Emulating Side Channel Attacks on *gem5*: lessons learned

Lilian Bossuet
UJM Saint-Etienne, CNRS,
Institut d'Optique Graduate School,
Lab. H. Curien UMR 5516, F-42023,
SAINT-ETIENNE, France
lilian.bossuet@univ-st-etienne.fr

Vincent Grosso
UJM Saint-Etienne, CNRS,
Institut d'Optique Graduate School,
Lab. H. Curien UMR 5516, F-42023,
SAINT-ETIENNE, France
vincent.grosso@univ-st-etienne.fr

Carlos Andres Lara-Nino
UJM Saint-Etienne, CNRS,
Institut d'Optique Graduate School,
Lab. H. Curien UMR 5516, F-42023,
SAINT-ETIENNE, France
carlos.lara@univ-st-etienne.fr

***Abstract*—Side channel attacks (SCA) have the potential of disrupting the trust of the users on computing platforms and cryptographic algorithms. The main challenge in the design of countermeasures against such threats is that an evaluation of their effectiveness can only be performed after they have been implemented. By that point, significant resources would have been invested in the creation of a prototype. Moreover, the large volume of combinations from all the potential target algorithms and computing systems complicates a systematical analysis. It is necessary to find strategies to simplify and systematize the study of SCAs and their countermeasures. *gem5* is a cycle-accurate simulator which offers the possibility to emulate a broad range of computing architectures. Beyond the functional verification, this tool computes multiple physical statistics from the simulated system. In this paper, we discuss the lessons learned from using *gem5* to simulate SCAs on an ARM system. Our work shows that while there is a correlation between the data and the reported statistics, there are significant challenges that must be addressed to improve the use of *gem5* for the emulation of physical phenomena.**

1. Introduction

Modern computing systems (like the IoT, wearable devices, and autonomous vehicles) exhibit characteristics that distinguish them from conventional computers. They run in large, unstructured environments to bring services closer to the user. They are networked and collaborate with each other to solve complex tasks with a limited investment of resources per device. They are ubiquitous and heterogeneous. But most importantly, they manage large volumes of data which may require to be protected from unauthorized access or modification. Cryptographic algorithms are the tools of choice for supplying security services such as confidentiality and integrity to the data. They may take different forms and offer distinct performance and cost tradeoffs as to be of use for many computing systems. But all of them, from a theoretical point of view, base their security on hard computing problems. Finding the correct solution [1] would be harder than looking for a particular grain of sand on earth [2].

Under formal security assumptions [3] it is necessary to consider that a potential attacker may gain access to the forward transformation of a cryptosystem (chosen plaintext attack) and even its inverse function (chosen ciphertext attack). To regard the system as secure, it must resist any potential attacks when these oracles are available to the adversary. That is, the attacker shall not gain any other information from the queries performed to the functional oracles. However, in practice, there are more sources of knowledge which may give the adversary an advantage to compromise the security of the system. Any computing device is, by nature, an agglomeration of physical phenomena. So goes the phrase "we have tricked a rock into thinking." Consequently, there are magnitudes which can be seen and quantified when the system runs. Some of these measurements are bound to be correlated with the data being processed by the platform [4]. If the system happens to perform cryptographic operations, these will also affect the different magnitudes of the device: supply power, electromagnetic emanation, clock frequency, heat dissipation, etc. Thus, it is said that data is *leaked* from the platform. If an attacker can tap into these side channel leakages, they will gain access to a large volume of data to aid in their venture. It is suspected that any cryptographic algorithm can be broken with enough traces of its operation [5].

The power [4] and electromagnetic [6] fingerprints of a circuit are commonly used to perform SCAs on its cryptographic algorithms. These magnitudes fluctuate quickly enough to give a good indicator of the status of the device. This characteristic, however, implies that sophisticated equipment may be needed to capture the information. After all, this is a typical problem of sampling [7]. Quick variations of small circuits may only be captured by expensive oscilloscopes and spectrum analyzers. Which difficulties the task of the attacker. They not only require direct access to the device; they must also count on the necessary equipment to mount an attack. Or so we thought. Recent works have shown that attackers do not require direct access to the target platform [8], [9] nor specialized analysis devices [10], [11] to mount power analysis attacks on internet-enabled platforms. If we remember that a current trend of technology is their hyper-connectivity [12], suddenly these devastating attacks become more concerning.

Cryptographic algorithms can be protected against SCAs by employing different obfuscation techniques such as masking [13] and shuffling [14]. These countermeasures, however, are adjusted for every algorithm and require extensive testing to confirm their effectiveness [15]. Additionally, some mitigation strategies may be favored for some platforms in function of the system constraints. This implies that multiple solutions may need to be evaluated to come up with the best choice. Developing and implementing multiple prototypes can burden any project with significant expenses. Computer aided design may supply a solution to some of these problems. This strategy could be adopted to perform many tests with small implementation costs.

The *gem5* simulator [16] can emulate the operation of multiple processors including ARM and RISC-V. This is done not only in a behavioral sense but modeling the platforms at the level of micro-architectural components. *gem5* has been used to emulate attacks against micro-architectures such as RowHammer [17] and Spectre [18]. The main interest for studying these attacks in *gem5* is that this software is open source and can be easily customized to model different hardware components. For example, it is possible to emulate custom hardware accelerators along complex processor systems. It is also possible to parameterize the simulation and customize the system to test multiple architectures. Therefore, it is a useful tool to test multiple target architectures and implement and test potential countermeasures.

A *gem5* simulation creates a large set of statistics associated with the operation of the system. These data can be used to study the behavior of the different components in the simulated platform. For example, we can estimate the instructions processed by the ALU and the memory access activity. These estimations will depend on the data being processed by the core or even on the activity of any custom accelerators that can perform direct-memory access. In our case, we focus on the first scenario. As the activity of the multiple underlying components will affect the power consumption of the entire system, we propose that the simulation statistics can be used to approach the electrical behavior of a simulated platform. Therefore, if there is any correlation between the metrics of the simulation and the application data, then *gem5* could be employed to emulate SCAs.

