Capacity of the EM Covert/Side-Channel Created by the Execution of Instructions in a Processor

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

***Abstract*— The goal of this paper is to answer how much information is "transmitted" by execution of particular sequence of instructions in a processor. Introducing such a measure would provide quantitative guidance for designing programs and computer hardware that minimizes inadvertent (side channel) information leakage, and would also help detect parts of a program or hardware design that have unusually high leakage (i.e. were designed to function as covert channel "transmitters"). To answer this question, we propose a new method to estimate the maximum information leakage through EM signals generated by execution of instructions in a processor. We start by deriving a mathematical relationship between electromagnetic side-channel energy (ESE) of individual instructions and the measured pairwise side-channel signal power. Then, we use this measure to calculate the transition probabilities needed for estimating capacity. Finally, we propose a new method to estimate side/covert channel capacity created by the execution of instructions in a processor and illustrate our results in several computer systems.**

***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

A covert/side-channel is a communication channel that is neither designed nor intended to transfer information [1]. Covert/side-channel signals are generated as a side effect of performing legitimate program activity on the hardware of the computer system. Since program activity and the resulting hardware activity are dependent on data processed by the program, the resulting side/covert channel signals can (and usually do) carry information about those data values. This allows attackers to obtain sensitive information by analyzing the side/covert channel signals produced by programs which use this sensitive information. Covert/side-channel attacks have been acknowledged as a serious security threat [2], and the severity of that threat dramatically increases as computer systems are increasingly mobile and may be physically controlled or placed close to potentially malicious entities. Examples of this are laptops and smart-phones used in coffee shops and other public spaces, servers in data centers controlled by third parties, robotic or remotely operated military vehicles and aircrafts in hostile territory, etc. In all these cases, the system is physically close to persons who should not have any access to its data. Fortunately, in most side-channel attacks, e.g. power analysis [3], [4], [5], [6], [7], [8], [9], [10], [11], temperature analysis [12], [13], caches-based attacks [14], [15], [16], etc. the "listener" still requires some degree of direct access to the system (to attach probes, run a "receiver" processes that measures cache activity, etc.), with some (often significant) risk of detection.

However, covert/side-channel attacks based on the system's electromagnetic (EM) emanations only require physical proximity, allowing sufficiently motivated attackers to carry out numerous attacks wirelessly and with little risk of detection [17], [18], [19], [20], [21], [22]. Previous work [21], [23] confirms that modulated EM emanations from laptop and desktop systems can be created by executing seemingly innocuous code, and EM covert channel transmission of thousands of bits per second has been demonstrated to distances of at least several meters and even through a wall.

The severity of a covert/side-channel is often measured in terms of bit rates, i.e. how rapidly it can transmit information. Millen was the first to establish a connection between Shannon's information theory and information flow models in computer systems [24] and calculated the capacity of such a covert channel. However, that model assumes a synchronous channel, which is not a realistic assumption for covert/side-channels. In contrast to most communication systems which are designed to avoid symbol loss and/or insertion with little or no overhead, the covert 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 [25], [26], [27], [28], [29], [30], [31], bounds on the capacity of channels corrupted with synchronization and substitution errors [32], [33], [34], or bounds on the capacity when codewords have variable length but no errors in the channel [32], [35], none of them provide 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. Introducing such a measure would provide

This work has been supported, in part, by NSF grants 1563991 and 1318934, AFOSR grant FA9550-14-1-0223, 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, AFOSR, and DARPA.

Baki Berkay Yilmaz, Robert Callan, 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.

quantitative guidance for designing programs and computer hardware that minimize inadvertent (side channel) information leakage, and would also help detect parts of a program or hardware design that have unusually high leakage (i.e. were designed to function as covert channel "transmitters").

To address this problem, we need to establish relationship between software activity, observed emanations, and side-channel capacity. The first attempt to quantify which instructions have the greatest potential to create side-channel vulnerabilities was reported in [22], where a measurement technique is devised to quantify the emanated power that can be attributed to the difference in execution between a pair of individual instructions.

In this paper, we derive a mathematical relationship between electromagnetic side-channel energy (ESE) of individual instructions and the measured pairwise side-channel signal power and use this measure to calculate the transition probabilities needed for estimating capacity. Then, we propose a new method to estimate side/covert channel capacity created by the execution of instructions in a processor. Finally, we illustrate how the proposed method works through evaluation of capacity in several practical systems.

The rest of the paper is organized as follows. Section II briefly reviews an approach for measuring the side-channel energy created by executing different processor instructions, Section III analytically quantifies the difference in energy available to an attacker between two instructions, Section IV describes a new method to quantify covert/side-channel capacity, Section V analyzes capacity of several systems, Section VI proposes some possible defensive methods for EM based attacks, and Section VII provides concluding remarks.

II. Review of a Method for Measuring Pairwise Side-channel Signal Power from Processor Instructions

In [22], it was assumed that an attacker has access to a program's source or executable code, and can observe EM emanations from the victim's system while this program is running. The attacker attempts to extract sensitive information by recording EM emanations, using them to infer which instructions are executed, and then infers sensitive data from knowledge of the executed instructions.

The most direct approach to quantifying the EM emanations from side-channel signal created by executing instruction X_1 vs. executing instruction X_2 is to measure the EM emanations while instruction X_1 is active, measure the EM emanations while instruction X_2 is active, and then take the difference between these two signals. This approach is impractical in high performance systems for several reasons. First, equipment capable of recording the $x_1(t)$ and $x_2(t)$ signals at greater than 10 Gsamples/sec (as required to test a processor using a GHz clock) and with thermal, quantization, etc. noise that is low enough to not obscure the (extremely small) difference between the two signals, is prohibitively expensive or non-existent. Second, complex processors heavily optimize the scheduling and execution of instructions, so determining the times where the test instructions X_1 or X_2 are actually active would be problematic. Third, some other instructions must be present around X_1 and X_2 to make the measurement practical (to trigger the measurement, setup the registers and memory used by instructions X_1 and X_2, etc.), and these unrelated components of the received signal can easily obfuscate the signal components created by the X_1 and X_2 instructions themselves.

```
1  while(1){
2    // Do some instances of the X1 instruction
3    for(i=0;i<n_inst;i++){
4      ptr1=(ptr1&~mask1)|((ptr1+offset)&mask1);
5      // The X1-instruction, e.g. a load
6      value=*ptr1;
7    }
8    // Do some instances of the X2 instruction
9    for(i=0;i<n_inst;i++){
10     ptr2=(ptr2&~mask2)|((ptr2+offset)&mask2);
11     // The X2-instruction, e.g. a store
12     *ptr2=value;
13   }
14 }
```

Fig. 1. The X_1/X_2 alternation pseudo-code.

To overcome these problems, a program has been designed in [22] that causes the system to execute X_1 and X_2 instructions in a controlled way that minimizes the effect of all other unrelated system activities, while concentrating the side channel energy into a narrow spectral band to minimize the impact of noise on the measurement. We produced these controlled emanations by choosing a repetition period $T_{\rm alt}$ and then create a small test program (microbenchmark) containing a `for` loop such that the first half of the loop does many repetitions of activity X_1 and the second half does many repetitions of activity X_2. The microbenchmark in Figure 1 implements this idea by executing X_1 and X_2 instructions $n_{\rm inst}$ times (denoted as $n_{\rm inst}$ in Figure 1) in each iteration of the outer loop. Lines 2 through 7 execute $n_{\rm inst}$ instances of the X_1 instruction, and then lines 8 through 13 execute the same number of instances of the X_2 instruction. Thus lines 2 through 13 represent one X_1/X_2 alternation, and this alternation is repeated (line 1) until the measurement of the side-channel signal is complete. It is critical to note that the value of $T_{\rm alt}$ is controlled directly by varying $n_{\rm inst}$. For example, increasing $n_{\rm inst}$ increases the time required to execute one iteration of the outer loop ($T_{\rm alt}$). The value of $T_{\rm alt}$ can be directly measured using counters available through processor instructions (e.g. the x86 `rdtsc` instruction) or the operating system (e.g. the Windows API $\mathrm{QueryPerformanceCounter}()$ function). We can then select the $n_{\rm inst}$ value that produces the desired alternation frequency ($f_{\rm alt} = 1/T_{\rm alt}$).

The eleven common x86 instructions are considered throughout the paper and listed in Figure 2, including loads and stores that go to different levels of the cache hierarchy, simple (ADD and SUB) and more complex (MUL and DIV) integer arithmetic, and the "No Instruction" case where the appropriate line in our alternation code (Line 6 or 12 in Figure 1) is simply left empty.

These microbenchmarks create EM emanations as shown

	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
NOI		No instruction

Fig. 2. x86 instructions for our X_1/X_2 ESE measurements.

in Figure 4. The spectrum was measured using a spectrum

Fig. 3. Measurement setup.

analyzer (Agilent MXA N9020A) and a magnetic loop antenna (AOR LA400) as shown in Figure 3. This antenna does not use a tuning capacitor and is terminated with a 50Ω load, so it has a flat frequency response between 10 kHz and 1 MHz. Unless otherwise noted, the measurements used an X_1/X_2 alternation frequency of 80 kHz (note that this frequency can be arbitrarily changed) and a measurement distance of 10 cm. A resolution bandwidth of 1 Hz was used to minimize noise. Intuitively we expect differences between the X_1 and X_2 instructions to appear at the frequency $f_{\text{alt}} = 1/T_{\text{alt}}$ where T_{alt} is the time required to execute one iteration of the outer loop in Figure 1.

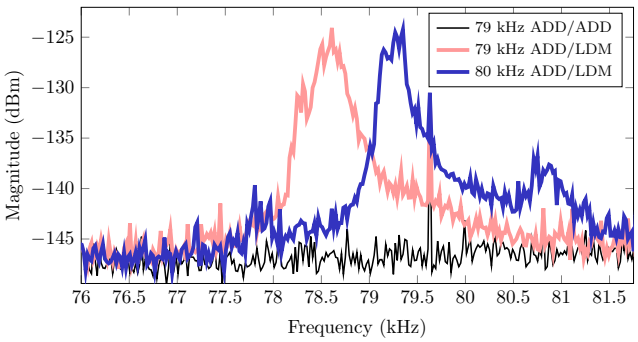

Fig. 4. Power spectrum of ADD/LDM instruction pair at 79 kHz and 80 kHz.

III. Mathematical Relationship Between ESE of Individual Instructions and The Measured Pairwise Side-channel Signal Power

Establishing mathematical relationship between ESE of individual instructions and the measured pairwise side-channel signal power available to an attacker [22] can not only help programmers and computer hardware designers to anticipate the vulnerability of the system, but can also help us relate codeword probabilities with energy levels of particular codewords, i.e., instructions.

