BIC: Blind Identification Countermeasure for Malicious Thermal Sensor Attacks in Mobile SoCs

Mostafa Abdelrehim*, Ahmad Patooghy†, Amin Malekmohammadi*, and Abdel-Hameed A. Badawy‡
* California State University, Bakersfield, CA 93309 USA
† North Carolina AT State University, Greensboro, NC 27411 USA
‡ New Mexico State University, Las Cruces, NM 88003 USA
{mabdelrehim, amalekm}@csub.edu, apatooghy@ncat.edu, badawy@nmsu.edu

2022 23rd International Symposium on Quality Electronic Design (ISQED) | 978-1-6654-9466-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISQED54688.2022.9806243

***Abstract*—Mobile System-on-Chips (SoCs) heavily rely on dynamic thermal management (DTM) methods in order to deal with their thermal and power density issues at runtime. The efficiency of any DTM method is directly related to the temperature data coming from the thermal sensors. For the first time, in this paper, we introduce a serious security attack on thermal sensors that can alter both the performance and reliability of the chip. We propose a Blind Identification Countermeasure (BIC) that successfully defeats the attack by identifying and isolating the infected sensor. In addition, the proposed method can accurately estimate the steady state temperature of the core associated with the isolated thermal sensor so that the DTM can continue its services with no interruption. Based on our wide range of evaluations, BIC can provide an excellent accuracy of 100% in detecting attacking sensors with a maximum temperature estimation error of ≈0.18°C. Also, BIC inflects a negligible performance overhead of 0.7% when tested with Geekbench 4.3.1 benchmark suite.**

I. Introduction

Mobile SoCs currently provide high performance multicore processors to fulfil the elevated demand in smartphone market [1]. Mobile devices have become a major tool to achieve a variety of tasks during their daily life. Mobile SoCs, however, has limited thermal-power budget despite the growing performance requirements. This is due to the limited cooling capabilities to maintain smaller form factor. Also the hand-held nature of smartphones requires less skin temperature to avoid burning sensation to the users [2]. In addition to that, generally higher operating temperatures reduce the lifetime of chips since it accelerates various failure mechanisms, e.g., electromigration. The increased leakage power is another issue that should be considered. For these reasons, Dynamic Thermal Management (DTM) techniques become essential in all mobile processors to reduce and control high operating temperature [3]–[5].

As DTM directly relies on thermal sensors, the accuracy and trustworthiness of thermal sensors are key factors. In this paper, we address the situations at which a thermal sensor is maliciously or benignly not functioning properly. This may happen due to either i) having unwanted modifications in the sensor circuit design in order to facilitate future security attacks (also known as Hardware Trojan) [6] or ii) permanent or transient faults occurrences in the sensor or its accompanying circuitry [7]. A malfunctioning thermal sensor would simply report $t \pm \Delta t_{error}$ where t is the actual temperature of the core and Δt_{error} is the error injected by the sensor. That may either cause performance degradation by, for example, asserting unnecessary frequency throttling by the DTM or accelerate aging processes reducing the lifetime of the chip.

A Hardware Trojan (HT) might be inserted to the design through various sources including having untrusted individuals in the design team [8], integrating unverified IP cores, which are designed out of the design team as a third-party IP (3rdPIP) [6], and utilization of untrusted design tools [9]. One possibility of having HT in the sensors is when using a 3rdPIP processing core with a built-in thermal sensor. The 3rd party designers know best the major sources of thermal hotspots, e.g. FPU, IPU, etc. and they should allocate a thermal sensor near the module that significantly heats up.

On top of the previous scenarios, even if the sensor is HT-free the possibility of having incorrect thermal measurements still exists. This may happen due to fabrication faults or because of aging effects. Another reason is that traditional sensors including the widely used ring oscillator thermal sensor [10] are very susceptible to noise and process variation [11]. In [12] it has been shown that, the error of the uncalibrated sensors used in IBM25PPC750L processors can be up to 34°C at an actual temperature of 95°C.

Our proposed technique, BIC, is derived from the Blind-Power Identification (BPI) algorithm [13]. BIC discovers the malicious sensor and protects the chip by offering a highly accurate estimation of the temperature that the sensor was supposed to report. The contributions of this paper are as follows:

- For the first time, we introduce a new attack model that affects thermal sensors of mobile processors. We also explore the negative consequences of the attack on both reliability and performance of the chip.
- We propose an efficient countermeasure for the attack based on detection, containment, and isolation of the malicious sensor.
- We have extensively tested this technique on multiple processor architectures to showcase its high accuracy and light overheads.

The rest of this paper is organized as follows: Section II briefly introduces BPI algorithm and its capabilities. Section III details the attack model and its negative consequences. BIC is analyzed in Section IV. The experimental setup and results of BIC accuracy and performance overhead are stated in Section V. Finally Section VI concludes the paper.