In this paper, we use *gem5* to perform the emulation of an ARM processor with a Linux-based operating system. We employ the platform described in [19] which features TrustZone [20] and OP-TEE [21] as logical protections. Using this system, we implement different cryptographic operations and analyze the process for retrieving and processing the simulation statistics. With these data we try to perform correlation power analysis, simple power analysis, and power-based covert channels. To the best of our knowledge, this is the first time such an analysis has been conducted. Our findings show that the data acquisition process is particularly challenging to conduct some of these evaluations. Nonetheless, we show points of utility for the improvement of the simulator to make the proposed approach more practical.

The rest of the paper is structured as follows. In Section 2 we discuss related works from literature. In Section 3 we describe our methods and the results obtained. Lastly, Section 4 presents a discussion of our findings and concludes this work. As an appendix, Section A has further details on our experimental work.

2. State of the Art

The literature reports multiple analysis tools which allow to study the energy footprint of a circuit before it is implemented. They intend to model the power dissipation of the design based on different approaches. These power estimations can be used, in some cases by the same tool, to detect information leakage in the design.

Some of the most used power analysis solutions have been created by technology vendors and thus are specialized for their products. We can mention, for example, *Incisive Palladium III* by *Cadence Design Systems* and *XPower Analyzer* by *AMD-Xilinx*. The former allows to study the dynamic power dissipation of ASIC designs and the latter supplies static and dynamic power estimation for FPGAs. *Incisive Palladium III* employs the gate model of the chip whereas *XPower Analyzer* employs the RTL model of the design. Therefore, it is necessary to design and describe the architecture to use these tools.

Other tools which employ the gate model of the circuit to study potential leakage sources include PARAM [22], ACA [23], and CASCADE [24]. These tools analyze the signals or gates in the design to estimate the points of interest for side channel analysis. The main drawback for ACA and PARAM is that they are closed-source projects, and while CASCADE claims to be open source it relies on commercial EDA tools such as *PrimeTime* by *Synopsys*. RTL-PSC [25] is a tool similar in functionality, however it only requires the RTL specification of the circuit. This system also relies on proprietary software, in particular *VCS* by *Synopsys*.

Leakage verification is another approach for evaluating the vulnerability of a circuit against SCAs. It relies on formal methods for assessing the information leakage such as Hamming models and TVLA. Several tools in the market employ the gate model of the platform for conducting such analyses. We can mention SCRIPT [26] and PATCH [27] in this category. Others like AMASIVE [28] and KARNA [29] employ higher abstractions and thus require a lower design investment for their use. All these systems are proprietary.

Open-source tools for leakage verification include MAPS [30] and COCO [31]. The former is exclusive for ARM Cortex M3 systems but is not cycle accurate. The latter can analyze any circuit but requires a gate-level description of the platform. Another tool which relies on an open-source initiative is SLEAK [32]. This system employs *gem5* to perform the emulation of ARM Cortex A8 processors. However, SLEAK itself is not readily available.

The general problem with these tools is that most of them are in some part closed-source projects. Furthermore, most of them are meant to study simple models of the system under test and not complete architectures. *gem5*, on the other hand, can emulate large architectures with relative ease. Another drawback of most of the tools in the literature is that they perform leakage analysis or verification over the gate model of the system under study. This means that it is necessary to design and describe the platform to later analyze it. However, this process implies a significant investment which we intend to prevent. In contrast *gem5* only requires a functional model of the architecture thus effectively reducing the design cost.

3. Methods

In this Section we describe the virtual architecture simulated in *gem5*. We illustrate our method for retrieving the statistics of operation and how to process the information. Finally, we describe our experiments in analyzing these data to emulate SCAs.

3.1. The simulated platform

This work uses the virtual and open platform from [19] that simulates the behavior of micro-architectural features and their interactions with the peripherals, like accelerators and memories in emerging technologies.

A *gem5* simulation is composed of Python configuration files and C++ program files, which are compiled into a *gem5* binary. The simulator behaves like a Python interpreter. The configuration files connect Python objects, which are either other Python objects or *gem5* primitives standing for hardware modules, through ports generally using a *gem5* memory packet protocol. The hardware primitives can be configured using parameters directly in the Python configuration file. We can, for example, specify the type of processors to be used, the size and technology of the cache, and even include *ad-hoc* components to stand for hardware accelerators.

One of the interesting characteristics of *gem5* is that it allows to model different processor architectures (`x86, ARM, RISC-V`) and specify multiple optimization levels (`debug, opt, fast`) to study multiple characteristics of the system. In our work, we focus on `ARM` platforms and use the `opt` level to conduct our experiments. Another detail to consider is that the simulator can perform either *system-call emulation* or *full simulation* of a platform. We have chosen the latter as this method allows us to obtain statistics which are closer to those found in a physical device.

We have simulated a single-core system with an `HPI` processor model, which extends the conventional `MinorCPU` (a *timing* model). We chose to use a single core to reduce the complexity of the simulation and the number of statistics produced. The system was provided with L1 and L2 caches as well as a memory management unit (MMU). We added a 2GB `DDR3_1600_8x8` unit which was emulated using Ramulator [33]. We employed the `VExpress_`*gem5*`_Foundation` machine type which allows to execute the `ArmTrustedFirmware` workload [34]. As in [19] we have also included the OP-TEE runtime in the simulation. This allowed us to study the behavior of a *protected* system and analyze its resilience against power attacks. Figure 1 illustrates the characteristics of the simulated platform.

From every execution of the simulation, *gem5* dumps statistics like the number of `IntAlu` instructions executed or the frequency of the processor, by default with a rate of $1E-3$. That is, every simulated millisecond a statistics file is updated with different measurements. This *sampling* rate can also be adjusted in the configuration file for the simulation. In this way it is possible to generate data at a rate consistent with the sampling theorem [7]. For example, if the core is running at 1MHz then it will be necessary to use a dump rate under $5E-7$.

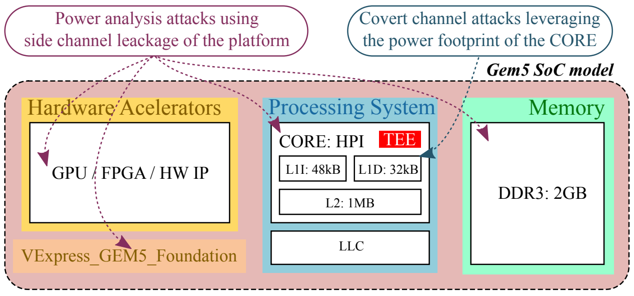


Figure 1: The architecture under analysis.