To achieve this goal, we define a voltage signal corresponding to one sequence of instructions as $s_1(t)$ and a voltage signal corresponding to another sequence of instructions as $s_2(t)$. Then, the total EM side-channel energy available to the attacker can be defined as

$$\mathcal{P}_{\text{A}}(s_1(t), s_2(t)) \equiv \frac{1}{R}\int_0^{T_s} (s_1(t) - s_2(t))^2 dt \qquad (1)$$

where $s_1(t)$ and $s_2(t)$ are voltages measured across a resistance R (typically instruments have $R = 50\Omega$), and $t = (0, T_s)$ is the time interval over which the difference in program execution occurs. This difference can be an if-then-else statement that causes one path or the other to be executed, a memory access that either does or does not suffer a cache miss, etc. Note that $\mathcal{P}_{\text{A}}$ represents the overall emitted power during the execution of the microbenchmark including many repetition of instructions X_1 and X_2. However, ESE is defined as the available energy for an attacker when two instructions are executed only one time by filtering any other EM signals and noise present in measurements.

Assume $x_1(t)$ and $x_2(t)$ is the characteristic signals belong to execution of instructions X_1 and X_2. Then, ESE is defined as

$$\text{ESE}(X_1, X_2) \equiv \frac{1}{R}\int_0^{T_x} \left(\mathbb{E}[x_1(t) - x_2(t)]\right)^2 dt \qquad (2)$$

where T_x is the maximum of the execution times of the instructions and $\mathbb{E}$ is the expectation operation.

To make our derivations mathematically traceable, we introduce following notations and assumptions.

1) The for-loops of the microbenchmark given in Fig. 1 generate $s_1(t)$ and $s_2(t)$ respectively. Because the emitted signal from each loop depends on the type of instruction used in the loop, to discard the ambiguities regarding the inserted instruction and the taken branch, we denote produced signals as $s_i^{(X_i)}(t)$ which is generated by the execution of the i^{th} inner-for-loop and X_i represents any instruction inserted into that loop.
2) $s_1^{(X_1)}(t)$ and $s_2^{(X_2)}(t)$ are voltages sampled at frequency $1/T_{\text{I}}$ to create the sequences $s_1^{(X_1)}[n]$ and $s_2^{(X_2)}[n]$ of length $N_s = T_{\text{s}}/T_{\text{I}}$.
3) The frequency content of $s_1^{(X_1)}(t)$ and $s_2^{(X_2)}(t)$ above $1/(2T_{\text{I}})$ is negligible (i.e. $s_1^{(X_1)}(t)$ and $s_2^{(X_2)}(t)$ have bandwidth $1/(2T_{\text{I}})$).
4) $s_1^{(X_1)}(t)$ and $s_2^{(X_2)}(t)$ are voltages measured across a resistance R.

5) The discrete energy available to the attacker $\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_2)}\right]$ is then defined as

$$\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_2)}\right] \equiv T_{\mathrm{I}} \sum_{l=0}^{N_s-1} \frac{\left(s_1^{(X_1)}[l] - s_2^{(X_2)}[l]\right)^2}{R}. \tag{3}$$

6) Sampled voltages can be represented as $s_1^{(X_1)}[l] = \mu_1[l] + w_1[l]$ and $s_2^{(X_2)}[l] = \mu_2[l] + w_2[l]$ where $\mu_1[l]$ and $\mu_2[l]$ are the mean voltage values, $w_1[l]$ and $w_2[l]$ are the additive noises which are i.i.d. $\mathcal{N}(0, \sigma_l^2)$. Here, σ_l^2, $\mu_1[l]$, and $\mu_2[l]$ are depended on the instruction. For example, if the sample is taken during the execution of the first-inner-loop and the embedded instruction is X_1, the sample can be written as

$$s_1[l] = \mu_1[l] + w_1[l] = X_1^v + w_1[l] \tag{4}$$

where $w_1[l] \sim \mathcal{N}(0, \sigma_{X_1}^2)$ and X_1^v is the average power instruction X_1 emits. We assume that additive noise describes all the variation in the signal and that noise power is dependent on the executed instruction since the electromagnetic emissions can vary according to the execution location.

7) Finally, the discrete ESE is defined as

$$\mathrm{ESE}\left[\mathrm{X}_1, \mathrm{X}_2\right] \equiv \frac{T_{\mathrm{I}}}{R} n_X \left(X_1^v - X_2^v\right)^2. \tag{5}$$

where $n_X = T_x/T_{\mathrm{I}}$ is the number of samples taken during only the execution of the instructions.

The micro-benchmark described in Section II creates an alternation signal at frequency f_{alt} by repeatedly executing instruction X_1 n_{inst} times, followed by n_{inst} executions of instruction X_2. We then measure $P(f_{\mathrm{alt}})$, the spectral power at frequency f_{alt}. Finally, $\mathrm{ESE}[\mathrm{X}_1, \mathrm{X}_2]$ can be calculated from the spectral power $P(f_{\mathrm{alt}})$ observed at f_{alt} (while running the X_1/X_2 alternation microbenchmark) as follows:

$$\mathrm{ESE}[\mathrm{X}_1, \mathrm{X}_2] \approx \left(\frac{\pi}{2}\right)^2 \frac{P(f_{\mathrm{alt}}) \cdot N}{n_X \cdot n_{\mathrm{inst}} \cdot f_{\mathrm{alt}}} + C(X_1, X_2), \tag{6}$$

where N is the number of samples taken during only one inner for-loop, n_X is the maximum of the number of samples taken during the execution of instructions X_1 and X_2 and $C(X_1, X_2)$ is a constant term that depends on the instruction pair, i.e.,

$$C(X_1, X_2) = -\frac{\mathcal{P}_{\mathrm{A}}\left(\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)}\right) + \mathcal{P}_{\mathrm{A}}\left(\mathrm{s}_1^{(\mathrm{X}_2)}, \mathrm{s}_2^{(\mathrm{X}_2)}\right)}{2 n_X n_{\mathrm{inst}}}. \tag{7}$$

The proof of the relationship in (6) is shown in Appendix I. Note that although expectation for $\mathcal{P}_{\mathrm{A}}$ value to be same when the same instruction is inserted into the for-loops given in Fig. 8, because of the additive noise given in (4), the observed ESE through the execution of the code results in a non-zero value.

IV. New Method for Evaluation of EM Side/Covert Channel Capacity Created by the Execution of Instructions in a Processor

In a covert/side channel, identifying the channel capacity is of some interest to attackers who desire to maximize the rate at which information is obtained, but it is of paramount interest to defensive actors (programmers and hardware designers) to gain an insight about how vulnerable a program or a computer system is to an attack and quantify how a potential design change would affect this vulnerability. Thus, in this section, we propose a new method to quantify the amount of information that can be obtained through unintended EM emanations of instructions in a computer system.

A. Quantifying the Side Channel Leakage

To quantify the leakage capacity through the execution of instruction sequences, we consider each instruction as an independent codeword, and the covert/side channel "transmission" consists of a sequence of codewords that corresponds to the sequence of instructions actually executed by the program. Note that some hardware events can significantly affect the duration and the emanated signal of specific instructions. We account for this by treating these as distinct instructions. For example, a load (read memory) instruction behaves very differently when it finds the desired value in the first-level cache, in the second level cache, or in main memory, which we view as three separate instructions: LDL1, LDL2, and LDM, respectively.

Fig. 5 illustrates the noisy channel model for a covert/side channel where the input codewords are instructions, P_i represents the probability of the i^{th} instruction occurrence and p_{ij} is the transition probability that given i^{th} instruction is used in the code but detected as the j^{th} instruction based on EM emanation observations.

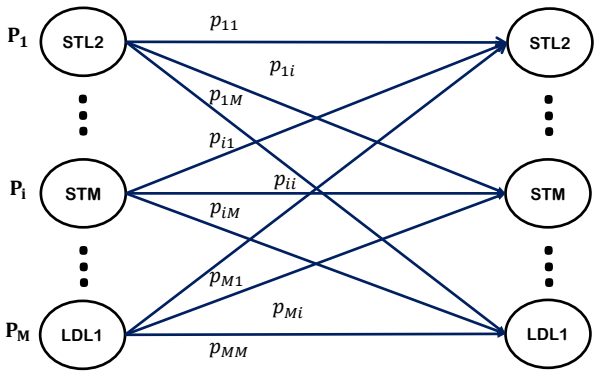

Fig. 5. Noisy channel model for covert/side channel.

There are several unique properties of this EM side/covert channel that require new methods for evaluating side/covert channel capacity.

First, all codewords are typically equally probable in traditional communication systems, but this is not true for the channel analyzed in this paper. Since our codewords are processor

instructions, we need to note that different instructions have different probability of occurrence in a program. For example, a typical program executes on-chip instruction (e.g. arithmetic operations) more often than memory accesses. Furthermore, processor designers put extra-efforts to ensure that among memory accesses the number of LDL1 executions is larger than the number of LDL2 occurrences and much larger than the number of LDM occurrences. Hence, it is realistic to expect that our codewords (i.e. instructions) do not have equal probability of occurrence.

Second, codewords in this channel do not have equal length. For example, it takes more energy and time to execute LDM in comparison to the time to execute ADD instruction.

Finally, this channel is inherently noisy due to variations in activity in the processor (i.e. different data values for a given instruction) and due to noise created by other electronic components near the processor.

All this implies that the Shannon-based channel capacity [34], [32], [35] overestimates the available capacity in the side/covert channel that transmits instructions as codewords. Therefore, we need to compute channel capacity in a way that considers the distinct instruction lengths, variable instruction probabilities, and maximum mutual information at the same time.

If we assume that we have a finite-length sequence where codewords can have different lengths, the goal is to invoke codewords into the sequence such that total information gathered from the sequence is maximized. Therefore, the length of the input and the entropy of this input are the issues critical to consider. Hence, we need to find a distribution for the input set such that total information achievable from the sequence is maximized, i.e., the information per sample is optimized. To achieve that, we propose the following optimization problem:

$$\underline{\textit{Setting I :}} \tag{8}$$

$$\text{maximize} \quad \frac{\sum\limits_{i,j} P_i p_{ij} \log\left(\frac{p_{ij}}{\sum\limits_{k} P_k p_{kj}}\right)}{\sum\limits_{i} P_i L_i}$$

$$\text{subject to}$$

$$\sum_{i} P_i = 1$$

$$P_i \geq 0 \text{ for } \forall i, j \in \{1, \cdots, K\}\,.$$

where K is the number of instructions in the input set. Through the optimization process, the variables that maximize the problem are $\{P_i \mid i \in \{1, \cdots, K\}\}$. These variables are the probabilities or frequencies of each instruction such that maximum information leakage occurs and we assume that occurrences of instructions are independent of each other. Transition probabilities p_{ij} are calculated based on ESE measurements as described in Section IV-B. The optimization problem is solved using gradient descent approach detailed in Appendix IV.

Note that the optimization problem given above simplifies to Shannon's channel capacity if the length of each instruction is the same and all instructions are equally probable.

B. A Practical Calculation of Transition Probabilities in EM Side/Covert Channel

In this section, we propose a method for finding transition probabilities defined in (8) using the measured pairwise side-channel signal power.

