# Electromagnetic Side Channel Information Leakage Created by Execution of Series of Instructions in a Computer Processor

Baki Berkay Yilmaz *Student Member, IEEE*
Milos Prvulovic *Senior Member, IEEE*, and Alenka Zajić *Senior Member, IEEE*

***Abstract*— The side-channel leakage is a consequence of program execution in a computer processor, and understanding relationship between code execution and information leakage is a necessary step in estimating information leakage and its capacity limits. This paper proposes a methodology to relate program execution to electromagnetic side-channel emanations, and estimates side-channel information capacity created by execution of series of instructions (e.g. a function, a procedure, or a program) in a processor. To model dependence among program instructions in a code, we propose to use Markov Source model, which includes the dependencies among sequence of instructions as well as dependencies among instructions as they pass through a pipeline of the processor. The emitted EM signals during instruction executions are natural choice for the inputs into the model. To obtain the channel inputs for the proposed model, we derive a mathematical relationship between the emanated instruction signal power (ESP) and total emanated signal power while running a program. Then, we derive leakage capacity of electromagnetic (EM) side channels created by execution of series of instructions in a processor. Finally, we provide experimental results to demonstrate that leakages could be severe and that a dedicated attacker could obtain important information.**

***Index Terms*— electromagnetic emanation security, electromagnetic information leakage, information security, security of modern processors, TEMPEST, side-channel attack, covert-channel attack, channel capacity.**

## I. Introduction

Vulnerabilities caused by side channels have gained more attention recently because attackers are getting more sophisticated and can exploit these channels to steal important information such as cryptokey [1], [2], password [3], or even key strokes on a laptop [4]. In the literature, many types of side/covert channel attacks are investigated. Some examples of these attacks could be related to power variation [5], [6], [7], [8], [9], [10], [11], [12], [13], temperature analysis [14], [15], cache-based analysis [16], [17], [18], etc. Detection probability of these types of attacks is pretty high because all these attacks require some degree of direct access to the victims' systems. On the other hand, attacks based on emanated EM signals only require close proximity, i.e. attacks based on power delivery and computational circuitry of a device [19], [20], [21], [22]. Therefore, detection of EM based side channel attacks is harder, which makes these attacks more serious side channel attacks.

Side-channel signals are generated as a side effect of performing legitimate program activity on a computer system. Since program activity and the resulting hardware activity are dependent on data processed by the program, the resulting side channel signals can (and usually do) carry information about those data values.

Often asked question is how serious is this type of information leakage. Millen was the first to establish a connection between Shannon's information theory and information flow models in computer systems [23], and calculated the capacity of such a covert channel. However, that model assumes a synchronous channel, which is not a realistic assumption for side-channels. In contrast to most communication systems, the side channel is not designed to transfer information at all, and its transmission is often corrupted by insertion, deletion and erroneous transfer of bits. While there is a large number of papers discussing bounds on the capacity of channels corrupted with synchronization errors [24], [25], [26], [27], [28], [29], [30], bounds on the capacity of channels corrupted with synchronization and substitution errors [31], [32], [33], or bounds on the capacity when codewords have variable length but no errors in the channel [31], [34], none of them provides the answer to how much information is "transmitted" by execution of particular sequence of instructions that do not have equal timing and are transmitted through erroneous channel. The first attempts to answer this question were presented in [35], [36], where covert channels are generated, and upper and lower leakage capacities were derived. In [37], a side-channel leakage capacity is derived for a discrete memoryless channel where it was assumed that each transmitted quantum of information (i.e. instruction in the code) is mutually independent but do not have equal length. Although all these papers make an important step toward assessing information leakage from side-channels, they fall short of considering the relationship among sequence of instructions, which is a result

This work has been supported, in part, by NSF grant 1563991 and DARPA LADS contract FA8650-16-C-7620. The views and findings in this paper are those of the authors and do not necessarily reflect the views of NSF and DARPA.

Baki Berkay Yilmaz and Alenka Zajić are with the School of Electrical and Computer Engineering, while Milos Prvulovic is with the School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA.

of program functionality as well as a processor pipeline depth, which impacts how much signal energy will be emanated.

To address this problem, this paper derives side-channel information capacity created by execution of series of instructions (e.g. a function, a procedure, or a program) in a processor. To model dependence among program instructions in a code, we propose to use Markov Source model, which includes the dependencies that exist in instruction sequence since each program code is written systematically to perform a specific task. The sources for channel inputs are considered as the emitted EM signals during instruction executions. To obtain the channel inputs for the proposed model, we derive a mathematical relationship between the emanated instruction signal power (ESP) as it passes through processor pipeline and total emanated signal power while running a program. This is in contrast to work in [37] where all energy emanated through side-channels is assigned to an instruction, without taking into account effect of processor pipeline depth, which significantly impacts the emanated signal. Finally, we provide experimental results to demonstrate that leakages could be severe and that a dedicated attacker could obtain important information.

The proposed framework considers processors as the transmitters of a communication system with multiple antennas. The antennas correspond to different pipeline stages of any processor. Moreover, inputs of the transmitter show dependency based on a Markov model which reflects the practicality of a program. Therefore, the goal in this paper is to obtain the channel capacity of a communication system, or the severity of the side channels.

The rest of the paper is organized as follows: Section II reviews capacity of Markov Sources over noisy channels, defines the proposed leakage capacity, and introduces the Markov Source model. Section III derives a mathematical relationship between the emanated instruction power (ESP) as it passes through processor pipeline and total emanated signal power while running a program. Section IV provides experimental results and leakage capacities for various devices. Finally, Section V provides a recipe for the leakage capacity calculation, and Section VI concludes the paper.

## II. Modeling Information Leakage from a Computer Program as a Markov Source Over a Noisy Channel

In this section, we propose a Markov source model whose states are series of instructions in a pipeline. We assume that channel inputs at each state are the emanated signal powers produced as combination of different instructions in a pipeline, and the channel outputs are the noise corrupted versions of the emitted signals. The reason for considering such a Markov model is that individual instructions are not independent from each other in the code as well as that ordering of instructions as they pass through pipeline significantly impacts emitted signal patterns.

### *A. Brief Overview of Markov Model Capacity over Noisy Channels*

Channel capacity provides the limit for a reliable information transmission in a communication system. Assuming $Y_1^n$ and $S_1^n$ represent the channel output and state sequences between $t = 1$ to $t = n$, the capacity of the Markov sources over noisy channels is defined as [38]

$$C = \max_{\substack{P_{ij} \\ (i,j)\in\mathcal{T}}} \lim_{n\to\infty} \frac{1}{n} I\left(S_1^n; Y_1^n | S_0\right) \tag{1}$$

where $I(\bullet)$ is the mutual information, $P_{ij}$ is the transition probability from state $i$ to $j$, and $\mathcal{T}$ is a set of valid state transitions. To maximize the overall mutual information between input and output sequences, we need to find the probability distribution of state transitions under the constraint that state transitions are only possible if $\mathcal{T}$ contains these paths. The equation given in (1) can be simplified further by using the chain rule, Markov, and stationary properties of the model. In [38], it is shown that the capacity can be simplified as

$$C = \max_{P_{ij}} \sum_{i,j:(i,j)\in\mathcal{T}} \mu_i P_{ij} \left[\log \frac{1}{P_{ij}} + T_{ij}\right]. \tag{2}$$

where

$$T_{ij} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \left[\log \frac{P_t(i,j|Y_1^n)^{\frac{P_t(i,j|Y_1^n)}{\mu_i P_{ij}}}}{P_t(i|Y_1^n)^{\frac{P_t(i|Y_1^n)}{\mu_i}}}\right], \tag{3}$$

and where $\mu_i$ is the stationary probability of state $i$, which satisfies $\mu_i = \sum\limits_{k\in\mathcal{S}} \mu_k P_{ki}, \forall i \in \mathcal{S}$, and $\mathcal{S}$ is the set of states. In this equation, $P_t(i|Y_1^n)$ is the probability that the state at time $t-1$ is $i$, and $P_t(i,j|Y_1^n)$ is the probability that the states at times $t-1$ and $t$ are $i$ and $j$ respectively, given the received sequence, $Y_1^n$.

There is no closed form solution to the optimization problem given in (2) because the calculation of $T_{ij}$ is still an open problem. However, in [38], a greedy algorithm to calculate $C$ is introduced. Although, the algorithm could not produce the exact results, the experimental findings show that the performance gap between the actual results and the algorithm's results is small.

In the following sections, we introduce our Markov Source model, obtain the channel inputs for the proposed model, and modify the expectation-maximization algorithm given in [38] to quantify the side-channel information leakage.

### *B. Proposed Markov Source Model for Modeling Information Leakage from a Sequence of Instructions*

Here, we describe a Markov source model that characterizes relationship among sequence of instructions as they pass through pipeline stages in a processor. Note that a processor pipeline is an assembly line for computing, and contains groups of activities related to computational tasks, i.e. fetching, decoding, executing, etc. [39]. We assume that channel inputs at each state are the emanated signal powers obtained as a combination of different power levels that instructions experience as passing through a pipeline, and the channel outputs are the noise corrupted versions of the emitted signals. To include the effect of pipeline depth, states are assumed to be all possible instruction combinations because each stage performs an operation on the instruction in the queue. For example, if a pipeline has a depth of $m$, and the cardinality of $\mathcal{S}$ is $q$, the number of states will be $q^m$.

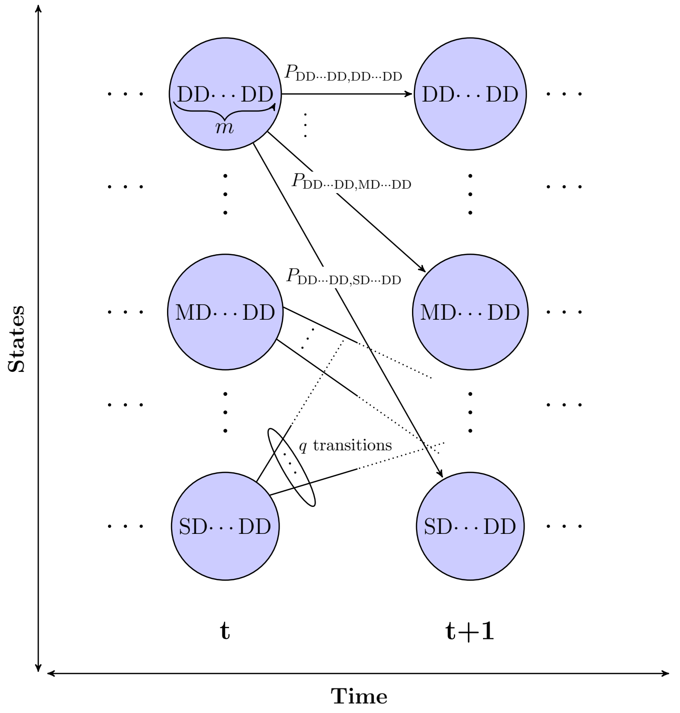
DD···DD
m
$P_{\text{DD}\cdots\text{DD},\text{DD}\cdots\text{DD}}$
DD···DD
$P_{\text{DD}\cdots\text{DD},\text{MD}\cdots\text{DD}}$
$P_{\text{DD}\cdots\text{DD},\text{SD}\cdots\text{DD}}$
MD···DD
MD···DD
States
$q$ transitions
SD···DD
SD···DD
t
t+1
Time

Fig. 1. Markov Source Model for the instruction execution when the pipeline depth is $m$, and the cardinality of the considered instruction set is three.

