Detection of Thermal Covert Channel Attacks Based on Classification of Components of the Thermal Signal Features

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

Abstract—In response to growing security challenges facing many-core systems imposed by thermal covert channel (TCC) attacks, a number of threshold-based detection methods have been proposed. In this paper, we show that these threshold-based detection methods are inadequate to detect TCCs that harness advanced signaling and specific modulation techniques. Since the frequency representation of a TCC signal is found to have multiple side lobes, this important feature shall be explored to enhance the TCC detection capability. To this end, we present a pattern-classification-based TCC detection method using an artificial neural network that is trained with a large volume of spectrum traces of TCC signals. After proper training, this classifier is applied at runtime to infer TCCs, should they exist. The proposed detection method is able to achieve a detection accuracy of 99%, even in the presence of the stealthiest TCCs ever discovered. Because of its low runtime overhead ($< 0.187\%$) and low energy overhead ($< 0.072\%$), this proposed detection method can be indispensable in fighting against TCC attacks in many-core systems. With such a high accuracy in detecting TCCs, powerful countermeasures, like the ones based on dynamic voltage and frequency scaling (DVFS), can be rightfully applied to neutralize any malicious core participating in a TCC attack.

Index Terms—Thermal Covert Channel Attack, Many-core System, Defense against Covert Channel Attack

1 Introduction

Among a wide range of security challenges facing today's many-core chips, thermal covert channel attacks are found particularly dangerous and difficult to deal with, and even more so, as numerous innovative methods and techniques have been attempted to enhance the transmission performance and/or stealthiness of TCCs. For instance, by switching from non-return-to-zero (NRZ) [1] to Manchester code [2] in encoding scheme, a TCC channel witnesses its bit error rate (BER) drops from 11% [3] to 1% [2], while the throughput gets a boost to 5 bits per second (bps). As hyper-threading becomes commonplace in modern many-core systems, a TCC channel described in [2] can even deliver a transmission rate higher than 45bps. In [3], a TCC attack over a real computer was demonstrated. In another study [4], cloud users who share the same FPGA chips can suffer from secret information leak with the help of TCC. These TCC attacks in real machines manifest the scope and seriousness of TCC attacks.

To fight against TCC attacks, the dynamic voltage and frequency scaling (DVFS) or noise jamming based countermeasures that are supposedly to block any meaningful thermal signal transmission can be applied [5], [6]. Both countermeasures rely on the threshold-based detection methods to detect the existence of a TCC. Essentially, a TCC is deemed as present if the amplitude of the thermal signal exceeds a well-defined threshold.

As TCC becomes stealthier with reduced signal amplitudes, the threshold-based detection methods find themselves inadequate to discover TCC attacks. For example, when a TCC needs to generate a high temperature within a signal period of t_b, if the heat up time is reduced from $0.5t_b$ (the case shown in Fig. 1(a)) to $0.1t_b$ (the case shown in Fig. 1(b)) and the cool down time is increased from $0.5t_b$ (the case shown in Fig. 1(a)) to $0.9t_b$ (the case shown in Fig. 1(b)), the signal amplitude is so low that the threshold-based detection schemes proposed in [5], [6] will fail to detect any TCC, pushing down the detection accuracy to almost 0 as indicated in Fig. 1(c). Countermeasures with such a low detection accuracy are unacceptable, since blindly generating thermal noise or applying DVFS to cores especially those running legitimate applications will lead to significant performance loss. In this paper, we define the radio of the heat up time to the cool down time as α, and the value of α

- X. Wang is with the School of Software Engineering, South China University of Technology, and also with State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences (e-mail: xiaohangwang@scut.edu.cn). Xiaohang Wang is the corresponding author.
- H. Huang and R. Chen are with the School of Software Engineering, South China University of Technology, China (e-mail: sehenry@mail.scut.edu.cn, rola@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.jiang@unlv.edu, mei.yang@unlv.edu)
- A. K. Singh is with the School of Computer Science and Electronic Engineering, University of Essex, UK. (e-mail: a.k.singh@essex.ac.uk)
- L. Huang is with the University of Electronic Science and Technology of China, Chengdu 610054, China. (e-mail: huanglt@uestc.edu.cn)

This research program was supported in part by the National Natural Science Foundation of China under Grant 61971200, in part by Fundamental Research Funds for the Central Universities South China University of Technology under Grant 2019MS087, in part by Open Research Grant of State Key Laboratory of Computer Architecture, Institute of Computing Technology Chinese Academy of Sciences under Grant CARCH201916, in part by Key Laboratory of Big Data and Intelligent Robot (South China University of Technology), Ministry of Education, in part by Zhejiang Lab under Grant 2021LE0AB01, in part by National Key Research and Development Program of China under Grant 2019QY0705.

is important for the improved stealthy TCC.

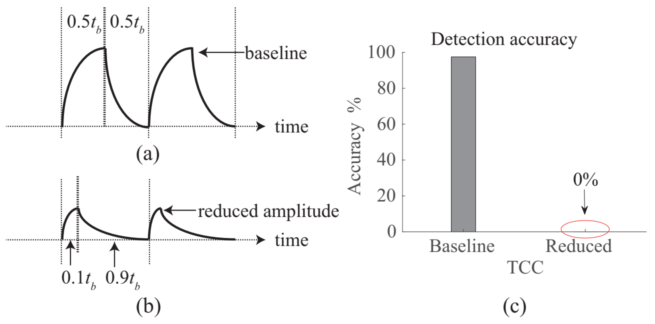


Fig. 1. (a) and (b) are the waveforms of temperatures signals when transmitting bits "11" under the return-to-zero (RZ) [7] encoding schemes. (a) The thermal pulse has a 50% duty cycle. (b) The thermal pulse has a 10% duty cycle with reduced peak value. (c) The detection accuracy of the baseline TCC that follows signaling in (a) vs. the detection accuracy of the TCC that has reduced signal amplitude as (b).

Although the improved stealthy TCC in Fig. 1(b) can circumvent the threshold-based detection, the TCC signals are found to have multiple side lobes of high amplitudes, a fairly consistent feature that can be explored for the sake of TCC detection. Correspondingly, we present a pattern recognition algorithm to classify (detect) TCC attacks out of thermal noise in this paper. This proposed pattern-classification-based TCC detection method is able to detect all the known TCC attacks and variants with a high detection accuracy. The contributions of this paper are as follows.

1) An improved stealthy TCC is designed such that the existing threshold-based detection methods are no longer useful.
2) The frequency components of the TCC signals are analyzed, and the features drawn from the side lobes are explored for TCC detection. A neural-network-based classifier is thus developed, trained, validated, and tested for TCC detection.
3) Experimental results show that the proposed detection scheme can detect TCCs with an accuracy of as high as 99%, and the runtime overhead is very low.

The remainder of this paper is organized as follows. Section 2 presents the previous works on thermal covert channel attacks. Sections 3 and 4 present the details regarding the possible designs and detection of the improved stealthy TCC, respectively. In section 5, the proposed detection scheme is experimentally verified, and the results are reported. Finally, section 6 concludes this paper.

2 PRELIMINARIES AND RELATED WORK ON DETECTION OF THERMAL COVERT CHANNELS

2.1 Baseline Thermal Covert Channels

With heat as communication media and no shared resources (*e.g.*, cache and memory), the thermal covert channel attacks can be launched in many-core systems [2], [3], [5], [7] more easily than other types of covert channels. A typical thermal covert channel is modeled to include a transmitter and a receiver as well as a defender, shown in Fig. 2.