We start the process by obtaining ESE values as illustrated in Table I. Each entry in this table is the ESE between the X_1 instruction (row) and X_2 instruction (column) computed using (6). The pairwise side-channel signal power needed to compute ESE is measured using method described in Section II. We measure a signal power of each instruction pair in the table 10 times over multiple days and take the mean to minimize the impact of changes in radio signal interference, room temperature, and slight differences in antenna position. For each pair of instructions, X_1 and X_2, we run the X_1/X_2 micro-benchmark and measure the power spectral density from 2.5 kHz above to 2.5 kHz below the alternation frequency. Then we integrate over this band to get the total power $P(f_{\text{alt}})$ generated by the difference between X_1 and X_2. Finally the $\text{ESE}[\text{X}_1, \text{X}_2]$ is calculated using (6).

The benchmarks were run as single-threaded Windows 7 32-bit user mode console applications on the Intel Core i7 (L1 Data Cache: 64 KB, 2 way L2 Cache: 1024 KB, 16 way). No other user-mode applications were active and wireless devices were disabled to minimize interference with the intentionally generated signals. Aside from this, the system was operating normally.

As an example, the ESE values for an Intel Core i7 laptop are given in Table I.

TABLE I

ESE VALUES (IN ZJ) FOR THE CORE I7 LAPTOP.

	LDM	**STM**	**LDL2**	**STL2**	**LDL1**	**STL1**	**ADD**	**SUB**	**MUL**	**DIV**
LDM	0	128	377	1344	237	246	283	279	247	1392
STM	53	0	944	1389	114	163	153	150	164	1213
LDL2	336	1394	0	24	84	112	149	143	109	736
STL2	1160	1248	21	0	42	56	99	100	114	536
LDL1	463	236	190	76	0	0	0	0	0	169
STL1	464	274	219	97	4	0	0	0	0	135
ADD	509	255	256	136	21	11	0	3	0	107
SUB	487	244	207	145	11	15	5	0	4	131
MUL	487	252	221	143	8	13	3	15	0	116
DIV	1188	812	651	293	82	90	92	93	134	0

1) Estimating the Voltage of a Specific Instruction: In this part, we provide the steps to estimate the voltage of an instruction. Please note that ESE can be viewed as a metric that measures the Euclidean distance between voltages produced by the execution of the instructions. Here, we introduce our approach to obtain these distances. As the first step, we are required to estimate the voltages generated by the instructions. However, since ESE is considered as an Euclidean distance and only provides the voltage differences, i.e. the distances in generated voltages among instructions, rather than generated actual voltage values of each instruction, we are required to obtain the distances among the instructions. In that respect,

since voltage is a one-dimensional quantity, i.e. the EM emanations caused by different instructions differ only in magnitude, we first focus on the locations of these instructions with respect to each other. Therefore, we are only interested in how instructions are ordered on a line with respect to the quantity of leakage they produce. We also need to emphasize that sorting the instructions can be from the instructions which cause more leakages to the ones which generate less leakages, or the other way around. Let us first consider an example of ESE shown in Table I. From the table, we can deduce that

* LDL1, STL1, ADD, MUL, and SUB have similar ESE values and, therefore, we consider them as one instruction which is called as G_6.
* Spacing between DIV and LDM is the largest (e.g., 1392). Therefore, they must be positioned at the corners, and the rest of the instructions lie between these two instructions.
* STM is the closest to LDM since LDM produces the smallest ESE value with STM among all instructions (e.g., 128).
* The closest instruction to LDL2 is STL2 since it produces the smallest EM leakage with LDL2.
* G_6 generates smaller ESE value with LDM and STM than STL2 and LDL2. In the same manner, ESE produced by the execution of LDM and LDL2 is larger than the one produced by LDM and G_6. Therefore, G_6 is positioned between STM and LDL2.

Based on the observations above, the order of instructions is provided in Fig. 6 and we denote this ordered sequence as $\mathbb{S}$.

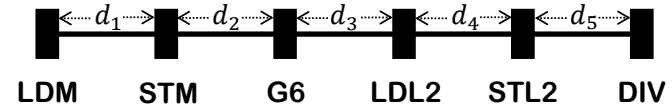

Fig. 6. An example of instruction ordering based on a measurements in Table I.

Here we note that the order of instructions does not give any intuition about the voltage values of instructions but clarifies the distances among the instructions. We do not give any importance to exact voltage values of each instruction due to the definition of ESE. To calculate the transition probabilities, we only need the distances between each instruction. Therefore, we set the following optimization problem to determine the distances when our input set includes {DIV, MUL, STM, LDM, LDL2, STL2} as follows:

$$\begin{aligned} \underline{\textit{Setting II :}} \quad & \underset{\mathbf{d},\epsilon}{\text{minimize}} && \|\epsilon\|_2 \\ & \text{subject to} \\ & \text{ESE}[\text{LDM},\text{STM}] - \kappa d_1^2 && = \epsilon_1 \\ & \text{ESE}[\text{LDM},\text{MUL}] - \kappa (d_1+d_2)^2 && = \epsilon_2 \\ & \qquad \vdots \\ & \text{ESE}[\text{STL2},\text{DIV}] - \kappa d_5^2 && = \epsilon_{15} \end{aligned}$$

where $\mathbf{d}(\mathbb{S}) = [d_1 \;\; d_2 \;\; d_3 \;\; d_4 \;\; d_5]^T$ is the distance vector required to be revealed, $\epsilon = [\epsilon_1 \;\; \epsilon_2 \;\; \cdots \;\; \epsilon_{15}]^T$ is the estimation error vector needed to be minimized and $\kappa = T_\text{I}/R$. We also note that to verify the given order of instructions is the optimum one, we run *Setting II* for different orders and checked the objective value $\|\epsilon\|_2$ is smallest for the order given in Fig. 6 among all of the orders of the instructions.

2) Calculation of Transition Probabilities: After obtaining the relative voltage differences, the next step is to determine the decision boundaries among instructions. Since our inputs lie on the same axis and only ordering of instructions and distances among them are important for transition probability calculations, we assume the mean of the instruction with the leftmost position in the ordered instruction sequence is zero. Then, means of the remaining instructions are calculated based on the obtained distances. Finally, the decision boundaries are calculated using maximum likelihood estimation between two consecutive instructions. For example, if instruction X_1 and X_2 are neighbors, the decision boundary is the point such that

$$\mathcal{N}(x|\mu_{X_1}, \bar{\sigma}^2_{X_1}) = \mathcal{N}(x|\mu_{X_2}, \bar{\sigma}^2_{X_2}) \tag{9}$$

where $\mathcal{N}(\bullet)$ is the normal pdf distribution, $\mu_{X_i} = X_i^v$ is the mean (position) of the i^{th} instruction and

$$\bar{\sigma}^2_{X_i} = \frac{\mathcal{P}_\text{A}\left(\text{s}_1^{(\text{X}_\text{i})}, \text{s}_2^{(\text{X}_\text{i})}\right) - \mathcal{P}_\text{A}\left(\text{s}_1^{(\text{NOI})}, \text{s}_2^{(\text{NOI})}\right)}{\kappa}. \tag{10}$$

The derivation of $\bar{\sigma}^2_{X_i}$ is given in Appendix II. $\bar{\sigma}^2_{X_i}$ characterizes the additive noise and impact of other program and hardware variabilities present in measurements.

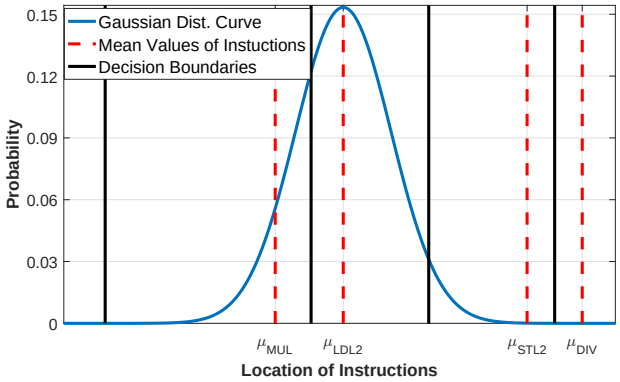

Fig. 7. An illustration of the process for calculation transition probabilities for a given instruction.

Thereafter, this Gaussian distribution is fit to the graph whose mean equals to the instruction X's mean, and whose standard deviation is $\bar{\sigma}^2_X$. Here we note that scaling mean and the variance of the distances and variances will not change probability distributions, hence we can simplify the notations by defining $\hat{\sigma}^2_X = \kappa \bar{\sigma}^2_X$ and $\hat{\mathbf{d}}(\mathbb{S}) = \sqrt{\kappa}[d_1 \;\; d_2 \;\; d_3 \;\; d_4 \;\; d_5]^T$.

Finally, the probability of each interval is assigned to each corresponding instruction as a conditional probability. An illustration of the process is given in Fig. 7.

V. Experimental Results and Discussions

In the previous section, we have introduced a method to compute side/covert channel capacity from noisy measurements of ESE. Here we note that the proposed algorithm is general and can be applied to any computational device with any set of instructions. In this section, we give a few examples explaining how to calculate the leakage capacity in (8).

Before presenting experimental results, we first summarize the steps for the proposed approach in this paper:

- To measure the ESE, first install the code given in Fig. 1 and measure the power at the frequency $f_{\rm alt}$ which will be depended on $n_{\rm inst}$.
- By exploiting the equation in (6), calculate the alternation energy of every two combinations of instructions.
- Apply *Setting II* as explained in Section IV-B.1 to estimate the emitted EM voltages by the execution of any instructions.
- Estimate the noise power w.r.t. an instruction based on (10), and then, calculate the decision boundaries to obtain the cross-transition probabilities of the proposed side/covert channel as described in Section IV-B.2.
- Obtain the probability distribution of instruction occurrences which maximizes mutual information rate of the modeled side/covert channel by exploiting *Setting I*.
- Use (8) to compute capacity.

A. Experimental Results of Core I7 Laptop

As the first example, we consider the ESE values in Table I. ESE values are from a Core i7 laptop with the 10 cm distance and the intended alternation frequency 80 kHz. We constrain the input set to $\{\mathrm{LDM}, \mathrm{STM}, \mathrm{DIV}, \mathrm{MUL}, \mathrm{LDL2}, \mathrm{STL2}\}$, using MUL as a representative of the "G6" group of instructions $\{\mathrm{LDL1}, \mathrm{STL1}, \mathrm{ADD}, \mathrm{SUB}, \mathrm{MUL}\}$.

TABLE II
Transition probabilities based on ESE measurement in Fig. I

	LDM	**STM**	**MUL**	**LDL2**	**STL2**	**DIV**
LDM	0.53	0.38	0.09	0	0	0
STM	0.49	0.26	0.19	0.04	0.01	0.01
MUL	0	0	0.94	0.06	0	0
LDL2	0	0	0.11	0.85	0.04	0
STL2	0	0	0	0.06	0.5	0.44
DIV	0	0	0	0	0.39	0.61

As the first step, cross-transition probabilities must be obtained. Therefore, we require to find the distances among the instructions with respect to emitted EM signal powers. The distances are acquired by feeding *Setting II* with Table I. The order of instruction is as given in Section IV-B.1. The resulting distance vector $\hat{\mathbf{d}}$, which contains the interval lengths between two neighbouring instructions in the magnitude space, is $\hat{\mathbf{d}}(\mathbb{S}) = [1 \;\; 18.8 \;\; 5.9 \;\; 9.5 \;\; 1]$.

