Instruction Level Program Tracking Using Electromagnetic Emanations

Baki Berkay Yilmaz[a*], Elvan Mert Ugurlu[a*], Alenka Zajic[a], and Milos Prvulovic[b]

[a]The School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

[b]The School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA

ABSTRACT

Monitoring computer system activities on the instruction level provides more resilience to malware attacks because these attacks can be analyzed better by observing the changes on the instruction level. Assuming the source code is available, many training signals can be collected to track the instruction sequence to detect whether a malware is injected or the system works properly. However, training signals have to be collected with high sampling rate to ensure that the significant features of these signals do not vanish. Since the clock frequencies of the current computer systems are extremely high, we need to have a commercial device with high sampling rate, i.e. 10GHz, which either costs remarkably high, or does not exist. To eliminate the deficiencies regarding the insufficient sampling rate, we propose a method to increase the sampling rate with the moderate commercial devices for training symbols. In that respect, we first generate some random instruction sequences which exist in the inspected source code. Then, these sequences are executed in a for-loop, and emanated electromagnetic (EM) signals from the processor are collected by a commercially available device with moderate sampling rate, i.e. sampling rate is much smaller than the clock frequency. Lastly, we apply a mapping of the gathered samples by utilizing modulo of their timings with respect to execution time of overall instruction sequence. As the final step, we provide some experimental results to illustrate that we successfully track the instruction sequence by applying the proposed approach.

Keywords: Side Channels, Instruction Tracking, Information Leakage, Electromagnetic Emanations, Malware Attacks

1. INTRODUCTION

As devices get more interconnected through the Internet of Things (IoT), it becomes imperative to find effective way of securing all these devices. Interest in tracking the program activity is drawing the attention of experts from various fields due to its possible applications in monitoring code flow, detecting malicious activities, reverse engineering, cryptanalysis, etc.[1–7]

Side channels are unintentional and asynchronous channels that can leak some sensitive information while performing a computer activity.[8] Many attacks are reported based on power analysis,[1, 2, 9–11] temperature analysis,[12] acoustic emanations,[13, 14] electromagnetic (EM) emanations,[7, 15–18] etc. One of the emerging applications of side-channel analysis is their use for tracking program activities on various code levels such as loops, paths, and basic blocks.[3–6, 19] Most of the previous work on single instruction code analysis is focused on building side-channel-based disassembler,[1, 20, 21] i.e. reconstructing the instructions from the side-channel signal. In particular, work in[1, 2, 20, 21] shows that when several samples per instruction are available to the observer (i.e. analyzed devices have 1, 4, or 16 MHz clock frequency) and the simplest form of pipelining (i.e. only prefetch and execute) is used, the instructions can be successfully reconstructed from side-channels. However, any device that is more advanced than microcontrollers, i.e., has the processor clock frequency higher than 16 MHz and has more than two stage pipeline architecture, requires signal receivers with very high sampling rate (on the order of many

* These two authors contributed to the paper similarly (author order does not reflect the extent of contribution).
Further author information: (Send correspondence to B.B.Y.)
B.B.Y.: E-mail: byilmaz7@gatech.edu, E.M.U.: E-mail: ugurlu@gatech.edu

times the processor clock frequency) to ensure that significant features of these signals are present in the received signal. Such receivers are either very expensive or non-existent.

To address this problem, we propose a method that utilizes existing receivers but allows for tracking of individual instructions on more advanced computer processors. To achieve this objective, we propose a new technique to collect training sequence and upsample it in a way that allows for single-instruction tracking using heavily undersampled received signals. Assuming the source code is available, many training signals can be collected to track the instruction sequence. We first generate some random instruction sequences which exist in the inspected source code. Then, these sequences are executed in a for-loop, and emanated electromagnetic (EM) signals from the processor are collected by a commercially available receiver with moderate sampling rate, i.e., sampling rate is much lower than the clock frequency. Then, we propose a new method that we refer to as *modulo operation*. The method is based on an idea that repeating the activity of interest periodically can be used for reconstructing the signal with higher effective sampling rate. Finally, we apply mapping of the gathered samples by utilizing modulo of their timings with respect to execution time of overall instruction sequence. Correlating single execution of individual instructions in a testing phase with the upsampled version of signal collected in training phase, we show that we can correctly track individual instructions even when received signals are undersampled.

The rest of the paper is organized as follows. Section 2 describes our implementation of single instruction tracking using EM signature generation, Section 3 describes how *modulo operation* allows for effectively increasing sampling rate, Section 4 presents experimental results, and Section 5 briefly summarizes the conclusions.

2. A METHOD FOR SINGLE INSTRUCTION TRACKING

One of the emerging applications of side-channel analysis is their use for tracking program activities on various code levels including individual instructions.

Two of the main challenges in tracking individual instructions on more advanced processors is lack of model that describes the pipeline effect on the emanated EM signal and heavily undersampled received signals. In order to address the first issue, we propose to monitor *sequences of instructions* instead of a *single instruction*. When an instruction is executed, pipeline is filled with neighboring instructions that are just before or after that instruction, and those neighboring instructions are included in the instruction sequence. As much as the individual contributions from the neighboring instructions that appear in different stages of pipeline are not known, using the whole sequence as a signature allows us to include their aggregate effect.