As shown in Fig. 2 (a), a TCC attack has a pair of transmitter and receiver programs. The transmitter is in the secure zone of the system and is able to obtain sensitive data. The receiver is in the unsecure zone and does not have direct access to the sensitive data. The transmitter encodes the bit stream of the sensitive data into temperature variations. For example, bit '1' is encoded by a rise and fall in temperature, and bit '0' is encoded by staying at a low temperature, as shown in Fig. 2 (b). The thermal signals are generated by running either computation-intensive codes for heating-up or keeping the core idle for cooling-down. The receiver on the other end of the communication link reads its thermal sensor and decodes the sensitive data originated from the transmitter.

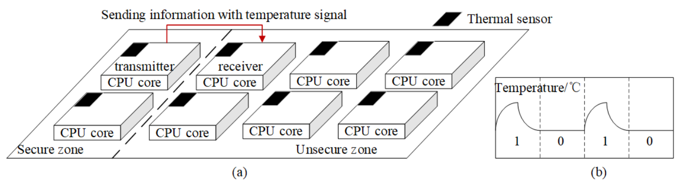


Fig. 2. (a) A TCC attack example showing a pair of transmitter and receiver. (b) Thermal signal after data encoding.

The transmitter program in a TCC can be implanted into a secure zone, which is supported by technologies like ARM TrustZone [1] and Intel software-guard extensions (SGX) [8] through software updates or other means [9]. For example, before a user application is loaded into its own SGX enclave, the transmitter codes can be injected into the user application as a Trojan [9], resulting in that the transmitter is able to run in the secure zone and has access to private data. After implanted into the secure zone, a TCC program can leak private data by deliberately manipulating chip temperatures [2], [3], [5], [6]. The thermal signals can be obtained either by directly reading the thermal sensors through MSR (*i.e.*, Model Specific Register) software interface [10] or by reading the temperature files exposed by some commonly installed temperature-monitoring utility tool (*e.g.*, CoreTemp [11] in Linux system). The thermal sensors nowadays are technologically fine-tuned, with several precisions like 1°C in [10] and 0.12°C in [12].

The TCC receiver program, which is outside the secure zone, reads the thermal signals from its local thermal sensor, decodes the bitstream back into sensitive data originated from the transmitter, and delivers the data to the hacker through the network.

In fighting against the TCC programs, a defender runs a program that can detect all cores' workload traces and can access all thermal sensors. The defender is granted ROOT privilege that it can apply countermeasures that it hopes to neutralize any TCC attacks.

Hereinafter, the TCC models described in [5] are adopted as the baseline TCC model in this paper. In paricular, both 0-hop and 1-hop TCCs are considered. Here the transmitter and receiver of a 1-hop channel are two cores that are one hop away from each other. There are two types of 0-hop channels: i) the receiver and transmitter threads run in the same physical core and they share the same thermal sensors; and ii) the receiver and the transmitter have access to the same thermal data files which are exposed by installed temperature monitoring software or sysfs system in Linux [11].

2.2 Threshold-based Detection Methods and Their Disadvantages

To fight against TCCs, threshold-based detection methods [5] have been proposed. The detection threshold, defined as ρ, ranges from ρ_l to ρ_h, where ρ_l is the average noise amplitude (*e.g.*, thermal signals generated by normal applications), and ρ_h is the average value of signal amplitudes at the transmission frequencies of TCCs.

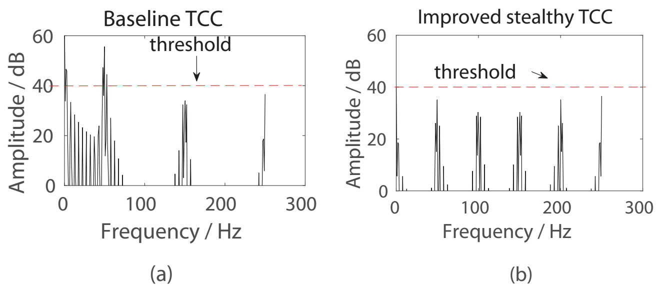


Fig. 3. (a) and (b) are the spectra of the baseline TCC and the improved stealthy TCC. The transmission frequency is 50Hz. The detection threshold against baseline TCCs is set to be 40dB.

The detection accuracy is related to the true positive rate, false positive rate, and true negative rate. Here a false positive case refers to the situation that an innocent thread is mistakenly recognized as a malicious one, while a true negative case refers to an innocent thread is not identified as so, and a true positive is when a TCC thread is correctly detected. From Figs. 3(a) and 3(b), one can see that when using a threshold-based detection which works fine against baseline TCCs, if the signal amplitudes of a TCC are below the threshold, the threshold-based detection methods will be unlikely to be able to detect such improved stealthy TCCs. On average, the true positive rate, true negative rate, and total accuracy are 0%, 47.5%, and 47.5%, respectively. Such a low true positive rate is unacceptable in detection.

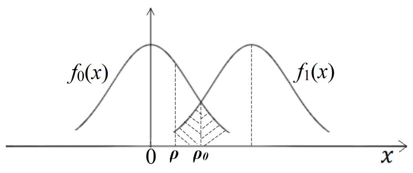


Fig. 4. The curves of $f_1(x)$ and $f_0(x)$.

Apparently, the error rate or detection accuracy depend on the selection of detection threshold. As shown in Fig. 4, let x be the signal amplitude, $f_1(x)$ be the probability density function of TCC's signal amplitudes, and $f_0(x)$ be the probability density function of noise's amplitudes. According to [13], the detection error rate, denoted as P_e, can be modelled by Eqn. (1), where $P(1)$ and $P(0)$ are the probabilities that the TCC signals and noise are sent, respectively; $P(0|1)$ is the probability that the signals from TCCs are not detected; and $P(1|0)$ is the probability that the noise is regarded as TCC signals. One can see from Eqn. (1), the detection error rate P_e or the detection accuracy $(1 - P_e)$ depend on the value of the detection threshold ρ.

$$\begin{aligned} P_e &= P(1)P(0|1) + P(0)P(1|0) \\ &= P(1)\int_{-\infty}^{\rho} f_1(x)dx + P(0)\int_{\rho}^{+\infty} f_0(x)dx \end{aligned} \tag{1}$$

Fig. 5 shows the distributions of $f_1(x)$ and $f_0(x)$, which are the probability density functions of the amplitudes of TCC signals and noise, respectively.

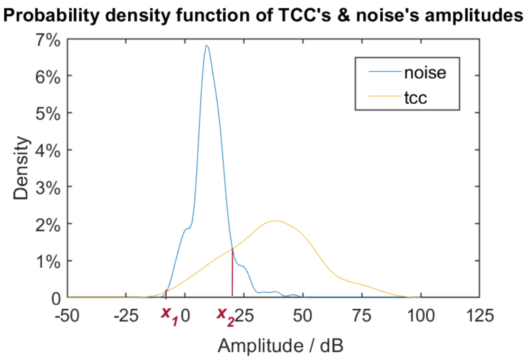


Fig. 5. the distribution of noise's and TCC's signal amplitudes.

The overlap (the shaded area in Fig. 4) between the two distribution functions, $f_1(x)$ and $f_0(x)$, denoted as P_o, is given by

$$P_o = \int_{-\infty}^{x_1} f_0(x)dx + \int_{x_1}^{x_2} f_1(x)dx + \int_{x_2}^{+\infty} f_0(x)dx \tag{2}$$

The percentages of P_o over $f_1(x)$ and $f_0(x)$ are defined in Eqns. (3) and (4) respectively, which are 30% and 28% from the experiments.

$$P_o{}^1 = \frac{P_o}{\int_{-\infty}^{+\infty} f_1(x)dx} \tag{3}$$

$$P_o{}^0 = \frac{P_o}{\int_{-\infty}^{+\infty} f_0(x)dx} \tag{4}$$

To study the impact of the threshold ρ on the detection accuracy of improved stealthy TCCs, Fig. 6 shows the numerical results of Eqn. (1) with the configurations detailed in Section 5.1.