The next step is to calculate the decision boundaries and noise deviations for each instruction. Noise powers for each instruction are calculated based on (10) such that $\overline{\sigma}^2(\mathbb{S}) = [7.87 \;\; 13.64 \;\; 1.47 \;\; 2.6 \;\; 3.21 \;\; 6.38]$. Since both noise deviations and distances in emanated powers are in hand, cross-transition probabilities can be calculated by following instructions given in Section IV-B.2. The resulting transition probabilities for the Core i7 laptop are provided in Table II. Once transition probabilities are computed, it is possible to find the maximum channel rate and achievable capacity. The Intel Core i7 processor is very complex, featuring out of order execution and many other microarchitectural optimizations. These optimizations often result in multiple instructions being executed in parallel, which results in reduced instruction length in our simplified model. Therefore, we define the length of each instruction as the excess amount of time caused by appending the same instruction to both for-loops given in Fig. 1 with respective to inserting nothing. For example, if T_{NOI} and T_X are the times, which takes to execute the microbenchmark in Fig. 1 when nothing is inserted and instruction X is inserted respectively, the length of the instruction X is

$$L_X = \frac{T_X - T_{NOI}}{2n_{\rm inst}}. \tag{11}$$

Then, the instruction lengths (execution times) are normalized and quantized such that the smallest length is equivalent to one and the lengths of all instructions are integers – this is reasonable because processor activity is in terms of clock cycles, with many instructions taking only one cycle to execute while several instructions take a number of cycles. Therefore, we consider the length vector for *Setting I* as $\mathbf{L}(\mathbb{S}) = [13 \;\; 20 \;\; 1 \;\; 1 \;\; 3 \;\; 8]^T$. The capacity rate is attained as **0.72 Bits/Quantum**, where quantum is defined as average instruction length per symbol. The resulting instruction probabilities are $\mathbf{P}_{\mathbb{S}} = [0.005 \;\; 0 \;\; 0.377 \;\; 0.354 \;\; 0.264 \;\; 0]^T$. The results are reasonable since the lengths of LDM, STM and DIV are longer and the deviations of these instructions are high, which increases uncertainty about the outputs of the instructions. On the other hand, although the entropy of STL2 is large, its probability is the largest because its uncertainty is provoked due to existence of DIV. Since the optimal probabilities of occurrence of DIV is zero and the length of STL2 is small, the uncertainty about STL2 is largely eliminated and having a smaller execution time makes it more favourable.

B. Experimental Results of Core 2 Laptop

In this section, we will provide the experimental results on an Intel Core 2 Duo laptop with 1.8GHz CPU clock, 333MHz DDR2 memory, 32KB 8 way L1 Data Cache and 4096 KB 16 way L2 cache. The considered instruction set is {LDM, LDL2, STL2, LDL1, STL1, ADD, SUB, MUL, DIV}. As the first step, we perform experiments to calculate the ESE power of instruction pairs at 10cm and 80kHz. The measured ESE values are given in Table III.

We observe that the ESE pattern observed in the Core I7 laptop also exists for the ESE values of the Core 2 Duo laptop. For example, all arithmetical instructions except DIV emit similar leakages whereas DIV behaves like a completely

different functional instruction. Therefore, for the leakage capacity calculation, one of these will be selected. Another observation is that the maximum leakage occurs when STM and DIV are embedded into the microbenchmark. Moreover, the tableau is almost symmetrical which supports our earlier observation that the instruction order in the microbenchmark does not affect the EM leakage.

TABLE III

ESE VALUES (IN ZJ) FOR THE CORE 2 DUO LAPTOP.

	STM	LDL2	STL2	LDL1	STL1	ADD	SUB	MUL	DIV
STM	0	61	100	19	25	21	21	21	56
LDL2	62	0	1	79	80	84	82	81	101
STL2	101	1	0	101	104	108	105	105	160
LDL1	19	70	103	0	0	0	0	0	2
STL1	26	79	104	0	0	0	0	0	2
ADD	21	84	109	0	0	0	0	0	1
SUB	20	82	107	0	0	0	0	0	1
MUL	21	83	108	0	0	0	0	0	1
DIV	57	99	161	2	2	1	1	1	0

To be able to calculate the leakage capacity through *Setting I*, we order the instructions with respect to their emitted voltage potentials. The sorted sequence of instructions which minimizes the objective function of *Setting II* is obtained as $\mathbb{S} = \{\mathrm{STL2}, \mathrm{LDL2}, \mathrm{STM}, \mathrm{ADD}, \mathrm{DIV}\}$ where ADD is chosen to represent the algorithmic operation set. The distances between the emitted voltages are $\hat{\mathbf{d}}(\mathbb{S}) = [2.1\ 7.3\ 1.45\ 1.85]$.

The next step is to obtain the cross-transition probabilities of the proposed model. In that respect, the noise powers conditioned on the execution locations of instructions are calculated as $\bar{\sigma}^2(\mathbb{S}) = [0.23\ 0.23\ 17.64\ 0.23\ 0.23]$ by the equation given in (10). Since we have both the noise powers and distances among the emitted powers resulting from the execution of instructions, it is time to calculate the cross translation probabilities of the model as described in Section IV-B.2. The transition probabilities are given in Table IV. We observe that the transition probability matrix is diagonally dominated expect the instruction STM as in the case of the Core I7 laptop.

TABLE IV

TRANSITION PROBABILITIES BASED ON ESE MEASUREMENT IN TABLE III

	STL2	LDL2	STM	ADD	DIV
STL2	0.99	0.01	0	0	0
LDL2	0.01	0.98	0.01	0	0
STM	0.02	0.05	0.49	0.15	0.29
ADD	0	0	0.07	0.9	0.03
DIV	0	0	0	0.03	0.97

For this processor, we obtain the relative instruction times as $\mathbf{L}(\mathbb{S}) = [3\ 1\ 31\ 1\ 8]$ by employing (11). We attain the capacity rate as **1.09 Bits/Qauntum** with the instruction occurrence probabilities $\mathbf{P}_{\mathbb{S}} = [0.09\ 0.44\ 0\ 0.47\ 0]$. We can observe again that the probabilities of longer instructions in terms of execution times are set to zero although these instructions have larger ESE values. Since execution times of these instructions are longer, the information, they provide at each quantum interval, is much smaller than the instruction with shorter execution times.

C. Experimental Results for Turion X2 Laptop

In this section, we provide the experimental results for AMD Turion X2 proccessor with 64 KB, 2 way L1 Data Cache and 1024 KB, 16 way L2 Cache. As usual, the first step is to measure the ESE values of instruction pairs which are given in Table V.

TABLE V

ESE VALUES (IN ZJ) FOR THE AMD TURION X2 LAPTOP.

	STM	LDL2	STL2	LDL1	STL1	ADD	SUB	MUL	DIV
STM	0	16	26	5	8	5	5	2	18
LDL2	14	0	0	31	33	34	33	34	53
STL2	30	0	0	40	40	42	41	42	70
LDL1	4	32	40	0	0	0	0	0	1
STL1	4	32	40	0	0	0	0	0	0
ADD	6	35	43	0	0	0	0	0	0
SUB	5	34	41	0	0	0	0	0	0
MUL	6	34	42	0	0	0	0	0	0
DIV	25	54	71	1	0	0	0	0	0

The order which minimizes the objective function of *Setting II* is same as for the Core 2 Duo laptop. As the side product of *Setting II*, the distance vector providing the distances between the neighbouring instruction voltages is $\hat{\mathbf{d}}(\mathbb{S}) = [1.01\ 3.74\ 1.97\ 1.74]$. The next step is to calculate the cross-transition probabilities of instructions. Thus, we calculate noise powers conditioned on instructions as $\bar{\sigma}^2(\mathbb{S}) = [0.4\ 0.34\ 24.22\ 0.34\ 0.34]$. By following the steps given in Section IV-B.2, we calculated the cross transition probabilities which are given in Table VI. The final step is to obtain the optimal solution for *Setting I*. Relative instruction execution times are $\mathbf{L}(\mathbb{S}) = [3\ 1\ 30\ 1\ 8]$. The rate is attained as **0.97 Bits/Quantum** with instruction occurrence probabilities $\mathbf{P}_{\mathbb{S}} = [0\ 0.498\ 0\ 0.502\ 0]$. If we compare the results with the Core 2 Duo laptop, this time we observe STL2 is set to zero. The reason can be justified as follows: We observe that the existence of STL2 and LDL2 creates more uncertainty, therefore, instead of keeping these two instructions in the codebook, removing one of these instructions increases the reliability of the communication channel. Since the execution time of STL2 is longer, forcing its occurrence probability to be zero enhances the overall rate obtained through *Setting I*.

TABLE VI

TRANSITION PROBABILITIES BASED ON ESE MEASUREMENT IN TABLE III

	STL2	LDL2	STM	ADD	DIV
STL2	0.79	0.21	0	0	0
LDL2	0.19	0.79	0.02	0	0
STM	0.19	0.11	0.27	0.14	0.28
ADD	0	0	0.04	0.89	0.07
DIV	0	0	0	0.07	0.93

D. Experimental Results for NIOS Processor on the DEI FPGA

As our last example, we use the ESE values from [36]. The measurements are done 10 cm above the NIOS processor

on the DE1 FPGA board. Assuming that an instruction could not have smaller magnitude than NOI, i.e. $\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X})}, \mathrm{s}_2^{(\mathrm{X})}\right] > \mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{NOI})}, \mathrm{s}_2^{(\mathrm{NOI})}\right]$ where X is any instruction rather than NOI, we assume that minimum ESE measurement with an instruction is 0.015 zJ. The main difference between FPGA and Core i7 measurements is that arithmetic operations and LDL1 are distinguishable in the FPGA case. This time, the order of instructions is $\mathbb{S}$ ={LDM-DIV-LDL1-MUL-ADD-SUB}. As the first step, we find the interval lengths among instructions which is $\hat{\mathbf{d}}(\mathbb{S}) = [2.29\ \ 0.39\ \ 0.27\ \ 3.39\ \ 0.91]$. Furthermore, noise power is calculated as $\overline{\sigma}^2(\mathbb{S}) = [0.008\ \ 0.0075\ \ 0.0075\ \ 0.0075\ \ 0.0075\ \ 0.0075]$. Then, the transition probabilities for FPGA are calculated and given in Fig. VII. For the *Setting I*, we assume $\mathbf{L}(\mathbb{S}) = [7\ \ 5\ \ 4\ \ 4\ \ 1\ \ 1]^T$ and obtain **1.14 Bits/Quantum** channel capacity rate where the probabilities of instructions are assigned as $\mathbf{P}_{\mathbb{S}} = [0.004\ \ 0.018\ \ 0.032\ \ 0.035\ \ 0.455\ \ 0.456]^T$. Observe that although the entropy of LDM and DIV is similar to ADD and SUB, assigned probabilities are much higher for ADD and SUB because their lengths are much smaller than other instructions, therefore, they can transmit more information in a second. Likewise, the optimum occurrence probabilities also explain which instructions create more vulnerability when an alternation created with them. For example, if a script generates an alternation due to ADD with another instruction, the leakage caused by this alternation will be huge presumably.