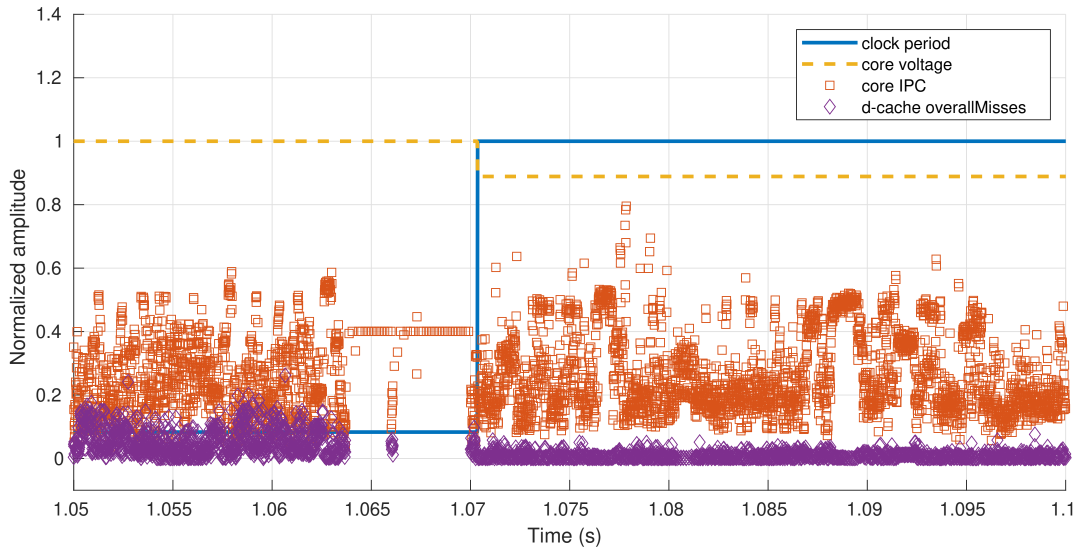
The main problem with this approach is that bumping the dump rate increases the complexity of the simulation and the volume of data produced. On the other hand, reducing the frequency of the core causes the simulation to behave erratically. The lowest frequency we managed to employ was 2MHz, but it was impossible to perform any significant computations with the system. With our evaluation platform (Ubuntu 20.04 LTS, 11th Gen Intel Core i7 @ 3.00GHz $\times$ 8, 32 GiB RAM, *gem5* v21) we found that it was possible to use a core frequency of 10MHz and an acquisition rate of 25MHz. Even then, some statistics files reached the size of 1TB. We developed basic Python and Matlab scripts to separate all the statistics into independent time-series and analyze the data.

The main idea of using the simulation statistics to approximate the power dissipation of the system comes from [35]. A simple non-representative power model can be approximated as shown in Equation 1. Evidently, the dynamic component of the model would be the most useful for power analysis. In Figure 2, we illustrate how the statistics retrieved from a *gem5* simulation can be used to approach a power estimation.

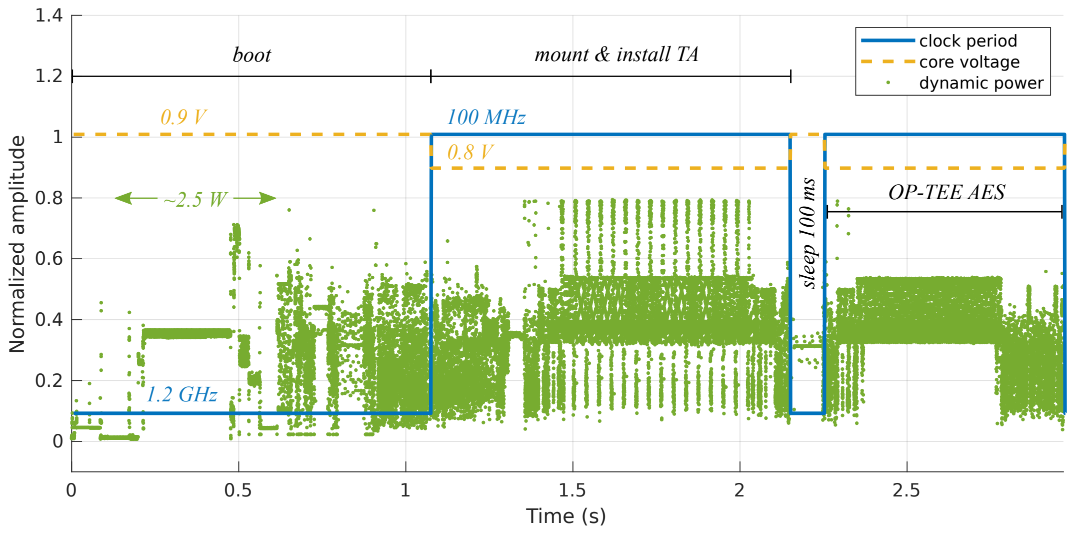
$$\begin{aligned} pow &= dyn + stat \\ dyn &= V \times (2\text{A} \times ipc + 3\text{pA} \times dentry\_misses) \\ stat &= 4 \times temperature \end{aligned} \tag{1}$$

An important challenge in the analysis of the samples obtained from the operation of a processor is to accurately find the different *parts* of its program. As illustrated in Figure 2b, we overcome this drawback by changing the core frequency at precise points of its operation. This is performed with the same mechanism we use to decrease the operational frequency to follow the sampling theorem.

Another problem with this approach is that the goodness-of-fit for a power estimation will depend greatly on the technology of the system, the architecture, and the mathematical model employed. Creating a function which can accurately describe the behavior of a physical device is equivalent to the design of its digital twin. Which is a problem outside of the scope of this work. However, as shown in Equation 1, even the most precise power model must be created from a composition of readily available statistics. If there is information concerning the operation of the system in these data, then this information would also be embedded in the power estimation. So, investigating the use of the raw statistics for emulating power attacks can be considered as an equivalent problem.



