spectrum of the temperature signals of a core when a typical application (e.g., Blackscholes from PARSEC) runs. From Fig. 4(b), the spectrum of a thermal covert channel can be divided into three bands: Band A that spans DC to 50Hz, Band B from 50 to 400Hz, and Band C that is higher than 400Hz (cut-off frequency) and considered unavailable for data transmission. A typical TCC channel, with signal amplitude higher than 0.01dB, will select transmission frequency from Band B rather from Band A to reduce noise interference, since TCC signal in Band A can be easily disrupted by the thermal signal generated by a normal application. Therefore, in the aspect of defense against TCC attack, we only detect possible TCC channels within Band B.

2) Frequency-scanning-based Detection of TCC Channels: Over a detection cycle, the system checks which signal frequencies that are available for TCC channels. Essentially, at the beginning of a detection cycle, the global manager sends a command to each core. Upon receiving the command, a core uses its own tunable bandpass filter, with its center frequency tunable within Band B and a fixed bandwidth, to process the thermal signals, and each core makes a decision based on the amplitudes of the output signal from the bandpass filter.

Algorithm 1 Frequency scanning based detection at core $i(1 \quad i \quad n_c)$.

Input: T_i, ρ, F, and $4f$.
Output: mp_i and mf_i.

1: Initialize mp_i and mf_i to be 0.
2: **for** f 2 F **do**
3: Run signal T_i through a bandpass filter with center (resonant) frequency of f, and bandwidth of $4f$, and get the signal amplitude $p(f)$.
4: **if** $p(f) > mp_i$ **then**
5: $mp_i = p(f)$ and $mf_i = f$
6: **end if**
7: $f = f + 4f$
8: **end for**
9: **if** $mp_i \quad \rho$ **then**
10: $mp_i = 0, mf_i = 0$, and send a message to the global manager that no TCC channel is found.
11: **end if**
12: Report mp_i and mf_i to the global manager.

As listed in Alg. 1, the frequency scanning based detection algorithm works as follows.

Step 1. The global manager initiates a detection cycle that tries to find if there is any possible TCC channel; upon receiving the command from the global manager to start a new detection cycle, each core uses a bandpass filter to scan the entire spectrum of Band B in a linear sweeping fashion with an increment of $4f$ (see line 2 and line 7 in Alg. 1). The best $4f$ is experimentally set to be 4Hz. Tuning to a center frequency f, this bandpass filter processes the thermal signals and outputs a signal with amplitude of $p(f)$, as defined in line 3 in Alg. 1.

Step 2. Each core determines the maximum amplitude of the signal from the frequency scanning process (see lines 4 to 6 in Alg. 1), and feeds it into a decision-making module. The decision-making module decides whether a signal is from a covert channel or not by comparing the signal's amplitude against a threshold ρ. If the signal amplitude is higher than ρ, an attack is deemed as present; otherwise, no attack is present. Statistically, the threshold ρ is chosen experimentally to be 0.02dB.

Step 3. At the end of a detection cycle (lines 9 to 12 in Alg. 1), if a core confirms that a TCC attack is present, it reports its findings, including the signal frequency and signal amplitude, to the global manager. Otherwise, it advises to the global manager that no TCC attack has been found in current detection cycle.

Step 4. If the global manager finds no TCC channel exists in any of the cores, it will hold back for a short time interval (e.g., 2 seconds) before initiating the next detection cycle that will start from Step 1 again.

If the global manager receives a notification of the existence of a TCC channel, it begins a process to locate the cores that have been affected by this TCC, as described in Section III-B.

B. Positioning Cores Affected by A Thermal Covert Channel Attack

When a core is transmitting, the core has the maximum signal amplitude within Band B among all the cores. Correspondingly, after each detection cycle, the global manager can get the position of the most suspicious core that has the highest signal amplitude. Note that there is a very low probability that a legitimate application may generate the same thermal signals as a TCC attack. As so, the global manager cannot immediately prescribe DVFS control to the suspicious core until it is confirmed to have a pairing core running on the same transmission frequency. To check if a suspicious core has a pairing core running on the same transmission frequency, the global manager initiates several detection cycles and each detection cycle is followed by one positioning phase that attempts to locate the core. This positioning phase (listed as Alg. 2) involves 3 major steps given below.

Step 1. The core that has the largest value among all the $mp_i (1 \quad i \quad n_c)$ is located (line 1), where mp_i is the maximum signal amplitude within Band B of core i, and n_c is the number of cores in the system. This core, referred as core mc, is considered as the most suspicious core that is transmitting TCC thermal signals, with a signal amplitude of mp_{mc} and the transmission frequency of mf_{mc}, when $mp_{mc} > \rho$. Here ρ is the preset threshold for the signal amplitude, mf_{mc} is the signal frequency that matches mp_{mc} of core mc in a detection cycle.