To illustrate how the proposed Markov Model works, Fig. 1 shows an example of Markov Source Model for the instruction execution when the pipeline depth is $m$, and the cardinality of the considered instruction set is three. In the figure, $P_{i,j}$ represents the state transition probability from state $i$ to state $j$, and circles denote the states of the model. The instruction set used in the example is {D, S, M}, which corresponds to division, subtraction, and multiplication, respectively. We utilize trellis diagram to explain the model explicitly although transitions are time invariant, i.e. $P_{i,j}$ does not vary in time. Moreover, the labels of the states are chosen as the combination of letters representing the instructions in the pipeline. Considering these three instructions, one of the states can be labeled as "DD$\mathbf{I}_S$DD" where $\mathbf{I}_S$ is a sequence of instructions whose length is $m-4$. Interpretation of the state corresponding to the label is that instructions in the $1^{th}$, $2^{nd}$, ..., $m-1^{th}$ and $m^{th}$ stages of the pipeline are D, D, ..., D, and D, respectively.

For each state, the number of possible paths is $q$, i.e. it is equal to the number of instructions in the set. For example, for the considered example, there exist only three paths from each state since the instruction set contains only three elements. For example, the possible states after "DD$\mathbf{I}_S$DD" could be "DDD$\mathbf{I}_S$D", "MDD$\mathbf{I}_S$D" or "SDD$\mathbf{I}_S$D". Furthermore, we assume that any instruction can be followed by any other instruction. This assumption helps the proposed model to be an indecomposable channel, therefore, the mutual information definition given in (2) is applicable to the proposed scheme.

We need to note that by considering the Markov source model, we can successfully capture the pipeline effect because it puts constraints on the state transitions. Moreover, $P_{i,j}$ explains the frequency of the instruction order encountered in the program. Therefore, the capacity of the proposed model provides the worst instruction sequence distribution which leaks information the most.

### C. Introducing Information Leakage Capacity for the Proposed Markov Source Model

The capacity definition given in (2) is well suited for Markov source models if the states take the same amount of time. In other words, the definition is valid for the models where the transitions last equal amount of time, and the transition time is not dependent on a given state. Unfortunately, applying the same capacity definition to the proposed scheme is not appropriate because different instructions take different time to execute. Therefore, we need a capacity definition which also accounts for instruction execution times. Hence, we propose a method to quantify the information leakage, which considers both execution time of each state and the mutual information between input and output sequences.

**Definition** *Assuming varying execution time of instructions, maximum possible information leakage through a processor is defined as*

$$C = \max_{\substack{P_{ij} \\ (i,j)\in\mathcal{T}}} \lim_{n\to\infty} \frac{I\left(S_1^n; Y_1^n | S_0\right)}{\sum_{i=1}^{n} \mathbf{L}(i)} \tag{4}$$

*where* $\mathbf{L}(i)$ *is the length of the state executed at the* $i^{th}$ *transition.*

Following the analogy between equations (1) and (2), we can rearrange the equation in (4) as follows

$$\lim_{n\to\infty} \frac{I\left(S_1^n; Y_1^n | S_0\right)}{\sum_{i=1}^{n} \mathbf{L}(i)} = \frac{\lim_{n\to\infty} \frac{1}{n} I\left(S_1^n; Y_1^n | S_0\right)}{\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{L}(i)} \tag{5}$$

$$= \frac{\sum_{i,j:(i,j)\in\mathcal{T}} \mu_i P_{ij} \left[\log \frac{1}{P_{ij}} + T_{ij}\right]}{\sum_{i\in\mathcal{S}} \mu_i L_i} \tag{6}$$

where $\mathcal{S}$ is the set containing all existing states, i.e. all instruction combinations, and $L_i$ is the execution length of the state $i$. Therefore, our definition can also be written as

$$C = \max_{\substack{P_{ij} \\ (i,j)\in\mathcal{T}}} \frac{\sum_{i,j:(i,j)\in\mathcal{T}} \mu_i P_{ij} \left[\log \frac{1}{P_{ij}} + T_{ij}\right]}{\sum_{i\in\mathcal{S}} \mu_i L_i}. \tag{7}$$

The result of this optimization provides the possible information leakage in bits per smallest number of clock cycles required to execute a state in $\mathcal{S}$ (which we call Bits/Quantum), not bits per second. The reason is that each instruction takes at least one clock cycle for any device, but clock frequencies can vary from one device to another. Since the goal is to analyze the leakage capacity on instruction level, we provide our results in Bits/Quantum. Please note here that even the leakage capacity of a device is small, the number of bits, a device can transmit in a second, could be large. Therefore, while examining the vulnerability of any device against side channel attacks, combining the leakage capacity with clock frequency leads to the most accurate results.

### *D. Reducing the Size of the Markov Source Model*

The main problem of the proposed Markov source model is the number of possible states and transitions. As the depth of the pipeline and the number of considered instructions increase, the number of states increases exponentially. This increase causes the iterative algorithm given in [38] to be more complex. Choosing states as individual instructions will simplify the proposed scheme. For these states, the channel input signal is assigned as the emanated EM signal while executing the corresponding instruction through all pipeline stages. With this approach, the number of states increases linearly, not exponentially, as the number of instructions increases.

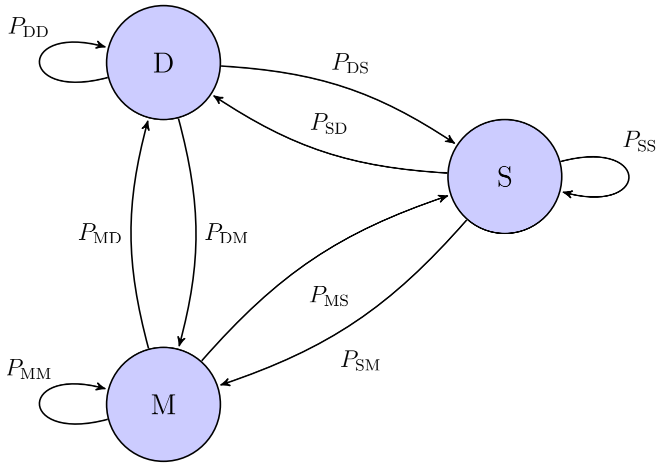

Fig. 2. Simplified version of Markov Source Model for the instruction execution when the cardinality of the considered instruction set is three.

In Fig. 2, we provide an example of the state diagram when the instruction set is $\{\text{D, M, S}\}$. This model is still indecomposable based on the assumption that each instruction can follow any other instruction. Therefore, the capacity definition given in (6) can be used to calculate leakage capacity limits. However, this definition also does not have a closed form solution, and an empirical algorithm similar to expectation-maximization (ExMa) algorithm in [38] is needed to solve the problem.

### *E. An Empirical Algorithm to Evaluate the Leakage Capacity*

To utilize the ExMa algorithm, we have to adjust the proposed model given in the previous section to remove the execution time of the instructions from the optimization problem. To achieve this goal, we propose to split the instructions into unit length sections, i.e., one clock cycle segments, and treat each of these segments as an individual state. To protect the overall framework and instruction sequence, we have to introduce some constraints for possible state transitions.

Let $K \in \mathcal{S}$ be a state whose length is $L_K > 1$. For the proposed model, we divide it into $L_K$ different states, where the states are named as $K_i$ where $i \in \{1, \cdots, L_K\}$. Each sub-state is called:

- *Initial state* if $i = 1$, i.e. $K_1$,
- *Exit state* if $i = L_K$, i.e. $K_{L_K}$,
- *Intra-state* if $i \in \{2, \cdots, L_K - 1\}$

of an instruction K. However, if the length of the instruction $K$ equals to one, we keep the instruction set unmodified. Note that the initial and exit states of $K$ will refer to full set $K$ for the scenario when $K$ takes only one clock cycle. Let $\mathcal{S}_M$ and $\mathcal{T}_M$ be the set of states and state transitions, respectively, after splitting the states to have a new instruction set whose members take same amount of time. Therefore, we can rewrite (6) as

$$C = \max_{\substack{\mathbb{P}_{ij} \\ (i,j)\in\mathcal{T}_M}} \sum_{(i,j)\in\mathcal{T}_M} \mathbb{u}_i \mathbb{P}_{ij} \left[\log \frac{1}{\mathbb{P}_{ij}} + \mathbb{T}_{ij}\right] \tag{8}$$

where $\mathbb{P}_{ij}$ refers the modified state transition probabilities, $\mathbb{u}_i$ is the stationary distribution of the new states, and $\mathbb{T}_{ij}$ is defined as in (3) in the new model.

Dividing the original states into substates is not enough to protect the duality between the optimization settings given in (6) and (8). We also have to make sure that the state transitions occur in a way that the instruction sequences for both settings follow the same path. For example, let $L_K$ be equal to 2. To ensure the duality, $\mathbb{P}_{K_1 j}$ must be nonzero only if $j$ is $K_2$. More formally, to guarantee the duality between the equations (6) and (8), we employ constraints on transitions which only allow state transitions in the following scenarios:

$\mathbf{R}_1$. An exit state of any instruction to an initial state of any instruction,

$\mathbf{R}_2$. $K_i$ to $K_{i+1}$ of instruction $K$ where $i \in \{1, \cdots, L_K - 1\}$.

Fig. 3 illustrates the proposed framework. This figure is a transformed version of the Markov source model given in Fig. 2 based on the rules imposed by $\mathbf{R}_1$ and $\mathbf{R}_2$. We assume that D and M take four and three times of the execution time of S, respectively. Here, $\text{M}_1$ and $\text{D}_1$ are the initial states, $\text{M}_3$ and $\text{D}_4$ are the exit states of M and D, respectively. $\text{D}_2$ and $\text{D}_3$ are the intra-states of DIV, and $\text{M}_2$ corresponds the intra-state of M. Note that these values are chosen arbitrarily, and only given as an illustration.

By applying the transformations introduced above, we have removed the problem of variable time of execution per instruction. The following theorem proves the models given in Section II-D and Section II-E are dual, and will lead to the same capacity results.

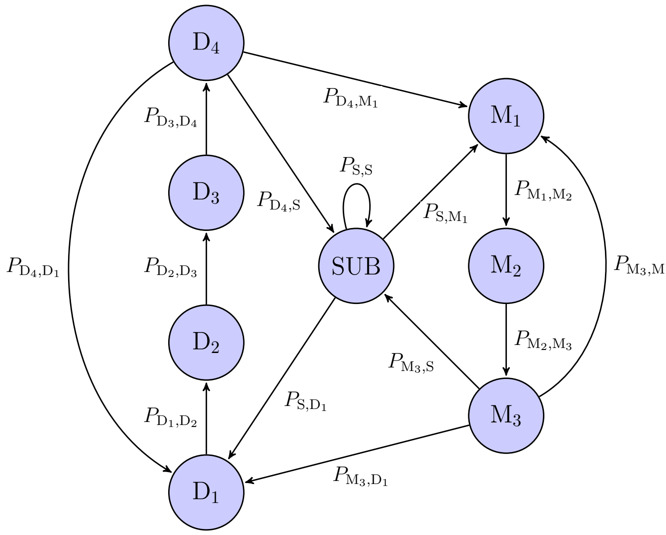

Fig. 3. Markov Model for the instruction execution as it goes through sub-states that take equal amount of time.