(a) *gem5* statistics



(b) estimation of dynamic power

Figure 2: Statistics obtained from a full-system *gem5* simulation of the architecture illustrated in Figure 1 with a sampling period of $1E-5$. The simulation included the boot sequence, the initialization of OP-TEE and the execution of `optee_example_aes`. The vertical axis shows the normalized amplitude since all the statistics have dynamic ranges which differ significantly; we simply normalize each trace for display purposes.

3.2. Power dissipation analysis

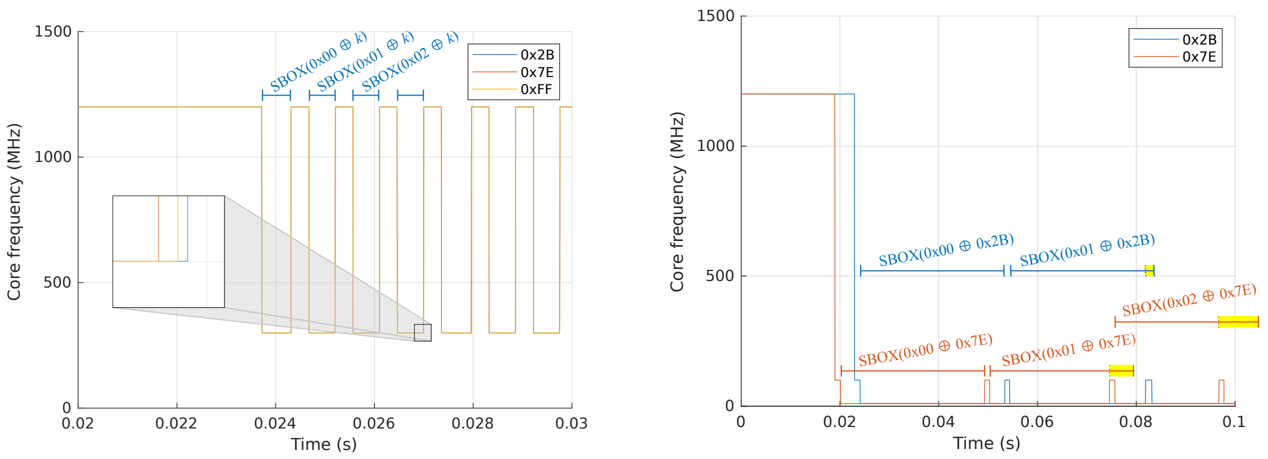
When we transition from the analysis of a power trace to the analysis of the statistics which may influence the power trace, we are confronted with a significant challenge: the volume of the data. In a regular *gem5* simulation we can find over 1,000 different metrics to analyze, which is time consuming. It is necessary to find out which statistics carry the most significant information for performing power analysis.

For this Subsection we used as case study the power analysis of AES [36], since it is a well-known subject in the literature. However, for a preliminary assessment, obtaining enough data from multiple *full* executions of AES seemed impractical. So, we simplified the algorithm under analysis to the essential operations used in SCA attacks. The pseudo-code for our simplified version of AES is provided in Algorithm 1.

Algorithm 1 Algorithm under power analysis

```
Require: $k$, an 8-bit random integer
Require: SBOX, the substitution box of AES
Require: $f, f'$, two frequencies of the core with $f' < 2f_s$

for $i = 0$ to $255$ do
    cpu_freq $\leftarrow f'$                {Pull trigger}
    SBOX($i \oplus k$)
    cpu_freq $\leftarrow f$                 {Release trigger}
    wait
end for
```



(a) sampling period is $2.5E-7$

(b) sampling period is $4E-8$

Figure 3: A simple analysis of the clock statistics from multiple experiments illustrates how under regular conditions (Figure 3a) the behavior of *gem5* is consistent. However, when the core frequency is decreased (to 10MHz in this case) the execution of the program becomes irregular (Figure 3b).

We selected two key values (`0x2B, 0x7E`) to analyze whether it was possible to find useful information in the *gem5* output. For that, we obtained the simulation statistics of the platform illustrated in Figure 1 when executing the program from Algorithm 1. From this, we obtained over 800 files holding multiple metrics from the simulation such as the number of instructions per cycle and the number of cache accesses; with some post-processing (removing incomplete and all-zero files) this number was reduced to 150. Evidently, it was not practical to process all these data with the usual statistical techniques employed for power analysis. So, we devised a strategy to reduce the number of targets for study:

STEP 1: Auto-correlation. Compute the correlation between every statistics file within a set (for example, the set of all the statistics obtained from the use of the key `0x2B`) to discard those with high similarity. Evidently, if some measurements are similar, they will have a high correlation and provide about the same information. We found 151 stat pairs with a correlation coefficient over 0.95, involving 23 files. We were still left with over 100 files to evaluate. This is shown in Figure A.1.

STEP 2: Cross-correlation. Compute the correlation between every statistics file in a set against the respective statistics in a distinct set (that is, the statistics for the key `0x2B` against the statistics for the key `0x7E`) as to drop those with high similarity. From this experiment we found only 30 stats with a correlation coefficient under 0.5 across sets, and only six stats with a correlation coefficient under 0.3 across sets. This is shown in Figure A.2.