II. Background: Blind Power Identification

Blind Power Identification (BPI) technique accurately cha acterizes the relation between the temperature and pow consumption of a given chip [13]. Assuming N core processc the method predicts the chip's temperature based on Eq. 1

$$\mathbf{T}(k) = \mathbf{A}\mathbf{T}(k-1) + \mathbf{B}\mathbf{P}(k), \quad (1)$$

where $\mathbf{T}(k)$ and $\mathbf{P}(k) \in \mathbb{R}^{N\times M}$ at which each column respe tively represents the temperature and power of the cores at instant of time $k \in [0\ 1\ \cdots\ M]$. $\mathbf{A}$ and $\mathbf{B} \in \mathbb{R}^{N\times N}$ capture the physical characteristics of the system and the mutual relations between the system's individual units. While estimating $\mathbf{A}$ is straightforward [13], the estimation of $\mathbf{B}$ needs an indirect approach. To estimate $\mathbf{B}$, BPI stresses each core such that its temperature reaches a steady state (SS) situation. At SS, $\mathbf{T}(k)\approx\mathbf{T}(k-1)=\mathbf{T_S}$, thus the relation between $\mathbf{T_S}$ and SS power $\mathbf{P_S}$ can be simplified to $\mathbf{T_S} \approx (\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{P_S} = \mathbf{R}\mathbf{P_S}$.

$\mathbf{R} \in \mathbb{R}^{N\times N}$ is in fact the SS thermal resistance transfer matrix that relates SS power and temperature in analogy to current and voltage in electrical circuits, respectively. Although all entries of $\mathbf{T_S}$ are known, only the sum of each column in $\mathbf{P_S}$, i.e. total power, is known. This is due to the big cost/size of power sensors. Chips generally have only one power sensor that measures the total power while they usually accommodate multiple thermal sensors [13]. BPI can estimate both $\mathbf{R}$ and $\mathbf{P_S}$ in three steps:

1) Using Nonnegative Matrix Factorization algorithm [14], $\mathbf{T_S}$ is factorized to $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{P}}_\mathbf{S}$ as first estimations for $\mathbf{R}$ and $\mathbf{P_S}$.
2) One characteristic of $\mathbf{R}$ is that the maximum element in each column should be on the diagonal. This requires reordering $\tilde{\mathbf{R}}$ columns and the corresponding $\tilde{\mathbf{P}}_\mathbf{S}$ rows such that the maximum elements are always positioned on the diagonal of $\tilde{\mathbf{R}}$.
3) The sum of each column in $\tilde{\mathbf{P}}_\mathbf{S}$ is not yet guaranteed to be equal to the actual total power measured. Therefore BPI estimates a scaling factor $\alpha=[\alpha_1\ \alpha_2\ \cdots\ \alpha_N]$ to justify this property. The final estimations are:

$$\mathbf{R} = diag(\alpha_1, \alpha_2, \cdots \alpha_N)\cdot\tilde{\mathbf{R}} \text{ and } \mathbf{P_S} = [\alpha_1\ \alpha_2\ \cdots\ \alpha_N]\cdot\tilde{\mathbf{P}}$$

III. Threat Model & Impacts

A. Threat Model

In this paper, the threat model is an HT-infected thermal sensor with the following characteristics:

- Able to identify the ambient temperature. This is possible during sleeping periods where no read requests would be received from the DTM in sleeping mode.
- Has a random back off time between attack triggers. Also, the first attack takes place after a long time of the first usage of the chip, e.g. weeks; otherwise the HT can be detected during chip testing and calibration phase before the final production.
- Triggers the attack only when SS temperature is reached and stops if the temperature declines to the ambient temperature or after some predefined delay.
- Similar to many other researchers [15, 16], we consider the existence of a single HT in the system. This assumption is reasonable even from adversaries' point of view as inserting more HTs makes the detection process easier.

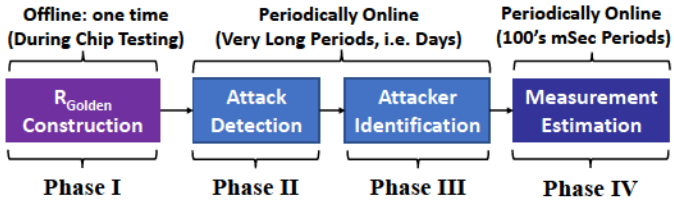


Fig. 1: The four phases of the proposed BIC countermeasure.

B. Negative Impacts

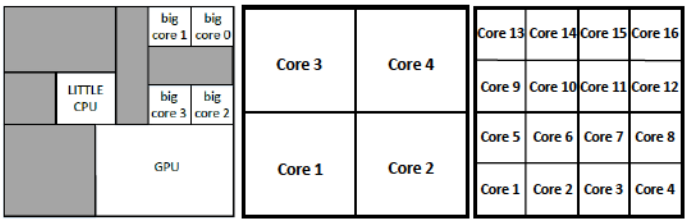
A malicious sensor reports a misleading temperature to DTM by adding $\pm\Delta t_{error}$ to the original correct temperature reading. In case of $+\Delta t_{error}$, the DTM might think the temperature is too high and exceeds the thermal threshold and try to reduce the power. The reaction might be throttling the clock frequency of the core, migrating the running thread to another colder core, or even power-gating the core completely, causing unnecessary performance loss. In case of $-\Delta t_{error}$, the situation is more dangerous. The reduced temperature would cause the DTM to believe the situation is safe and mistakenly take no action. In that case, high temperatures would deteriorate the QoS provided to the user in many ways. First it might cause a burning sensation, secondly, it increases the leakage power which accelerate the battery drainage. Finally, this also affects the reliability of the chip by reducing its lifetime [17]. It is even widely accepted in industry that 10°C increase in operating temperature would reduce the average lifetime of the chip to half (10°C half-life rule of thumb) [18].