To address the problem of undersampled signals, we propose a *modulo operation* as a method to upsample received EM side-channel signals. Assuming we have a signal that is composed of samples from a periodic signal, *modulo operation* can be used to upsample that signal (reader can refer to Section 3 for detailed analysis of *modulo operation*). One should note that, when the instruction sequence is executed, the emanated EM signal occurs only once and hence it is not a periodic signal. However, since we have access to the device during training phase, we can artificially mimic the required periodicity condition by executing the same sequence repeatedly. It should also be noted that the repetition of the instruction sequence is only used in the training phase and not in the testing phase, where instruction sequence is executed only once. Considering the requirements above, we can summarize the setups used for training and testing phases as follows.

- **Training Phase:** This phase is used to generate EM signatures. Therefore, the instruction sequence is executed N consecutive times by utilizing a for loop. The pseudo code for this setup is shown in Figure 1a. The empty for-loop before and after the execution of the instruction sequence are used as markers to detect the beginning and ending of the instruction sequence.
- **Testing Phase:** In this phase, the unknown sequence of individual instructions is tested against signals collected in a training phase. Unlike in training phase, in testing phase, the instruction sequence is executed only once. The pseudo code for this phase can be seen in Figure 1b. The empty for-loops before and after the execution of the instruction sequence are similarly used as markers to detect the beginning and ending of the instruction sequence.

```
for
  #empty for loop
end

for N times
  #Instruction sequence
end

for
  #empty for loop
end
```

(a) Pseudo code for **Training Phase**

```
for
  #empty for loop
end


  #Instruction sequence


for
  #empty for loop
end
```

(b) Pseudo Code for **Testing Phase**

Figure 1: Pseudo codes for **Training** and **Testing** phases.

Implementing *modulo operation* while generating the EM signatures does not only increase the sampling rate, it also results in EM signatures that are averaged versions of N executions of the same instruction sequence. This helps with reducing noise as well as avoiding aliasing. By using sequences of instructions instead of single instructions, we are inevitably increasing the total possible number of EM signatures that needs to be generated. On the other hand, the longer the sequence, the better the pipeline effect is included in the generated EM signature. This situation indicates the trade off between the total number of EM signatures and the ability to address the pipeline effect. Finally, the training and testing EM signals are correlated against each other and different sequences of instructions are correctly classified.

3. MODULO OPERATION AS A METHOD FOR UPSAMPLING SIDE-CHANNEL SIGNALS

In this section, we first explain the theory behind the *modulo operation*, then illustrate the technique through simple examples, and finally discuss the required conditions for this operation to be applicable.

Theoretical Perspective: Let's assume that $y(t)$ is a long periodic signal with period T, and T_s is the sampling period of the measuring instrument, which (without loss of generality) can be defined as

$$T_s = kT + \Delta_m. \tag{1}$$

Here, k is a fixed non-negative integer, T is the period of the target signal $y(t)$, and Δ_m (that will be referred to as *modular offset*) is defined as:

$$\Delta_m = \mathrm{mod}(T_s, T). \tag{2}$$

Let $y_s[n]$ be a discrete time sequence which is constructed by sampling $y(t)$ with the sampling period T_s as

$$y_s[n] = y(nT_s), \text{ for } \forall n \text{ where } n \in \{0, 1, 2, ..., N-1\}. \tag{3}$$

Since T_s is known, sampling times, t_n's, for each sample n are also known:

$$t_n = nT_s, \text{ for } \forall n \text{ where } n \in \{0, 1, 2, ..., N-1\}. \tag{4}$$

Let t_n^{mod} be the time indicating where the n^{th} sample corresponds in the fundamental period of $y(t)$ and it can be written as

$$t_n^{mod} = \mathrm{mod}\left(t_n, T\right) = \mathrm{mod}\left(n\left(kT + \Delta_m\right), T\right) = \mathrm{mod}\left(n\Delta_m, T\right), \text{ for } \forall n \text{ where } n \in \{0, 1, 2, ..., N-1\}. \tag{5}$$

It should be noted that $0 \leq t_n^{mod} < T$. Hence, the value of the n^{th} sample ($y_s[n]$) can be wrapped around and used to fill in the value for t_n^{mod}. After repeating this process for all samples and sorting t_n^{mod}'s accordingly, we can obtain several new samples in $y(t)$'s fundamental period $[0, T]$. The next step is determining the exact number of the new samples in $y(t)$'s fundamental period $[0, T]$.