**Theorem 1 (Duality)** *The optimization settings given in (6) and (8) are dual problems if the constraints imposed by* $\mathbf{R}_1$ *and* $\mathbf{R}_2$ *are satisfied.*

*Proof:* Please see Appendix I. ■

Figure 3 illustrates that although we pose some constraints on the possible state transitions, the state transition diagram is still indecomposable. Therefore, the capacity definition and corresponding iterative algorithms given in [38] can be utilized. However, to apply the algorithm, the channel inputs have to be known. In the following section, we introduce a methodology to calculate the channel input power, i.e., emitted signal power while processing an instruction through the pipeline.

## III. Estimating Channel Input Power in the Proposed Markov Model

To obtain the channel inputs for the proposed model, in this section, we derive a mathematical relationship between the emanated instruction power as it passes through processor pipeline and total emanated signal power while running a program. This is in contrast to work in [37] where all energy emanated through side-channel is assigned to an instruction, without taking into account effect of processor pipeline, which significantly impacts the emanated signal. Another advantage of this approach to calculate emanated energy per instruction is that capacity can be directly related to signal to noise ratio (SNR).

### *A. Definition for Emanated Signal Power (ESP) of Individual Instructions as They Pass Through Pipeline*

In this section, we define **E**manated **S**ignal **P**ower (ESP) which is the channel input power for the proposed Markov source model.

For activity $\mathcal{A}_1$, let assume $\mathtt{T}_{\mathcal{A}_1}$ is the execution time, $\mathtt{T}^{\mathtt{P}}_{\mathcal{A}_1}$ is the total time spent in the pipeline except the execution stage, $a_{\mathcal{A}_1}(t)$ is the characteristic signal emanated only when $\mathcal{A}_1$ is executed, and $a^{P}_{\mathcal{A}_1}(t)$ is the signal emanated as a consequence of processing the activity throughout the pipeline excluding the execution stage. We define $\mathrm{ESP}(\mathcal{A}_1)$ as:

$$\mathrm{ESP}(\mathcal{A}_1) = \frac{\int_0^{\mathtt{T}^{\mathtt{P}}_{\mathcal{A}_1}} |a^{P}_{\mathcal{A}_1}(t)|^2 dt + \int_0^{\mathtt{T}_{\mathcal{A}_1}} |a_{\mathcal{A}_1}(t)|^2 dt}{R} \tag{9}$$

where we assume the activity $\mathcal{A}_1$ stays in the pipeline for the time interval $(0, \mathtt{T}_{\mathcal{A}_1} + \mathtt{T}^{\mathtt{P}}_{\mathcal{A}_1})$ only once, $R$ is the resistance of the measuring instrument, and the execution step is the last step of the pipeline. Here, we need to emphasize that $a_{\mathcal{A}_1}(t)$ and $a^{P}_{\mathcal{A}_1}(t)$ are desired signals emanated while processing activity $\mathcal{A}_1$ through the pipeline only. They do not contain any components from any other signals and interrupts ideally. We also need to note that although we assume that the execution of an instruction happens at the very end of the pipeline, it is only for better illustration of the equation given in (9), and the execution could be done at any stage of a pipeline. We need to note that ESP provides the mean available power while executing an instruction, therefore, we assume that the noise term comprises all variations in the emanated power.

Although ESP is defined in continuous time domain, we have to alter this equation to cope with discrete time analysis since measurements are done on digital devices. Let assume sampling frequency of the measuring instrument is $f_s = 1/T_s$. We also assume that the number of samples taken during the execution of the instruction $\mathcal{A}_1$ is $N_I = \mathtt{T}_{\mathcal{A}_1}/T_s$, and the number of samples taken, when the instruction $\mathcal{A}_1$ is processed in a pipeline except for execution stage, is $P_S = \mathtt{T}^{\mathtt{P}}_{\mathcal{A}_1}/T_s$. Then, ESP in discrete time can be written as

$$\mathrm{ESP}[\mathcal{A}_1] = \frac{\sum\limits_{m=0}^{P_S-1} |a^{P}_{\mathcal{A}_1}[m]|^2 + \sum\limits_{m=0}^{N_I-1} |a_{\mathcal{A}_1}[m]|^2}{R/T_s}. \tag{10}$$

### *B. Estimating ESP From The Total Emanated EM Signal Power Created by a Program*

Measuring ESP is not a trivial task. Execution of any instruction is overlapped with execution of other instruction in the code as well as other activities in the other stages of the pipeline. Therefore, we need a method to separate signal components that do not belong to the considered instruction from the desired signals related to a particular instruction. In [40], a program is designed to calculate the emanated energy difference between two instructions.

In this paper, we modify the work in [40] to evaluate energy emanated by a single instruction. For ease of explanation, we show the code from [40] in Fig. 4. The code has two inner for-loops such that the first for-loop repeats the execution of Activity A, and the second for-loop repeats the execution of Activity B. Work in [40] shows that given the activities in the inner for-loops are non-identical, a spectral component at the alternation frequency, $f_{\mathrm{alt}} = 1/T_{\mathrm{alt}}$, is generated where $T_{\mathrm{alt}}$ is the one period of outer for-loop.

Instead of inserting two different activities into for-loops of the code, we insert instruction under observation in the first

```
for(i=0;i<n_out;i++){
  // Do some instances of the A instruction
  for(i=0;i<n_inst;i++){
    ptr1=(ptr1&~mask1)|((ptr1+offset)&mask1);
    // The A-instruction, e.g. an add
    value+=ptr1;
  }
  // Do some instances of the B instruction
  for(i=0;i<n_inst;i++){
    ptr2=(ptr2&~mask2)|((ptr2+offset)&mask2);
    // The B-instruction, e.g. a multiplication
    value*=ptr2;
  }
}
```

Fig. 4. The A/B alternation pseudo-code in [40].

for-loop of the code, and NOP instruction into the second for-loop of the code. We note here that NOP instruction keeps the processor idle for one clock cycle. Hence, if the execution time of the activity in the first for-loop takes more than one clock cycle, the number of NOPs in the second for-loop has to be chosen carefully so that both loops take equal amount of time. In other words, the number of iterations of the first for-loop, n_inst, has to be equal to number of iterations of the second for-loop n_inst2=$n_{inst}$. Here, we assume the emitted signal power at all stages of a pipeline for NOP forms the baseline that we use to normalize the power consumption of other instructions relative to NOP. Therefore, for the mathematical tractability of the derivations given in Appendix II, we assume that the signal measured while execution of NOP is a consequence of additive Gaussian white noise.

After running the modified code in [40] and measuring the power at the alternation frequency, the next step is to derive the relationship between the total emitted power and ESP. Let $s(t)$ be the emanated signal when the outer loop iterates for one time. We assume that the frequency content of $s(t)$ is negligible for the frequencies above $f_s/2$, and lasts for $\mathtt{T_E}$ seconds. Therefore, the total number of samples taken during the experiment is equal to $N_T = \mathtt{T_E}/T_s$. Let $T_L$ be the execution time of any inner for-loop only for one period. Then, the number of samples taken in a period can be written as $N_L = T_L/T_s$. Therefore, the relationship between $N_T$ and $N_L$ becomes $N_T = 2 \times n_{inst} \times N_L$.

Now, let the power measured around this frequency be $\mathcal{P}_{\mathcal{A}_1}(f_{\text{alt}})$ while executing the code under the assumptions stated above. The following theorem gives the relationship between the total emanated signal power and the instruction power.

**Theorem 2 (ESP)** *Let $\boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}})$ be the normalized emanated power which is defined as*

$$\boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}}) = \mathcal{P}_{\mathcal{A}_1}(f_{\text{alt}}) - \mathcal{P}_{\text{NOP}}(f_{\text{alt}}) \tag{11}$$

*where $\mathcal{P}_{\text{NOP}}(f_{\text{alt}})$ is the measured emanated power when both for-loops of the code are employed with* NOP. *The mathematical relationship between* $\text{ESP}[\mathcal{A}_1]$ *and $\mathcal{P}_{\mathcal{A}_1}(f_{\text{alt}})$ while running the activity $\mathcal{A}_1$ in the first for-loop can be written as:*

$$\text{ESP}[\mathcal{A}_1] \quad = \quad \left(\frac{\pi}{2}\right)^2 \frac{\boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}}) \cdot N_L}{(N_I + P_S) \cdot f_{\text{alt}} \cdot n_{inst}}. \tag{12}$$

*Proof:* Please see Appendix II. ■

## IV. Experimental Results and Information Leakage Analysis

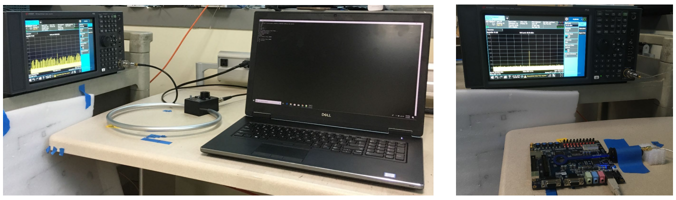

Fig. 5. Measurement setups used in the experiments.

In this section, we provide the experimental results for emanated signal power of each instruction, and evaluate leakage capacity of various computer platforms.

The experimental setup is shown in Fig. 5. We used a spectrum analyzer (Agilent MXA N9020A), and magnetic loop probe (AAronia H field probe PBS-H3) for FPGA board and a magnetic loop antenna (AOR LA400) for other devices. We performed our measurements by setting the alternation frequency, $f_{\text{alt}}$, to 80 kHz. We keep the distance as close as possible to the processor since our goal is to reveal the input powers of the transmitter, i.e. ESP. The activities used in this section correspond to x86 instructions given in Fig. 6.

| | Instruction | Description |
|---|---|---|
| LDM | mov eax,[esi] | Load from main memory |
| STM | mov [esi],0xFFFFFFFF | Store to main memory |
| LDL2 | mov eax,[esi] | Load from L2 cache |
| STL2 | mov [esi],0xFFFFFFFF | Store to L2 cache |
| LDL1 | mov eax,[esi] | Load from L1 cache |
| STL1 | mov [esi],0xFFFFFFFF | Store to L1 cache |
| ADD | add eax,173 | Add imm to reg |
| SUB | sub eax,173 | Sub imm from reg |
| MUL | imul eax,173 | Integer multiplication |
| DIV | idiv eax | Integer division |
| NOP | | No operation |

Fig. 6. x86 instructions for our setup.

To obtain the experimental results, the steps we follow are:

- Run the program given in Fig. 4 as described in Section III-B to measure the available total signal power at the alternation frequency.
- Calculate ESP of each instruction for all available devices based on the equation given in (12).
- Transform the Markov Chain of instructions, and define the new constraints for the new model in terms of allowable paths as in Section II-E.