IV. BIC: Blind Identification Countermeasure

The thermal resistance transfer matrix, $\mathbf{R}$, is a function of the layout geometry and the materials of the chip and also the heatsink or cooling properties. Most modern mobile SoCs adopt passive phase change material cooling strategies [19]–[21]. That ensures constant heatsink and cooling properties at all times. Therefore, once $\mathbf{R}$ is obtained, any trial to re-estimate it should result in the same matrix. This is the main idea behind our Blind Identification Countermeasure (BIC) that consists of four major phases as shown in Fig. 1.

1) Phase I - $\mathbf{R_{Golden}}$ Construction: This phase runs during chip testing before the final production. In testing, engineers check for fabrication faults through scan-chain, monitor the target performance, and calibrate the on-chip thermal sensors using off-chip thermal measurements with significantly higher accuracy such as a high resolution IR camera. The objective of phase I is to estimate the correct $\mathbf{R}$; henceforth $\mathbf{R_{Golden}}$. To estimate $\mathbf{R}$, thermal measurements are required. The testing engineer can still depend on the sensor measurements including the malicious sensor. As assumed earlier in Section III the attacker will trigger for the first time after a long time from the first usage of the chip. However, even if for some tested chips the attacker triggers early, the test engineer can determine $\mathbf{R_{Golden}}$ by voting between all the estimations from different test chips where in the majority the malicious sensors should not trigger early.

2) Phase II - Attack Detection: This step occurs during runtime of the multicore processor when BIC re-estimates



(a) Heterogeneous. **(b) 2x2.** **(c) 4x4.**

Fig. 2: Different processor layouts used in this study. Similar to [22], in (a) we exclude the non thermally limited modules in the shaded areas.

$\mathbf{R}$ using the on-chip thermal sensors. If the attack triggers during this phase, the resulting $\mathbf{R}$ (henceforth $\mathbf{R_{Runtime}}$) will be different from $\mathbf{R_{Golden}}$. Assuming N core system and ξ is a small tolerance set by the designer to judge on the difference between $\mathbf{R_{Runtime}}$ and $\mathbf{R_{Golden}}$, then the condition for equality can be set as follows:

$$(|\mathbf{R_{Golden}} - \mathbf{R_{Runtime}}|)_{ij} \preceq \xi \Rightarrow \mathbf{R_{Golden}} = \mathbf{R_{Runtime}}. \quad \forall\, i,j \in (1:N) \tag{2}$$

Otherwise, BIC assumes an attack has happened. The accuracy of attack detection is dependent on the fine tuning of ξ. The fine tuning should take place during chip testing. The testing engineer can add manually an error to the temperature reading and observe the change in $\mathbf{R_{Runtime}}$ versus $\mathbf{R_{Golden}}$ and based on that, ξ can be tuned to achieve maximum accuracy. Finally, this phase should run periodically in the final product but the period can be relaxed to minimize the performance overhead. For example, it can run before a sleep state of the system, as a system security update, or even during user reboot.

3) Phase III - Attacker Identification: This phase runs directly after discovering the attack in phase II. To identify the malicious sensor, we refer to the fact that BPI can build the thermal matrix for a subset of cores in the main system. For example, if there is a big cache block that minimally contributes to the thermal profile, this cache can be excluded when estimating $\mathbf{R}$. There are two ways to get the thermal submatrix for a subsystem: 1) if the whole system $\mathbf{R}$ is known then all subsystem matrices are contained within this thermal matrix, and 2) if $\mathbf{R}$ is unknown, then BPI can build directly a subsystem by running stress workloads only on the modules under consideration while keeping all other modules idle so that they do not contribute to the temperatures of the considered modules. The main idea behind identification is that if we already know $\mathbf{R_{Golden}}$ then the submatrices are already part of it. For example, for N=3 if:

$$\mathbf{R_{Golden}} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}.$$

The submatrices can be derived directly as follows:

$$\mathbf{R_{Golden\text{-}Sub1}} = \begin{bmatrix} R_{22} & R_{23} \\ R_{32} & R_{33} \end{bmatrix}, \quad \mathbf{R_{Golden\text{-}Sub2}} = \begin{bmatrix} R_{11} & R_{13} \\ R_{31} & R_{33} \end{bmatrix},$$

$$\mathbf{R_{Golden\text{-}Sub3}} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}.$$