Number of New Samples: As we repeat calculating t_n^{mod} for increasing number of n's, every sample fills in a distinct point within $[0, T]$ until t_n^{mod} becomes zero again (other than $n = 0$ for which t_n^{mod} is always zero). Let M denote the smallest nonzero sample index n for which $t_n^{mod} = 0$. For any sample with index number M or greater than M, it is not possible to fill in a distinct point within $[0, T]$ because the value at t_n^{mod} is already filled by a previous sample. In other words, any subsequent sample with index $n \geq M$ does not increase the total number of distinct sample points within the fundamental period. Next step is determining the exact number of new samples M. In order to find that, we use the definition in (5): $t_n^{mod} = \text{mod}\,(n\Delta_m, T)$. Trivial solution is $t_n^{mod} = 0$ for $n = 0$, but we are looking for the sampling index $n > 0$. Then, we can observe that $\text{mod}\,(n\Delta_m, T)$ becomes zero when $n\Delta_m$ is the same as the least common multiple of Δ_m and T. Let $lcm(x, y)$ correspond to an operator whose result is the least common multiple of its operands x and y. It should be noted that we are using less strict definition of least common multiple operator, i.e, the result and operands x and y do not necessarily have to be integers, i.e., $lcm(0.01, 0.02) = 0.02$.

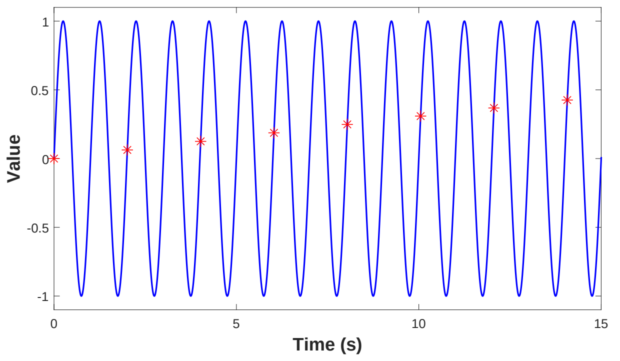
Hence, the total number of samples in one period of $y(t)$ is equal to M which can be found as:

$$M = \frac{lcm(\Delta_m, T)}{\Delta_m}. \tag{6}$$

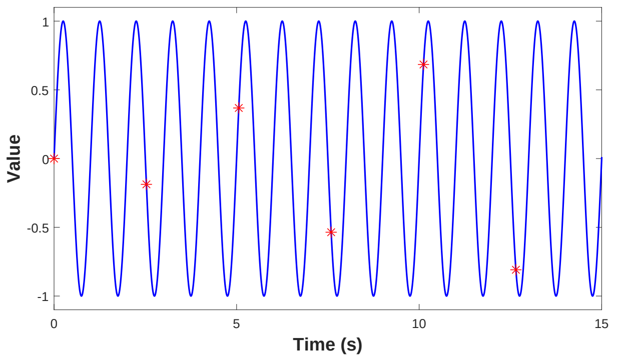
New samples are spaced by $T_s^n = T/M$, therefore, the new sampling frequency f_s^n is:

$$f_s^n = \frac{1}{T_s^n} = \frac{M}{T} = \frac{lcm(\Delta_m, T)}{T\Delta_m}. \tag{7}$$

If the total number of samples, N, is equal to or larger than M, we will have at least one sample that represents the value of the signal at $t = 0, T_s^n, 2T_s^n, ..., (T - T_s^n)$. Therefore, when this condition is satisfied, new sampling time of the signal becomes T_s^n. Consequently, it can be concluded that the effective sampling rate has been increased from $\frac{1}{T_s}$ to $\frac{1}{T_s^n} = \frac{lcm(\Delta_m, T)}{T\Delta_m}$.



(a) $y(t)$ (solid curve) and $y_{s_1}[n]$ (markers) for $T_{s_1} = 2.01$.



(b) $y(t)$ (solid curve) and $y_{s_2}[n]$ (markers) for $T_{s_2} = 2.53$.

Figure 2: The solid curves represent the continuous target signal $y(t) = \sin(2\pi t)$, whereas the markers represent $y_s[n]$ (samples obtained from $y(t)$ with sampling rates $T_{s_1} = 2.01$ (a) and $T_{s_2} = 2.53$ (b)). One should note that this is a case where the signal is heavily undersampled, which causes aliasing.

3.1 *Modulo Operation* Through Examples

In order to illustrate how *modulo operation* works, consider the following signal:

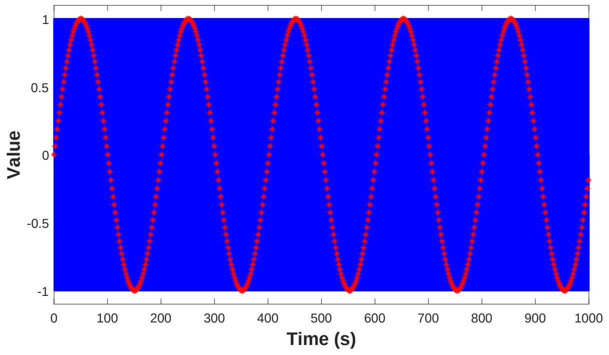
$$y(t) = \sin(2\pi t). \tag{8}$$

We consider two different cases of T_s: $T_{s_1} = 2.01$ and $T_{s_2} = 2.53$. Since the period of $y(t)$ is 1, the *modular offsets* are $\Delta_{m_1} = 0.01$ and $\Delta_{m_2} = 0.53$, respectively. After sampling $y(t)$ with sampling periods T_{s_1} and T_{s_2}, we obtain the following discrete time signals:

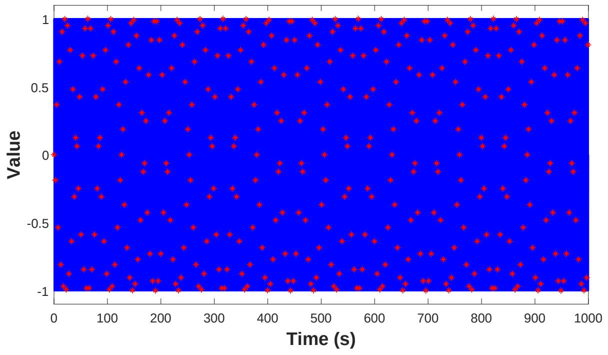
$$y_{s_1}[n] = y(nT_{s_1}) = \sin(2\pi nT_{s_1}) \quad \text{and} \quad y_{s_2}[n] = y(nT_{s_2}) = \sin(2\pi nT_{s_2}), \text{ for } \forall n \tag{9}$$

where $n \in \{0, 1, 2, ..., N-1\}$.

The solid curves in Figure 2 illustrate $y(t)$, whereas the markers in Figure 2a and 2b represent $y_{s_1}[n]$ and $y_{s_2}[n]$, respectively. One should note that for both cases $T_s > T$ and we have less than 1 sample for each period of $y(t)$. This situation presents a scenario where the sampling rate is very low with respect to the frequency of the signal under investigation. Due to aliasing, which occurs as a consequence of undresampling, $y_{s_1}[n]$ and $y_{s_2}[n]$ are completely different from each other even though they are obtained from the same signal. Without any further assumptions, $y_{s_1}[n]$ and $y_{s_2}[n]$ cannot be used to recover the target signal $y(t)$.



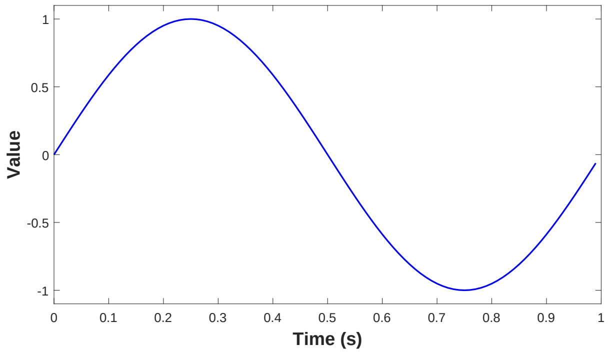
(a) $y_s[n]$ for $T_{s_1} = 2.01$.



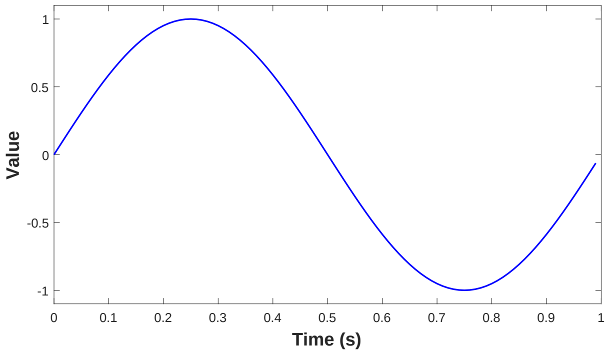
(b) $y_s[n]$ for $T_{s_2} = 2.53$.

Figure 3: Time axis is extended to $[0, 1000]$ s for Figure 2, and the markers start to resemble $y(t)$ in (a) but not in (b).

In Figure 3, time axis is extended to $[0, 1000]$ s, and one can note that the waveform that is generated by the markers in Figure 3a resembles the sinusoidal target signal $y(t)$ on a different timing scale. However, Figure 3b does not resemble $y(t)$ at all. Aliasing in Figure 3b is, therefore, very obvious, but one should also realize that aliasing occurs in Figure 3a as well. Although the shape of the markers resemble the target signal, the period of the signal generated by markers is completely different than the period of the target signal.



(a) Samples sorted (t_n^{mod}) for $T_{s_1} = 2.01$.



(b) Samples sorted by (t_n^{mod}) for $T_{s_2} = 2.53$.

Figure 4: When samples are sorted by their modular sampling timing (t_n^{mod}), the reconstructed signal fully resembles y(t) in its fundamental period for both (a) and (b).

The results obtained after applying the *modulo operation* are presented in Figure 4 and it can be noted that the reconstructed signals from both $y_{s_1}[n]$ and $y_{s_2}[n]$ are identical to $y(t)$ for $t \in [0, T]$. These figures expose the strength of the proposed approach to increase the sampling rate. We have first showed that $y_{s_1}[n]$ and $y_{s_2}[n]$ are samples obtained from the same target signal $y(t)$, but they are completely different signals due to aliasing. However, applying the *modulo operation* helped us to recover the underlying target signal in both cases. Therefore, in addition to increasing the sampling rate, *modulo operation* is very useful in resolving aliasing present in undersampled signals.