- Define the signal to noise ratio (SNR) as:

$$\mathrm{SNR} = \frac{\sum\limits_{i\in\mathcal{S}} (\mathrm{ESP}[i])^2}{|\mathcal{S}| \times N_0/2} \tag{13}$$

  where $|\mathcal{S}|$ is the cardinality of instruction set $\mathcal{S}$.
- For a given SNR, run the algorithm given in [38] to obtain the stationary probabilities of each sub-state and corresponding leakage capacities.
- If the stationary probability of instructions is required, solve the following equations

$$\mu_i = \mathtt{L} \times \mathrm{u}_1^i, \quad \forall i \in \mathcal{S} \tag{14}$$

  where $\mu_i$ is the stationary probability of $i^{th}$ instruction for the original case, and $\mathrm{u}_1^i$ is the initial sub-state of $i^{th}$ instruction for the transformed scenario, and $\mathtt{L}$ is a constant which can be written as

$$\mathtt{L} = \left(\sum_{k\in\mathcal{S}} \mathrm{u}_1^k\right)^{-1}. \tag{15}$$

- Define "Quantum" as the ratio between number of required clock cycles to execute an instruction and minimum number of clock cycles to execute at least one instruction.

Please note that for some experiments, Quantum is equivalent to a clock cycle, but for some experiments, it can correspond to a couple of clock cycles. Additionally, abbreviations used in this section can be listed as follows:

- $\boldsymbol{C}_P$: Capacity in Bits/Quantum obtained with the proposed scheme.
- $\boldsymbol{C}_0$: Capacity in Bits/Instruction obtained by assuming execution time of all instruction takes only one clock cycle and using capacity definition given in (1). We also assume that the optimal stationary distribution for this capacity definition is denoted as $\boldsymbol{\mu}_0$.
- $\boldsymbol{C}_N$: Capacity in Bits/Quantum which is calculated as

$$\boldsymbol{C}_N = \frac{\boldsymbol{C}_0}{\sum\limits_{i\in\mathcal{S}} \boldsymbol{\mu}_0[i] L_i}. \tag{16}$$

  This capacity definition maps $\boldsymbol{C}_0$ into Bits/Quantum for a fair comparison.
- $\boldsymbol{C}_\infty$: Capacity in Bits/Quantum obtained by setting $\mathrm{SNR} = \infty$ and exploiting the proposed scheme to obtain the maximum possible leakage.

### A. Experimental Results and Leakage Capacity for FPGA

This section presents the experimental results and leakage capacity for NIOS Processor on DE1 FPGA board. The ESP and corresponding execution length of each instruction are provided in Table I. Please note that length of an instruction means total execution time of each instruction in terms of Quantum.

TABLE I

ESP values (in zJ) for DE1 FPGA board.

| | LDM | LDL1 | DIV | ADD | SUB | MUL |
|---|---|---|---|---|---|---|
| ESP | 139.38 | 69.98 | 87.60 | 0.32 | 6.10 | 55.14 |
| Length | 7 | 4 | 5 | 1 | 1 | 4 |

In Fig. 7, we plot the leakage capacity for FPGA as a function of SNR. We observe that $\boldsymbol{C}_0$ exceeds $\boldsymbol{C}_P$ because $\boldsymbol{C}_0$ considers that each instruction takes only one clock cycle. However, if we normalize $\boldsymbol{C}_0$ to obtain $\boldsymbol{C}_N$, we can observe that applying traditional Shannon theory underestimates available leakage capacity and that proposed leakage capacity estimation $\boldsymbol{C}_P$ is needed to establish relationship between sequence of instructions as they pass through pipeline and leakage capacity.

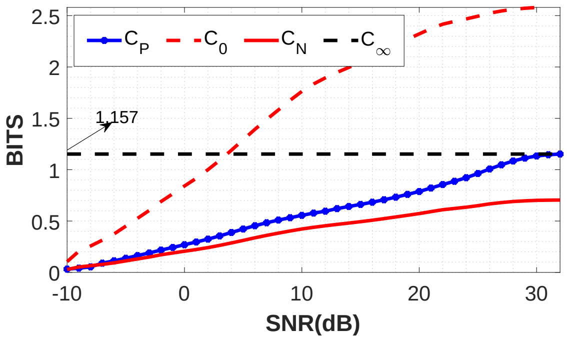

Fig. 7. Leakage Capacity for NIOS Processor on the DEI FPGA.

Additionally, we observe that leakage capacity for SNR = 59.96 dB in [37] is 1.14 Bits/Quantum. Please note that the method in [37] does not allow for capacity calculation as a function of SNR. On the other hand, with the proposed scheme, the estimated leakage capacity is higher and reaches 1.157 Bits/Quantum when SNR is around 30 dB. This result indicates that considering the pipeline depth and the dependence between instructions, which are not included in [37], more realistically estimates leakage capacity. We also note that the leakage capacity is high even for low SNR regimes allowing for transmission of thousands of bits per second because the clock frequencies of the current devices are high. Therefore, software and hardware designers need to consider side-channels and devise countermeasures to decrease side-channel leakages as much as possible.

### B. Experimental Results and Leakage Capacity for AMD Turion X2 Laptop

This section provides the leakage capacity for a laptop with AMD Turion X2. It has 64 KB 2 way L1 Cache and 1024 KB 16 way L2 Cache. ESP values and execution lengths are given in Table II.

TABLE II

ESP values (in zJ) for AMD Turion X2 Laptop.

| | LDL2 | LDM | STM | STL2 | STL1 | MUL | DIV |
|---|---|---|---|---|---|---|---|
| ESP | 150.08 | 84.66 | 64.74 | 188.17 | 0.49 | 0.21 | 7.26 |
| Length | 1 | 26 | 30 | 3 | 1 | 1 | 8 |

We need to note here that LDL1, ADD, and SUB are not included into our analysis because ESP values and execution lengths of these instructions are almost equal to STL1. Therefore, including these instructions does not affect overall leakage capacity. However, if we consider STL1, LDL1, ADD, and SUB as a sub-instruction set whose members are almost identical, STL1 could be thought as a representative of this set.

We observe that the deviation of the execution length of instructions is much larger compared to FPGA. The effect of having such a deviation can be seen from Fig. 8 where the gap between $C_0$ and $C_P$ is significantly larger. Additionally, the leakage capacity given in [37] is 0.97 Bits/Quantum when SNR is 23.78 dB, but the new proposed leakage capacity $C_P$ shows that the leakage can be up to 1.36 Bits/Quantum for the same SNR region. This result indicates that all signals emanated from all stages of a pipeline carry some information, therefore, ignoring these signals can cause underestimation of the leakages.

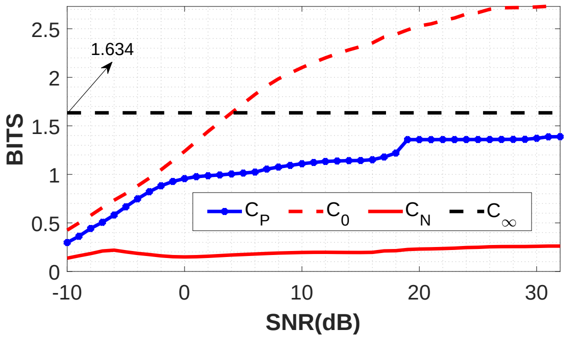

Fig. 8. Leakage Capacity for AMD Turion X2 Laptop.

The results also show that the capacity of the laptop is moderately high even for low SNR regimes. For example, we observe that the leakage capacity of this system is approximately 1 Bits/Quantum around 0 dB SNR. Unfortunately, if the attacker is in very close proximity, and has the ideal decoder to reveal the secret information, $C_P$ could raise up to 1.634 Bits/Quantum, which corresponds to 1.634*$10^9$ bits/second for a processor with 1 GHz processor clock and all instructions taking one clock cycle. We also observe that $C_P$ could not achieve the data rate of $C_\infty$ in the given SNR regime. For $C_P$ to achieve maximum rate, it requires about 57 dB SNR. However, for the consistency among figures, we keep the considered SNR regime same for each plot.

### C. Experimental Results and Leakage Capacity for Core 2 DUO Laptop

In this section, we provide the results for Core 2 DUO laptop. It has 1.8 GHz CPU clock, 32 KB 8 way L1 and 4096 KB 16 way L2 caches. ESP values and lengths of instructions are given in Table III. Similar to AMD laptop, the deviation of the instruction length is large, which causes the capacity gap between the proposed and Shannon based methods to be larger.

We do not consider the results for STL1, SUB and ADD because the lengths and ESP values of these instructions are almost same with LDL1. For this device, we assume that LDL1, STL1, SUB and ADD form a sub-instruction set, and LDL1 as the representative of this set. We observe that $C_P$ can be up to 1.634 Bits/Quantum if the attacker can find a way to capture emanated signals with high SNR. Furthermore, at 23.82 dB SNR, $C_P$ is 1.36 Bits/Quantum, again higher then 1.09 Bits/Quantum capacity predicted in [37]. The difference between these results reveals the importance of considering both pipeline depth and ordering of instructions.

TABLE III
ESP VALUES (IN zJ) FOR CORE 2 DUO LAPTOP.

| | STL2 | LDM | STM | LDL2 | LDL1 | MUL | DIV |
|---|---|---|---|---|---|---|---|
| **ESP** | 422.16 | 181.58 | 79.94 | 320.48 | 0.75 | 0.06 | 7.02 |
| **Length** | 1 | 26 | 31 | 3 | 1 | 1 | 8 |

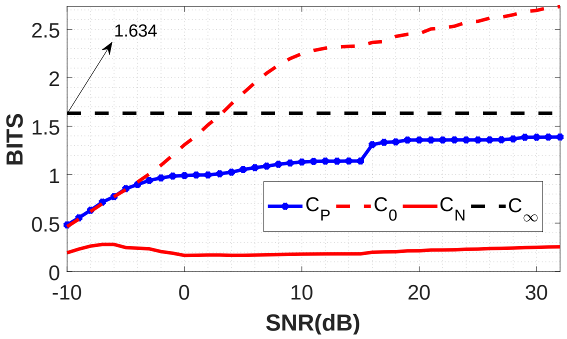

Fig. 9. Leakage Capacity for Core 2 DUO Laptop

We also observe that the required SNR for $C_P$ to achieve $C_\infty$ must be at least 56 dB. However, with a moderate gain antenna and proximity to the laptop, the attacker can steal sensitive information since the leakage capacity is 0.5 Bits/Quantum when SNR is around -10dB. Considering the clock frequency of the computer, the side channel can have a transmission rate of thousand of bits per second under ideal circumstances.

### D. Experimental Results and Leakage Capacity for Core I7 Laptop

The last example we provide is for Core I7 laptop which has 3.4 GHz CPU clock with 64 KB 2 way L1 Data and 1024 KB 16 way L2 caches. Table IV provides ESP and execution length of each instruction. The first observation here is that the deviation of the execution length of instructions is not as large as the other laptops, which causes the gap between $C_P$ and $C_0$ results to decrease as given in Fig. 10.