At runtime, since we already know $\mathbf{R_{Runtime}}$ is erroneous, all submatrices of all subsystems are directly estimated using the second method by setting one core idle at a time. For example, to estimate $\mathbf{R_{Runtime\text{-}Sub1}}$ core 1 is set idle and then only the thermal sensors of cores 2 and 3 beside the total power measurements are used. Therefore, if the thermal sensor of core 1 is the malicious one, then we should find $\mathbf{R_{Golden\text{-}Sub1}}$=$\mathbf{R_{Runtime\text{-}Sub1}}$. Also, we should find $\mathbf{R_{Golden\text{-}Sub2}}$≠$\mathbf{R_{Runtime\text{-}Sub2}}$ and $\mathbf{R_{Golden\text{-}Sub3}}$≠$\mathbf{R_{Runtime\text{-}Sub3}}$ because the estimation of both $\mathbf{R_{Runtime\text{-}Sub2}}$ and $\mathbf{R_{Runtime\text{-}Sub3}}$ depends on the malicious readings from the thermal sensor connected to core 1. Based on these observations, BIC estimates all submatrices during runtime and compares each one to its corresponding submatrix in $\mathbf{R_{Golden}}$. The one with minimum error difference allows BIC to identify the malicious sensor. Later in Section V-B, numerical examples from our simulations are provided to better clarify how the runtime submatrices and the original $\mathbf{R_{Runtime}}$ would differ from the correct golden values after introducing errors.

4) Phase IV - Measurement Estimation: After identifying the malicious sensor, it is still important to estimate the core temperature associated to it. BIC uses $\mathbf{R_{Golden}}$ to estimate the SS value of that missing measurement. BIC can do that if it could estimate the power consumption of each core. Without loss of generality, let's assume N=3 with core 1 connected to the malicious sensor. If $\mathbf{t_S}$ is the SS temperature vector of the system then $\mathbf{t_S}$=$\mathbf{R_{Golden}}$·$\mathbf{p_S}$, where $\mathbf{p_S}$ is the SS power vector (Eq. 3).

$$\mathbf{t_S} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \mathbf{R_{Golden}} \cdot \mathbf{p_S} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{3}$$

The red-colored symbols in Eq. 3 are unknowns since we only know the total power and aim to estimate core 1 SS temperature. To solve for $\mathbf{p_S}$ and t_1, BIC performs two steps: (1) estimating $\mathbf{p_S}$ by least-square minimization of the following system assuming P_{tot} is the measured total power:

$$\min \left\| \begin{bmatrix} t_2 \\ t_3 \end{bmatrix} - \begin{bmatrix} R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \right\|_2^2$$

$$\text{s.t.} \quad \|\mathbf{p}\|_1 = P_{tot} \quad \& \quad \mathbf{p} \geq \mathbf{0}$$

(2) finding t_1 where $t_1 = [R_{11}\ R_{12}\ R_{13}]\,[p_1\ p_2\ p_3]^\intercal$.
This case example can simply be extended to any N with no change in the rationale.

V. Evaluations

A. Simulation and Experimental Setup

We choose the three different mobile processor layouts shown in Fig. 2 to evaluate the security achievements and performance overheads of BIC: i) a heterogeneous 6-cores processor [23], ii) a 4-cores processor (2×2), and iii) 16-cores processor (4×4). The heterogeneous and 2×2 layouts represent the state-of-the-art of application processors in the market while the 4×4 layout can be considered a logical near future step of application processors. The setup of each scenario can be summarized as follows:

1) Power traces are randomly generated based on predetermined power budgets. We set the power budgets and geometry information of each layout similar to [13, 22].

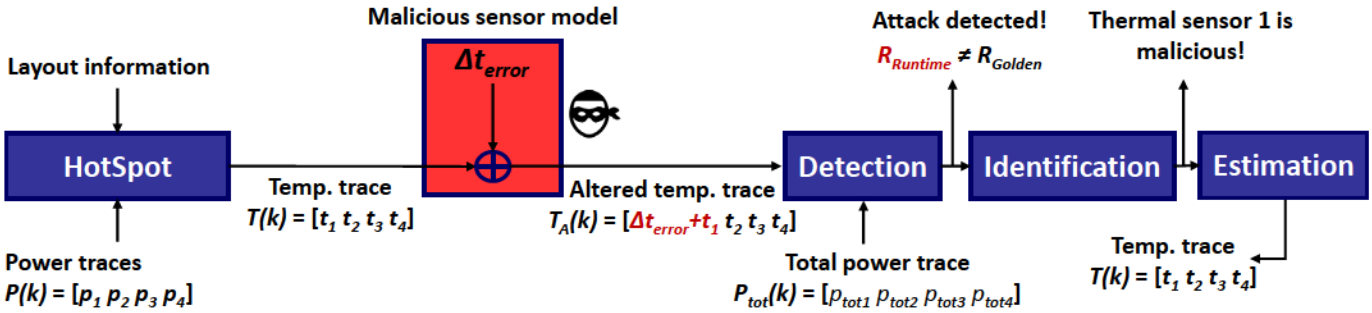


Fig. 3: The simulation setup to show how the malicious thermal sensor is simulated assuming a 4-cores processor. In that example, thermal sensor 1 is assumed malicious thus an error is introduced to t_1 which is core 1 estimated temperature.