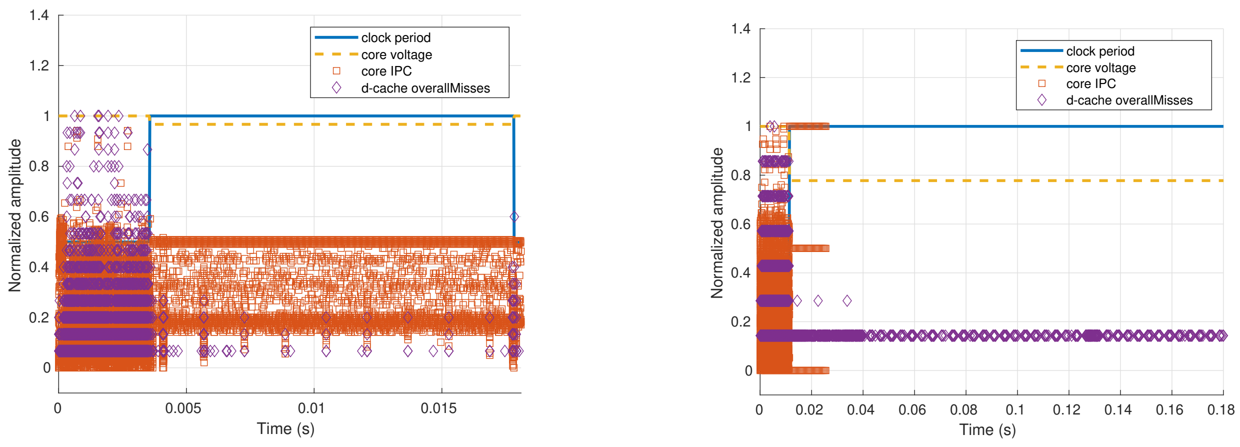
Then, we computed the correlation matrix of the statistics for one of the keys against their own set (auto-correlation) and the correlation matrix between the statistics in both sets (cross-correlation. The first experiment would give us information about the dependence between statistics and the second would highlight which statistics held more information related to the data processing. These findings are detailed in Section A.

At first sight these results seemed promising as it was possible to find which statistics were more interesting to evaluate. However, in practice, when we set $f' = 10MHz$ and $t_s = 4E-8$, as to follow the sampling theorem, the fidelity of the analysis decreased. As shown in Figure 3, the behavior of the simulation became inconsistent. This could be further corroborated with an analysis of the cross-correlation between statistics for different key values. For the experiment in Figure 3a the main diagonal of the correlation matrix showed a mean $\delta = 0.72$ and standard deviation $\sigma = 0.24$, while for the experiment in Figure 3b the respective values were $\delta = 0.47$ and $\sigma = 0.28$. Notably, the correlation coefficient between the clock signals decreased from 1.00 to 0.56. Another issue we met was that for the experiment in Figure 3b we could only capture 20 traces. This task took over 72h for each key value and yielded 1TB of statistics in each case.

Despite these findings, we decided to test whether it was possible to perform Test Vector Leakage assessment [37] over the data (evaluating the complete sets of stats). This should have helped to find out if the leakages we saw were dependent or not of the secret value. For this experiment, the secret values were the key values `0x2B` and `0x7E` (these values are found in the generic AES key used to obtain test vectors) and we performed a `Student's` t`-test` to see if the leakages' distribution differed from one another. However, using the raw leakages, no statistics overtook the leakage detection threshold of 4.5. This confirmed the null hypothesis of the t`-test` that both sets followed the same distribution. Supposing leakages are independent of the key used, it seems impossible to mount a successful differential attack using single *gem5* statistics. These negative results may change if we can get a larger set of traces for each key, apply signal processing techniques to improve the synchronization of the traces, or consider attacks on various statistics at the same time.

3.3. Timing analysis

Even though we were unable to show a clear relationship between the *gem5* statistics and the secret key



(a) target frequency of 600MHz

(b) target frequency of 10MHz

Figure 4: A timing analysis of the execution of the program in Algorithm 2 with a sampling frequency of 25 MHz. The vertical axis shows the normalized amplitude since all the statistics have dynamic ranges which differ significantly; we simply normalize each trace for display purposes.

of AES, it was clear from Figure 2b that there was some influence of the algorithm in the model. So, we decided to evaluate whether we could perform simple power analysis using the simulator.

It is well known that unprotected implementations of algorithms like RSA [38] and elliptic curve cryptosystems can leak the secret key via a simple timing analysis. This is most commonly due to the use of naive multiplication algorithms which perform branching operations as a function of the key value.

Given the glaring drawbacks of *gem5* for simulating complex operations we decided to skip on RSA. Instead, we analyzed a simple binary-field double-and-add 1024-bit multiplication implemented in C. The algorithm under analysis for this scenario is provided in Algorithm 2. We conducted two experiments to evaluate our hypothesis, see Figure 4.

Algorithm 2 Algorithm under timing analysis

```
Require: $g(t)$, an irreducible polynomial for $\mathbb{K}$
Require: $a, b \in \mathbb{K}$, two random integers
Require: $f, f'$, two frequencies of the core with $f' < 2f_s$

  cpu_freq ← $f'$
  for $i = 0$ to $5$ do
    $a \times b \in \mathbb{K}$
  end for
  cpu_freq ← $f$
```

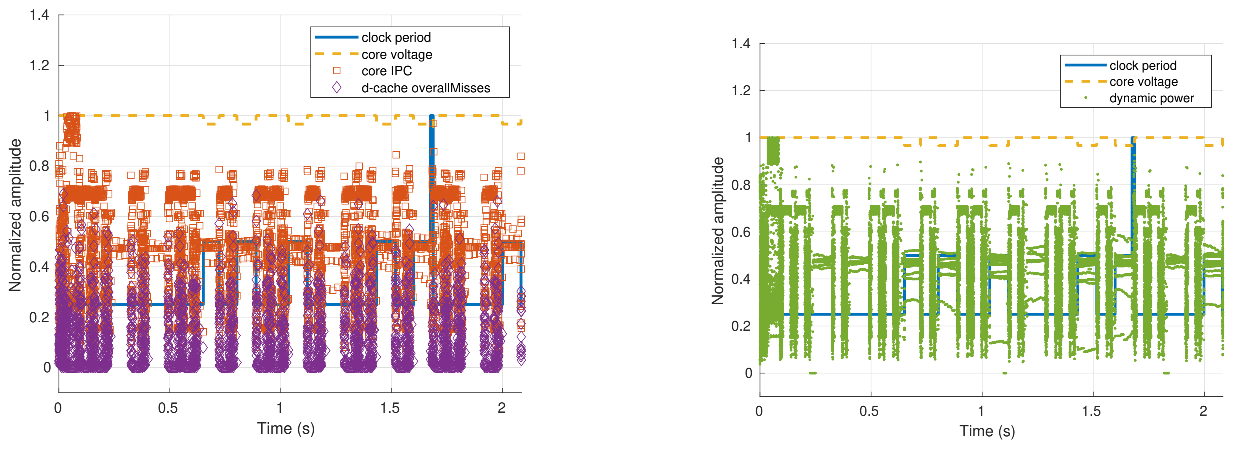
In the first experiment (Figure 4a) we set the core frequency to 600 MHz and used an acquisition rate of $4E-8$. This allowed us to see the activity of the core during processing. However, we could not find any timing pattern which corresponded with the execution of the field multiplications. So, we decreased the core frequency to investigate whether following the sampling theorem was necessary to find any timing behavior. But the result was just the opposite. As illustrated in Figure 4b the statistics produced were of inferior quality. For example, in the case of the IPCs, the number of NAN values increased from less than 1% to over 60%. Once again, we arrived at the conclusion that *gem5* simulations behave inconsistently when low frequencies are used.