This example shows how *modulo operation* can be used to reconstruct a periodic signal in its fundamental period with higher effective sampling rate. One should note that we assume a few conditions that are required for *modulo operation*:

- The target signal is a periodic signal and its period is known or estimated.
- Modular offset ($\Delta_m = mod(T, T_s)$) is nonzero, in other words, the sampling period T_s is not an integer multiple of the period of the target signal.

For any signal that satisfies the aforementioned conditions, *modulo operation* can be applied. We now show that the signal obtained by using the setup of **Training Phase** in Section 2 indeed satisfies those conditions:

- Same instruction sequence is executed several times consecutively by using a for loop, and therefore, the signal becomes periodic during the execution of the for loop. Also, because the total number of executions of the instruction sequence is determined by us (thus, known to us), the period of the instruction sequence (the time it takes for the instruction sequence to be executed only once) can be estimated through dividing the total execution time by the number of iterations. With this specific setup, the periodicity condition can be satisfied and the period can be easily estimated.
- Sampling period, T_s, is determined by the sampling equipment. Usually, it is very unlikely to have T_s be an integer multiple of T, but even if this occurs, sampling time of the equipment can be slightly adjusted in a way to avoid T_s from being an integer multiple of T.

The discussion above shows that setup used in **Training Phase** of our experiments satisfies the required conditions for applying *modulo operation.* It is worth re-emphasizing that this setup is used for generating the EM signatures (where we have much more degree of freedom in modifying the source code), but it is not repeated for the actual instruction tracking stage (**Testing Phase**), because, the instructions are executed only once in the actual tracking stage.

Finally, it may be worthwhile to note that *modulo operation* is not increasing the real-time sampling rate. Instead, *modulo operation* uses the samples that are recorded from several repetitions of a periodic signal and creates a new signal whose time scope is limited to one period of the signal, but the new signal contains samples that are spaced much closer to each other in time.

4. EXPERIMENTAL RESULTS

In all our experiments we use Altera DE1 Cyclone II, a high-density, low cost FPGA (Field Programmable Gate Array). For implementing our code on the device, we use Altera's embedded processor Nios II that employs a relatively advanced pipeline architecture with 6 pipeline stages (Fetch, Decode, Execute, Memory, Align and Writeback) and operates at 50 MHz clock frequency. The considered instructions are given in Table 1.

In this section, we consider 20 instruction sequences listed in Table 2. Note that instructions include the basic arithmetic operations such as `ADD,SUB,MUL,DIV`, as well as load `(LDM)` and store `(STM)` instructions.

We measure the emanated EM signals with a near field magnetic probe (AAronia PBS H3) that is located above the processor of the target device as shown in Figure 5. For recording the measured data, we use a spectrum analyzer (Keysight N9020A MXA). Main reason behind using this device is the built-in features of the device to downconvert the recorded signal and the visualize the signal around the clock frequency. However, any

Table 1: Instructions used for the experiments (Altera DE1 Cyclone II).

Instruction		Description
LDM	`ldw rA, 0(rB)`	Load register rA with the memory word located at the the effective byte address
STM	`stw rA, 0(rB)`	Store rA to the memory location by the effective byte address
ADD	`addi rA, rB, IMM16`	Sum the 16-bit immediate and rB values, and store in rA
SUB	`sub, rA, rB, IMMED`	Subtract IMMED from the value of rB, and store in rA
MUL	`muli rA, rB, IMM16`	Multiply 16-bit immediate and rB values, and store in rA
DIV	`div rA, rB, rC`	Divide rB by rC and then store to rC

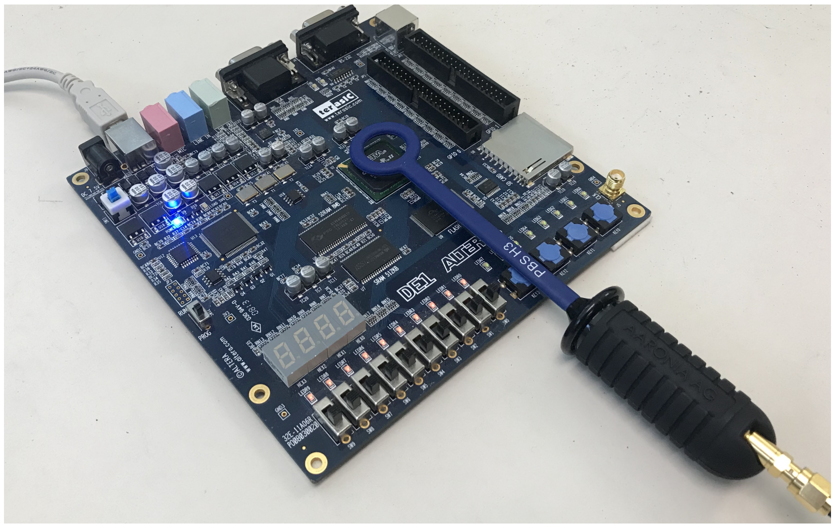


Figure 5: Measurement Setup: Magnetic near field probe is located on top of the processor of the FPGA.

other less expensive device with similar sampling rate features can be used to repeat the same experiments. We record the signal with 25.6 MHz sampling rate around 50 MHz center frequency and 10 MHz bandwidth.