2) We feed the power traces and the layout information to HotSpot 6.0 thermal simulator [24] to generate the corresponding thermal traces.
3) One core is chosen iteratively and its thermal sensor is considered malicious to test BIC in all different situations.
4) To model the malicious sensor, the HotSpot reported temperature is altered by adding Δt_{error}.
5) Δt_{error} iteratively changes in two ranges (−15:−1) and (1:15) with 1°C step. Although big errors, e.g. ±10, are more reasonable to achieve more damage, we included small errors to test the limits of BIC.
6) ξ changes based on the layout under test; Heterogeneous ξ=(0.1:1.5) with 0.1 step, 2×2 ξ=(0.01:0.15) with 0.01 step, and 4×4 ξ=(0.1:0.2) with 0.01 step.

For each tuple $(\xi, \Delta T)$ the simulator counts the number of failed attempts. A failure counts if either BIC failed to detect the attack or identify the attacker. Therefore the maximum number of failures at any tuple is the number of cores N. An example of how the simulation is running is shown in Fig. 3 assuming core 1 thermal sensor is malicious.

B. Results

1) BIC Attack Detection & Identification Accuracy: As shown in Fig. 4, BIC shows high accuracy in detecting sensor attacks for small ξ. At a fixed Δt_{error}, the number of failures is proportional to ξ and plateaued at certain values depending on the scenario. The Plateau is at N, meaning that in all tests BIC failed to discover either the attack or to identify the attacker. The justification behind that behavior is that based on the introduced Δt_{error} the elements of $\mathbf{R_{Runtime}}$ will change with some certain errors from the original corresponding elements in $\mathbf{R_{Golden}}$ and if the tolerance exceeds that difference then BIC will not be able to detect the error.

It is also noticeable in Fig. 4 and Table I, that the accuracy is affected by Δt_{error}. BIC shows no failures when $|\Delta t_{error}|$ is high (Heterogeneous: -5≥Δt_{error}≥4, 4×4: -4≥Δt_{error}≥-3, and 2×2: -4≥Δt_{error}≥2). Also, out of these cases, higher failure percentages occur at even much smaller Δt_{error}. For example, as shown in Table I in 2×2 case 100% of identification failures occurs at Δt_{error}=1 and ≈55% of attack detection failures takes place at Δt_{error}=±1. The same conclusions can be derived under other scenarios as well.

2) Detection & Identification Case Study: To justify more the detection and identification phases of BIC we shed light on some numerical examples. We assume the 2×2 case with core 1 connected to the malicious sensor with ξ=0.1 and Δt_{error}=5°C. The following are both $\mathbf{R_{Golden}}$ and $\mathbf{R_{Runtime}}$ that BIC estimated:

$$\mathbf{R_{Golden}} = \begin{bmatrix} 0.7214 & 0.1045 & 0.1046 & 0.0995 \\ 0.0988 & 0.7068 & 0.0947 & 0.1044 \\ 0.0917 & 0.0874 & 0.7028 & 0.0959 \\ 0.0959 & 0.1065 & 0.1052 & 0.7046 \end{bmatrix},$$

$$\mathbf{R_{Runtime}} = \begin{bmatrix} 1.1600 & 0.2115 & 0.2178 & 0.1946 \\ 0.0814 & 0.7582 & 0.0862 & 0.0911 \\ 0.0608 & 0.0643 & 0.7373 & 0.0747 \\ 0.0712 & 0.0927 & 0.0972 & 0.6911 \end{bmatrix}.$$

The maximum absolute error is 0.4386 is found between $\mathbf{R_{Runtime-11}}$ and $\mathbf{R_{Golden-11}}$ (|1.1600−0.7214|=0.4386⇒ 60.8% error). Since ξ=0.1 BIC can successfully detect the attack. The identification can be achieved from the estimation of $\mathbf{R_{Runtime}}$ submatrices:

$$\mathbf{R_{Runtime\text{-}Sub1}} = \begin{bmatrix} 0.6842 & 0.0839 & 0.0975 \\ 0.0712 & 0.6710 & 0.0856 \\ 0.0886 & 0.0898 & 0.7000 \end{bmatrix},$$

$$\mathbf{R_{Runtime\text{-}Sub2}} = \begin{bmatrix} 1.1965 & 0.2436 & 0.2171 \\ 0.0323 & 0.7633 & 0.0498 \\ 0.0060 & 0.0371 & 0.6685 \end{bmatrix},$$

$$\mathbf{R_{Runtime\text{-}Sub3}} = \begin{bmatrix} 1.1540 & 0.2365 & 0.2149 \\ 0.0437 & 0.7415 & 0.0527 \\ 0.0188 & 0.0401 & 0.6647 \end{bmatrix},$$

$$\mathbf{R_{Runtime\text{-}Sub4}} = \begin{bmatrix} 1.3167 & 0.2386 & 0.2521 \\ 0.0102 & 0.7085 & 0.0278 \\ 0.0007 & 0.0133 & 0.7358 \end{bmatrix}.$$