TABLE VII
TRANSITION PROBABILITIES BASED ON ESE MEASUREMENT IN [36]

	LDM	DIV	LDL1	MUL	ADD	SUB
LDM	1	0	0	0	0	0
DIV	0	0.99	0.01	0	0	0
LDL1	0	0.01	0.93	0.06	0	0
MUL	0	0	0.06	0.94	0	0
ADD	0	0	0	0	1	0
SUB	0	0	0	0	0	1

Since a modern processor executes several billion instructions per second, the computed EM side/covert channel capacity of 1.14 Bits/Quantum implies that the attacker might obtain several gigabytes of information per second. Although this is an extremely high information leakage rate, the rate actually achieved by practically demonstrated side channel attacks on cryptographic implementations is much lower. This apparent discrepancy is primarily caused by different assumptions about how the program is designed and different definitions of what constitutes information. Our capacity derivations are for the worst-case scenario where the program is specifically designed to leak information, whereas cryptographic implementations are designed to have significant resilience to side channel attacks. Furthermore, cryptographic attacks only consider the rate of leakage for encryption keys, whereas our capacity derivations account for any information about program execution.

E. Effect of Alternation Time T_{alt} on ESE

In this section, the effect of measurement frequencies on leakage capacity is investigated. The measurements are done at the side of the NIOS processor on the DE1 FPGA board with 10 cm distance. For each frequency, we follow the procedure given in the beginning of Section V. Occurrence probabilities of instructions are provided in Table VIII. Furthermore, $\mathbf{R}(\mathbb{S}) = [1.12\ \ 1.14\ \ 1.16\ \ 1.14]$ Bits/Quantum is the vector containing resulting rates of the setups with varying frequencies where $\mathbb{S}$ = [40 kHz, 50 kHz, 70 kHz, 80 kHz]. These results indicate that as long as the same setup is considered but different values of T_{alt} are used, the change on the capacity and probability distribution of instructions are minimal.

TABLE VIII
OCCURRENCE PROBABILITIES OF INSTRUCTIONS FROM MEASUREMENTS COLLECTED AT DIFFERENT FREQUENCIES

	LDM	DIV	LDL1	MUL	ADD	SUB
40 kHz	0.005	0.019	0.024	0.034	0.459	0.459
50 kHz	0.004	0.017	0.035	0.036	0.454	0.454
70 kHz	0.004	0.018	0.041	0.041	0.447	0.449
80 kHz	0.004	0.018	0.032	0.035	0.455	0.456

F. Justification of the Proposed Model

In this part, we compare our results with classical Shannon capacity and provide the underlying reasons why the proposed methodology is needed. Table IX provides the capacities for different platforms. Note that Shannon capacity presumes that the codewords have the same length which means the number of channel uses by each codeword is same.

In Table IX, PS, SC and USC represent the capacity results obtained by the proposed framework, by assuming equal codeword length and exploiting Shannon's capacity, and by Shannon's capacity divided by the average code-length calculated via the occurrence probabilities which are utilized to obtain Shannon's capacity. These results clarify why the proposed method is more informative in terms of leakage capacity.

TABLE IX
CAPACITY COMPARISON WITH CLASSICAL SHANNON'S CAPACITY

	AMD Turion	Core 2 Duo	Core I7	NIOS
PS (Bits/Quantum)	0.97	1.09	0.72	1.14
SC (Bits/Symbol)	1.46	1.86	1.64	2.46
USC (Bits/Quatum)	0.42	0.48	0.27	0.68

Executed instructions take different amount of time, therefore, the number of channel uses for each instruction varies which could not be reflected by the classical Shannon capacity. On the other hand, if we normalize the Shannon's capacity with the average channel uses, we obtain the information per channel uses. If we compare the results, which are USC and the proposed method, we observe that the proposed method increases the effectiveness of the channel per use.

Therefore, we can claim that the proposed method provides better information on the severity of the leakage.

VI. Potential Defense Mechanisms

Our technique can help defensive efforts for both covert and side-channels, although in somewhat different ways. For covert channels, the assumption is that the developer of the program is crafting the code so as to maximize the leakage. Since our technique allows potential leakage to be computed given the instruction frequencies (how often each type of instruction executes in the program), one defensive approach would be to determine the instruction mix (there are a number of program execution profiling tools that can do that), compute the potential leakage and use that to drive the decision about what to do (e.g. subject high-potential-leakage programs to additional scrutiny, "sandbox" them to limit their access to potentially sensitive information, etc.).

For side-channels, the assumption is that the developer is interested in reducing the leakage produced by the code they are developing. Our technique allows the programmer to quantify the potential leakage of various parts of the program (or the program as a whole), identify parts of the program whose leakage may be higher than tolerable, and then make sure that these parts of the program do not operate on sensitive data, change instruction mix in a way that reduces the potential leakage (e.g. reduce the use of high-ESE instructions, or even use our technique to quantify the reduction in leakage that would be obtained through each potential code change), or surgically apply one of the known techniques for reducing information leakage through side-channels.

In terms of insight that we can offer to programmers and hardware designers, our results indicate that most of the potential information leakage is a result of using a very small number of instructions that are much easier to correctly distinguish. For software designers, this means that a program's use of these instructions should not be dependent on sensitive data values. For hardware designers, this means that reduction of a hardware design's overall vulnerability to EM side channel attacks largely depends on addressing the EM side channel signals produced by this very small subset of the processor's overall instruction set.

The goal of this paper is to find out which instructions are the most promising to be used as codewords in a covert channel or to be checked for in side-channel. One may assume that instructions that have the strongest EM signatures (e.g. cache misses) are the best to be used as codewords for information transmission. However, as our analysis shows, that is not the case because cache misses also take the longest time to be executed, hence carrying much smaller Bits/Quantum information. The future work will be on how to design a codebook that consists of these codewords and achieves capacity limits derived in this paper. Some simple examples of an attack based on the selection of instructions are shown in [20] and [23], but more systematic approach is needed and is part of our future work.

VII. Conclusions

This paper offers an answer to how much information is "transmitted" by execution of particular sequence of instructions in a processor. We first propose a new method to estimate the maximum information leakage through EM signals generated by execution of instructions in the processor. Then, we derive a mathematical relationship between electromagnetic side-channel energy (ESE) of individual instructions and the measured pairwise side-channel signal power. Furthermore, we use this measure to calculate the transition probabilities needed for estimating capacity, and propose a new method to estimate side/covert channel capacity created by the execution of instructions in a processor. Finally, we illustrate how the proposed method works on several practical examples. The reason behind naming the proposed structure as covert/side channel can be claimed as follows: Covert channels are the channels can be exploited for a hidden communication. Therefore, by taking advantage of side channels generated by EM leakage, a covert channel communication system can be created. However, how to exploit this side channel as a communication system is still a question.

Appendix I
The Relationship between ESE and Measured Spectral Power of a Microbenchmark

In this section, we show that $\mathrm{ESE}\left[\mathrm{X}_1, \mathrm{X}_2\right]$ defined in (2) is related to the alternation power $P(f_{\mathrm{alt}})$ as in (6) where $s_m^{(X_m)}[n]$ denotes the sampled signal when instruction X_m is inserted into the m^{th} loop given in Fig. 1. The assumptions given in Section III will be considered through derivations.

We start by noting that the signals generated by the ESE benchmarks can be represented as a specific mixture of two *periodic* signals with period N. For $n = 0, ..., N-1$, the first signal obtained from the first iteration of the first inner loop is denoted as

$$\hat{s}_1^{(X_1)} = [o_1[0], o_1[1], \cdots, o_1[n_O - 1], x_1[0], x_1[1], \cdots, x_1[n_X - 1]]$$

such that $N = n_O + n_X$. Note that $\mathbb{E}\left[s_1^{(X_1)}[n+N]\right] = \mathbb{E}\left[s_1^{(X_1)}[n]\right]$ because $s_1^{(X_1)}[n]$ is periodic which leads that we also assume the additive noise are also periodic. Following the same procedure for the second inner loop, we have

$$\hat{s}_2^{(X_2)} = \left[o_2[0], o_2[1], \cdots, o_2[n_O - 1], x_2[0], x_2[1], \cdots, x_2[n_X - 1]\right].$$

We denote the sampled voltage at the time points where instruction X_1 is active as $x_1[i]$ and the sampled voltage at the time points where instruction X_2 is active as $x_2[i]$ where $i \in \{0, 1, \cdots, n_X - 1\}$. Similarly $o_m[n]$ represents the other instructions in the benchmark necessary to make the benchmark practical (e.g. to create a loop around instruction X_1 or instruction X_2) for the m^{th} loop.

First, we derive the ESE between two instructions from the measurement given in (3) as follows:

$$\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_2)}\right] \equiv T_{\mathrm{I}} \sum_{l=0}^{N_s-1} \frac{\left(s_1^{(X_1)}[l] - s_2^{(X_2)}[l]\right)^2}{R}$$

$$= \frac{T_{\mathrm{I}}}{R}\sum_{k=0}^{Nn_{\mathrm{inst}}-1}\left(s_1^{(X_1)}[k]-s_2^{(X_2)}[k]\right)^2$$
$$\approx \frac{\kappa}{2}\left[\sum_{k=0}^{N-1}\left(\hat{s}_1^{(X_1)}[k]-\hat{s}_2^{(X_2)}[k]\right)^2\right] \quad (12)$$

where $\kappa = (2T_{\mathrm{I}}n_{\mathrm{inst}})/(R)$ and (12) follows the periodicity of $s_1^{(X_1)}[n]$ and $s_2^{(X_2)}[n]$. Based on the assumption in Section III, we can write $x_1[i] = X_1^v + w_1^{(X_1)}[i]$, $x_2[i] = X_2^v + w_2^{(X_2)}[i]$ and $o_m[i] = O^v + w_m^{(O)}[i]$ where $w_1^{(X_1)}[i] \sim \mathcal{N}(0,\sigma_{X_1}^2)$, $w_2^{(X_2)}[i] \sim \mathcal{N}(0,\sigma_{X_2}^2)$, $w_m^{(O)}[i] \sim \mathcal{N}(0,\sigma_O^2)$ and $m \in \{1,2\}$. Therefore,

$$\begin{aligned}
&\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)},\mathrm{s}_2^{(\mathrm{X}_2)}\right]\\
\approx\ & \frac{\kappa}{2}\textstyle\sum_{k=0}^{n_X-1}\left(X_1^v - X_2^v + w_1^{(X_1)}[k] - w_2^{(X_2)}[k]\right)^2\\
&-\frac{\kappa}{2}\textstyle\sum_{k=0}^{n_O-1}\left(w_1^{(O)}[k]-w_2^{(O)}[k]\right)^2\\
=\ & \frac{\kappa n_X}{2}\frac{1}{n_X}\textstyle\sum_{k=0}^{n_X-1}\left(X_1^v - X_2^v + w_1^{(X_1)}[k] - w_2^{(X_2)}[k]\right)^2\\
&-\frac{\kappa n_O}{2}\frac{1}{n_O}\textstyle\sum_{k=0}^{n_O-1}\left(w_1^{(O)}[k]-w_2^{(O)}[k]\right)^2 \quad (13)
\end{aligned}$$