TABLE IV
ESP VALUES (IN aJ) FOR CORE I7 LAPTOP.

| | LDL2 | LDM | STM | STL2 | SUB | STL1 | ADD | MUL | DIV |
|---|---|---|---|---|---|---|---|---|---|
| **ESP** | 1.03 | 1.38 | 1.23 | 0.56 | 0.05 | 0.09 | 0.08 | 0.06 | 0.54 |
| **Length** | 1 | 12 | 15 | 4 | 1 | 1 | 1 | 1 | 8 |

We observe that LDL1 and SUB have approximately same ESP. Therefore, SUB is considered as the representative of the group of these instructions. For ideal scenarios, $C_P$

can go up to 2.32 Bits/Quantum. To achieve this rate, the setup must ensure at least 47 dB SNR. In addition, when SNR is 23.84 dB, the leakage capacity with the model in [37] is 0.72 Bits/Quantum, although it is obtained as 1.65 Bits/Quantum with the proposed model. Hence, including both pipeline depth and dependencies between instructions helps better quantification of leakage capacity.

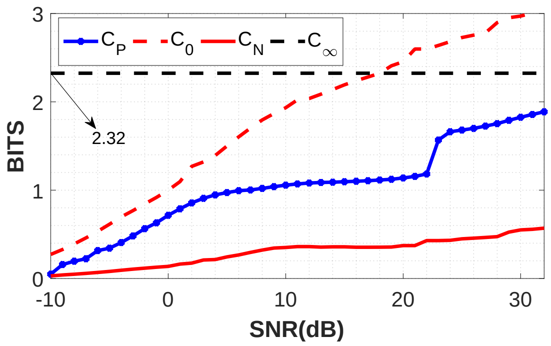

Fig. 10. Leakage Capacity for Core I7 Laptop

Also, for the low SNR scenarios, $C_P$ is high enough, i.e. 0.7 Bits/Quantum around 0 dB. Considering the clock frequency of the laptop, the capacity values given in Fig. 10 could be a messenger to warn any users about the possible vulnerabilities that computer systems might have.

Another evaluation methodology to assess the severity of side channels is given in [41]. They define success rate to demonstrate the performance of an adversary attack. It is possible to establish a connection between the success rate and the proposed information leakage. This connection is achieved if the probabilities which lead to maximum information leakages are utilized to calculate the success rate. As a simple example, if the goal of an attack is to reconstruct instructions (although our paper does not consider a specific attack, but aims to provide a universal upper bound for EM side channels), we can define the success rate as

$$\mathbf{Succ}^{sc-ir-1,\mathbf{I}}_{A_{\mathbf{I},\mathrm{ESP}}}(\mathbf{D},\sigma) = \sum_{i\in\mathbf{I}} \left[ \mu[i] \int_{d_{i_L}}^{d_{i_U}} f(x|\mathrm{ESP}(i),\sigma)dx \right] \tag{17}$$

where $\mathbf{D}$ is the set of decision boundaries for all instructions, $\mathbf{I}$ is the set containing all considered instructions, $d_{i_U}$ and $d_{i_L}$ are the upper and lower decision boundaries for the $i^{th}$ instruction, $f(x|\alpha,\sigma)$ is the pdf of white Gaussian noise distribution with mean $\alpha$ and standard deviation $\sigma$, $\mu[i]$ is the stationary probability of $i^{th}$ instruction that is the result of the optimization problem given in (7). The decision boundaries are calculated based on ESP values of neighboring instructions. Therefore, if the target of an attack is known, it is possible to provide success rate of an attack by exploiting the parameters which optimize (7).

Welch's T-test is an evaluation methodology which is heavily exploited in the security assessment of cryptographic implementations against side channel attacks [42], [43], [44]. The proposed framework can be also associated with T-test assessment methodology if we assume there exists an attack which can separate emitted signals of different pipeline stages, and depends on the emitted signal power of individual instructions when the same instruction is executed successively. If these assumptions hold, the attacker will receive samples which will be the noise added version of emitted signal power while performing activity $i$, i.e, $y^i_j = \mathrm{ESP(i)} + noise$, where $y^i_j$ denotes the $j^{th}$ successive execution of the instruction $i$. Let $\mathbf{y}_i$ be

$$\mathbf{y}_i = \left[ y^i_1 \; y^i_2 \; \cdots \; y^i_{N_i} \right]$$

where $N_i$ is the number of successive execution of the instruction $i$. Let also $\Delta_{m,n}$ be the T statistic of instruction $m$ and $n$, which is given as

$$\Delta_{m,n} = \frac{\mathbb{E}\left(\mathbf{y}_m\right) - \mathbb{E}\left(\mathbf{y}_n\right)}{\sqrt{\frac{\mathrm{var}(\mathbf{y}_m)}{N_m} + \frac{\mathrm{var}(\mathbf{y}_n)}{N_n}}} \tag{18}$$

where $\mathbb{E}(\bullet)$ is the expectation operation, and $var(\bullet)$ gives the variance of its input. The T statistic for the instruction $m$ will be significant only if $\Delta_{m,n}$ is above a threshold for any $n \in \mathbf{I}$. Therefore, the T statistic could be an empirical methodology, which can provide required repetition of an instruction for a successful side channel attack.

## V. Utilizing the Proposed Framework for Security Assessment

The leakage capacity definition given in this paper provides the maximum leakage amount that any EM side/covert channel can achieve on a given device. This capacity can help designers to predict possible vulnerabilities of their products at the design-stage and provide the opportunity to design countermeasures, or to redesign their systems to prevent possible side-channel attacks. Comparing with the evaluation method based on success rate, which quantifies accurate retrieval rate of an attack's target (i.e., secret key bit estimation), leakage capacity defines the maximum information leakage through side channels without specifying the attack itself. Therefore, it provides a universal upper bound for EM side channels. This section provides a recipe to check whether the considered system is secure enough against side channel attacks, and explains steps to justify why they are required. The procedure for the assessment is given in Fig. 11, and can be explained as follows:

- The first step is to collect emanated EM signal power available to an attacker while executing an instruction. Considering the clock frequency of modern computer systems, measuring the single instruction power could be problematic because of synchronization, complex pipeline structure, etc. To handle these problems, the designed microbenchmark given in Fig. 4 is run to obtain both $\mathcal{P}_{A_1}(f_{\mathrm{alt}})$ and $\mathcal{P}_{\mathrm{NOP}}(f_{\mathrm{alt}})$ where $\mathcal{P}_i(f_{\mathrm{alt}})$ is the total emanated signal power when instruction $i$ and NOP are inserted into the first and second inner-for-loops, respectively.
- The measurements to obtain $\mathcal{P}_i(f_{\mathrm{alt}})$ are done from near-field because the goal is to capture all emanated

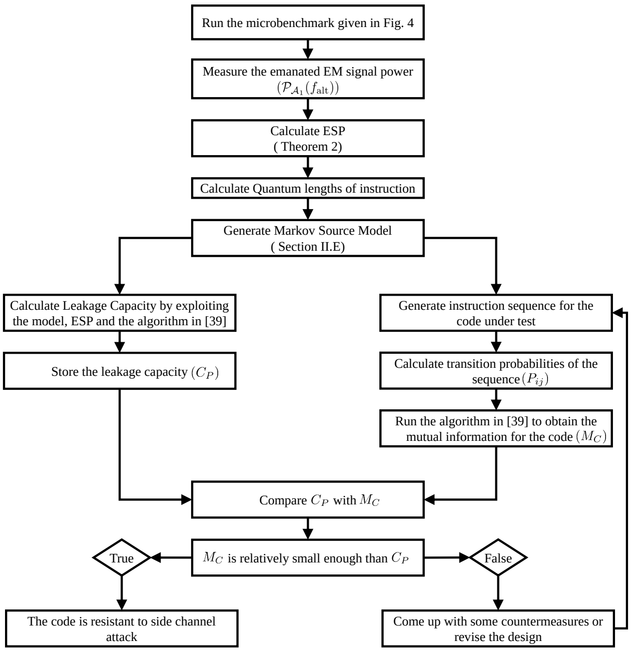
Run the microbenchmark given in Fig. 4
Measure the emanated EM signal power ($\mathcal{P}_{A_1}(f_{\text{alt}})$)
Calculate ESP ( Theorem 2)
Calculate Quantum lengths of instruction
Generate Markov Source Model ( Section II.E)
Calculate Leakage Capacity by exploiting the model, ESP and the algorithm in [39]
Store the leakage capacity ($C_P$)
Generate instruction sequence for the code under test
Calculate transition probabilities of the sequence($P_{ij}$)
Run the algorithm in [39] to obtain the mutual information for the code ($M_C$)
Compare $C_P$ with $M_C$
$M_C$ is relatively small enough than $C_P$
True
False
The code is resistant to side channel attack
Come up with some countermeasures or revise the design

Fig. 11. The methodology to assess information leakage.

signal as much as possible. This approach helps to have close empirical results for signal power because actual emanated instruction power is not available. Then, ESP of each considered instruction is calculated based on the formula given in (12).

- Because of the functionality of a program, a script, etc., and the complex pipeline structure of modern computer systems, instructions shows dependency to each other. To consider the dependency among instructions, a Markov Model is created as given in Section II.
- The next step is to calculate the limit for information leakage. To obtain the limit, the algorithm given in [38] will be exploited. For the algorithm, it is required that the channel inputs from each source have to last for the same amount of time. However, instructions can take different number of clock cycles to execute. Therefore, the Quantum length of each instruction has to be revealed. Please note that the Quantum length is defined as the ratio between actual execution time of an instruction and the minimum execution time within instruction set.
- After having the execution length and utilizing the duality given in Theorem 1, the next step is to apply the transformation in Section II-E. This transformation makes sure that each channel input takes same amount of time so that the algorithm given in [38] can be utilized to calculate the leakage capacity for a targeted SNR regime.
- The result of the algorithm provides the leakage capacity which is denoted as $C_P$. We use this number later as the baseline to compare with the leakages of designs to understand the relative resistance of them against any possible side channel attack.
- To find the leakage of any code, program, design, etc., the number of transitions from $i^{th}$ to $j^{th}$ instructions is counted. These numbers are normalized to calculate $P_{ij}$ for the inspected source code.
- Having the state transition probabilities, $P_{ij}$, our next goal is to reveal the available mutual information for the test code. Please note that our goal is to find the mutual information with the given $P_{ij}$, therefore, we do not need to update the state transition probabilities. Hence, we run the algorithm given in [38] only once without updating the state transition probabilities. The mutual information obtained as a result of the algorithm is denoted as $M_C$.
- As the last step, we compare $C_P$ with $M_C$. If $M_C$ is much smaller then $C_P$, and very close to zero, the designer can conclude that the source code is secure. Otherwise, a new design or some countermeasures, i.e. shielding, etc., has to be considered. Then, the same steps given in this section have to be followed again until achieving $M_C \ll C_P$.

Please note that the procedure given here does not specify the attack methodology, but provides the worst case scenario for a victim in terms of information leakage. It is still an ongoing research to have an attack that achieve the limits given in this paper. However, designers can utilize the procedure to prevent any future attacks.