Figure 6 displays the recorded signal for a specific instruction sequence (`DIV-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD-DIV`). The top and bottom plots in Figure 7 represent the *modulo operation* result and the single execution of the same instruction sequence, respectively. One should note that the plot on the top is much more smooth due to higher resulting sampling rate and the averaging nature of *modulo operation*. In fact, for this specific instruction sequence, the sampling rate is increased by a factor of 125. In order to compare those two signals, we use Pearson correlation coefficient.[22] Clearly, these two discrete time signals have different number of samples. Therefore, the *modulo operation* result is re-sampled at the same time instances of the single execution signal. Re-sampled *modulo operation* result and the original single execution result are plotted on top of each other in Figure 8 and the resulting correlation coefficient is 0.96. Note that when the same *modulo operation* result is correlated with single execution of other instruction sequences, the correlation coefficient is less than 0.9 for all different sequences.

Table 3 shows the correlation matrix between the EM signatures that are obtained with *modulo operation* and the single execution of the instruction sequences. We can note that the dominant terms in the matrix are the diagonal terms, which supports the claim that the generated EM signatures by *modulo operation* can be used to detect the corresponding instruction sequences. One should also note that, similar instruction sequences (such as instruction sequence 10 and 11, which differ only in one instruction) have higher cross correlation coefficients, but none of the cross correlation coefficients are above 0.9. Therefore, the corresponding threshold for these instruction sequences can be set as 0.9.

Table 2: Instruction sequences that are used in the FPGA experiments

Seq. No.	Sequence
1	SUB-DIV-STM-DIV-STM-MUL-LDM-MUL-SUB-MUL-ADD-DIV-LDM-DIV-ADD-MUL
2	LDM-MUL-ADD-MUL-LDM-DIV-SUB-MUL-ADD-DIV-SUB-DIV-STM-DIV-STM-MUL
3	STM-MUL-STM-DIV-SUB-MUL-LDM-MUL-SUB-DIV-ADD-MUL-LDM-DIV-ADD-DIV
4	MUL-ADD-ADD-SUB-DIV-SUB-DIV-ADD-MUL-SUB
5	ADD-SUB-ADD-SUB-DIV-SUB-DIV-ADD-MUL-MUL
6	ADD-ADD-MUL-SUB-ADD-DIV-DIV-MUL-SUB-SUB
7	SUB-DIV-SUB-DIV-MUL-ADD-MUL-SUB-ADD-ADD
8	SUB-DIV-ADD-DIV-ADD-MUL-SUB-MUL-ADD-SUB
9	ADD-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD-DIV-SUB
10	DIV-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD-DIV-SUB
11	MUL-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD-DIV-SUB
12	ADD-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD
13	DIV-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD
14	MUL-DIV-SUB-DIV-ADD-MUL-SUB-MUL-ADD
15	DIV-SUB-DIV-MUL-SUB-MUL-ADD-DIV-SUB-MUL-ADD-DIV-SUB-SUB
16	DIV-SUB-DIV-MUL-ADD-DIV-SUB-MUL-ADD-DIV-SUB-SUB
17	DIV-SUB-DIV-DIV-SUB-MUL-ADD-DIV-SUB-SUB
18	DIV-SUB-DIV-MUL-ADD-DIV-SUB-SUB
19	DIV-SUB-DIV-DIV-SUB-SUB
20	DIV-SUB-DIV-SUB

These results illustrate that single instruction level tracking is possible, even when received signals are heavily undersampled.

5. CONCLUSIONS

Earlier work have illustrated the usage of EM emanations for instruction tracking on processors with simple pipeline architectures and low operating frequencies. In this paper, we show how EM emanations can be used to track instructions on a more advanced device (an FPGA) with a processor that has a more complex pipeline structure and higher operating clock frequency.

In order to track the instructions, we generate reference signals (called as EM signatures) to be compared with the emanated signals. The complex structure of the target device has motivated us to develop a new technique and modified experimental setups to generate EM signatures. Firstly, in order to incorporate the pipeline effect, EM

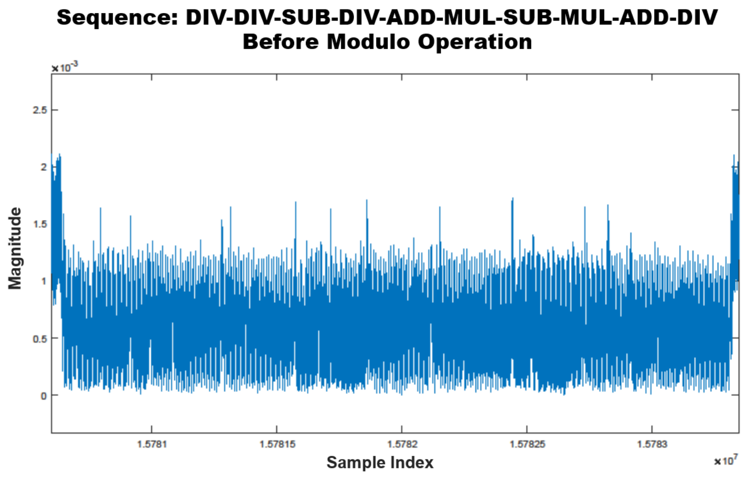


Figure 6: Recorded signal for the given sequence in **Training Phase** before *modulo operation*.