Assuming the number of samples taken during executions of instructions are large enough, sum operations given in (13) can be considered as the expectation operation. Therefore, we can claim

$$\begin{aligned}
&\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)},\mathrm{s}_2^{(\mathrm{X}_2)}\right]\\
\approx\ & \frac{\kappa n_X}{2}\mathbb{E}\left[\left(X_1^v - X_2^v + w_1^{(X_1)}(k) - w_2^{(X_2)}(k)\right)^2\right]\\
&-\frac{\kappa n_O}{2}\mathbb{E}\left[\left(w_1^{(O)}[k]-w_2^{(O)}[k]\right)^2\right]\\
=\ & \tfrac{\kappa}{2}\Big(n_X(X_1^v - X_2^v)^2 + n_X\sigma_{X_1}^2\\
&+n_X\sigma_{X_2}^2 + 2n_O\sigma_O^2\Big). \quad (14)
\end{aligned}$$

Furthermore, observe that (14) can be modified in terms of ESE as follows:

$$\begin{aligned}
\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)},\mathrm{s}_2^{(\mathrm{X}_2)}\right] &\equiv n_{\mathrm{inst}}\mathrm{ESE}[\mathrm{X}_1,\mathrm{X}_2]\\
&+\frac{1}{2}\Big(\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)},\mathrm{s}_2^{(\mathrm{X}_1)}\right]+\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_2)},\mathrm{s}_2^{(\mathrm{X}_2)}\right]\Big). \quad (15)
\end{aligned}$$

If we define

$$\hat{C}(X_1,X_2) = \mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)},\mathrm{s}_2^{(\mathrm{X}_1)}\right]+\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_2)},\mathrm{s}_2^{(\mathrm{X}_2)}\right], \quad (16)$$

ESE of two instructions in time domain can be written as

$$\mathrm{ESE}[\mathrm{X}_1,\mathrm{X}_2] = \frac{2\mathcal{P}_{\mathrm{A}}\left[\mathrm{s}_1^{(\mathrm{X}_1)},\mathrm{s}_2^{(\mathrm{X}_2)}\right]-\hat{C}(X_1,X_2)}{n_{\mathrm{inst}}}. \quad (17)$$

Although, we derive ESE power of two instructions in time domain, measuring ESE of two sequences can be cumbersome because the number of samples required can be huge. In that perspective, we derive the equation given in 6 which clarifies the relation between the ESE power of two instructions and the power at f_{alt}.

To relate $s_1^{(X_1)}[n]$ and $s_2^{(X_2)}[n]$ to our benchmarks, we define a square wave $p[n]$ with a 50% duty cycle as

$$p[0 \le n < Nn_{\mathrm{inst}}] = 1 \quad (18)$$
$$p[Nn_{\mathrm{inst}} \le n < 2Nn_{\mathrm{inst}}] = 0, \quad (19)$$

where $p[n]$, $s_1^{(X_1)}[n]$ and $s_2^{(X_2)}[n]$ are periodic with period $2Nn_{\mathrm{inst}}$, since we assume the additive noise is also periodic. Then, we can take the discrete Fourier series of these signals over $2Nn_{\mathrm{inst}}$ samples. We refer to $S_1^{(X_1)}[k]$, $S_2^{(X_2)}[k]$, and $P[k]$ as the discrete Fourier series (DFS) of $s_1^{(X_1)}[n]$, $s_2^{(X_2)}[n]$ and $p[n]$ respectively, defined for $0 \le k < 2Nn_{\mathrm{inst}}$.

We next define

$$v[n] = p[n]s_1^{(X_1)}[n] + (1-p[n])s_2^{(X_2)}[n], \quad (20)$$

which represents the signal created by the sequence of instructions executed by the microbenchmarks.

We start the derivation of relationship between ESE and measured spectral power by observing that the DFS of $v[n]$ defined in (20) can be written as

$$\begin{aligned}
V[k] &= P[k] * S_1^{(X_1)}[k] + (1-P[k]) * S_2^{(X_2)}[k]\\
&= S_2^{(X_2)}[k] + P[k] * \left(S_1^{(X_1)}[k] - S_2^{(X_2)}[k]\right), \quad (21)
\end{aligned}$$

where $*$ denotes periodic convolution, defined as in Appendix III.

Now we consider $V[1]$, the 2^{nd} Fourier coefficient (the first harmonic) of the $v[n]$ sequence:

$$V[1] = S_2^{(X_2)}[1] + \frac{\sum_{m=0}^{2Nn_{\mathrm{inst}}-1} P[1-m]\left(S_1^{(X_1)}[m] - S_2^{(X_2)}[m]\right)}{2Nn_{\mathrm{inst}}}. \quad (22)$$

Using the equation (51) from Appendix III, we can observe that $S_1^{(X_1)}[k]$ and $S_2^{(X_2)}[k]$ are non-zero only for $k = 2n_{\mathrm{inst}}l$ for $l = 0, 1, ..., N-1$. Then $V[1]$ simplifies to

$$V[1] = \frac{\sum_{l=0}^{N-1} P[1-2n_{\mathrm{inst}}l]\left(S_1^{(X_1)}[2n_{\mathrm{inst}}l] - S_2^{(X_2)}[2n_{\mathrm{inst}}l]\right)}{2Nn_{\mathrm{inst}}} \quad (23)$$

Then, $V[1]$ can be further expanded as follows

$$\begin{aligned}
V[1] = &\frac{P[1]\left(S_1^{(X_1)}[0] - S_2^{(X_2)}[0]\right)}{2Nn_{\mathrm{inst}}}\\
&+\frac{P[1-2n_{\mathrm{inst}}]\left(S_1^{(X_1)}[2n_{\mathrm{inst}}] - S_2^{(X_2)}[2n_{\mathrm{inst}}]\right)}{2Nn_{\mathrm{inst}}} \quad (24)\\
&+\dots
\end{aligned}$$

Here we note that next few higher order odd harmonics can be similarly expanded while the even harmonics are zero. Additionally, we note that $P[k]$ is the k^{th} coefficient of the discrete Fourier series for a square wave with period $2Nn_{\mathrm{inst}}$

and can be written as ([39], Example 8.3)

$$\begin{aligned}\frac{|P[k]|}{2Nn_{\text{inst}}} &= \frac{\sin(\pi k/2)}{2Nn_{\text{inst}}\cdot\sin(\frac{\pi k}{2Nn_{\text{inst}}})}\\ \Rightarrow \frac{|P[k]|}{2Nn_{\text{inst}}} &\approx \frac{\sin(\pi k/2)}{\pi k}\\ \Rightarrow \frac{|P[1]|}{2Nn_{\text{inst}}} &\approx \frac{1}{\pi}.\end{aligned} \tag{25}$$

The last two steps follow by recognizing that

$$2Nn_{\text{inst}}\cdot\sin\left(\frac{\pi k}{2Nn_{\text{inst}}}\right) = \pi k\cdot\frac{\sin\left(\frac{\pi k}{2Nn_{\text{inst}}}\right)}{\frac{\pi k}{2Nn_{\text{inst}}}} \tag{26}$$

and noting that $\sin(x)/x \to 1$ as $x \to 0$ (i.e. large n_{inst}). Since n_{inst} is typically > 100, this approximation is valid. For $n_{\text{inst}} > 100$, $|P[1]| > 100|P[1-n_{\text{inst}}]|$, so the higher order terms in (24) can be neglected, giving

$$\begin{aligned}|V[1]| &\approx \frac{|P[1]|}{2Nn_{\text{inst}}}\cdot\left|S_1^{(X_1)}[0]-S_2^{(X_2)}[0]\right|\\ \Rightarrow \pi\,|V[1]| &\approx \left|S_1^{(X_1)}[0]-S_2^{(X_2)}[0]\right|.\end{aligned} \tag{27}$$

The next step is to calculate the difference between $S_1^{(X_1)}[0]$ and $S_2^{(X_2)}[0]$. Here we note that , we decompose $s_1^{(X_1)}[n] = \hat{o}_1[n] + i_1^{(X_1)}[n]$ where the first N samples of $\hat{o}_1[n] = [o_1[0], o_1[1], ..., o_1[n_O-1], 0, \cdots, 0]$ and the first N samples of $i_1^{(X_1)}[n] = [0, ..., 0, x_1[0], x_1[1], \cdots, x_1[n_X-1]]$. We can decompose $s_2^{(X_2)}[n]$ similarly. By the linearity of the Fourier transform

$$\begin{aligned}S_1^{(X_1)}[k]-S_2^{(X_2)}[k] &= I_1^{(X_1)}[k]+\hat{O}_1[k]-\left(I_2^{(X_2)}[k]+\hat{O}_2[k]\right)\\ &= I_1^{(X_1)}[k]-I_2^{(X_2)}[k]+\left(\hat{O}_1[k]-\hat{O}_2[k]\right).\end{aligned} \tag{28}$$

The DFS coefficient $I_1^{(X_1)}[0]$ is

$$\begin{aligned}I_1^{(X_1)}[0] &= \sum_{n=0}^{2Nn_{\text{inst}}-1} i_1^{(X_1)}[n]\\ &= \sum_{r=0}^{2n_{\text{inst}}-1}\sum_{s=0}^{n_X-1} i_1^{(X_1)}[rn_{\text{inst}}+n_O+s]\\ &= \sum_{r=0}^{2n_{\text{inst}}-1}\sum_{s=0}^{n_X-1}\left(X_1^v+w_1^{(X_1)}[rn_{\text{inst}}+n_O+s]\right)\\ &= 2n_{\text{inst}}n_X X_1^v + 2n_{\text{inst}}\sum_{n=n_O}^{N} w_1^{(X_1)}[n].\end{aligned} \tag{29}$$

where the last equality follows the assumption that noise is also circular. Similarly, $I_2^{(X_2)}[0] = 2n_{\text{inst}}n_X X_2^v + 2n_{\text{inst}}\sum_{n=n_O}^{N} w_2^{(X_2)}[n]$ and $O_i[0] = 2n_{\text{inst}}n_O O^v + 2n_{\text{inst}}\sum_{l=0}^{n_O-1} w_i^{(O)}$ where $i \in \{1,2\}$. Therefore,

$$\begin{aligned}S_1^{(X_2)}[0]-S_2^{(X_2)}[0] &= 2n_{\text{inst}}\Big[n_X(X_1^v-X_2^v)\\ &\quad+\sum_{l=0}^{n_O-1}\left(w_1^{(O)}[l]-w_2^{(O)}[l]\right)\\ &\quad+\sum_{l=0}^{n_X-1}\left(w_1^{(X_1)}[l]-w_2^{(X_2)}[l]\right)\Big]\end{aligned} \tag{30}$$