## VI. Conclusions

This paper proposed a methodology to relate program execution to electromagnetic side-channel emanations and estimate side-channel information capacity created by execution of series of instructions (e.g. a function, a procedure, or a program) in a processor. To model dependence between program instructions in a code, we have proposed to use Markov Source model, which includes the dependencies that exist in instruction sequence since each program code is written systematically to perform a specific task. The sources for channel inputs are considered as the emitted EM signals during instruction executions. To obtain the channel inputs for the proposed model, we derive a mathematical relationship between the emanated instruction power (IP) and total emanated signal power while running a program. Then, we have derived leakage capacity of electromagnetic (EM) side channels created by execution of series of instructions in a processor. Finally, we have provided experimental results to demonstrate that leakages could be severe enough for a dedicated attacker to obtain some prominent information.

## Appendix I
## Establishing the Duality Between (6) and (8)

Transforming the optimization problem given in (6) to the problem given in (8) helps to utilize the ExMa algorithm presented in [38]. However, the necessary step for that is to show that the duality holds between (6) and (8).

Let $\mathbb{Y}_1^{\mathbf{n}_M}$ be the adjusted version of $Y_1^n$ for the transformation of the proposed model in Section II-D to the model in Section II-E where $\mathbf{n}_M$ is the number of states after dividing each $n$ state properly. We assume the leakage occurs at the exit state, and the rest of the states do not emit any signal for instructions which take more than one clock cycle. Actually, most accurate approach is to split the available leakage power to all sub-states. However, we note that for any intra/initial state, we can write $\mathbb{T}_{ij}$ as

$$\mathbb{T}_{ij} \quad = \quad g\left(\mathbf{T}_{ij}\right) \tag{19}$$

where $g\left(\bullet\right)$ is a function of $\mathbf{T}_{ij}$, and $\mathbf{T}_{ij}$ can be written as

$$\begin{aligned}\mathbf{T}_{ij} \quad &= \quad \log \frac{P_t(i,j|\mathbb{Y}_1^{\mathbf{n}_M})^{\frac{P_t(i,j|\mathbb{Y}_1^{\mathbf{n}_M})}{\mathtt{u}_i\mathbb{P}_{ij}}}}{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})^{\frac{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})}{\mathtt{u}_i}}} && (20)\\ &= \quad \frac{P_t(i,j|\mathbb{Y}_1^{\mathbf{n}_M})}{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})\mathbb{P}_{ij}} \log \frac{P_t(i,j|\mathbb{Y}_1^{\mathbf{n}_M})}{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})}. && (21)\end{aligned}$$

Applying Bayesian rule, we have

$$\begin{aligned}\mathbf{T}_{ij} &= \frac{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})P_t(j|i,\mathbb{Y}_1^{\mathbf{n}_M})}{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})\mathbb{P}_{ij}} \log \frac{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})P_t(j|i,\mathbb{Y}_1^{\mathbf{n}_M})}{P_t(i|\mathbb{Y}_1^{\mathbf{n}_M})}\\ &= \frac{P_t(j|i,\mathbb{Y}_1^{\mathbf{n}_M})}{\mathbb{P}_{ij}} \log P_t(j|i,\mathbb{Y}_1^{\mathbf{n}_M}). && (22)\end{aligned}$$

For the intra/initial states, we have

$$P_t(j|i,\mathbb{Y}_1^{\mathbf{n}_M}) \overset{(a)}{=} P_t(j|i) \overset{(b)}{=} \mathbb{P}_{ij}$$

where $(a)$ follows that there exists only one path from state $i$, where $i$ is an intra/initial state, to any other state independent of any given sequence, and $(b)$ follows that the transition probability for the Markov chain provides sufficient information to describe any transition probability from one state to another at any given time. Therefore, assigning an arbitrary power values for these states do not affect the transition probability at time $t$ given the output sequence. For the tractability of the mathematical derivations, we assume these states produce no signal at all. Moreover, for these states, we can simplify (22) further as

$$\mathbf{T}_{ij} = \frac{P_t(j|i)}{\mathbb{P}_{ij}} \log P_t(j|i) = \frac{\mathbb{P}_{ij}}{\mathbb{P}_{ij}} \log \mathbb{P}_{ij} = 0 = \mathbb{T}_{ij} \tag{23}$$

for any $j \in \mathcal{S}_M$, which means

$$\mathtt{u}_i \sum_{j\in\mathcal{S}_M} \mathbb{T}_{ij} = 0.$$

Therefore, given an instruction with an execution time larger than one, the intra/initial states of this instruction do not contribute to the equation given in (8) in terms of $\mathbb{T}_{ij}$. As the second step, we have to check the contribution of these intra/initial states to the definition of leakage capacity. In that respect, we have

$$-\mathtt{u}_i \sum_{j\in\mathcal{S}_M} \mathbb{P}_{ij} \log \mathbb{P}_{ij} = 0$$

since $\mathbb{P}_{ij}$ is equal to zero or one. Therefore, for the intra/initial states, we have

$$\mathtt{u}_i \sum_{j\in\mathcal{S}_M} \mathbb{P}_{ij}\left(\mathbb{T}_{ij} - \log \mathbb{P}_{ij}\right) = 0$$

which means total contribution is zero if the considered state is an intra/initial state of an instruction whose execution time takes more than one clock cycle.

Now, let's check the values obtained from the exit states. Here, our analysis is based on the assumption that the leakage occurs at exit states. To proceed further, we define $\mathbb{T}_{ij}$ as

$$\mathbb{T}_{ij} = \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \left[\log \frac{P_t(k,l|Y_1^n)^{\frac{P_t(k,l|Y_1^n)}{\mathtt{u}_i\mathbb{P}_{ij}}}}{P_t(k|Y_1^n)^{\frac{P_t(k|Y_1^n)}{\mathtt{u}_i}}}\right] \tag{24}$$

where $i$ is the exit state of instruction $k$ and $j$ is the initial state of instruction $l$. The reason to redefine $\mathbb{T}_{ij}$ is to satisfy the duality between the problems because it is obvious that $T_{kl} = \mathbb{T}_{ij}$ after redefining $\mathbb{T}_{ij}$. Moreover, the transition probabilities for an exit state to an initial state are kept exactly the same with the corresponding instruction to instruction transition probabilities, i.e. $P_{DIV,SUB} = \mathbb{P}_{D_4,SUB}$, $P_{MUL,SUB} = \mathbb{P}_{M_3,SUB}$, etc. (Here, transitions are based on Fig. 2 and Fig. 3), to preserve the duality.

Therefore, for the exit state $k$ of instruction $i$, we can write the following equality:

$$\sum_{j\in\mathcal{S}} P_{ij}\left[\log \frac{1}{P_{ij}} + T_{ij}\right] = \sum_{j\in\mathcal{S}_M} \mathbb{P}_{kj}\left[\log \frac{1}{\mathbb{P}_{kj}} + \mathbb{T}_{kj}\right]. \tag{25}$$

Note that the number of states in the original Markov Model is the same as the number of exit states in the transformed Markov Model.

Since the transformed Markov Model is also an indecomposable model, it has a stationary distribution which can be written as

$$\mathbf{u} = \mathbf{u}\mathbf{P}$$

where $\mathbf{u}$ is the state probabilities, and $\mathbf{P}$ is the matrix containing the state transition probabilities. To derive mathematical results, we utilize the classical probability constraints. For that, let $\mathtt{u}_i$ be the stationary distribution of $k^{th}$ sub-state of instruction $M$, and $\mathtt{u}_k^M$ be its mapped version. The constraints for the transformed model are

$$\sum_i \mathtt{u}_i = \sum_{i,j} \mathtt{u}_i^j = 1$$

and

$$\mathtt{u}_i^j = \mathtt{u}_k^j,\ \forall i,k \in \{1,\cdots,L_j\} \text{ and } j \in \mathcal{S},$$

i.e., $\mathtt{u}_{M_1} = \mathtt{u}_{M_2} = \mathtt{u}_{M_{L_{MUL}}}$. Therefore, we have

$$\sum_{i\in\mathcal{S}_M} \mathtt{u}_i = \sum_{i\in\mathtt{E}(\mathcal{S}_M)} L_i \mathtt{u}_{i_{L_i}} = 1 \tag{26}$$

where $\mathtt{E}\left(\mathcal{S}_M\right)$ is the set containing exit states of instructions. Let us rewrite the capacity definition for the transformed model as

$$\begin{aligned}C_T \quad &= \quad \max_{\mathbb{P}_{ij}} \sum_{i,j:(i,j)\in\mathcal{T}_M} \mathtt{u}_i \mathbb{P}_{ij}\left[\log \frac{1}{\mathbb{P}_{ij}} + \mathbb{T}_{ij}\right] && (27)\\ &= \quad \max_{P_{ij}} \sum_{i,j:(i,j)\in\mathtt{E}(\mathcal{T}_M)} \mathtt{u}_i P_{ij}\left[\log \frac{1}{P_{ij}} + T_{ij}\right] && (28)\end{aligned}$$

where (28) follows the equality given in (25), and $\mathtt{E}(\mathcal{T}_M)$ represents the state transition set of all exit states. To proceed forward, we define

$$\mathrm{u}^j_{L_j} = \frac{\mu_j}{\sum\limits_{k\in\mathcal{S}} L_k\mu_k}$$

which obeys the probability constraint that

$$1 = \sum_{i\in\mathcal{S}_M} \mathrm{u}_i = \sum_{i\in\mathtt{E}(\mathcal{S}_M)} L_i\mathrm{u}_i = \sum_{j\in\mathcal{S}} L_j\mathrm{u}^j_{L_j} \tag{29}$$

$$= \sum_{j\in\mathcal{S}} \frac{L_j\mu_j}{\sum\limits_{k\in\mathcal{S}} L_k\mu_k} = \frac{\sum\limits_{i\in\mathcal{S}} L_i\mu_i}{\sum\limits_{k\in\mathcal{S}} L_k\mu_k} = 1 \tag{30}$$

where (29) follows the equality given in (26). Therefore, we have

$$C_T = \max_{\mathbb{P}_{ij}} \sum_{(i,j)\in\mathcal{T}_M} \mathrm{u}_i\mathbb{P}_{ij}\left[\log\frac{1}{\mathbb{P}_{ij}} + \mathbb{T}_{ij}\right] \tag{31}$$

$$= \max_{\mathbb{P}_{ij}} \sum_{(i,j)\in\mathtt{E}(\mathcal{T}_M)} \mathrm{u}_i\mathbb{P}_{ij}\left[\log\frac{1}{\mathbb{P}_{ij}} + \mathbb{T}_{ij}\right] \tag{32}$$

$$= \max_{P_{ij}} \sum_{(i,j)\in\mathcal{T}} \frac{\mu_i}{\sum\limits_{k\in\mathcal{S}} L_k\mu_k} P_{ij}\left[\log\frac{1}{P_{ij}} + T_{ij}\right] \tag{33}$$

$$= \max_{P_{ij}} \frac{\sum\limits_{(i,j)\in\mathcal{T}} \mu_i P_{ij}\left[\log\frac{1}{P_{ij}} + T_{ij}\right]}{\sum\limits_{k\in\mathcal{S}} L_k\mu_k} \tag{34}$$

where (33) follows the equality given in (25). Since (34) is exactly same with (6), the proposed transformation preserves the duality which ends the proof.