3.4. Covert channel attacks

SCAs do not only focus on retrieving side channel information. Another vulnerability reported in the literature [39] is to use the power footprint of the platform to transfer information covertly. In this attack model, the adversary leverages the use of components in the platform to produce a discernible pattern in the energy consumption of the circuit. For example, hardware components, accelerators, and memory elements can be activated or accessed with a pattern to transfer a message. The receiver is generally another component within the system which would not normally be allowed to share a communications channel with the sender. In the presence of trusted execution environments like OP-TEE, the exclusion level of different peripherals or accelerators can mitigate such attacks.

However, to which extent different components can affect the power dissipation of the device is unknown until tested. As we illustrated in Figure 2b, even the execution of cryptographic components can produce a noticeable impact on the statistics of the simulation. We conducted an experiment to illustrate the issue, see Figure 5.

In this scenario we simply executed the `optee_example_aes` with a basic "test message" input once or twice, repeatedly. Each instance of the example performed encryption and decryption requests to the AES TA to encrypt the data and verify if the computation was correct. As can be seen from Figure 5a, the operations performed by the AES TA have a significant impact on the statistics reported by the simulator, even with a low sampling frequency. The activation pattern is reflected even in simple power models like the one included in Figure 5b. So, we expect that using the OP-TEE trusted applications as power wasters would have a similar effect on a physical device.



(a) *gem5* statistics (b) estimation of dynamic power

Figure 5: Statistics obtained from the execution of `optee_example_aes` with a sampling period of $1E-5$. The vertical axis shows the normalized amplitude since all the statistics have dynamic ranges which differ significantly; we simply normalize each trace for display purposes.

It is interesting to note that for this experiment we didn't request any frequency changes to the kernel. The frequency variation seen in Figure 5 is the result of the decisions taken by the `schedutil` governor of Linux. But despite this interference the pattern produced to encode a secret message is clearly distinguishable and could be easily decoded with the application of some filters and basic heuristics.

4. Conclusions

In this paper we have investigated the feasibility of using the *gem5* simulator to emulate SCAs on microprocessor systems. These devices are commonly found in a wide range of platforms which makes them an interesting case study. In recent years, their security resilience against physical attacks has been scrutinized as this kind of vulnerabilities has shown prowess for compromising the security of such systems. It has been proven that SCAs need more study with the emergence of novel attack models like remote power analysis and covert channels.

The use of a simulator would allow us to solve important challenges found in the study of power attacks. It would allow the security auditors to review a large combination of computing platforms, architectures, and security algorithms. All without the need to invest time and resources in the creation of prototypes and the implementation of the solutions. *gem5* has the potential to fill this niche as a simulator which allows to study multiple aspects of the platform. It can be used to derive precise statistics about the operation of the core and then combine them to obtain more complex estimators for the system. However, some challenges must be overcome to improve the usability of this platform:

- it is necessary to improve the production of statistics to make their reporting more concise
- it is necessary to implement mechanisms for enabling or disabling statistics with fine granularity

While the current distribution of the simulator includes some rudimentary support to achieve these goals, this code could very well be considered legacy. Its use is obscure, not documented, and requires us to manually edit core scripts of the simulator. This is followed by a recompilation of the software, which is a lengthy process. Despite these issues, we investigated the possibility of emulating power SCAs using *gem5*.

In relation to power analysis (correlation power analysis, simple power analysis) we showed a relationship between the algorithms and the simulation statistics. This allowed us to find certain statistics of interest. However, upon further testing, we could not verify that they held a statistical relationship with the secret key of the algorithms under study. The main challenge for conducting these experiments was the simulation time and the data volume. As a possible solution we explored the idea of decreasing the operational frequency of the virtual platform, but we found out that *gem5* has problems emulating low-frequency systems. Alternatively, if it was possible to select which statistics are produced it would not be necessary to decrease the target frequency. Reducing the number of statistics reported would also contribute to reducing the simulation time. On the other hand, our study on power-based covert channels clearly showed the potential of the simulator for studying architectural and software vulnerabilities.

Data availability

The multiple sets of statistics used in our experiments as well as the scripts created for processing the data can be accessed freely on https://github.com/CarlosAndresLARA/power-gem5.

Acknowledgements

This work has been supported by the French government through the *Agence Nationale de la Recherche* under project ARCHISEC (ANR-19-CE39-0008) and in the framework of the *France 2030* initiative under project ARSENE (ANR-22-PECY-0004).

References

[1] D. J. Bernstein, "Understanding brute force." ECRYPT STVL Workshop on Symmetric Key Encryption, Apr 2005.

[2] R. Krulwich, "Which Is Greater, The Number Of Sand Grains On Earth Or Stars In The Sky?." NPR, Sept 2012. [Online] https://www.npr.org/sections/krulwich/2012/09/17/161096233.

[3] M. Bellare and P. Rogaway, "Introduction to Modern Cryptography." University of California, May 2005. [Online] https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf.

[4] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *19th Annual International Cryptology Conference (CRYPTO)*, pp. 388–397, Springer, Aug 1999.

[5] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Advances in information security, Springer, 2008.

[6] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM Side—Channel(s)," in *4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 29–45, Springer, Aug 2003.

[7] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice Hall, 1996.

[8] M. Zhao and G. E. Suh, "FPGA-Based Remote Power Side-Channel Attacks," in *2018 IEEE Symposium on Security and Privacy*, pp. 229–244, IEEE, May 2018.

