High-Level Simulation of Embedded Software Vulnerabilities to EM Side-Channel Attacks

Aditya Thimmaiah, Vishnuvardhan V. Iyer,
Andreas Gerstlauer, and Michael Orshansky

Dept. of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX 78712, USA
{auditt,vishnuv.iyer,gerstl,orshansky}@utexas.edu

Abstract. Attacks on embedded devices using the electromagnetic (EM) side channel have proliferated. Predicting software vulnerability to such attacks requires an ability to simulate EM fields during software development rather than relying on expensive lab-based measurements. We propose a modeling approach capable of synthesizing instruction-level EM traces for arbitrary software, using a one-time pre-characterization of a processor. Reducing the cost of dictionary construction is a major contribution of this paper. Results on a set of benchmarks show that synthesized traces are accurate in estimating EM emanations with less than 5% mean absolute percentage error (MAPE) compared to measurements. Furthermore, synthesized traces predict control flow leakage with an accuracy of 87% or more based on the side-channel vulnerability factor (SVF) metric.

Keywords: Embedded security · EM side-channel · EM simulation.

1 Introduction

Evaluating the security risks of software running on embedded and IoT systems has become essential. Attacks through the electromagnetic (EM) side-channel are an especially pernicious threat due to their non-invasive nature [5]. EM fields emanating from a chip during program execution can be used for recovery of cryptographic keys [4] and tracking program control flow [6]. Analysis of EM vulnerabilities are typically performed using lab-based measurements. However, lab-based EM measurement setups can be costly, time-consuming and require expertise not available to typical software developers. Therefore, an EM simulator capable of synthesizing EM traces corresponding to on-chip activity with accuracy that is sufficient for security assessment is highly desirable.

A security-focused EM simulator needs to represent information pertinent to leakage analysis. Prior work has approached the construction of such simulators specifically for application-specific integrated circuits (ASICs) concerned with implementation of cryptographic algorithms [16; 8; 9]. However, they require detailed layout simulation or rely on the working of the cryptographic algorithm to simplify simulator complexity thus limiting generality. Most of the literature on

side-channel analysis of software running on general purpose micro-controllers has been in the power domain [13; 15; 10]. These works construct a dictionary by modeling power consumption of the micro-controller as a function of switching activity in selected architectural blocks, and then using the pre-characterized dictionary to synthesize traces for arbitrary programs running on the micro-controller. However, construction of such dictionaries may become infeasible as the number of selected blocks and their interactions and dependencies increases. Current approaches therefore compromise between complexity in dictionary construction and accuracy of synthesized traces. Moreover, the additional spatial component increases complexity in the EM versus power domain.

In this paper, we seek to develop an EM simulator that is capable of synthesizing instruction-level EM traces to assess the vulnerability of arbitrary software running on embedded micro-controllers against EM side channel attacks (EMSCAs). We present a methodology that significantly reduces the cost of dictionary construction without compromising EM trace synthesis accuracy. We achieve this by hypothesizing that in a class of non-pipelined micro-controllers possessing a single bus architecture, a condensed set of architectural blocks can be identified to characterize the EM emanation during different cycles of each instruction's execution. We identify these dominant architectural blocks to model for every cycle of all instructions in the instruction set architecture (ISA) using a statistical feature selection algorithm. By constructing dictionaries at the cycle level for dominant blocks and dependencies, only the switching activity of a small subset of architectural blocks is required to be modeled, thereby combating combinatorial explosion. We demonstrate that dominant blocks and dependencies identified at one spatial location remain consistent across others, thereby simplifying feature selection overhead. We also observe that instructions accessing the same micro-architectural components may share similarities in their EM traces and hence can be clustered during a post-processing step. This allows to share cycle dictionaries across instructions and further reduce the cost of dictionary construction. In summary, the contributions of the paper are as follows:

- We propose a simulation platform that is capable of synthesizing EM traces corresponding to arbitrary software running on embedded micro-controllers, enabling testing of software against EMSCAs at development stage;
- We propose novel feature selection and redundancy removal algorithms to identify the dominant architectural blocks for each cycle and cluster cycles of different instructions with similar EM signatures to reduce the cost of dictionary construction;
- We demonstrate our synthesis approach by implementing it on an AT89S51 micro-controller belonging to the 8051-family, where synthesized traces are evaluated on five benchmark programs and shown to be in agreement with measurements with better than 95% accuracy; and
- We use the simulator to predict the side-channel vulnerability factor (SVF) of a benchmark program. We validate the result against the SVF extracted from measured traces and show that they agree with >87% accuracy.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 details the proposed synthesis flow including the feature selection and redundancy removal algorithms for dictionary construction and synthesis. Section 4 presents an accuracy evaluation of our synthesis flow and a case study of utilizing the proposed EM trace simulator for control-flow prediction. Finally, Section 5 concludes the paper with a summary and an outlook on future work.