## Appendix II
## Mathematical Derivation of ESP

In this section, we show how ESP is related to the alternation power at the corresponding frequency. For measurement and derivation purposes of ESP, the code given in Fig. 4 is used. Here, we assume that the sampled sequence of $s(t)$ is $s[m]$, and each sample can be written as $s[m] = i[m] + w[m]$ where $i[m]$ is the emanated signal sample and $w[m]$ is additive independent and identically distributed (i.i.d.) white noise with zero mean and variance $\sigma^2_w$. We assume that the noise term contains all disruptive signal powers and their variations.

Let $s^{L_1}_1[m]$ be the sequence corresponding to only one period of the first for-loop signal, and the length of $s^{L_1}_1[m]$ is $N_L$. We can decompose $s^{L_1}_1[m]$ into three different sequences. Assuming the depth of the pipeline is $P_S$, these sequences are:

* The samples of the considered instruction including all pipeline stages:

$$a_{\mathcal{A}_1}[m] = \big[0, \ \cdots, \ 0, a^1_{\mathcal{A}_1}, a^2_{\mathcal{A}_1}, \cdots, a^p_{\mathcal{A}_1}, \ a_{\mathcal{A}_1}[0], \ \cdots, \ a_{\mathcal{A}_1}[N_I-1], a^{p+1}_{\mathcal{A}_1}, \cdots, a^{P_S}_{\mathcal{A}_1}\big]$$

where $a^i_{\mathcal{A}_1}$ is the $i^{th}$ sample of the emitted signal when $\mathcal{A}_1$ is in a pipeline stage rather than execution.

* The samples of other activities rather than $\mathcal{A}_1$ to make the microbenchmark practical including the pipeline effect:

$$o_{L_1}[m] = [o[0], \ o[1], \ \cdots, \ o[N_L-2], \ o[N_L-1]] \,.$$

Here, we need to note that the samples taken for the first iteration of the inner for-loop will be different than the other iterations even for the ideal case due to pipeline depth. Although it looks like the periodicity is not valid for $o_{L_1}[m]$, we can able to ignore it thanks to the assumption that $n_{inst}$ is large.

* Finally, the last sequence compromises all other components which are assumed to be Gaussian and given as

$$w_{L_1}[m] = [w[0], \ w[1], \ \cdots, \ w[N_L-1]] \,.$$

Combining all these sequences, we have

$$s^{L_1}_1[m] = a_{\mathcal{A}_1}[m] + o_{L_1}[m] + w_{L_1}[m].$$

Following the same decomposition for the second for-loop signal, called $s^{L_2}_2[n]$, we have

* $o_{L_2}[m] = [o[0], \ o[1], \ \cdots, \ o[N_L-2], \ o[N_L-1]]$,
* $w_{L_2}[m] = [w[0], \ w[1], \ \cdots, \ w[N_L-1]]$,

which leads to

$$s^{L_2}_2[m] = o_{L_2}[m] + w_{L_2}[m].$$

Here, we assume that NOP consumes very little energy as it passes through the stages of a pipeline, which means it produces a signal whose power is close to zero. Observe here that since both loops are almost identical except the part where $\mathcal{A}_1$ is inserted, we assume that $o_{L_1}[m]$ and $o_{L_2}[m]$ are identical to each other, therefore, we refer both sequences as $\mathrm{o}[m]$. Let $p[m]$ be a square wave with 50% duty cycle and period of $2N_L n_{inst}$ samples, and $\mathbf{s}[m]$ be the one period signal of the outer for-loop. Let also $\mathbf{a}_{\mathcal{A}_1}[m]$ and $\mathbf{o}[m]$ be generated by concatenating $a_{\mathcal{A}_1}[m]$ and $o_{L_1}[m]$ by $2\cdot n_{inst}$ times, respectively. Furthermore, we can simply assume that the noise components are i.i.d. for both for-loops. Therefore, we have

$$\mathbf{s}[m] = p[m]\mathbf{a}_{\mathcal{A}_1}[m] + \mathbf{o}[m] + \mathbf{w}[m].$$

The first harmonic of $\mathbf{s}[m]$ can be written as

$$\mathbf{S}[1] = \frac{\sum\limits_{\gamma=0}^{2N_L n_{inst}-1} P[1-\gamma]\mathbf{A}_{\mathcal{A}_1}[\gamma]}{2N_L n_{inst}} + \mathbf{O}[1] + \mathbf{W}[1]. \tag{35}$$

We know that $\mathbf{O}[k]$ and $\mathbf{A}_{\mathcal{A}_1}[k]$ have nonzero frequency components only if $k = 2\cdot n_{inst}\cdot l$, $\forall l \in \{0, \cdots, N_L-1\}$, and $|P[1]| \gg |P[1-2n_{inst}]|$. Therefore, (35) can be approximately written as

$$\mathbf{S}[1] \approx \frac{P[1]}{2N_L n_{inst}}\mathbf{A}_{\mathcal{A}_1}[0] + \mathbf{W}[1]. \tag{36}$$

If we take the magnitude square of both sides, we have

$$\begin{aligned} |\mathbf{S}[1]|^2 &= \left| \frac{P[1]}{2N_L n_{inst}} \mathbf{A}_{\mathcal{A}_1}[0] + \mathbf{W}[1] \right|^2 \\ &= \left| \frac{P[1]}{2N_L n_{inst}} \mathbf{A}_{\mathcal{A}_1}[0] \right|^2 + |\mathbf{W}[1]|^2 \\ &\quad - \frac{\Re e \left\{ P[1] \mathbf{A}_{\mathcal{A}_1}[0] \mathbf{W}^*[1] \right\}}{N_L \cdot n_{inst}} \end{aligned} \tag{37}$$

where $(\cdot)^*$ is conjugation and $\Re e\{\cdot\}$ takes the real part of its argument. Assuming

$$\left| \frac{P[1]}{2N_L n_{inst}} \mathbf{A}_{\mathcal{A}_1}[0] \right| \gg \Re e \left\{ P[1] \mathbf{A}_{\mathcal{A}_1}[0] \mathbf{W}^*[1] \right\},$$

the first harmonic of $\mathbf{s}[m]$ can be simplified further as

$$|\mathbf{S}[1]|^2 \approx \left| \frac{P[1]}{2N_L n_{inst}} \mathbf{A}_{\mathcal{A}_1}[0] \right|^2 + |\mathbf{W}[1]|^2. \tag{38}$$

To proceed further, we need to have the expression for $\mathbf{A}_{\mathcal{A}_1}[0]$. Utilizing the DFS, we have

$$\mathbf{A}_{\mathcal{A}_1}[0] = \sum_{\gamma=0}^{2N_L n_{inst}} \mathbf{a}_{\mathcal{A}_1}[\gamma] \overset{(a)}{=} 2n_{inst} \sum_{\gamma=0}^{N_L} \mathbf{a}_{\mathcal{A}_1}[\gamma] \tag{39}$$

where $(a)$ follows the fact that $\mathbf{a}_{\mathcal{A}_1}[m]$ is periodic with $N_L$ samples. Since, at each period, only $N_I + P_S$ of $\mathbf{a}_{\mathcal{A}_1}[m]$ have nonzero values, and assuming $N_I + P_S$ is large enough, (39) can be written as

$$\begin{aligned} \mathbf{A}_{\mathcal{A}_1}[0] &= 2(N_I + P_S) n_{inst} \mathbb{E}\left[\mathbf{a}_{\mathcal{A}_1}[m]\right] \\ &= 2(N_I + P_S) n_{inst} \mu_{\mathcal{A}_1} \end{aligned} \tag{40}$$

Note that exploiting (10), ESP$[\mathcal{A}_1]$ can also be written as

$$\begin{aligned} \mathrm{ESP}[\mathcal{A}_1] &= \frac{T_s(N_I + P_S)}{R} \mathbb{E}\left[ |\mathbf{a}_{\mathcal{A}_1}[m]|^2 \right] \\ &= \frac{T_s(N_I + P_S)}{R} \left( \mu_{\mathcal{A}_1}^2 + \sigma_{\mathcal{A}_1}^2 \right) \\ &\approx \frac{T_s(N_I + P_S)}{R} \mu_{\mathcal{A}_1}^2 \end{aligned} \tag{41}$$

where $\sigma_{\mathcal{A}_1}$ is the standard deviation of the samples while an instruction signal is executed, and (41) follows the assumption that the variation in measured signal during the execution of an instruction is much smaller than its mean value. Combining (40) with (41), we have

$$\mathrm{ESP}[\mathcal{A}_1] \approx \frac{T_s}{4R(N_I + P_S) n_{inst}^2} |\mathbf{A}_{\mathcal{A}_1}[0]|^2. \tag{42}$$

The final step is to show how ESP$[\mathcal{A}_1]$ and the alternation power $\mathcal{P}(f_{\text{alt}})$ are related to each other. The relation between the first harmonic of the signal and the power measure through the spectrum analyzer is given as [45], [46]

$$\mathcal{P}(f_{\text{alt}}) = \frac{2}{R} \left( \frac{|\mathbf{S}[1]|}{2 \cdot N_L \cdot n_{inst}} \right)^2. \tag{43}$$

Let $\mathcal{P}_{\mathcal{A}_1}(f_{\text{alt}})$ be the measured alternation power when $\mathcal{A}_1$ is inserted into first for-loop, and the second loop is kept empty. On the other hand, let $\mathcal{P}_0(f_{\text{alt}})$ be the measured power when both for-loops are kept empty (Here, we need to remark that keeping the loops empty means inserting NOP as many as the total number of clock cycles required to execute $\mathcal{A}_1$). Finally, let $\boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}})$ be the normalized alternation power for the instruction $\mathcal{A}_1$ which is defined as

$$\boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}}) = \mathcal{P}_{\mathcal{A}_1}(f_{\text{alt}}) - \mathcal{P}_0(f_{\text{alt}}).$$

The critical observation is that the term related to $\mathcal{A}_1$ in (38) is zero when both for loops are kept empty. Assume $\mathbf{S}_{\mathcal{A}_1}[1]$ and $\mathbf{S}_0[1]$ denote the first harmonics of the signal when 1) $\mathcal{A}_1$ is inserted, and 2) both loops are kept empty, respectively. Considering this setup, we can write

$$|\mathbf{S}_{\mathcal{A}_1}[1]|^2 - |\mathbf{S}_0[1]|^2 \approx \frac{1}{\pi^2} |\mathbf{A}_{\mathcal{A}_1}[0]|^2 \tag{44}$$

where we utilize the approximation that $\pi|P[1]| \approx 2N_L n_{inst}$. Exploiting the definition of normalized alternation power, and employing the equations given in (42), (43), and (44), we can write

$$\begin{aligned} \boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}}) &= \mathcal{P}_{\mathcal{A}_1}(f_{\text{alt}}) - \mathcal{P}_0(f_{\text{alt}}) \\ &= \frac{2/R}{(2N_L n_{inst})^2} \left( |\mathbf{S}_{\mathcal{A}_1}[1]|^2 - |\mathbf{S}_0[1]|^2 \right) \\ &= \frac{2/R}{(2N_L n_{inst})^2} \frac{1}{\pi^2} |\mathbf{A}_{\mathcal{A}_1}[0]|^2 \\ &= \frac{\mathrm{ESP}[\mathcal{A}_1]}{(\pi N_L)^2} \frac{2(N_I + P_S)}{T_s}. \end{aligned} \tag{45}$$

To emphasize the relation between the power at the alternation frequency and ESP, we can write

$$f_{\text{alt}} \cdot n_{inst} = \frac{1}{2 \cdot N_L \cdot T_s}. \tag{46}$$

Plugging the equation (46) into (45), we have

$$\begin{aligned} \boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}}) &= \left( \frac{2}{\pi} \right)^2 \frac{\mathrm{ESP}[\mathcal{A}_1]}{N_L/(N_I + P_S)} \frac{1}{2N_L T_s} \\ &= \left( \frac{2}{\pi} \right)^2 \frac{\mathrm{ESP}[\mathcal{A}_1]}{N_L/(N_I + P_S)} f_{\text{alt}} n_{inst}. \end{aligned} \tag{47}$$

To finalize our proof, we need to keep ESP$[\mathcal{A}_1]$ alone on the one side. Therefore, we have

$$\mathrm{ESP}[\mathcal{A}_1] = \left( \frac{\pi}{2} \right)^2 \frac{\boldsymbol{\mathcal{P}}_{\mathcal{A}_1}(f_{\text{alt}}) \cdot N_L}{(N_I + P_S) \cdot f_{\text{alt}} \cdot n_{inst}} \tag{48}$$

which concludes the proof.