[9] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An Inside Job: Remote Power Analysis Attacks on FPGAs," *IEEE Design & Test*, vol. 38, pp. 58–66, Mar 2021.

[10] J. Gravellier, J.-M. Dutertre, Y. Teglia, and P. Loubet-Moundi, "High-Speed Ring Oscillator based Sensors for Remote Side-Channel Attacks on FPGAs," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–8, IEEE, Dec 2019.

[11] J. Gravellier, J.-M. Dutertre, Y. Teglia, and P. L. Moundi, "Side-Line: How Delay-Lines (May) Leak Secrets from Your SoC," in *2021 International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 3–30, Springer, Oct 2021.

[12] P. Chanak and I. Banerjee, "Internet-of-Things-Enabled SmartVillages: An Overview," *IEEE Consumer Electronics Magazine*, vol. 10, pp. 12–18, Jul 2021.

[13] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pp. 142–159, Springer, May 2013.

[14] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert, "Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note," in *18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pp. 740–757, Springer, Dec 2012.

[15] A. G. Bayrak, F. Regazzoni, P. Brisk, F.-X. Standaert, and P. Ienne, "A First Step towards Automatic Application of Power Analysis Countermeasures," in *48th Design Automation Conference*, pp. 230—235, ACM, Jun 2011.

[16] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The Gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, pp. 1—-7, Aug 2011.

[17] L. France, F. Bruguier, M. Mushtaq, D. Novo, and P. Benoit, "Implementing Rowhammer Memory Corruption in the gem5 Simulator," in *2021 IEEE International Workshop on Rapid System Prototyping*, pp. 36–42, Oct 2021.

[18] P. Ayoub and C. Maurice, "Reproducing Spectre Attack with Gem5: How To Do It Right?," in *14th European Workshop on Systems Security*, pp. 15—20, Association for Computing Machinery, Apr 2021.

[19] Q. Forcioli, J.-L. Danger, C. Maurice, L. Bossuet, F. Bruguier, M. Mushtaq, D. Novo, L. France, P. Benoit, S. Guilley, and T. Perianin, "Virtual Platform to Analyze the Security of a System on Chip at Microarchitectural Level," in *5th Workshop on Security of Software/Hardware Interfaces*, pp. 96–102, IEEE, Sept 2021.

[20] T. Alves and D. Felton, "Trustzone: Integrated hardware and software security," *Information Quarterly*, vol. 3, no. 4, pp. 18–24, 2004.

[21] J. Bech, "How to create and run Trusted Applications on OP-TEE," Jan 2014. [Online] https://www.slideshare.net/linaroorg/lcu14103-how-to-create-and-run-trusted-applications-on-optee.

[22] M. Arsath K F, V. Ganesan, R. Bodduna, and C. Rebeiro, "PARAM: A Microprocessor Hardened for Power Side-Channel Attack Resistance," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 23–34, Dec 2020.

[23] Y. Yao, T. Kathuria, B. Ege, and P. Schaumont, "Architecture Correlation Analysis (ACA): Identifying the Source of Side-channel Leakage at Gate-level," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 188–196, Dec 2020.

[24] D. Šijačić, J. Balasch, B. Yang, S. Ghosh, and I. Verbauwhede, "Towards efficient and automated side-channel evaluations at design time," *Journal of Cryptographic Engineering*, vol. 10, pp. 305–319, Jun 2020.

[25] M. He, J. Park, A. Nahiyan, A. Vassilev, Y. Jin, and M. Tehranipoor, "RTL-PSC: Automated Power Side-Channel Leakage Assessment at Register-Transfer Level," in *IEEE 37th VLSI Test Symposium*, pp. 1–6, Jul 2019.

[26] A. Nahiyan, J. Park, M. He, Y. Iskander, F. Farahmandi, D. Forte, and M. Tehranipoor, "SCRIPT: A CAD Framework for Power Side-Channel Vulnerability Assessment Using Information Flow Tracking and Pattern Generation," *ACM Transactions on Design Automation of Electronic Systems*, vol. 25, May 2020.

[27] V. Samadi Bokharaie and A. Jahanian, "Power side-channel leakage assessment and locating the exact sources of leakage at the early stages of ASIC design process," *The Journal of Supercomputing*, pp. 1–26, Jun 2022.

[28] S. A. Huss, M. Stöttinger, and M. Zohner, "Amasive: an adaptable and modular autonomous side-channel vulnerability evaluation framework," *Number Theory and Cryptography*, pp. 151–165, 2013.

[29] P. Slpsk, P. K. Vairam, C. Rebeiro, and V. Kamakoti, "Karna: A Gate-Sizing based Security Aware EDA Flow for Improved Power Side-Channel Attack Protection," in *2019 IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, Nov 2019.

[30] Y. Le Corre, J. Großschädl, and D. Dinu, "Micro-architectural power simulator for leakage assessment of cryptographic software on ARM Cortex-M3 processors," in *9th International Workshop Constructive Side-Channel Analysis and Secure Design (COSADE)*, pp. 82–98, Springer, Apr 2018.

[31] B. Gigerl, V. Hadzic, R. Primas, S. Mangard, and R. Bloem, "Coco: Co-Design and Co-Verification of Masked Software Implementations on CPUs," in *30th USENIX Security Symposium*, pp. 1469–1468, USENIX Association, Aug 2021.

[32] D. Walters, A. Hagen, and E. Kedaigle, "SLEAK: A side-channel leakage evaluator and analysis kit," Tech. Rep. AD1107774, MITRE CORP BEDFORD MA, Jan 2014.

[33] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible DRAM simulator," *IEEE Computer Architecture Letters*, vol. 15, pp. 45–49, Mar 2015.

[34] A. Herrera, "Running Trusted Firmware-A on gem5." arm Community, Jun 2020. [Online] https://community.arm.com/arm-research/b/articles/posts/running-trusted-firmware-a-on-gem5.

[35] T. E. Hansen, "ARM Power Modelling." gem5 Documentation, Nov 2022. [Online] https://www.gem5.org/documentation/learning_gem5/part2/arm_power_modelling/.