Step 2. The position of a suspicious core is decided, following the rules defined in lines 2 through 5. If the signal amplitude is strong enough, that is, mp_{mc} is higher than the preset threshold, and there is another core l 2 L which is listed in the most suspicious cores list L in the history positioning phases and meets the following conditions,

Algorithm 2 Locate the positions of the TCC transmitter core and receiver core.

Input: L, SF_i, δ_t, mp_i, and mf_i.

Output: The positions of transmitter (l_1) and receiver (l_2) cores, and transmission frequency (f_t).

1: $mc = \arg\max_i mp_i$.
2: **if** $mp_{mc} > \rho$ **then**
3: **if** 9l 2 L and 9f 2 SF_l and mc 6= l and jmf_{mc} fj $<$ δ_t **then**
4: $l_1 = l, l_2 = mc, f_t = (f + mf_{mc})/2$
5: **end if**
6: Add mf_{mc} to SF_{mc} and add mc to L.
7: **end if**
8: **if** l_1 6= $NULL$,and l_2 6= $NULL$ **then**
9: **return** $l_1, l_2,$ and f_t
10: **else**
11: Initiate a new detection cycle.
12: **end if**
13: **return**

9f 2 SF_l and jmf_{mc} fj $< \delta_t$

then the transceiver cores, l and mc, are considered as being found. Here L is a list that stores the most suspicious cores in the history positioning phases; SF_l is a list that adds a thermal signal frequency, which matches the maximum amplitude within Band B, of the core l in every positioning phase; δ_t is the estimation error of the transmission frequency. The transmission frequency f_t is set as the average of f and mf_{mc} (line 4).

Step 3. At the end of each detection cycle, the most suspicious core position mc is added to list L and the transmission frequency mf_{mc} is added to SF_{mc} (line 6 in Alg. 2). If the global manager manages to locate the transmitter core and the receiver core as well as the transmission frequency, it stops the positioning process (lines 8 through 9), and starts the blocking phase, as described in Section III-C.

The positioning process is suitable for all communication protocols since a transmitter and the pairing receiver must use the same transmission frequency to communicate.

C. Blocking the Thermal Covert Channel Attack

Once a core is identified to be part of a TCC, a countermeasure shall be implemented to block any future TCC transmission. Note that a countermeasure that completely shuts down the cores participating in a TCC attack is not always a viable solution. In today's multi-core processors that multi-threading is becoming commonplace, more than likely, both the transmitter core and the receiver core in a TCC may also run legitimate threads besides the TCC threads. In this case, turning off these cores indiscriminately also kill those legitimate threads, and thus lead to an unacceptable performance loss when all the computing resources are so much needed to handle heavy workload.

Here we adopt a different countermeasure strategy that takes into account of both security requirements and system

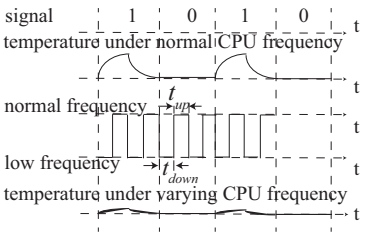

Fig. 5. An example showing the blocking scheme.

performance issues. Specifically, the thermal covert channel transmission is blocked by altering the frequency level of the transmitter core and also that of the receiver core. Scaling down the frequency level can effectively change the temperature pattern so that the bit error rate (BER) of the receiver will jump to an extremely high level, say over 90%, so that no meaningful TCC communication is possible (no correct data is possibly received). Lowering down frequency level of a core certainly has a negative implication on the chip's performance. As a tradeoff, we experimentally choose 500MHz as the lowest level since the TCC communication sees a high BER (e.g., more than 60%) when the frequency level of the transmitter core is lower than or equals to 500MHz. Then, two blocking methods are proposed: 1) A simple method that lowers down the frequency level of the TCC core at 500MHz all the time. 2) An advanced method that further reduces performance loss. As for the advanced method, a variable down-up rate, β, is introduced, and it is defined as the ratio of the temporal duration t_{down} of the lowest frequency level to temporal duration t_{up} of the normal frequency level. That is,

$$\beta = t_{down}/t_{up}$$

As shown in Fig. 5, when the frequency level of the transmitter core is running at its typical frequency level (e.g., 2GHz), the temperature signal may see a sharp rise and fall transition when encoding signal '1'; however, the signal amplitude is much lower than before when the frequency of the transmitter core drops to a lower level (e.g., 500MHz) for a duration of t_{down} and then returns to its typical value for a duration of t_{up}. By enforcing the down-up rate, the frequency level goes down (to 500 MHz in Fig. 5) and then up (back to 2 GHz) periodically.