.

signatures are generated for sequences of instructions rather than single instructions. Furthermore, we propose and implement a technique called *modulo operation*, which addresses the issue introduced by low sampling rate. Results show that *modulo operation* is successful not only to obtain an EM signature with higher sampling rate, but also to eliminate possible measurement noise. Results demonstrate that a specific EM signature is highly correlated with the single execution of the same sequence and much less correlated with all other sequences. This confirms the applicability of EM signals that are generated by using *modulo operation*.

As much as the EM signatures for instruction sequences used in our experiment can be used to detect the corresponding sequences, achieving instruction tracking at instruction level necessitates generating EM signatures for all possible sequences. To this end, our experimental results show that using *modulo operation* for generating EM signatures for all possible sequences is a feasible approach to obtain high success rate.

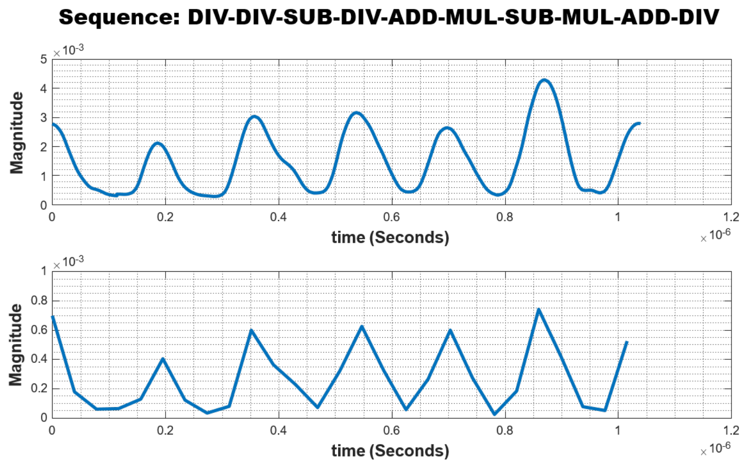


Figure 7: The plot on the top presents the result of the *modulo operation* obtained by using **Training Setup**, plot on the bottom presents the single execution of the same instruction sequence obtained by **Testing Setup**.

.

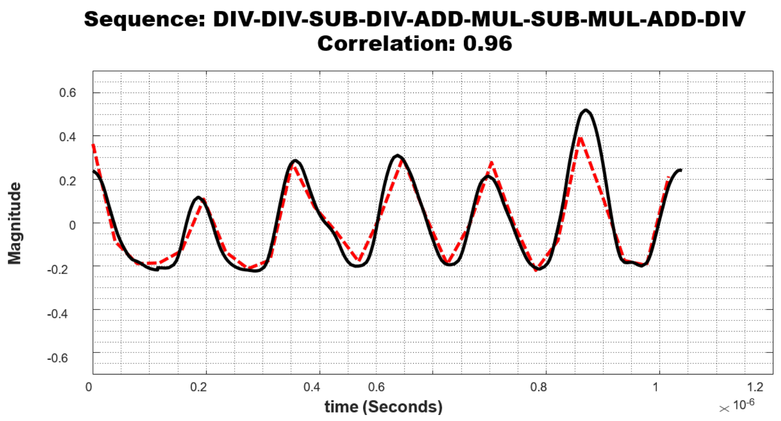


Figure 8: Resampled *modulo operation* result (solid curve) vs. single execution of the same instruction sequence (dashed curve)

Table 3: Correlation between the EM signatures and their one-time-run versions for Altera DE1 Cyclone II. The columns denote the EM signatures and the rows denote the one-time-run versions. The diagonal entries dominate the other terms, therefore, the generated EM signatures can identify the executed sequences and corresponding instructions (The values given in the table is correlation coefficient $\times$ 100).