By calculating the maximum errors we find that:
$\max(\{|\mathbf{R_{Runtime-sub1-ij}} - \mathbf{R_{Golden-sub1-ij}}|\}) = 0.0318$,
$\max(\{|\mathbf{R_{Runtime-sub2-ij}} - \mathbf{R_{Golden-sub2-ij}}|\}) = 0.4751$,
$\max(\{|\mathbf{R_{Runtime-sub3-ij}} - \mathbf{R_{Golden-sub3-ij}}|\}) = 0.4326$, and
$\max(\{|\mathbf{R_{Runtime-sub4-ij}} - \mathbf{R_{Golden-sub4-ij}}|\}) = 0.5953$
$\forall\ i, j \in \{1, 2, 3\}$.
The minimum absolute error occurs in $\mathbf{R_{Runtime-sub1}}$. That means the thermal traces used to estimate this matrix, i.e. t_2, t_3, and t_4, were error-free. This is the correct conclusion because in that scenario the sensor of core 1 is the adversary. Therefore BIC could also identify correctly the malicious sensor.

3) Estimation Accuracy and Performance Overhead: BIC shows great accuracy in estimating the missing thermal sensor reading. As shown in Table II, for the three multicore layouts,

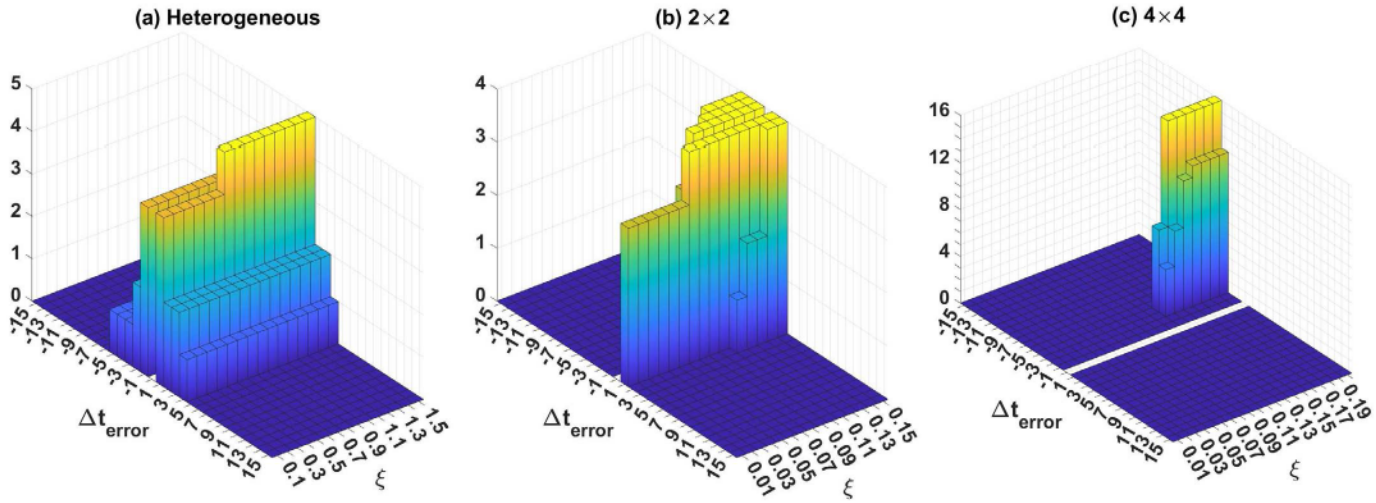


Fig. 4: Failure counts versus (Δt_{error}, ξ). As shown BIC failures only occur at small $|\Delta t_{error}|$ in all the three tested scenarios.

TABLE I: % Detection and identification failures versus Δt_{error} for different multicore layouts. The shaded cells are the only cases when BIC fails in either detection or identification otherwise BIC shows 100% success.

Δt_{error}	Heterogeneous		2×2		4×4	
	Detection Failure	Identification Failure	Detection Failure	Detection Failure	Detection Failure	Detection Failure
-6:-15	0%	0%	0%	0%	0%	0%
-5	0%	6.33%	0%	0%	0%	0%
-4	0%	6.33%	2.33%	0%	59.54%	0%
-3	0%	6.33%	13.95%	0%	40.46%	0%
-2	0%	12.66%	16.28%	0%	0%	0%
-1	28.57%	20.25%	25.58%	0%	0%	0%
1	52.38%	19.83%	31.78%	100%	0%	0%
2	19.05%	9.3%	10.08%	0%	0%	0%
3	0%	12.66%	0%	0%	0%	0%
4	0%	6.33%	0%	0%	0%	0%
5	0%	0%	0%	0%	0%	0%
6:15	0%	0%	0%	0%	0%	0%

TABLE II: Temperature estimation accuracy of BIC phase IV.

Processor Layout	Max. Err.	Avg. Err.
6 Cores (Heterogeneous)	0.0097°C	0.0029°C
4 Cores (2×2)	0.1836°C	0.0787°C
16 Cores (4×4)	0.0302°C	0.0080°C

the maximum absolute error is ≤0.18°C. Thanks to BPI's excellent accuracy in estimating $\mathbf{R_{Golden}}$ [13].