## References

[1] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. L. Callan, A. G. Zajic, and M. Prvulovic, "One&done: A single-decryption em-based attack on openssl's constant-time blinded RSA," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, 2018, pp. 585–602.

[2] M. Backes, M. Dürmuth, S. Gerling, M. Pinkal, and C. Sporleder, "Acoustic side-channel attacks on printers," in *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, 2010, pp. 307–322.

[3] M. Guri, A. Kachlon, O. Hasson, G. Kedma, Y. Mirsky, and Y. Elovici, "Gsmem: Data exfiltration from air-gapped computers over GSM frequencies," in *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, 2015, pp. 849–864. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/guri

[4] A. Zajic and M. Prvulovic, "Experimental demonstration of electromagnetic information leakage from modern processor-memory systems," in *IEEE Transactions on Electromagnetic Compatibility, Volume: 56, Issue: 4,*, 2014, p. 885893.

[5] A. G. Bayrak, F. Regazzoni, P. Brisk, F.-X. Standaert, and P. Ienne, "A first step towards automatic application of power analysis countermeasures," in *Proceedings of the 48th Design Automation Conference (DAC)*, 2011.

[6] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound countermeasures to counteract power-analysis attacks," in *Proceedings of CRYPTO'99, Springer, Lecture Notes in computer science*, 1999, pp. 398–412.

[7] D. Boneh and D. Brumley, "Remote Timing Attacks are Practical," in *Proceedings of the USENIX Security Symposium*, 2003.

[8] B. Coppens, I. Verbauwhede, K. D. Bosschere, and B. D. Sutter, "Practical Mitigations for Timing-Based Side-Channel Attacks on Modern x86 Processors," in *Proceedings of the 30th IEEE Symposium on Security and Privacy*, 2009, pp. 45–60.

[9] L. Goubin and J. Patarin, "DES and Differential power analysis (the "duplication" method)," in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 1999*, 1999, pp. 158–172.

[10] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of CRYPTO'96, Springer, Lecture notes in computer science*, 1996, pp. 104–113.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis: leaking secrets," in *Proceedings of CRYPTO'99, Springer, Lecture notes in computer science*, 1999, pp. 388–397.

[12] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks of modular exponentiation in smart cards," in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 1999*, 1999, pp. 144–157.

[13] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs," in *Cryptographic Hardware and Embedded Systems - CHES 2014*, ser. Lecture Notes in Computer Science, L. Batina and M. Robshaw, Eds. Springer Berlin Heidelberg, 2014, vol. 8731, pp. 242–260. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-44709-3_14

[14] M. Hutter and J.-M. Schmidt, "The temperature side channel and heating fault attacks," in *Smart Card Research and Advanced Applications*, ser. Lecture Notes in Computer Science, A. Francillon and P. Rohatgi, Eds. Springer International Publishing, 2014, vol. 8419, pp. 219–235. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08302-5_15

[15] J. Brouchier, T. Kean, C. Marsh, and D. Naccache, "Temperature attacks," *Security Privacy, IEEE*, vol. 7, no. 2, pp. 79–82, March 2009.

[16] D. Gullasch, E. Bangerter, and S. Krenn, "Cache games–bringing access-based cache attacks on aes to practice," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 490–505.

[17] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 494–505.

[18] Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi, "Cryptanalysis of block ciphers implemented on computers with cache," 01 2002.

[19] J. Quisquater and D. Samyde, "Electromagnetic analysis (EMA): measures and counter-measures for smart cards," in *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, 2001, pp. 200–210.

[20] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, no. Generators, 2001, pp. 251–261.

[21] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, 2002, pp. 29–45.

[22] E. D. Mulder, S. B. Örs, B. Preneel, and I. Verbauwhede, "Differential power and electromagnetic attacks on a FPGA implementation of elliptic curve cryptosystems," *Computers & Electrical Engineering*, vol. 33, no. 5-6, pp. 367–382, 2007.

[23] J. K. Millen, "Covert channel capacity," in *Security and Privacy, 1987 IEEE Symposium on*, April 1987, pp. 60–60.

[24] Z. Wang and R. Lee, "Capacity estimation of non-synchronous covert channels," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, June 2005, pp. 170–176.

[25] R. J. Anderson and F. A. Petitcolas, "On the limits of steganography," *IEEE Journal on selected areas in communications*, vol. 16, no. 4, pp. 474–481, 1998.

[26] V. Crespi, G. Cybenko, and A. Giani, "Engineering statistical behaviors for attacking and defending covert channels," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 124–136, 2013.

[27] M. C. Davey and D. J. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 687–698, 2001.

[28] R. Venkataramanan, S. Tatikonda, and K. Ramchandran, "Achievable rates for channels with deletions and insertions," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 6990–7013, 2013.

[29] A. Kirsch and E. Drinea, "Directly lower bounding the information capacity for channels with iid deletions and duplications," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 86–102, 2010.

[30] J. Hu, T. M. Duman, M. F. Erden, and A. Kavcic, "Achievable information rates for channels with insertions, deletions, and intersymbol interference with iid inputs," *IEEE Transactions on Communications*, vol. 58, no. 4, 2010.

[31] S. Verdú and S. Shamai, "Variable-rate channel capacity," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2651–2667, 2010.

[32] M. Rahmati and T. M. Duman, "Bounds on the capacity of random insertion and deletion-additive noise channels," *IEEE Transactions on Information Theory*, vol. 59, no. 9, pp. 5534–5546, 2013.

[33] H. Mercier, V. Tarokh, and F. Labeau, "Bounds on the capacity of discrete memoryless channels corrupted by synchronization and substitution errors," *IEEE Transactions on Information Theory*, vol. 58, no. 7, pp. 4306–4330, 2012.

[34] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[35] B. Yilmaz, A. Zajic, and M. Prvulovic, "Modelling jitter in wireless channel created by processor-memory activity," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018*, 04 2018, pp. 2037–2041.

[36] B. B. Yilmaz, M. Prvulovic, and A. Zajić, "Capacity of deliberate side channels created by software activities," in *Military Communications Conference (MILCOM), MILCOM 2018-2018 IEEE*. IEEE, 2018.

[37] B. B. Yilmaz, R. Callan, A. Zajic, and M. Prvulovic, "Capacity of the em covert/side-channel created by the execution of instructions in a processor," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 605–620, 2018.

[38] A. Kavcic, "On the capacity of Markov sources over noisy channels," in *2009 IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 5, 2001, pp. 2997–3001.

[39] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design MIPS Edition: The Hardware/Software Interface*. Newnes, 2013.

[40] R. Callan, A. Zajic, and M. Prvulovic, "A Practical Methodology for Measuring the Side-Channel Signal Available to the Attacker for Instruction-Level Events," in *Proceedings of the 47th International Symposium on Microarchitecture (MICRO)*, 2014.

[41] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2009, pp. 443–461.

[42] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi *et al.*, "Test vector leakage assessment (tvla) methodology in practice," in *International Cryptographic Module Conference*, vol. 1001, 2013, p. 13.

[43] T. Schneider and A. Moradi, "Leakage assessment methodology," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2015, pp. 495–513.

[44] F.-X. Standaert, "How (not) to use welch's t-test in side-channel security evaluations," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2018, pp. 65–79.

[45] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes in c," *Cambridge University Press*, vol. 1, p. 3, 1988.

[46] G. Heinzel, A. Rüdiger, and R. Schilling, "Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new at-top windows," 2002.

**Baki B. Yilmaz** (S'16) received the B.Sc. and M.Sc. degrees in Electrical and Electronics Engineering from Koc University, Turkey in 2013 and 2015 respectively. He joined Georgia Institute of Technology in Fall 2016 and he is currently pursuing his PhD in School of Electrical and Computer Engineering, focusing on quantifying covert/side-channel information leakage and capacity. Previously, he worked on channel equalization and sparse reconstruction. His research interests span areas of electromagnetic, signal processing and information theory.

**Milos Prvulovic** (S'97-M'03-SM'09) received the B.Sc. degree in electrical engineering from the University of Belgrade in 1998, and the M.Sc. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 2001 and 2003, respectively. He is a Professor in the School of Computer Science at the Georgia Institute of Technology, where he joined in 2003. His research interests are in computer architecture, especially hardware support for software monitoring, debugging, and security.

He is a past recipient of the NSF CAREER award, and a senior member of the ACM, the IEEE, and the IEEE Computer Society.

**Alenka Zajic** (S'99-M'09-SM'13) received the B.Sc. and M.Sc. degrees form the School of Electrical Engineering, University of Belgrade, in 2001 and 2003, respectively. She received her Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology in 2008. Currently, she is an Associate Professor in the School of Electrical and Computer Engineering at Georgia Institute of Technology. Prior to that, she was a visiting faculty member in the School of Computer Science at Georgia Institute of Technology, a post-doctoral fellow in the Naval Research Laboratory, and a design engineer at Skyworks Solutions Inc. Her research interests span areas of electromagnetic, wireless communications, signal processing, and computer engineering.

Dr. Zajić was the recipient of the 2017 NSF CAREER award, 2012 Neal Shepherd Memorial Best Propagation Paper Award, the Best Student Paper Award at the IEEE International Conference on Communications and Electronics 2014, the Best Paper Award at the International Conference on Telecommunications 2008, the Best Student Paper Award at the 2007 Wireless Communications and Networking Conference, and the Dan Noble Fellowship in 2004, which was awarded by Motorola Inc. and the IEEE Vehicular Technology Society for quality impact in the area of vehicular technology. Currently, she is an editor for IEEE Transactions on Wireless Communications.