In our experiments, half of the applications generate TCC signals, and the remaining half of the applications generate noise. Therefore, the ideal case of the true positive rate, the true negative rate, the false positive rate, and the total accuracy are supposed to be 50%, 50%, 0%, and 100%, respectively. From Fig. 6, one can see that when the threshold ρ is lower than 40dB, the true positive rate increases, but the true negative rate decreases. When the threshold drops to 30dB, the total detection accuracy reaches the highest at 70%, with a true positive rate of 40% and a true negative rate of 30%. Note that the false negative rate of 10% means 10% of the TCC attacks are not detected. In addition, when this threshold drops even further, the

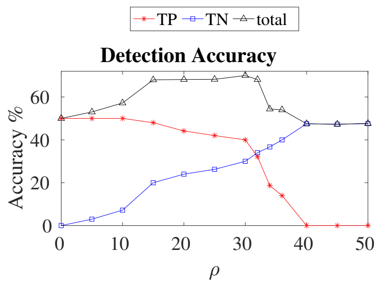


Fig. 6. The accuracies of detecting the improved stealthy TCC under different detection thresholds. 'TP', 'TN', and 'total' are the percentage of true positive cases over all cases, the percentage of true negative cases over all cases, and the sum of 'TP' and 'TN', respectively.

false positive rate goes up to a level higher than 20% (i.e., 30%∼50%). With such a high false positive rate, applying a DVFS-based countermeasure, like the one described in [5], will lead to unacceptable performance loss.

3 IMPROVED THERMAL COVERT CHANNEL

3.1 Design of improved stealthy TCCs

To circumvent the threshold-based detection, the signal amplitude of TCC needs to be reduced to the level lower than the detection threshold. As shown in Fig. 1(b), the signal amplitude reduction can be done by reducing the time to boost up the temperatures as follows.

Based on the RZ encoding scheme, when transmitting a bit '1', hereinafter, the ratio of the time to heat up over the time to cool down is defined as α. Note that α is almost always set to 1 (*e.g.*, $\alpha = 0.5t_b/0.5t_b$ as shown in Fig. 1(a)) in the baseline TCCs in Section 2.1. The value of α is selected in an iterative manner that targets to fail the threshold-based detection method.

1) In each iteration, build a TCC based on a specific value of α (intial value is set to 1, and in every iteration its value is linearly decremented by ω from the previous iteration).
2) Then we apply the threshold-based detection to detect the just built TCC and get the detection accuracy as well as the packet error rate (PER) when no countermeasure is adopted.
3) The iterative method should be stopped if the minimum detection accuracy is obtained and the PER of the TCC is lower than an acceptable level, say 10%.

Once α is determined, the improved stealthy TCC attack is designed as follows.

The transmitter uses the above method to compute α. The bit stream to be transmitted is RZ encoded with a period of t_b. For bit '1', the transmitter core runs CPU-intensive code for a duration of $\epsilon \times t_b$, and cools down for a duration of $(1-\epsilon) \times t_b$ within the same period. For bit '0', the core keeps idle for the entire period. On the receiver side, a finite impulse response (FIR) filter with center frequency of $1/t_b$ is used to filter out the signal after reading the thermal sensors. If the signal amplitude is over a decision threshold which is half of the maximum signal amplitude, it is deemed as bit '1', otherwise bit '0'.

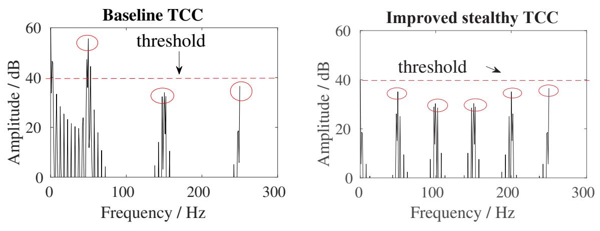
The transmitter and receiver use a handshake protocol [5] for communication. That is, the transmitter first sends a request packet (REQ) to the receiver to setup the connection. Upon receiving the REQ packet, the receiver replies an ACK packet to the transmitter. The transmitter then starts data transmission. Finally, the transmitter sends a TER packet to terminate the connection once the data transmission is done.

3.2 Spectrum Analysis of TCC Signals

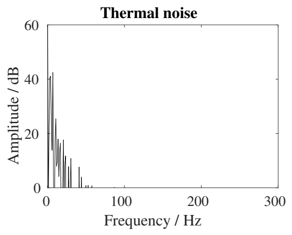
We analyze the above proposed improved stealthy TCC in frequency domain. A TCC signal is set to be band-limited, ranging from 10Hz to 500Hz [5]. Below 10Hz, thermal noise dominates, and since the thermal sensors have a refresh rate (sampling) of 1000 times per second, signal transmission frequency has to be cut at 500Hz.

From Figs. 7(a) and 7(b), one can see that the signal amplitude at 50Hz of the improved stealthy TCC is much lower than that of the baseline TCC. Besides, the improved stealthy TCC have additional high-amplitude side lobes at 100, 150, 200, and 250Hz, a feature not seen in the baseline TCC. By comparing the spectra of the baseline and improved stealthy TCCs and that of thermal noise [see Fig. 7(c)], one can see that these high-amplitude side lobes can be used as the key feature to distinguish the TCCs from noise.

Correspondingly, a pattern-classification-based detection method is proposed in the next subsection. Even with adoption of different types of encoding schemes (*e.g.*, on-off keying, RZ, and Manchester code), the TCCs with improved stealthiness still exhibit high-amplitude side lobes when PER is lower than 10%. As a result, a TCC can not be escaped from being detected by the proposed detection method.



(a) (b)



(c)

Fig. 7. (a) and (b) are the spectra of the baseline TCC and the improved stealthy TCC, respectively. The transmission frequencies in (a) and (b) are both 50Hz. The red circles in (a) and (b) are the characteristics of the baseline TCC and the improved stealthy TCC, respectively. (c) The spectrum of thermal noise (*e.g.*, generated by running the 'blackscholes' application from PARSEC [14]).

4 CLASSIFICATION-BASED DETECTION OF THERMAL COVERT CHANNEL ATTACKS

To fight against the improved TCC, the features of TCCs in Section 3.2 is explored by a neural network based detection to classify (detect) TCCs from noise. The neural network model that is trained offline is applied to the system at runtime to classify the TCC signals and noise. For TCC, the pattern recognition algorithm generates an output 1, and for thermal noise, the output is 0.

4.1 Neural Network Model

The neural network based model is experimentally compared against two other models based on classification tree [15] and logistic regression [15]. The validation accuracies of all the three models are compared in Table 1. One can see that the proposed neural network model has the highest validation accuracy (the sum of true positive rate and true negative rate on the validation data set). Therefore, we used the proposed neural network model in all our subsequent experiments.

TABLE 1
Validation accuracies of different machine learning techniques

Machine learning techniques	Neural network	Classification tree [15]	Logistic regression [15]
Validation accuracies	0.99	0.91	0.95

As shown in Fig. 8, the neural network model has k middle layers and 1 output layer, which is called $(k+1)$-layer neural network model.