Due to the relaxed nature in running phases II and III, their performance overheads can be neglected. Although phase IV might incur performance overhead. A typical DTM technique periodically samples the sensor measurements. Sampling incurs delay and therefore the sampling frequency should be carefully chosen. Since the temperature has an analogy to capacitance charging, the temperature reaches SS in hundreds of mSecs. Therefore usually sampling is done with hundreds of mSecs rates [25].

We adopted the Snapdragon 865 Mobile Hardware Development Kit (HDK) as a realistic mobile computing platform to test the performance overhead of BIC phase IV. This Android HDK contains a Qualcomm Snapdragon SM8250 – 64-bit octa-core Kryo 585 Processor [26]. We test the overhead by running Geekbench 4.3.1 computing benchmarks [27] once without running phase IV and then one more run with phase IV in the background. To test the overhead, we set the sampling to be 250mSec. We repeat this process four times to reduce the variance. BIC estimation algorithm is written in Matlab based on the original BPI source code [28] and then converted to C using Matlab Coder. The benchmarks scores are shown in Fig. 5 before and after running phase IV. As shown the average performance overhead is negligible just ≈0.7% confirming the lightweight overhead of BIC phase IV.

VI. Conclusions

In this paper, we resolve a serious situation at which a thermal sensor either intentionally or naturally reports wrong temperature readings. We show that a malicious reported temperature can result in significant QoS degradation in either performance or reliability of multicore chips. We also present a blind identification method to detect, locate, and isolate the malicious thermal sensor. The proposed method; BIC, shows 100% accuracy in discovering the attack and identifying the attacker when the absolute value of the added temperature error is ≥6°C. Also, the proposed method estimates the steady state missing temperature with an error ≤0.18°C. The steady state temperature estimation imposes a very small average performance overhead of 0.7% when running Geekbench 4.3.1 benchmark suite on real mobile processor.

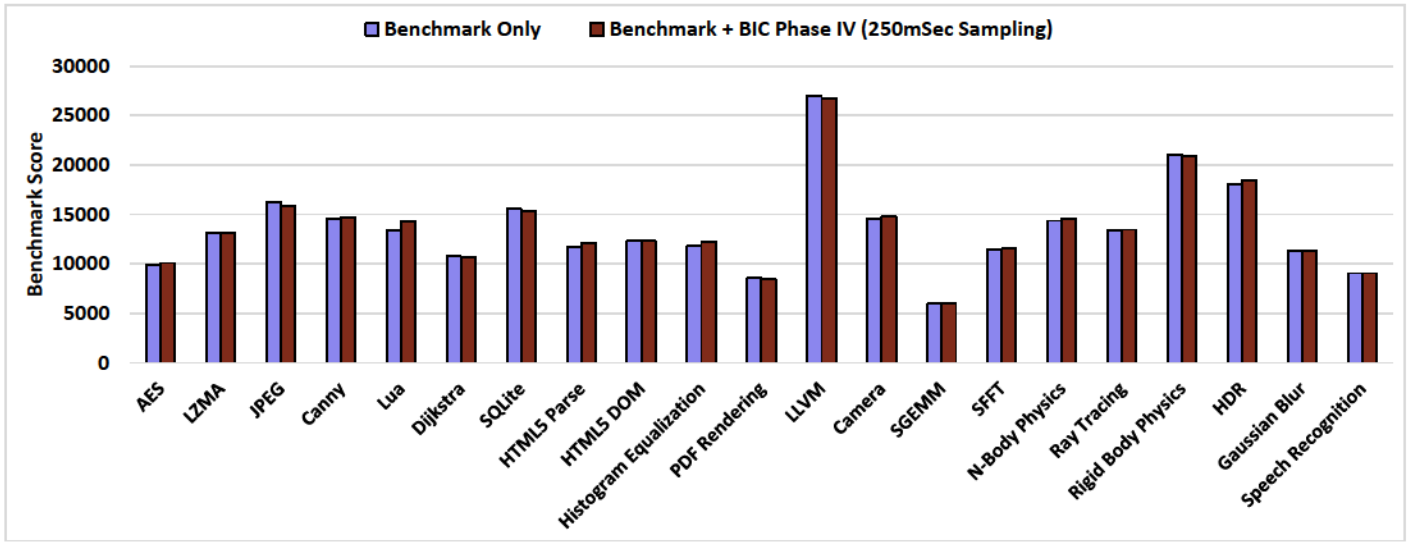


Fig. 5: The scores of Geekbench 4.3.1 multithreaded benchmarks running on Snapdragon 865 octa-core before and after running BIC phase IV. As shown the average overhead is very small ≈0.7%.

VII. Acknowledgement

The authors would like to thank Professor Sherief Reda and his PhD candidate Sofiane Chetoui from SCALE lab at Brown University. Professor Sherief has supported the work with with technical feedback and by providing the Qualcomm Snapdragon SM8250 HDK for experimentation of that work.

References

[1] D. Pandiyan, S.-Y. Lee, and C.-J. Wu, "Performance, energy characterizations and architectural implications of an emerging mobile platform benchmark suite - mobilebench," in *2013 IEEE International Symposium on Workload Characterization (IISWC)*, 2013, pp. 133–142.