Assuming n_O and n_X are large enough and additive noises are independent, we have

$$\begin{aligned}\left|S_1^{(X_2)}[0]-S_2^{(X_2)}[0]\right|^2 &\approx (2n_{\text{inst}})^2\Big[(n_X(X_1^v-X_2^v))^2\\ &\quad+2n_O\sigma_O^2+n_X\sigma_{X_1}^2+n_X\sigma_{X_2}^2\Big]\\ &= 4n_{\text{inst}}^2 D_{ab}\end{aligned} \tag{31}$$

where

$$D_{ab} = n_X\left(n_X(X_1^v-X_2^v)^2+\sigma_{X_1}^2+\sigma_{X_2}^2\right)+2n_O\sigma_o^2. \tag{32}$$

As the next step, we need to relate (31) to the power observed with the spectrum analyzer. The power observed with the spectrum analyzer is described by ([40], [41])

$$P(f_{\text{alt}}) = \frac{2}{R}\Big(\frac{|V[1]|}{2Nn_{\text{inst}}}\Big)^2, \tag{33}$$

where $2Nn_{\text{inst}}$ is the number of samples taken in one period T_{alt}. We also note that

$$n_{\text{inst}}f_{\text{alt}} = \frac{1}{2NT_I}. \tag{34}$$

So, by plugging (27) and (31) into (33), we have

$$\begin{aligned}P(f_{\text{alt}}) &= \frac{2}{R}\left(\frac{\left|S_1^{(X_1)}[0]-S_2^{(X_2)}[0]\right|}{2Nn_{\text{inst}}\cdot\pi}\right)^2\\ &= \frac{2}{R}\left(\frac{4n_{\text{inst}}^2 D_{ab}}{4N^2n_{\text{inst}}^2\cdot\pi^2}\right)\\ &= \frac{2}{R}\left(\frac{D_{ab}}{N^2\pi^2}\right).\end{aligned} \tag{35}$$

As mentioned before, since working on time domain is cumbersome, our main goal is to measure ESE in frequency domain. Therefore, using (34), (33) and (35), we obtain the relationship between ESE and $P(f_{\text{alt}})$ as follows:

$$P(f_{\text{alt}}) = \frac{2}{R}\left(\frac{D_{ab}}{N^2\cdot\pi^2}\right) \tag{36}$$

$$\Rightarrow \quad \frac{D_{ab}}{R} = \frac{P(f_{\text{alt}})\cdot N^2\cdot\pi^2}{2\cdot} \tag{37}$$

$$\Rightarrow \quad \frac{D_{ab}}{RNn_{\text{inst}}} = \frac{P(f_{\text{alt}})\cdot N\cdot\pi^2}{2\cdot n_{\text{inst}}} \tag{38}$$

$$\Rightarrow \quad \frac{2\cdot f_{\text{alt}}T_{\text{I}}D_{ab}}{R} = \frac{P(f_{\text{alt}})\cdot N\cdot\pi^2}{2\cdot n_{\text{inst}}} \tag{39}$$

$$\Rightarrow \quad \frac{2T_{\mathrm{I}} n_{\mathrm{inst}}}{R} D_{ab} = \frac{P(f_{\mathrm{alt}}) \cdot N \cdot \pi^2}{2 \cdot f_{\mathrm{alt}}} \tag{40}$$

$$\Rightarrow \qquad \kappa D_{ab} = \pi^2 \frac{P(f_{\mathrm{alt}}) \cdot N}{2 \cdot f_{\mathrm{alt}}} \tag{41}$$

where (39) follows the equality given in (34). Relating the equation in (41) with (14) and (15), we have

$$\begin{aligned} \kappa D_{ab} &= \kappa \left(n_X \left(n_X (X_1^v - X_2^v)^2 + \sigma_{X_1}^2 + \sigma_{X_2}^2 \right) + 2 n_O \sigma_o^2 \right) \\ &= 2 n_X n_{\mathrm{inst}} \ \mathrm{ESE}[\mathrm{X}_1, \mathrm{X}_2] \\ &\quad + \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)} \right] + \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_2)}, \mathrm{s}_2^{(\mathrm{X}_2)} \right]. \end{aligned} \tag{42}$$

To simplify the notation, we define

$$C(X_1, X_2) = - \frac{\mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)} \right] + \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_2)}, \mathrm{s}_2^{(\mathrm{X}_2)} \right]}{2 n_X n_{\mathrm{inst}}} \tag{43}$$

which can be considered as the noise term added to ESE of (X_1, X_2) instruction pair because of the measurement done in frequency domain. Therefore, we have

$$\mathrm{ESE}[\mathrm{X}_1, \mathrm{X}_2] = \frac{1}{2 n_X n_{\mathrm{inst}}} \pi^2 \frac{P(f_{\mathrm{alt}}) \cdot N}{2 \cdot f_{\mathrm{alt}}} + C(X_1, X_2) \tag{44}$$

$$= \left(\frac{\pi}{2} \right)^2 \frac{P(f_{\mathrm{alt}}) \cdot N}{n_X \cdot n_{\mathrm{inst}} \cdot f_{\mathrm{alt}}} + C(X_1, X_2) \tag{45}$$

which concludes the proof.

Appendix II
Execution Location Based Noise Power Estimation

In this section, we provide a method to estimate the additive noise power corresponding to the execution location of any instruction. Note that when the same instruction is inserted into both for-loops, we have

$$\begin{aligned} \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)} \right] &= \tfrac{\kappa}{2} \Big(n_X (X_1^v - X_1^v)^2 + n_X \sigma_{X_1}^2 \\ &\qquad + n_X \sigma_{X_1}^2 + 2 n_O \sigma_O^2 \Big) \\ &= \tfrac{\kappa}{2} \Big(2 n_X \sigma_{X_1}^2 + 2 n_O \sigma_O^2 \Big). \end{aligned} \tag{46}$$

On the other hand, if we do not insert any instruction into these loops, we have

$$\mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{NOI})}, \mathrm{s}_2^{(\mathrm{NOI})} \right] = \frac{\kappa}{2} \Big(2 n_O \sigma_O^2 \Big). \tag{47}$$

Therefore, to obtain the noise power related to any instruction can be found by combining (46) and (47) as

$$\begin{aligned} & \kappa n_X \sigma_{X_1}^2 = \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)} \right] - \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{NOI})}, \mathrm{s}_2^{(\mathrm{NOI})} \right] \\ \Rightarrow \quad & n_X \sigma_{X_1}^2 = \frac{\mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)} \right] - \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{NOI})}, \mathrm{s}_2^{(\mathrm{NOI})} \right]}{\kappa} \\ \Rightarrow \quad & \bar{\sigma}_{X_1}^2 = \frac{\mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{X}_1)}, \mathrm{s}_2^{(\mathrm{X}_1)} \right] - \mathcal{P}_{\mathrm{A}} \left[\mathrm{s}_1^{(\mathrm{NOI})}, \mathrm{s}_2^{(\mathrm{NOI})} \right]}{\kappa}. \end{aligned} \tag{48}$$

Appendix III
Discrete Fourier Series

The discrete Fourier series pair of a periodic signal $x[n]$ with period M can be defined as [39]

$$\begin{aligned} X[k] &= \sum_{n=0}^{M-1} x[n] e^{-j(2\pi/M)kn} \qquad \text{(DFS)} \\ x[n] &= \frac{1}{M} \sum_{n=0}^{M-1} X[k] e^{-j(2\pi/M)kn} \quad \text{(IDFS)}. \end{aligned} \tag{49}$$

Multiplying two sequences $x_1[n]$ and $x_2[n]$ in time domain is equivalent to their periodic convolution in the frequency domain and can be written as [39]

$$x_1[n] x_2[n] \xrightarrow{\mathrm{DFS}} X_1[k] * X_2[k] = \frac{1}{M} \sum_{l=0}^{M-1} X_1[l] X_2[k-l]. \tag{50}$$

The discrete Fourier series of a periodic signal $x[n]$ taken *over a period of ML samples* is equal to

$$\begin{aligned} X[k] &= \sum_{n=0}^{ML-1} x[n] e^{-j \frac{2\pi}{ML} kn} \\ &= \sum_{m=0}^{M-1} \left(x[m] \sum_{l=0}^{L-1} e^{-j \frac{2\pi}{ML} k(m+Ml)} \right) \\ &= \sum_{m=0}^{M-1} \left(x[m] e^{-j \frac{2\pi}{ML} km} \sum_{l=0}^{L-1} e^{-j 2\pi k \frac{l}{L}} \right) \\ &= \sum_{m=0}^{M-1} \left(x[m] e^{-j \frac{2\pi}{ML} km} L \sum_{l=-\infty}^{\infty} \delta[k - lL] \right) \\ &= \begin{cases} L \sum_{m=0}^{M-1} x[m] e^{-j \frac{2\pi}{ML} km} & \text{for } k = Ll \\ 0 & \text{for } k \neq Ll \end{cases} \end{aligned} \tag{51}$$

The discrete time periodic impulse train with period M can be written as [39]

$$\sum_{l=-\infty}^{\infty} \delta[k - Ll] = \frac{1}{L} \sum_{l=0}^{L-1} e^{-j 2\pi k \frac{l}{L}}. \tag{52}$$

Appendix IV
Gradient Descent Approach for Capacity Calculation

In this section, we provide the gradient descent algorithm to solve the optimization problem in (8). We also derive a closed form solution to attain the probability of each instruction. First, we take the derivative of (8) with respect to an instruction probability P_m. Therefore,

$$\frac{\partial f(\mathbf{P})}{\partial P_m} = \frac{\partial(\frac{\Omega}{\Psi})}{\partial P_m} \tag{53}$$

$$\frac{\partial(\frac{\Omega}{\Psi})}{\partial P_m} = \frac{\partial\left(\frac{\sum_{i,j} P_i p_{ij} \log p_{ij} - \sum_{i,j} P_i p_{ij} \log \sum_k P_k p_{kj}}{\sum_i P_i L_i}\right)}{\partial P_m}$$

$$= \frac{\left(\sum_j p_{mj} \log p_{mj} - G_m\right)\Psi - \Omega L_m}{\Psi^2} \tag{54}$$

where L_m is the length of m^{th} instruction, and

$$\mathbf{P} = [P_1 \quad P_2 \quad \cdots \quad P_K]^T \tag{55}$$

and

$$\begin{aligned} G_m &= \sum_j p_{mj} \log \sum_k P_k p_{kj} + \sum_{i,j} P_i p_{ij} \frac{p_{mj}}{\sum_k P_k p_{kj}} \\ &= \sum_j p_{mj} \log \sum_k P_k p_{kj} + \sum_j p_{mj} \sum_i P_i p_{ij} \frac{1}{\sum_k P_k p_{kj}} \\ &= \sum_j p_{mj} \log \sum_k P_k p_{kj} + \sum_j p_{mj} \\ &= \sum_j p_{mj} \log \sum_k P_k p_{kj} + 1. \end{aligned} \tag{56}$$

Since transition probabilities are calculated based on ESE measurements as explained in Appendix IV-B, we can calculate entropy for each instruction beforehand. Let $H_m = -\sum_j p_{mj} \log p_{mj}$ be the entropy for the m^{th} instruction. So, the gradient vector for the optimization problem can be given as

$$\nabla_{\mathbf{P}}\left(\frac{\Omega}{\Psi}\right) = \nabla_{\mathbf{P}} f(\mathbf{P}) = \begin{bmatrix} \frac{\partial(\frac{\Omega}{\Psi})}{\partial P_1} & \frac{\partial(\frac{\Omega}{\Psi})}{\partial P_2} \cdots & \frac{\partial(\frac{\Omega}{\Psi})}{\partial P_K} \end{bmatrix}^T \tag{57}$$

where $(\bullet)^T$ is the transpose operation and

$$\frac{\partial(\frac{\Omega}{\Psi})}{\partial P_m} = -\frac{(H_m + G_m)\Psi + \Omega L_m}{\Psi^2}. \tag{58}$$

Fig. 8 delivers the insights about the algorithm to achieve the optimal solution. In the algorithm, $\mathbf{I}_K$ in line 12 stands for the identity matrix with size K and $sum(\bullet)$ in line 13 returns the sum of its vector argument.