1) Input data: the input data is a vector with 491 elements, with each element representing a signal amplitude (the signal frequencies span from 10Hz to 500Hz with an incremental of 1Hz) of TCC or noise. The input vector is denoted as $\vec{x}$ [*e.g.*, $\vec{x}=(x_1,x_2,\ldots,x_{491})$].
2) Middle layers: the l-th layer contains n_l neural nodes, with each node value being denoted as an element of vector $\vec{a^{[l]}}$ [*e.g.*, $\vec{a^{[l]}} = ({a_1}^{[l]}, {a_2}^{[l]}, \ldots, {a_{n_l}}^{[l]})$]. The activation function of layer l is denoted as $\delta^{[l]}(\cdot)$.
3) Output layer: this layer has a node with its output value (denoted as $\hat{y}$) to be either '1' or '0', indicating whether the input is a TCC signal or not. The 'sigmoid' activation function is adopted in this layer before the final result is generated. When the output value from 'sigmoid' is higher than a threshold (*e.g.*, 0.5 in our experiments), it indicates that the input data is possibly from TCC and the final output of the model is 1, otherwise 0.

4.2 Data Preprocessing and Training

For the training data of the neural network model, we generate TCC signals or noise from a logical core for each t seconds (*e.g.*, 2 seconds) with a sampling frequency of 1000Hz. That is, each data sample contains $1000\times t$ temporal

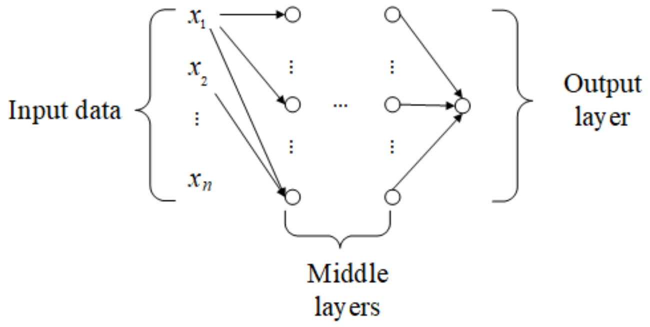


Fig. 8. The architecture of the neutral network model for classification.

signal values over t seconds. Before training the parameters of the model, each data sample is transformed into frequency domain representation. That is, through discrete Fourier transform (DFT), each data sample is made of a sequence of the amplitudes of 491 signal components. We also provide a supervised label (denoted as y) for each data sample to the neural network model for parameter training. If the signals are from thermal covert channels, the supervised label is set to be '1', otherwise '0'.

The data samples are divided into three datasets: the training set, validation set, and test set. As their names suggest, the data from the training set is used to train the parameters (*i.e.*, the weights of the network edges) of the neural network model; the data from the test set is used to evaluate the ability of model generalization, that is, how well the training model performs on new data samples; and the data from the validation set is used to choose a model that has the best ability of generalization among different hyper-parameters (*i.e.*, learning rate, number of training iterations, number of neural layers or nodes, *etc.*). The parameters (weights of edges) of the model during the training are randomly initialized. After φ iterations in training, the gradient descent based method [15] obtains the neural network model parameters with learning rate ε and cost function $J(y,\widehat{y}) = -\sum_{i=1}^{n} y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})$, by the following three-step procedure.

Step 1 - Forward Propagation: From the input layer to the output layer, the weights of each neuron $\boldsymbol{a}^{[l]}$ are computed layer by layer,

$$\boldsymbol{a}^{[l]} = \delta^{[l]}\left(\boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}\right) \tag{5}$$

where $\boldsymbol{a}^{[0]} = \boldsymbol{x}$; that is, we have the input vector, $\widehat{y} = \boldsymbol{a}^{[R]}$ with R as the number of layers of the neural network.

Step 2 - Backpropagation: From the output layer to the first layer, the gradient of the cost function to the parameters of each layer is calculated layer by layer by the chain rule following Eqn. (6) to Eqn. (9),

$$dz^{[l]} = \frac{\partial J}{\partial \boldsymbol{a}^{[l]}} \times \delta^{[l]\prime}\left(\boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}\right) \tag{6}$$

$$\frac{\partial J}{\partial \boldsymbol{W}^{[l]}} = \frac{1}{c} \times dz^{[l]} \times \boldsymbol{a}^{[l-1]T} \tag{7}$$

$$\frac{\partial J}{\partial \boldsymbol{b}^{[l]}} = \frac{1}{c} \times \sum_{h=0}^{J-1} dz^{[l]}[:, j] \tag{8}$$

$$\frac{\partial J}{\partial \boldsymbol{a}^{[l-1]}} = \boldsymbol{W}^{[l]T} \times dz^{[l]} \tag{9}$$

where $\frac{\partial J}{\partial \boldsymbol{W}^{[l]}}$ is the partial derivative of the cost function $J(y, \widehat{y})$ with respect to $\boldsymbol{W}^{[l]}$, $\frac{\partial J}{\partial \boldsymbol{b}^{[l]}}$ is the partial derivative of the cost function $J(y, \widehat{y})$ with respect to $\boldsymbol{b}^{[l]}$, $dz^{[l]}$ is an intermediate term, $\delta^{[l]'}(\cdot)$ is the derivative of $\delta^{[l](\cdot)}$, $\boldsymbol{a}^{[l]T}$ and $\boldsymbol{W}^{[l]T}$ are the transpose of $\boldsymbol{a}^{[l]}$ and $\boldsymbol{W}^{[l]}$ respectively, and c is the sample size. $dz^{[l]}[:, j]$ is the j-th column vector of matrix $dz^{[l]}$.

Step 3, the parameters of the neural network model are updated following Eqns. (10) and (11).

$$\boldsymbol{W}^{[l]} = \boldsymbol{W}^{[l]} - \frac{\partial J}{\partial \boldsymbol{W}^{[l]}} \tag{10}$$

$$\boldsymbol{b}^{[l]} = \boldsymbol{b}^{[l]} - \frac{\partial J}{\partial \boldsymbol{b}^{[l]}} \tag{11}$$

4.3 Online Detection

After the neural network model is trained offline, it can be used to detect the existence of a TCC attack in a many-core system at runtime. Herein a detection cycle and a global manager are the time unit of each detection and the thread to initiate a detection cycle, respectively. In addition, the detection is performed at the logical core level since the hyper-threading multi/many-core systems have become commonplace. To reduce runtime overhead, a distributed detection architecture can be adopted. That is, the global manager assigns the detection jobs to each individual logical core, and each logical core performs the detection and reports their individual results back to the global manager regarding whether there is a possible attack or not.

Note that since there is a strong thermal correlation between neighboring cores, detecting thermal signals generated by each core's activities cannot easily distinguish a TCC core from those running normal applications. For instance, in the case that a transmitter core and a core running normal applications are physically next to each other in the vertical stack of a 3D many-core system, the thermal signals collected from both cores exhibit the same transmission frequency. Essentially, a TCC program running in a secure zone does not have direct access to the cores to change their voltage and/or frequency; rather a TCC controls the CPU workloads by either running computation-intensive codes or keeping the cores idle as an indirect means to generate thermal signals. Therefore, instead of using thermal signals, the CPU workloads measured by the number of instructions per cycle (IPC) are used to exactly pin down the TCC cores.

In each detection cycle, the global manager samples each logical core's IPC profiles over a time window, say t_1 seconds (*i.e.*, 2 seconds in our experiments), and it then commands each logical core to execute a pattern-classification-based detection to test whether there is a TCC attack or not.