[2] Y. Hang and H. Kabban, "Thermal management in mobile devices: challenges and solutions," in *2015 31st Thermal Measurement, Modeling Management Symposium (SEMI-THERM)*, 2015, pp. 46–49.

[3] A. Prakash, H. Amrouch, M. Shafique, T. Mitra, and J. Henkel, "Improving mobile gaming performance through cooperative cpu-gpu thermal management," in *Proceedings of the 53rd Annual Design Automation Conference*, ser. DAC '16, 2016.

[4] S. Chetoui and S. Reda, "Coordinated self-tuning thermal management controller for mobile devices," *IEEE Design Test*, vol. 37, no. 5, pp. 34–41, 2020.

[5] O. Sahin, L. Thiele, and A. K. Coskun, "Maestro: Autonomous qos management for mobile applications under thermal constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1557–1570, 2019.

[6] J. Zhang, F. Yuan, and Q. Xu, "Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 153–166.

[7] A. Webber and A. Haj-Omar, "Calculating useful lifetimes of temperature sensors," *TI Application Report*, July 2018.

[8] S. M. Sebt, A. Patooghy, H. Beitollahi, and M. Kinsy, "Circuit enclaves susceptible to Hardware Trojans insertion at gate-level designs," *IET Computers & Digital Techniques*, vol. 12, no. 6, pp. 251–257, 2018.

[9] D. Mukhopadhyay and R. S. Chakraborty, *Hardware security: design, threats, and safeguards*. CRC Press, 2014.

[10] C.-A. Lefebvre, L. Rubio, and J. Montero, "Digital thermal sensor based on ring-oscillators in Zynq SoC technology," 09 2016, pp. 276–278.

[11] X. Li, X. Ou, Z. Li, H. Wei, W. Zhou, and Z. Duan, "On-Line Temperature Estimation for Noisy Thermal Sensors Using a Smoothing Filter-Based Kalman Predictor," *Sensors (Basel, Switzerland)*, vol. 18, no. 2, p. 433, Feb 2018.

[12] S. Remarsu and S. Kundu, "On Process Variation Tolerant Low Cost Thermal Sensor Design in 32nm CMOS Technology," in *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI '09, New York, NY, USA, 2009, p. 487–492.

[13] S. Reda, K. Dev, and A. Belouchrani, "Blind Identification of Thermal Models and Power Sources From Thermal Measurements," *IEEE Sensors Journal*, vol. 18, no. 2, pp. 680–691, Jan 2018.

[14] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct 1999.

[15] M. M. Ahmed, A. Ganguly, A. Vashist, and S. M. Pudukotai Dinakarrao, "AWARe-Wi: A jamming-aware reconfigurable wireless interconnection using adversarial learning for multichip systems," *Sustainable Computing: Informatics and Systems*, vol. 29, p. 100470, 2021.

[16] I. H. Abbassi, F. Khalid, S. Rehman, A. M. Kamboh, A. Jantsch, S. Garg, and M. Shafique, "Trojanzero: Switching activity-aware design of undetectable hardware trojans with zero power and area footprint," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 914–919.

[17] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, "The case for lifetime reliability-aware microprocessors," in *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, 2004, pp. 276–287.

[18] A. Webber, "Calculating useful lifetimes of embedded processors," *TI Application Report*, March 2020.

[19] A. Kurhade, V. Talele, T. Venkateswara Rao, A. Chandak, and V. Mathew, "Computational study of pcm cooling for electronic circuit of smart-phone," *Materials Today: Proceedings*, vol. 47, pp. 3171–3176, 2021, 3rd International Conference on Advances in Mechanical Engineering and Nanotechnology.

[20] Y. Tomizawa, K. Sasaki, A. Kuroda, R. Takeda, and Y. Kaito, "Experimental and numerical study on phase change material (PCM) for thermal management of mobile devices," *Applied Thermal Engineering*, vol. 98, pp. 320–329, 2016.

[21] Z. Luo, H. jung Cho, X. Luo, and K. R. Cho, "System thermal analysis for mobile phone," *Applied Thermal Engineering*, vol. 28, pp. 1889–1895, 2008.

[22] M. Said, S. Chetoui, A. Belouchrani, and S. Reda, "Understanding the Sources of Power Consumption in Mobile SoCs," in *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*, 2018, pp. 1–7.

[23] Y.-H. Gong, J. J. Yoo, and S. W. Chung, "Thermal Modeling and Validation of a Real-World Mobile AP," *IEEE Design Test*, vol. 35, no. 1, pp. 55–62, 2018.

[24] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotSpot: a compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[25] F. J. Mesa-Martinez, E. K. Ardestani, and J. Renau, "Characterizing processor thermal behavior," *SIGARCH Comput. Archit. News*, vol. 38, no. 1, p. 193–204, Mar. 2010.

[26] Snapdragon Mobile HDK Tech Specs:. [Online]. Available: https://www.lantronix.com/products/snapdragon-865-mobile-hdk/#tab-overview

[27] Geekbench 4.3.1. [Online]. Available: https://www.geekbench.com/blog/2018/11/geekbench-431/

[28] BPI Matlab Source. [Online]. Available: https://github.com/scale-lab/BPI