Another goal of the section is to derive a closed-form solution to the defined problem in (8). We exploit the result in (54) and set the derivative equals to zero. Let us write the overall optimization problem in Lagrangian form such that

$$\mathcal{L}(\mathbf{P}) = f(\mathbf{P}) + \lambda(1 - \sum_i P_i) + \mu^T \mathbf{P} \tag{59}$$

where μ and λ are the dual variables. For the rest of the derivations, we will assume that each instruction occurs with

```
1  //Initialization of the state probabilities
2  P(0) = 1_K\K;
3  //-- K is the number of instruction
4  //-- 1_K is the vector with full of zeros
5  //and whose length is K.
6  converge = false;
7  while(!converge){
8    ∇_P ← Calculate the gradient at time k
9    P(k+1) = P(k+1) + μ∇_P f(P(k))
10   // μ is the step size
11   // Project into the feasible region
12   P(k+1) = (I_K − 1_K 1_K^T)P(k+1) + 1_K;
13   P(k+1) = P(k+1) / sum(P(k+1));
14   if [(|f(P(k+1)) − f(P(k))|) /|f(P(k))|] < ϵ{
15   converge = true;
16   }
17 }
```

Fig. 8. Gradient descent approach to achieve optimal solution

nonzero probability so that, from KKT (Karush-Kuhn-Tucker) conditions, μ must equal to a zero vector. So,

$$\frac{\partial \mathcal{L}(\mathbf{P})}{\partial P_m} = 0 \tag{60}$$

$$\Rightarrow \quad \frac{(H_m + G_m)\Psi + \Omega L_m}{\Psi^2} - \lambda = 0 \tag{61}$$

$$\Rightarrow \quad \frac{(H_m + G_m) + R L_m}{\Psi} = \lambda \tag{62}$$

where $R = \max_{\mathbf{P}} f(\mathbf{P})$ which represents the value for the optimum solution. The reason for the insertion of R instead of the ratio Ψ/Ω is simply because calculations are done in derivative space and the optimization setting ensures convexity. Observe that left hand side of (62) simple equals to partial derivative of $f(\mathbf{P})$. Hence, for the optimal solution, the partial derivative of the objective function with respect to each instruction probability must equal to each other. By keeping that in mind and plugging the exact definition of G_m, we have

$$(H_m + G_m) + R L_m = \lambda\Psi \tag{63}$$

$$\Rightarrow \quad G_m = -H_m - R \cdot L_m + \lambda\Psi \tag{64}$$

$$\Rightarrow \quad \sum_j p_{mj} \log \sum_k P_k p_{kj} + 1 = -H_m - R \cdot L_m + \lambda\Psi \tag{65}$$

$$\Rightarrow \quad \sum_j p_{mj} \log \sum_k P_k p_{kj} = -(H_m + R \cdot L_m - \lambda\Psi + 1) \tag{66}$$

If we define the equality $\sum_m h_{mi} p_{mj} = \delta_{ij}$ where δ is the Kronocker delta function such that $\delta_{ij} = \delta[i-j]$, introduce the notation $\sigma_m = (H_m + R \cdot L_m - \lambda\Psi + 1)$, multiply both sides with h_{mi} and sum over m, we have

$$\sum_{m,j} h_{mi} p_{mj} \log \sum_k P_k p_{kj} = -\sum_m \sigma_m h_{mi} \tag{67}$$

$$\Rightarrow \quad \sum_j \delta_{ij} \log \sum_k P_k p_{kj} = -\sum_m \sigma_m h_{mi} \tag{68}$$

$$\Rightarrow \quad \log \sum_k P_k p_{ki} = -\sum_m \sigma_m h_{mi} \tag{69}$$

$$\Rightarrow \quad \sum_k P_k p_{ki} = exp(-\sum_m \sigma_m h_{mi}). \tag{70}$$

Finally, let us define $\sum_i z_{ti} p_{ki} = \delta_{tk}$ and perform the same

operations as before, we have

$$\sum_{i,k} P_k z_{ti} p_{ki} = \sum_i z_{ti} exp(-\sum_m \sigma_m h_{mi}) \quad (71)$$

$$\Rightarrow \quad \sum_k P_k \delta_{kt} = \sum_i z_{ti} exp(-\sum_m \sigma_m h_{mi}) \quad (72)$$

$$\Rightarrow \quad P_t = \sum_i z_{ti} exp(-\sum_m \sigma_m h_{mi}) \quad (73)$$

$$\Rightarrow \quad P_t = \sum_i z_{ti} exp(-\sum_m (H_m + R \cdot L_m - \lambda\Psi + 1) h_{mi}) \quad (74)$$

Hence, we can extract the instruction probabilities if we have the optimal rate R. Since our purpose is to obtain the maximum rate and the point which maximizes this rate, we can scan for a value of R in the range between 0 to $\log_2 K$ and λ such that it satisfies the conventional probability distribution constraints i.e. $\sum P_i = 1$. Our search is limited by $\log_2 K$ for the rate since the lengths of each instruction are at least unity, but unlimited for λ. Fortunately, proposed algorithm given in Fig. 8 provides the rate, λ and the corresponding point, therefore, we do not need to perform such a scan. However, we can use the closed-form solution for inspection purposes.

REFERENCES

[1] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, Oct. 1973. [Online]. Available: http://doi.acm.org/10.1145/362375.362389

[2] J. Millen, "20 years of covert channel modeling and analysis," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, 1999, pp. 113–114.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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

[12] 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

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

[14] E. Bangerter, D. Gullasch, and S. Krenn, "Cache games - bringing access-based cache attacks on AES to practice," in *Proceedings of IEEE Symposium on Security and Privacy*, 2011.

[15] Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi, "Cryptanalysis of block ciphers implemented on computers with cache," in *Proceedings of the International Symposium on Information Theory and its Applications*, 2002, pp. 803–806.

[16] Z. Wang and R. B. Lee, "New cache designs for thwarting software cache-based side channel attacks," in *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*. ACM, 2007, pp. 494–505.

[17] D. Agrawal, B. Archambeult, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 2002*, 2002, pp. 29–45.

[18] ——, "The EM side-channel(s): attacks and assessment methodologies," in *http://www.research.ibm.com/intsec/emf-paper.ps*, 2002.

[19] M. G. Khun, "Compromising emanations: eavesdropping risks of computer displays," *The complete unofficial TEMPEST web page: http://www.eskimo.com/~joelm/tempest.html*, 2003.

[20] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer, "Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation," in *Cryptographic Hardware and Embedded Systems – CHES 2015*, ser. Lecture Notes in Computer Science, T. Gneysu and H. Handschuh, Eds. Springer Berlin Heidelberg, 2015, vol. 9293, pp. 207–228. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-48324-4_11

[21] A. Zajic and M. Prvulovic, "Experimental demonstration of electromagnetic information leakage from modern processor-memory systems," *Electromagnetic Compatibility, IEEE Transactions on*, vol. 56, no. 4, pp. 885–893, Aug 2014.

[22] 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.

[23] 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, Aug. 2015, pp. 849–864. [Online]. Available: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/guri

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

[25] 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.

[26] R. Anderson and F. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 474-481, May 1998.

[27] 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, Feb. 2013.
[28] M. Davey and D. MacKay, "Reliable communication over channels with insertions, deletions, and substitutions," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 687–698, Feb 2001.
[29] R. Venkataramanan, S. Tatikonda, and K. Ramchandran, "Achievable rates for channels with deletions and insertions," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, July 2011, pp. 346–350.
[30] A. Kirsch and E. Drinea, "Directly lower bounding the information capacity for channels with i.i.d. deletions and duplications," *Information Theory, IEEE Transactions on*, vol. 56, no. 1, pp. 86–102, Jan 2010.
[31] J. Hu, T. Duman, M. Erden, and A. Kavcic, "Achievable information rates for channels with insertions, deletions, and intersymbol interference with i.i.d. inputs," *Communications, IEEE Transactions on*, vol. 58, no. 4, pp. 1102–1111, April 2010.
[32] S. Verdu and S. Shamai, "Variable-rate channel capacity," *Information Theory IEEE Transactions on*, vol. 56, no. 6, pp. 2651–2667, June 2010.
[33] M. Rahmati and T. Duman, "Bounds on the capacity of random insertion and deletion-additive noise channels," *Information Theory, IEEE Transactions on*, vol. 59, no. 9, pp. 5534–5546, Sept 2013.
[34] H. Mercier, V. Tarokh, and F. Labeau, "Bounds on the capacity of discrete memoryless channels corrupted by synchronization and substitution errors," *Information Theory, IEEE Transactions on*, vol. 58, no. 7, pp. 4306–4330, July 2012.
[35] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.
[36] R. Callan, N. Popovic, A. Daruna, E. Pollmann, A. Zajic, and M. Prvulovic, "Comparison of electromagnetic side-channel energy available to the attacker from different computer systems," *Proceedings of the IEEE Symposium on Electromagnetic Compatibility (EMC)*,pp. 1-5, Dresden, Germany, August 2015.
[37] R. L. Callan, *Analyzing software using unintentional electromagnetic emanations from computing devices*. PhD thesis, Georgia Institute of Technology, 2017.
[38] S.W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing," *California Technical Publishing*, 1997, pp. 255–259.
[39] A.V. Oppenheim, R.W. Schafer, and J.R. Buck, "Discrete-time signal processing," *Prentice Hall*, 1999.
[40] 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.
[41] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P Flannery, "Numerical recipes in C," *Cambridge University Press*, 1996, pp. 550–551.
[42] R. L. Dobrushin, "Translated from problemy peredachi informatsii," *Probl. Inf. Transmiss.*, vol. 3, no. 4, pp. 11–26, 1967.

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.

Robert L. Callan (S'14-M'17) received the B.Sc. degree in electrical engineering from the University of Pennsylvania in 2007, the M.Sc. degree in electrical engineering from the University of Southern California in 2008, and the Ph.D. degree from the School of Electrical and Computer Engineering at the Georgia Institute of Technology in 2016. He is currently a post-doctoral researcher in the Electromagnetic Measurements in Communications and Computing (EMC^2) Lab at the Georgia Institute of Technology focusing on analyzing software using EM emanations. Previously, he characterized high speed serial interfaces at IBM and Altera. His research interests span areas of electromagnetics, VLSI, and computer engineering.

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 an Associate 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, the Best Paper Award at MICRO 2016, 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.