Algorithm 1: Pattern-classification-based detection at core $i (1 \leq i \leq n_c.)$

```
   Input: $IPC_i$ and $L$;
   $IPC_i$: The traces of CPU workload (IPC) of logical
   core $i$ recorded in a detection cycle;
   $L$: A list, with a capacity of $n_c$ (the number of logical
   cores in the SoC), of the transmitter logical cores
   during a detection cycle. It is initially set to be
   empty.
   Output: $L$
1  begin
2      Calculate the spectrum of signal $IPC_i$ using the
         discrete fast Fourier transform (FFT) algorithm;
3      Feed the signal amplitudes (from 10Hz to 500Hz
         with a frequency incremental of 1Hz) to the
         input layer of the neural network model and get
         the model result $\hat{y}$;
4      if $\hat{y} = 1$ then
5          Add $i$ to $L$;
6          Report logical core $i$ to the global manager;
7          return.
8      end
9      Send a message to the global manager that no
         TCC channel is found in core $i$.
10 end
```

This detection task is set to supersede any other tasks of the logical core that has been engaged with.

The pattern-classification-based detection algorithm in Algorithm 1 works as follows.

Step 1. The global manager initiates a detection cycle to see if there is any possible TCC attack. Upon receiving the command, each logical core extracts the spectrum of its IPC signals (see line 2 in Algorithm 1). Then each logical core feeds the signal amplitudes (from 10Hz to 500Hz with a linear frequency incremental of 1Hz) to the detection model and gets the output $\hat{y}$ when the model calculation is finished (see line 3 in Algorithm 1).

Step 2. After getting the output $\hat{y}$ from the detection model, one can decide whether the signals are actually from a covert channel or not (see line 4 in Algorithm 1). Once a suspicious channel is detected in logical core i, core i is added to list L (see line 5 in Algorithm 1). Note that a normal application running on a core may be mistakenly deemed as a suspicious one, which is a standard false positive. There is a low probability of false positive, typically in the range of 5% as demonstrated in [5], which is acceptable in this step.

Step 3. At the end of a detection cycle, if a logical core confirms that a TCC attack is present, it reports its findings, including the position of the detected logical core, to the global manager (see line 6 in Algorithm 1). Otherwise, the logical core can conclude that no TCC attack has been found in the current detection cycle and reports so to the global manager (see line 9 in Algorithm 1).

Step 4. If the global manager finds no TCC channel exists in any of the logical cores, it initiates a new detection cycle, after which the process starts all over again from step 1. Otherwise, if the address space of a thread listed in L can be accessed, that thread is removed from list L. Here only a detected thread running in a secure zone is deemed a threat to system security. Note that supported by processor reserved memory [16], only self-signed applications can access the secure zone.

If one or more TCC logical cores are detected, *i.e.*, list L is not empty, the global manager begins to block the transmission from the cores listed in L. The transmission blocking is supported by applying the DVFS-based countermeasure proposed in [5] to the cores detected with a TCC transmitter or receiver (essentially the physical CPU cores that the detected logical cores belong to). Since the DVFS-based countermeasure dynamically changes the voltage and frequency level of the detected transmitter core (*e.g.*, scaling down from 2.5GHz to 500MHz), the thermal signals generated by the transmitter can be severely distorted, leading to a very high error transmission rate that essentially shuts down a TCC.

4.4 Overhead of the Proposed Detection Method

Similar to the threshold-based detection method in [5], a detection cycle is initiated repeatedly; that is, a new detection cycle will be initiated after the global manager applies DVFS countermeasure to the detected cores. A detection cycle spans 2 phases: t_1, and t_2 (see section 4.3), where t_1 (*i.e.*, 2s) is the time for the global manager to calculate all logical cores' IPC values, and t_2 is the time for each logical core to perform discrete fast Fourier transform and neural network model inference. Compared with the threshold-based detection method in [5], the proposed detection method has additional runtime overhead and energy overhead for the neural network model inference at runtime.

As indicated in [5], the length of a detection cycle on average is 2s (see section 4.3). During t_1, only the global manager takes $57344 \times n_c$ clock cycles or $28 \times n_c \mu s$ for a core running at 2GHz. During t_2, each core takes 901,120 clock cycles or 0.45ms to perform the discrete Fourier transform. The neural network inference needs 4920 real number multiplications, which corresponds to 9.84×10^4 clock cycles [17], or a total of runtime of $49.1\mu s$ for a core clock running at 2GHz.

Although the detection works periodically, the system (except the global manager) runs normal tasks as well as the TCC tasks during most of the time of a detection cycle (*i.e.*, during t_1). When $n_c \leq 100$, the inference time of the proposed detection accounts for lower than 0.17% of the execution time of the normal applications. The energy consumption overhead of the proposed detection is only about 0.039% of the total energy consumption of the whole system. In general, the runtime overhead in terms of cycle count and energy consumption of our proposed detection method is fairly low, given its high detection accuracy.

Besides, the global manager (running a thread that has root privilege) needs to broadcast a control packet to other cores to initiate the detections in parallel. With a 2-pipeline-stage router architecture, when $n_c \leq 100$, the communication overhead is lower than 200 clock cycles, which is negligible compared to execution times of most applications.

In terms of storage overhead, each logical core needs to store a copy of the weights of the neural network with each weight in double precision (64 bits). Altogether, when $n_c \leq 100$, the storage overhead is lower than 3.936MB (*i.e.*, $100 \times 64 \times 4920 \div 8 \div 10^6$), which is considered fairly low.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

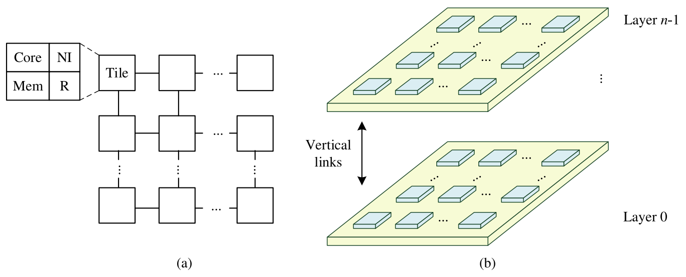


Fig. 9. Examples of (a) 2D and (b) 3D many-core systems.

To demonstrate diverse applicability of our approach, we have considered two sets of experiment, and they are performed on respective 2D and 3D many-core systems. In these many-core systems, each tile is composed by a processor core, memory units (L1 I/D caches and an L2 cache bank), network interface (NI), and a router, as shown in Fig. 9. Tiles are connected by networks-on-chip. All the experiments are either run on a many-core simulator, Sniper-v7.2 [18], or directly run on two real machines featuring a 2D multi-core system.

In the simulator, to dynamically generate temperatures for all the cores, McPAT-v1.0 [20] and Hotspot-v6.0 [21] are adopted as the power and thermal models, respectively. The temperatures from TCC cores are deemed as TCC signals while the temperatures generated by normal applications made of a few benchmarks from PARSEC [14] and SPLASH-2 [22] are treated as the thermal noise. The detailed configurations of Sniper, Hotspot, TCC programs, and real machines are tabulated in Table 2.

5.1.1 *Experimental Configurations of the Many-core Systems*

As for a 2D 1-hop channel, two physically separated cores form a TCC pair, one as the transmitter and the other as the receiver, while all the other cores are running legitimate threads from the selected benchmarks. Specifically, each physical core runs two simultaneous multithreading (SMT) threads, and both the transmitter core and the receiver core run a TCC thread and a thread spawned by the benchmarks. As for a 0-hop channel, the two logical cores sitting in the same physical core are able to run the transmitter and receiver programs.

As for a 3D many-core system where its floorplan follows the one used in [23], the receiver core of a 1-hop channel is right below the transmitter core, while the remaining configurations are set to be the same as those of the 2D many-core case. In a 3D many-core system, the vertical layers are connected by the TSV's (Through-Silicon-Vias).