To first ensure the security, the frequency level of the transmitter core should be scaled down at most of the time when the transmitter generates high temperatures to transmit signal '1'. According to the NRZ scheme and Eqn. 1, which is mentioned in section II-C and shown in Fig. 3(c), one can see that the duration t_h to generate high temperatures when transmitting signal '1' is less than $0.5t_b$, thus we can ensure that the DVFS control is applied at every interval when the transmitter generates high temperatures by limiting the sum of t_{down} and t_{up} to $0.5t_b$.

The blocking process has the following major steps.

Step 1. The global manager calculates the duration t_b of one signal duty cycle according to the dectected transmission frequency f_t. (i.e., $t_b = 1/f_t$)

Step 2. During each time $0.5t_b$, the global manager first scales down the frequency level of the transmitter core and receiver core for a duration of t_{down}, and then lets the frequency level of that core back to its normal level.

Step 3. The global manager initiates a detection process every t seconds (e.g., 2 seconds), to check whether there is another TCC channel or not. If it finds another TCC channel, it goes back to the positioning process while keeping the current TCC transmission to remain blocked.

TABLE II
CONFIGURATIONS OF THE EXPERIMENTS

Sniper configuration	
Instruction set architecture	x86-64
Number of cores	3 3 / 4 4 / 8 8
Number of SMT threads per core	2
Technology node (nm)	22
Frequency and voltage levels of CPUs (MHz/V)	2000/1.0, 1000/0.7, 700/0.68, 600/0.66, 500/0.64, 250/0.6
Configuration for integrated Hotspot	
Chip thickness	$0.15mm$
Silicon thermal conductivity	$100W/(m \quad K)$
Silicon specific heat capacity	$1.75 \quad 10^6 J/(m^3 \quad K)$
Heat sink thickness	$6.9mm$
Heat sink thermal conductivity	$400W/(m \quad K)$
Specific heat capacity of heat sink	$3.55 \quad 10^6 J/(m^3 \quad K)$
Configuration for TCC programs	
Transmission frequency	100 Hz, 150 Hz, 200Hz, etc.
Number of bits per packet	20
Preamble of a packet	101010
Preset transmission bit rate	10 bps
Distance between transmitter and receiver	1 hop

IV. EXPERIMENTAL EVALUATION

Our experiments are performed on a many-core simulator, Sniper [7], with McPAT [8] integrated as the power model. We also integrate Hotspot [9] as the thermal model to dynamically generate temperatures for all the cores. We choose 10 benchmarks from PARSEC [10] and SPLASH2 [11] and they will be treated as the thermal noise that may interfere a TCC. Two separated cores are set to be part of a TCC, while all the other cores run the legitimate threads of the selected benchmarks. Specifically, each core runs two simultaneous-multithreading (SMT) threads, and both the transmitter core and the receiver core run a TCC thread and a thread of the benchmarks. The detailed configuration of Sniper, Hotspot and TCC programs are tabulated in Table II. The floorplan of the processor cores follows the one reported in [12].

A. Experimental Results

From Fig. 6(a), one can see that the average BERs of a TCC communication, with a transmission frequency of 100Hz and CPU frequency level of 2000MHz, are below 7% for all the system sizes when the DVFS control is not applied to the TCC cores. In this case, the TCC poses a major threat to the many-core systems.

From Fig. 6(b), one can see that, the average accuracy of positioning transceiver (P_{acc}) is 100% and the average accuracy of estimating detection transmission frequency (f_{acc}) is 98%. The accuracy of estimating the transmission frequency is denoted as f_{acc}, and the positioning accuracy is denoted as P_{acc},

$$f_{acc} = (1 \quad j(f_t \quad f_{detected})/f_t j) \quad 100\% \qquad (2)$$

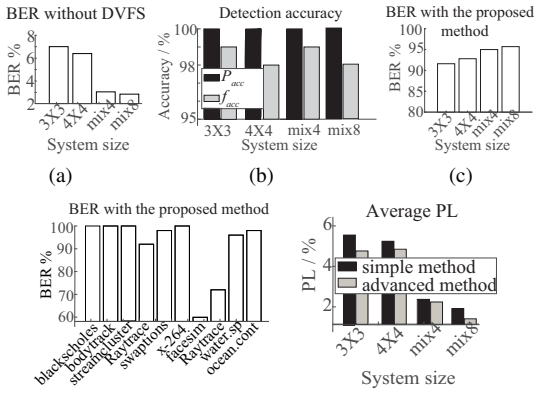

Fig. 6. (a) The average BERs of a TCC communication with different system sizes. Mix4 (Mix8) is the case that the thermal noise seen by a TCC in an 8 8 system is from a mixture of 4 (8) multi-threaded benchmarks. (b) Detection accuracy, including the accuracy of positioning the transmitter core (P_{acc}) and estimating transmission frequency (f_{acc}), of the TCC communication under different system sizes. (c) The average BER results with the proposed method under different system sizes. (d) The average BER results with the proposed method under different benchmarks' noise in a 4 4 system. The former seven benchmarks are from PARSEC and the latter three are from SPLASH2. (e) The average performance loss (PL) results with different system sizes. The setting of transmission (signal) frequency of TCC programs is 100Hz.