		Generated EM signatures																			
		1A	**2A**	**3A**	**4A**	**5A**	**6A**	**7A**	**8A**	**9A**	**10A**	**11A**	**12A**	**13A**	**14A**	**15A**	**16A**	**17A**	**18A**	**19A**	**20A**
One-time-run versions	**1B**	**91**	54	73	15	23	11	21	19	28	43	21	23	7	54	18	37	27	24	14	2
	2B	12	**95**	50	16	57	43	50	21	39	56	58	6	65	58	50	57	10	46	9	18
	3B	61	67	**93**	6	55	42	50	2	18	62	66	28	52	68	24	55	8	46	5	16
	4B	19	6	20	**94**	46	61	42	73	62	10	29	73	50	14	44	4	67	27	78	66
	5B	9	53	18	52	**97**	79	80	75	72	30	49	39	55	32	53	50	55	62	61	67
	6B	11	31	2	66	80	**91**	87	85	75	16	50	67	61	16	59	43	70	72	68	79
	7B	2	43	20	32	70	80	**92**	64	73	24	54	48	50	38	61	55	55	72	54	76
	8B	27	8	34	41	30	47	38	**97**	66	19	26	83	42	21	56	2	76	24	77	66
	9B	41	3	31	57	44	62	57	80	**98**	19	37	82	62	19	76	0	90	34	68	69
	10B	22	57	53	20	55	40	44	11	20	**96**	88	22	70	76	33	83	3	47	4	11
	11B	20	49	33	44	57	53	50	50	62	65	**93**	26	82	58	58	70	46	51	47	47
	12B	12	41	48	42	10	18	7	80	52	47	7	**94**	10	50	47	33	68	14	75	66
	13B	20	41	20	59	61	60	57	69	80	36	80	45	**98**	40	62	39	75	58	61	57
	14B	33	65	72	11	65	44	46	8	31	76	83	30	81	**95**	29	62	7	49	7	10
	15B	39	20	25	34	33	46	47	64	80	3	45	66	54	4	**96**	17	70	25	56	55
	16B	13	53	41	26	53	57	66	25	36	82	78	1	60	60	48	**96**	12	66	17	36
	17B	32	25	41	57	28	51	36	78	85	38	15	82	42	32	64	21	**95**	20	74	56
	18B	12	42	27	49	61	81	91	49	58	38	48	32	54	38	45	63	45	**97**	31	53
	19B	23	11	20	46	40	42	25	77	59	23	19	66	32	16	51	9	70	4	**92**	70
	20B	8	6	17	40	46	55	59	72	66	13	23	71	34	8	57	14	58	25	77	**94**

ACKNOWLEDGMENTS

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.

REFERENCES

[1] T. Eisenbarth, C. Paar, and B. Weghenkel, *Building a Side Channel Based Disassembler*, pp. 78–99. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[2] M. Msgna, K. Markantonakis, and K. Mayes, "Precise instruction-level side channel profiling of embedded processors," in *Information Security Practice and Experience*, X. Huang and J. Zhou, eds., pp. 129–143, Springer International Publishing, (Cham), 2014.

[3] R. Callan, F. Behrang, A. Zajic, M. Prvulovic, and A. Orso, "Zero-overhead profiling via em emanations," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, pp. 401–412, ACM, 2016.

[4] N. Sehatbakhsh, A. Nazari, A. Zajic, and M. Prvulovic, "Spectral profiling: Observer-effect-free profiling by monitoring em emanations," in *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, p. 59, IEEE Press, 2016.

[5] A. Nazari, N. Sehatbakhsh, M. Alam, A. Zajic, and M. Prvulovic, "Eddie: Em-based detection of deviations in program execution," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 333–346, IEEE, 2017.

[6] H. A. Khan, M. Alam, A. Zajic, and M. Prvulovic, "Detailed tracking of program control flow using analog side-channel signals: a promise for iot malware detection and a threat for many cryptographic implementations," in *Cyber Sensing 2018*, **10630**, p. 1063005, International Society for Optics and Photonics, 2018.

[7] M. Alam, H. A. Khan, M. Dey, N. Sinha, R. Callan, A. 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 18)*, pp. 585–602, 2018.

[8] B. Lampson, "A note on the confinement problem," 1973.

[9] J. Balasch, B. Gierlichs, R. Verdult, L. Batina, and I. Verbauwhede, "Power analysis of atmel cryptomemory - recovering keys from secure eeproms," in *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pp. 19–34, 2012.

[10] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pp. 104–113, 1996.

[11] J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, pp. 292–302, 1999.

[12] J. Brouchier, T. Kean, C. Marsh, and D. Naccache, "Temperature attacks," *IEEE Security & Privacy* **7**(2), pp. 79–82, 2009.

[13] 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*, pp. 307–322, 2010.

[14] D. Genkin, A. Shamir, and E. Tromer, "RSA key extraction via low-bandwidth acoustic cryptanalysis," in *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pp. 444–461, 2014.

[15] 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*, pp. 29–45, 2002.

[16] R. Callan, A. G. Zajic, and M. Prvulovic, "A practical methodology for measuring the side-channel signal available to the attacker for instruction-level events," in *47th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2014, Cambridge, United Kingdom, December 13-17, 2014*, pp. 242–254, 2014.

[17] B. B. Yilmaz, M. Prvulovic, and A. Zajić, "Capacity of deliberate side-channels created by software activities," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pp. 237–242, IEEE, 2018.

[18] B. B. Yilmaz, R. L. Callan, M. Prvulovic, and A. Zajić, "Capacity of the em covert/side-channel created by the execution of instructions in a processor," *IEEE Transactions on Information Forensics and Security* **13**(3), pp. 605–620, 2018.

[19] R. Rutledge, S. Park, H. Khan, A. Orso, M. Prvulovic, and A. Zajic, "Zero-overhead path prediction with progressive symbolic execution," in *International Conference on Software Engineering (ICSE) 2019*, ACM, 2019.

[20] J. Park, X. Xu, Y. Jin, D. Forte, and M. Tehranipoor, "Power-based side-channel instruction-level disassembler," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

[21] D. Strobel, F. Bache, D. Oswald, F. Schellenberg, and C. Paar, "Scandalee: a side-channel-based disassembler using local electromagnetic emanations," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp. 139–144, EDA Consortium, 2015.

[22] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, pp. 1–4, Springer, 2009.