TABLE 2
Experimental configurations

Sniper configuration	
Instruction set architecture	x86-64
Network topology	Mesh
Network topology	3D mesh, with each layer having its own 2D mesh and the layers are connected vertically through Through-Silicon-Vias (TSV) [19].
Number of cores (2D)	3×3 / 4×4 / 8×8 / 16×4
Number of cores (3D)	3×3×3 / 4×4×3 / 8×8×3
Number of SMT threads per core	2
CMOS technology node (nm)	22
Frequency and voltage levels of CPUs	2500/1.0, 1000/0.7, 800/0.68, 700/0.66, 600/0.64, 500/0.6, 400/0.5
Benckmarks of PARSEC	Blackscholes, Canneal, Fluidanimate, Streamcluster, Swaptions, X-264, Dedup, Freqmine
Benckmarks of SPLASH-2	Raytrace, Barnes
Configuration of the integrated Hotspot	
Chip thickness	$0.15mm$
Silicon thermal conductivity	$100W/(m \cdot K)$
Silicon specific heat capacity	$1.75 \times 10^6 J/(m^3 \cdot K)$
Heat sink side	$0.06m$
Heat sink thickness	$6.9mm$
Heat sink thermal conductivity	$400W/(m \cdot K)$
Specific heat capacity of heat sink	$3.55 \times 10^6 J/(m^3 \cdot K)$
Configuration for TCC programs	
Transmission frequency	10Hz, 20Hz, 50Hz, 80Hz, 100 Hz, 150 Hz
Preamble of a packet	101010
Packet size in bits	64
Distance between a transmitter and a receiver	0/1 hop
Bandpass filter using by the receiver	Window-based FIR (finite impulse response) filter
Bandwidth of bandpass filter	4Hz

Configurations of the real machines		
ID	1	2
Processor	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz	Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
Memory	16 Gbytes	16 Gbytes
DRAM Frequency	1,200MHz	2,133MHz
Mainboard	MSI Z170-A PRO	20HNA01PCD
# of Physical Cores	4	2
# of Logical Cores	8	4
Operate System	Ubuntu 16.04 LTS	Windows 10

For real machines with 2D multi-core chips adopted in this study, one has a quad-core eight-thread Intel Core i7-7700HQ processor clocked at 2.8GHz, and the other has a dual-core four-thread Intel Core i7-6700U processor clocked at 2.7GHz. We fix the fan speed to the maximum and let other cores sleep, and only the transmitter core and receiver core are active, as the case in [2], and all the other cores are set to sleep.

In the real machines with a coarse-grained sensor resolution of 1°C, since a 0-hop channel does not need to transfer heat between two neighbor cores, the transmission frequency of a 0-hop channel can be much higher than that of 1-hop channels. That is, the upper bound of transmission frequencies for 0-hop channels and 1-hop channels are experimentally found to be 100Hz and 20Hz, respectively.

In the simulations, the precision of temperatures is set to be 1°C [10] and 0.12°C [12] for simulations in 3D and 2D many-core systems, respectively. Note that the thermal correlation in the vertical direction is higher than that in the horizontal, thus, the vertically placed 3D 1-hop channels are more efficient than those channels in a 2D many-core system.

5.1.2 *Experimental Scenarios of the Many-core Systems*

The TCC communications are point to point in nature. For each experiment, packets are transmitted randomly for 1000 times and the result is then averaged. Effectiveness of a TCC attack is measured in terms of the PER, which is defined as follows.

$$PER = N_e/N \times 100\% \tag{12}$$

where N is the total number of packets transmitted, and N_e is the number of packets failed to be correctly recognized. Note that when a few bits of a control packet (*e.g.*, the connection request packet REQ) are flipped, say 1 in 5 bits, the bit error rate (BER) in this case is 20%; but since the packet cannot be recovered, the PER is actually 100%. The experiments mainly include the following scenarios:

- Measuring the PERs of the improved stealthy TCC communications under different system sizes when threshold-based detection and DVFS-based countermeasure [5] are applied.
- Measuring the detection accuracy when applying the proposed pattern-classification-based detection method.
- Measuring the PER of the improved stealthy TCC communication when the proposed pattern-classification-based detection and DVFS-based countermeasure [5] are in place.
- Evaluating the performance loss of legitimate applications when exploiting the proposed detection and DVFS control to the transmitter core.
- Evaluating the performance loss of legitimate applications when exploiting DVFS control to the transmitter core.

5.2 Finding the value of α for the improved stealthy TCC

The improved stealthy TCC not only needs to circumvent the threshold-based detection, but also needs to ensure a low PER (*e.g.*, $< 10\%$). To better measure how well our proposed detection method is, the TCC with the best stealthiness is firstly tested in a 1-hop channel.

Fig. 10 shows the PERs of a 1-hop TCC under different α's with and without the threshold-based detection [5] and DVFS-based countermeasure [5] applied. When the threshold-based detection is in place, the DVFS-based countermeasure is applied to the detected CPU cores. As α decreases from 1 to 1/9, the PER under the threshold-based detection decreases significantly, dropping from 82% to 8.5%. When α is 1/12 or even smaller, the PER of TCC reaches an unacceptably high level (*i.e.*, $> 88\%$), since the signal noise rate (SNR) is too low to sustain a TCC communication. Therefore, the best α is set to be 1/9 for an improved stealthy TCC.

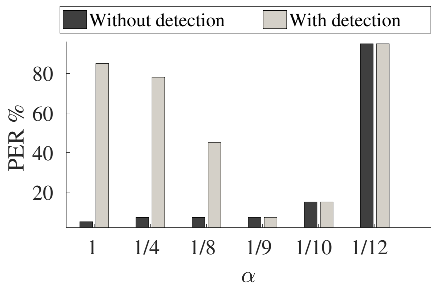


Fig. 10. The average PER results of 1-hop TCC communications with different α's with and without the threshold-based detection [5] and the DVFS-based countermeasure [5] applied.

5.3 Finding the Parameters of the Neural Network Model

In order to find the parameters and train the neural network model for TCC detection, we collect 350,000 samples of IPC signals that are from the TCC programs running with different α values (*e.g.*, from 1/12 to 1), encoding schemes, transmission frequencies and packet bits. We also collect 350,000 samples of IPC noise that are from the legitimate applications. The collected data samples are split among the training (5/7 of the samples), test (1/7 of the samples), and validation (1/7 of the samples) sets.

The parameters (weights of the edges) of the neural network are automatically learned by the gradient-descent-based training [24], while the hyper-parameters are manually tuned offline to get a model with better generalization ability. Besides detection accuracy, another important consideration is the network complexity in terms of the number of layers and/or neurons, as complexity of a neural network should be preferably kept low to reduce runtime overhead. In this case, various neural network models described in Section 4 are built with varying complexities and experimentally compared for their detection accuracy.

The validation accuracies, defined as the sum of true positive rate and true negative rate, of different neural network models on the validation set are reported in Table 3. All the models in Table 3 have 491 inputs and 1 output. Model A, whose inference time is 984 clock cycles, is a two-layer neural network model with 2 nodes in its middle layer. The numbers of nodes, layers, and inference times of

TABLE 3
Validation accuracies of different neural network models

Neural networks	A	B	C	D	E
Number of layers	2	2	2	2	3
Number of nodes	2	5	10	100	10, 2
Inference time (cycle)	984	2460	4920	49200	4932
Validation accuracies	0.99956	0.99957	0.99996	0.99997	0.99996

the other neural network modes can be found in Table 3. Model E has two middle layers, whereas they have 10 and 2 nodes, respectively. One can see that model C achieves a high validation accuracy with moderate inference time. Therefore, model C with 10 nodes in its middle layer is adopted in the following experiments.

5.4 Experimental Results

5.4.1 Results of the improved stealthy TCC Attacks

To measure how well the improved stealthy TCCs (described in Section 3.1) can be detected by the threshold-based detection, we run another experiment. Once a TCC core is detected by the threshold-based detection [5], the DVFS-based countermeasure [5] is applied to that core to block the communication. The average PERs of experiments from the simulations and measurements of the two real machines are shown in Fig. 11 and Table 4, respectively.