$$P_{acc} = \begin{cases} 100\% & P_{detected} = P_{transmitter|receiver} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where f_t is the real transmission frequency of TCC signals and $f_{detected}$ is our detected signal frequency; $P_{detected}$ is our detected position (core id) of TCC core and $P_{transmitter|receiver}$ is the actual position (core id) of transmitter core or receiver core. By using the proposed detection strategy, we can always figure out whether there is a TCC attack or not in a many-core system and precisely detect its position and transmission frequency.

With the proposed countermeasure, once a TCC attack is detected, the TCC transmitter core's frequency level is changed with a down-up rate β of 9. From Fig. 6(c), the average BER of the TCC attacks across systems with different sizes is higher than 92%. Specifically, from Fig. 6(d), one can see that, in a 4 4 system, the average BER of TCC increases to 60%, and, in most cases, it even reaches 100%. With such high BER, no meaningful communications can be sustained; that is, our defense strategy can effectively shut down TCC attacks. In this experiment, the global manager confirms that it can no longer find any TCC channel. Fig. 6(e) shows the average performance loss (PL) of normal applications. Our proposed advanced blocking method (in section III-C) with a down-up rate β of 9 (β=9) is compared against our simple blocking method (in section III-C) that keeps the transmitter core at the lowest frequency level (i.e., 500MHz) all the time once a TCC attack is detected. The performance loss (PL) is defined as:

$$PL = \frac{\pi_{avg} \quad \pi_0}{\pi_{avg}} \quad 100\% \quad (4)$$

where π_{avg} is the average performance (instruction per cycle) of an application when no DVFS control is applied, and π_0 is the average performance of an application when DVFS control is in effect. One can see that the average PL results of our two blocking methods are all below 6% for different system sizes. Further, the advanced method reduces PL of 15% on average than that of the simple method. In a large many core system (e.g., 8 8), the average performance loss of our proposed countermeasure is at a low level (below 3%, for example) and it is acceptable.

The main runtime overhead is incurred by the detection phase. In our experiments, the average runtime overhead of a detection cycle is below 0.05 second, which is several orders of magnitude lower than the break interval (e.g., 2 seconds) between two detection cycles. Therefore, the runtime overhead of our proposed countermeasure is acceptable.

V. CONCLUSION

In this paper, a countermeasure was proposed to defend against TCC attacks that may impose severe security threats to many-core systems. The proposed countermeasure scheme includes three major steps: 1) A frequency scanning method is applied to examine the temperature traces to identify any possible TCC attack by comparing the signal amplitudes with the thresholds. 2) The TCC cores are located as these cores need to communicate using the same transmission frequency. 3) DVFS is applied to the cores participating in a TCC to block its communications. Experimental results have confirmed that the proposed countermeasures could cause TCC attacks to suffer extremely high BER ($> 92\%$), while the multi-core system only experienced a very modest loss ($< 3\%$) in an 8 8 many-core system. The proposed countermeasure is thus a suitable scheme that can be deployed in many-core systems facing possible TCC attacks.

REFERENCES

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

[2] Z. Long, X. Wang, Y. Jiang, G. Cui, L. Zhang, and T. S. T. Mak, "Improving the efficiency of thermal covert channels in multi-/many-core systems," in *DATE*, pp. 1459–1464, 2018.

[3] D. B. Bartolini, P. Miedl, and L. Thiele, "On the capacity of thermal covert channels in multicores," in *EuroSys*, pp. 24:1–24:16, 2016.

[4] J. Shor, K. Luria, and D. Zilberman, "Ratiometric bjt-based thermal sensor in 32nm and 22nm technologies," in *ISSCC*, pp. 210–212, 2012.

[5] 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 *ISSCC*, pp. 260–261, 2013.

[6] "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.

[7] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *SC*, pp. 52:1–52:12, 2011.

[8] 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 *MICRO*, pp. 469–480, 2009.

[9] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *ISCA*, pp. 2–13, 2003.

[10] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *PACT*, pp. 72–81, 2008.

[11] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *ISCA*, pp. 24–36, 1995.

[12] X. Wang, P. Liu, M. Yang, M. Palesi, Y. Jiang, and M. C. Huang, "Energy efficient run-time increment mapping for 3-d networks-on-chip," *J. Comput. Sci. Technol.*, vol. 28, no. 1, pp. 54–71, 2013.