[36] NIST, "FIPS 197: Advanced Encryption Standard (AES)," Standard, National Institute of Standards and Technology, Nov 2001. [Online] https://csrc.nist.gov/publications/detail/fips/197/final.

[37] T. Schneider and A. Moradi, "Leakage Assessment Methodology," in *17th International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 495–513, Springer, Sept 2015.

[38] NIST, "FIPS 186-5: Digital Signature Standard (DSS)," Standard, National Institute of Standards and Technology, Feb 2023. [Online] https://csrc.nist.gov/publications/detail/fips/186/5/final.

[39] D. R. E. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-Based Covert Channels Using FPGAs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 26, pp. 1–25, Jun 2021.

A. Correlation analysis for the statistics

We performed a correlation analysis for a reduced set of statistics (sorted alphabetically). From the autocorrelation shown in Figure A.1 we could figure out (by looking outside the main diagonal) that there seem to be some statistics with high similarity (*SimdAdd, SimdAlu, SimdCmp*) which may carry redundant information.

From the cross-correlation shown in Figure A.2 we can figure out (also by looking at the main diagonal) that there are statistics with low similarity (*FloatMisc, No_OpClass*) which may be influenced by data dependency. In the cross-correlation matrix we can observe a slight asymmetry resulting from the analysis of equivalent pairs produced by taking one statistic from each set (e.g. $\mathit{corr}(\texttt{set}_A\texttt{.stat}_A,\ \texttt{set}_B\texttt{.stat}_B)$ vs $\mathit{corr}(\texttt{set}_B\texttt{.stat}_A,\ \texttt{set}_A\texttt{.stat}_B)$).

It is also interesting to note how the *clock* statistic (which carries the *alignment* information) has a high cross-correlation value, which lends validity to the proposed method for segmenting the traces.

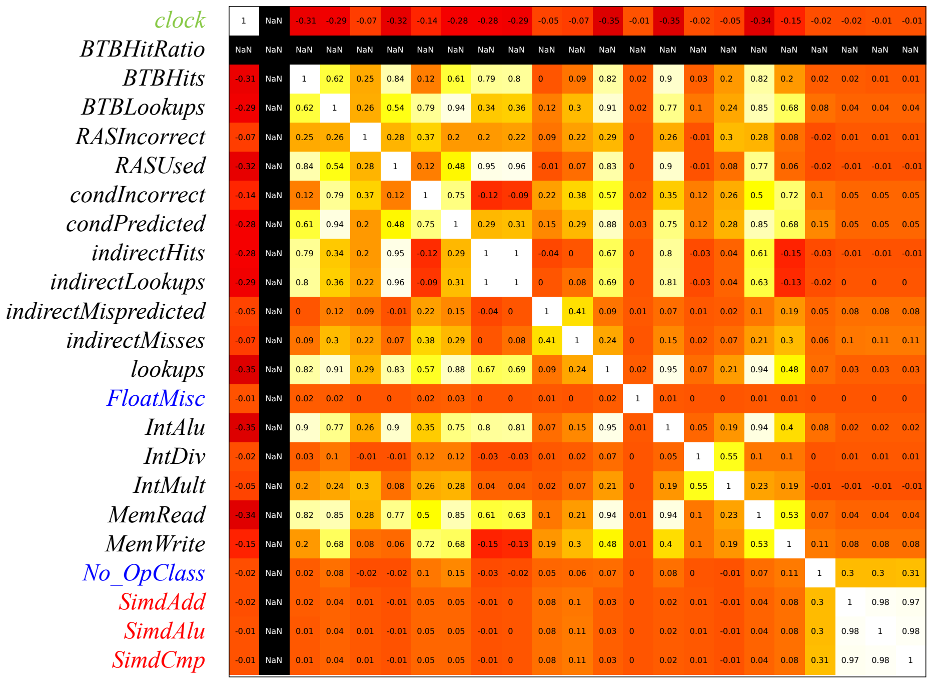


Figure A.1: Correlation within the set of statistics generated from the usage of the key `0x2B`.

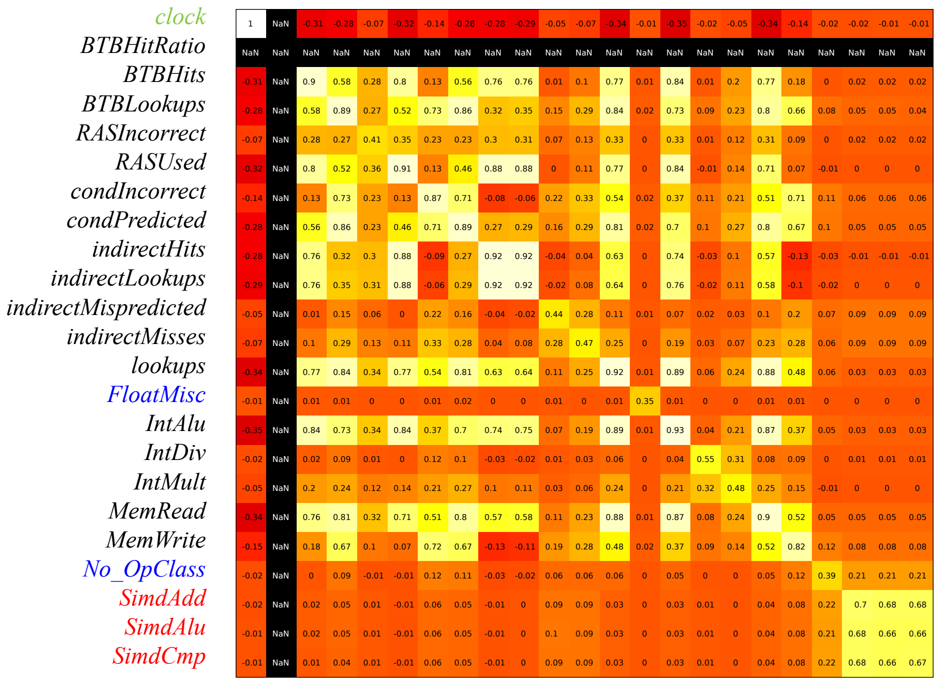


Figure A.2: Correlation between two sets of statistics generated from the usage of the keys `0x2B` and `0x7E`.