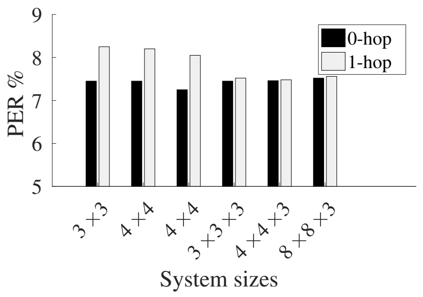


Fig. 11. The average PERs of the improved stealthy TCC under different system configurations and scenarios when the threshold-based detection [5] and DVFS-based countermeasure [5] are applied together.

As for the simulations, the TCCs can work with a transmission frequency of 100Hz and a CPU running at 2,500 MHz. From Fig. 11, one can see that the average PERs of TCC transmission in the 2D and 3D many-core systems of different sizes are all below 8.5%, which means that the improved stealthy TCC can barely be detected using the threshold-based detection and thus most of the time, the DVFS-based countermeasure is not triggered.

A similar result obtained for the TCCs running in the two real machines is shown in Table 4. The average PERs of a 0-hop channel and a 1-hop channel are both lower than 8%, which means the threshold-based detection method [5] can not detect the improved stealthy TCC.

In a simple word, as indicated by both simulation results and real machine measurements, the improved stealthy

TABLE 4
The average PERs of the improved stealthy TCC by applying the threshold-based detection [5] and DVFS-based countermeasure [5] in the two real machines.

Distance	PER	Distance	PER
0-hop	3.5%	1-hop	7%

TCC certainly poses a serious threat to any system, when threshold-based detection method [5] be applied.

5.4.2 Evaluation of the Proposed Detection Scheme

In the end of each detection cycle, if a TCC attack is detected by the global manager, the CPU core associated with the TCC will be located. The accuracy is the combination of true positive rate and true negative rate of detection.

$$Accuracy = \begin{cases} 1 & P_{detected} = P_{transmitter} \text{ or } P_{not} = P_{normal} \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

where $P_{detected}$ are the positions (core ID) of the detected cores, $P_{transmitter}$ are the actual positions of the transmitter cores, P_{not} are the positions of the cores that are not detected as TCC cores, and P_{normal} are the positions of the cores that are not running TCC transmitter threads.

To measure the average detection accuracies of the threshold-based detection and the proposed detection, both detection methods are respectively adopted in 100,000 experiments grouped as 1,000 sets. Each experiment involves 4 logical cores, with 2 logical cores running the transmitter and receiver programs of TCC, and the other 2 logical cores running normal threads (threads from PARSEC [14] or SPLASH2 [22]).

From Fig. 12, one can see that when applying the proposed detection method, the average detection accuracies of the baseline TCCs and the improved stealthy TCCs are at 99% (*i.e.*, 50% of true positive rate and 49% of true negative rate). In sharp contrast, although the threshold-based detection method works reasonably well to detect the baseline TCCs with an accuracy about 96%,the accuracy drops to only about 45% (*i.e.*, 0% of true positive rate and 45% of true negative rate) when the improved stealthy TCC is present.

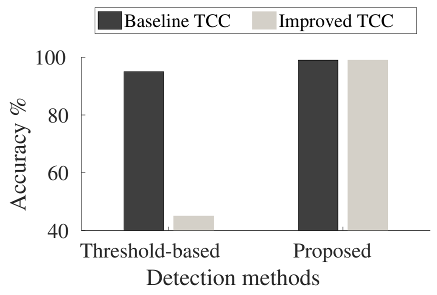


Fig. 12. The detection accuracies of both the improved stealthy TCC and the baseline TCC under the threshold-based and the proposed detection methods.

From Fig. 13, in the two real machines, our proposed detection method can effectively detect both the baseline and improved stealthy TCCs with average accurary of higher than 95%. In contrast, the threshold-based detection method in [5] fails to detect the improved stealthy TCC with a detection accuracy of lower than 50%.

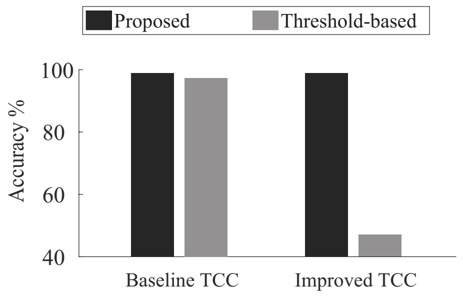


Fig. 13. Average accuracies of the proposed detection method and the threshold-based detection method [5] against the baseline and improved stealthy TCCs in the two real machines.

The false positive rates of the proposed detection method and the threshold-based detection method [5] are shown in Table 5. One can see that when the threshold-based detection method [5] is applied, the false positive rate of the improved stealthy TCC is unacceptably high at 49%. But with our proposed detection method applied, the false positive rates of detecting both the baseline TCC and improved stealthy TCC are very low.

TABLE 5
The false positive rates of the proposed detection method and the threshold-based detection method [5]

Proposed detection method		Threshold-based detection method [5]	
TCC type	False positives rate	TCC type	False positives rate
Baseline TCC	0.004%	Baseline TCC	4%
Improved TCC	0.004%	Improved TCC	49%

Therefore, by using the proposed pattern-classification-based detection strategy, we can almost always identify a TCC attack, should it ever exist, and correspondingly, the location(s) of the transmitter core(s) can be accurately determined.

Once a TCC attack (including the baseline TCC and the improved stealthy TCC) is detected, the frequency level of the TCC transmitter core is changed by the DVFS-based countermeasure proposed in [5]. As shown in Table 6 (averaged measurements in the two real machines) and Fig. 14 (simulation results), with our proposed pattern-classification-based detection and DVFS-based countermeasure, the average PERs of the baseline TCCs and the improved stealthy TCCs are all higher than 75%. Such a high PER (*i.e.*, $> 70\%$) really denies any meaningful communications in practice; that is, our proposed detection with the DVFS-based countermeasure can effectively shut down TCC attacks.

5.4.3 Average Performance Loss

Scaling down the V/F level of a physical core running the TCC transmitter program will also negatively impact performance of a legitimate logical core. Fortunately, we do not need to apply DVFS-based countermeasure to the TCC

TABLE 6
The average PERs of the baseline and improved stealthy TCCs with the proposed detection method and DVFS-based countermeasure [5] applied in the two real machines.

Baseline TCC		Improved stealthy TCC	
Distance	PER	Distance	PER
0-hop	75%	0-hop	77%
1-hop	86%	1-hop	88%

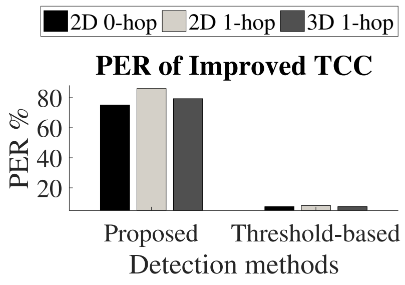


Fig. 14. The average PER results of TCCs (*i.e.*, baseline TCCs and improved stealthy TCCs) communications with the proposed detection method and the DVFS-based countermeasure.

transmitter core all the time since the TCC programs return to be inactive after finishing transmission. We denote τ as the ratio of the time of TCC being inactive to the time of TCC being active. In practice, τ is set to be much higher than 4 [5].

We assume that a TCC thread may share the same physical core with a thread of a legitimate application. The performance loss (PL) of a legitimate application is 0%, if DVFS is not applied to block TCC. When DVFS is applied to the physical cores participating in a TCC, the performance loss of that legitimate application is calculated by

$$PL = \frac{\pi_{avg} - \pi_0}{\pi_{avg}} \times 100\% \tag{14}$$

As shown in Fig. 15, we compare the average PLs under the threshold-based detection and the proposed detection with the DVFS-based countermeasure and τ set to be 4. The true positive and false positive rates of the threshold-based detection are 40% and 20%, respectively, as given in Section 3.2. The true positive and false positive rates of the proposed detection are 50% and 1%, respectively, as given in Section 5.4.2. One can see that with the false positive rate of 20% achieved by the threshold-based detection (see Section 3.2) and thus DVFS is excessively applied, the PL of the threshold-based detection and DVFS is at least $3\times$ higher than the PL of the proposed detection method. For a large many-core system (*e.g.*, number of cores $\geq 8 \times 8$), the PL of the proposed detection and DVFS is lower than 2%, which is considered very low in any practical sense.

Table 7 compares our proposed detection and defense method with related countermeasures. As in Table 7, the task migration-based method [3] and pre-heating method [4] can detect neither the baseline TCC nor the improved stealthy TCC, as these two methods do not involve detection. The threshold-base detection method [5] fails to detect the improved stealthy TCC with the reason stated in Section

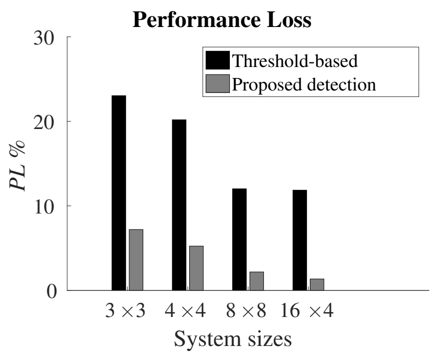


Fig. 15. The average performance loss (PL) of normal applications with different system sizes with τ of 4.

TABLE 7
Comparison of related detection and/or countermeasure methods

Detection method	Detection accuracy against the baseline TCC	Detection accuracy against the improved stealthy TCC	Performance loss in a large-scale multi-core system (number of cores $\geq 8 \times 8$)
The task migration based method [3]	0%	0%	Not mentioned
The pre-heating method [4]	0%	0%	Not mentioned
The threshold-based detection method [5]	97%	47%	3%
The proposed method	99%	99%	2%

2.2. In contrast, our proposed detection method can detect both the baseline TCC and the improved stealthy TCCs.

6 CONCLUSION

In this paper, a pattern-classification-based detection was proposed to fight against improved stealthy TCC which employs reduced signal amplitude that fails the threshold-based detection methods. This proposed pattern-classification-based detection can achieve a detection accuracy of 99% for both the baseline TCC and improved stealthy TCC. After applying the DVFS-based countermeasure in the detected CPU cores, the PERs of both the baseline TCC and the improved stealthy TCC are higher than 70%, but at a low runtime overhead ($< 0.187\%$) and low energy overhead ($< 0.072\%$). With its low complexity and overhead, the proposed detection and DVFS-based countermeasure are able to work seamlessly together to thwart any known TCC attacks.

REFERENCES

[1] ARM, "Building a secure system using trustzone technology," 2021. [Online]. Available: https://developer.arm.com/documentation/genc009492/c/

[2] D. B. Bartolini, P. Miedl, and L. Thiele, "On the capacity of thermal covert channels in multicores," in *Proc. European Conf. Computer Systems*, 2016, pp. 24:1–24:16.

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

[4] Shanquan Tian and Jakub Szefer, "Temporal thermal covert channels in cloud FPGAs," in *Proc. Int'l Symp. Field-Programmable Gate Arrays*, 2019, pp. 298–303.

[5] H. Huang, X. Wang, Y. Jiang, A. K. Singh, M. Yang, and L. Huang, "Detection of and countermeasure against thermal covert channel in many-core systems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 252–265, 2022.

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

[8] V. Costan and S. Devadas, "Intel SGX explained," 2016. [Online]. Available: http://eprint.iacr.org/2016/086

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

[10] Intel, "8th gen intel core processor family datasheet," 2018. [Online]. Available: https://www.intel.com/content/www/us/en/products/docs /processors/core/8th-gen-core-datasheet-vol-1.html

[11] A. Liberman, "Coretemp," 2011. [Online]. Available: https://www.alcpu.com/CoreTemp/

[12] 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.

[13] R. E. Ziemer and W. H. Tranter, *Principles of communications*. Houghton Mifflin Co International Inc., 1998.

[14] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *Proc. Int'l Conf. Parallel Architectures and Compilation Techniques*, 2008, pp. 72–81.

[15] Jiawei Han and Micheline Kamber and Jian Pei, *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011.

[16] B. Lapid and A. Wool, "Cache-attacks on the ARM trustzone implementations of AES-256 and AES-256-GCM via gpu-based analysis," 2018. [Online]. Available: https://eprint.iacr.org/2018/621

[17] L. Literák, "80x86 instruction set," 1999. [Online]. Available: http://www.penguin.cz/ literakl/intel/intel.html

[18] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 3, pp. 28:1–28:25, 2014.

[19] Kai-Yuan Jheng, Chih-Hao Chao, Hao-Yu Wang, and An-Yeu Wu, "Traffic-thermal mutual-coupling co-simulation platform for three-dimensional network-on-chip," in *Proc. Int'l Symp. VLSI Design Automation and Test*, 2010, pp. 135–138.

[20] 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*, 2009, pp. 469–480.

[21] R. Zhang, M. R. Stan, and K. Skadron, "Hotspot 6.0: validation, acceleration and extension," 2015. [Online]. Available: http://www.cs.virginia.edu/ skadron/Papers/HotSpot60_TR.pdf

[22] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," in *Proc. Int'l Symp. Computer Architecture*, 1995, pp. 24–36.

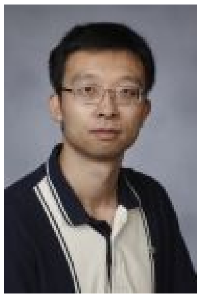
[23] B. Li, X. Wang, A. K. Singh, and T. S. T. Mak, "On runtime communication- and thermal-aware application mapping in 3d noc," *IEEE Trans. Parallel Distributed Syst.*, vol. 30, no. 12, pp. 2775–2789, 2019.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

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.

Hengli Huang received the bachelor's degree in software engineering from South China Normal University, Foshan, China, in 2018. He got the master's degree from the School of Software Engineering, South China University of Technology, Guangzhou, China, in 2021. His research interests include system security and NoC-based systems.

Ruolin Chen received the bachelor's degree in software engineering from South China University of Technology, Guangzhou, China, in 2021. She is currently working as a college counselors in South China University of Technology, and is going to pursue the master's degree with the School of Software Engineering, South China University of Technology. Her research interests include system security and NoC-based systems.



Yingtao Jiang received the Ph. D. degree in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2001. He joined the Department of Electrical and Computer Engineering (ECE), University of Nevada, Las Vegas, NV, USA, where he was promoted to Full Professor in 2013, and subsequently served as the ECE Department Chair between 2015 and 2018. He is currently an Associate Dean of the College of Engineering, University of Nevada. His research interests include algorithms, computer architectures, VLSI, networking, nanotechnologies.

Amit Kumar Singh is an Associate Professor at University of Essex, UK. He received the B. Tech. degree from IIT, Dhanbad, India, in 2006, and the Ph. D. degree from Nanyang Technological University (NTU), Singapore, in 2013. 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 has served on the TPC of IEEE/ACM conferences like DAC, DATE, CASES and CODES+ISSS.

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.

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.