2 Related Work

EM trace simulators specifically for ASICs concerned with cryptographic algorithms have been studied in [16] and [8]. The authors investigated a pre-silicon approach by using 3D full-wave EM field solvers on a given chip layout to simulate the EM fields. Although these approaches enable prediction of susceptibility to EMSCAs during the hardware design stage, they are limited by the large overhead of simulating the entire chip layout [11]. Moreover, the study in [16] dealing with ASICs for the advanced encryption standard (AES) simplifies the simulation complexity by only considering the circuit state corresponding to the final round of AES targeted by key recovery attacks. This dependency of the simulation on the working of the target algorithm limits their applicability. An extension of this approach to a 16-bit general purpose micro-controller was presented in [9]. However, they were only able to show that the differential traces between simulation and measurement at the instruction level (difference between the traces of an instruction for its extreme values) share a similar trend. Furthermore, all such approaches require-layout related information, which may not always be available.

In the power domain, a post-silicon approach to modeling security-related power consumption of a general purpose micro-controller was presented in [13]. Power consumption at the instruction level was modeled via the switching activity of specific architectural blocks, which were assumed to be sufficient for capturing the power consumed by the micro-controller as a whole. However, the presented approach may become computationally infeasible as the number of considered architectural blocks increases, since the traces required to simulate the power consumption would need to be modeled under all combinations of the considered blocks and their dependencies. An attempt to address these issues was made in [15]. Rather than model the power consumption at the assembly language instruction level as in [13], they modeled power consumption only during assignment, arithmetic, relational and bitwise operations of a higher-level programming language (C++) used to program the micro-controller. Dictionaries for synthesizing traces corresponding to these operations were then constructed by identifying the most suitable leakage model (Hamming Weight or Hamming distance between old and new value of the variable manipulated by these operations) through correlation with the measured power trace. Therefore, accuracy in simulation of power consumed was traded off for simplicity of dictionary construction (since only a change in a single software variable as opposed to changes in several different hardware architectural blocks is considered). Our approach

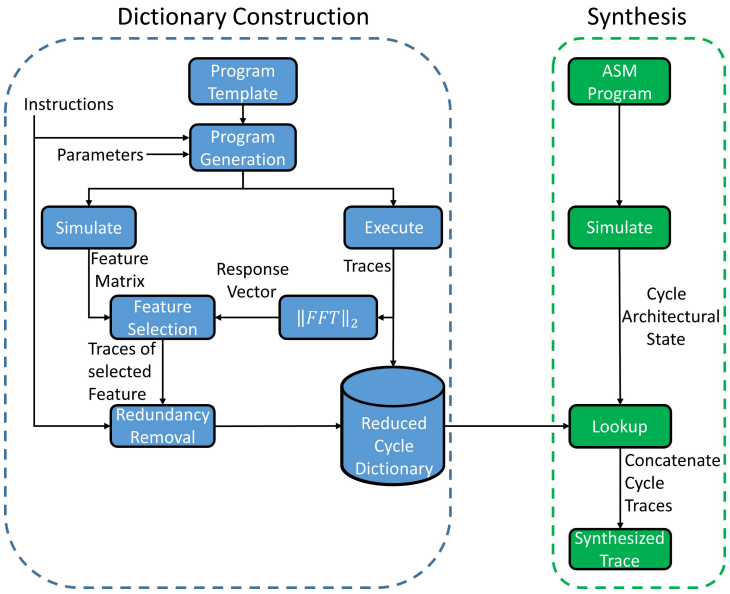

Fig. 1. Overview of our proposed EM trace synthesis approach.

combats combinatorial explosion of [13] by refining the initially selected architectural blocks using feature selection to identify the dominant blocks to model for each cycle. Moreover, since we construct the dictionary at a lower level of abstraction than [15], the degradation in accuracy of synthesis is not significant.

An EM trace simulator was presented in [10] that uses a grey box modeling approach (combining hardware implementation knowledge and statistical analysis of leakage traces) similar to ours. They introduce a post-processing step to cluster dictionary entries sharing a similar leakage profile, which we similarly utilize in our work to reduce the cost of dictionary construction. However, their investigation of synthesis of EM traces is limited to one spatial location, specifically the one closest to the power pins. Moreover, only results documenting correlation between locations (in cycles) of peaks of measured and simulated EM traces is presented. By contrast, we demonstrate that our approach is capable of synthesizing EM traces across multiple spatial locations and document correlation between synthesized and measured traces for benchmarks at arbitrary cycle-level granularity.

3 EM Synthesis Flow

Fig. 1 presents an overview of our proposed synthesis flow. Our approach consists of two stages, *Dictionary Construction* and *Synthesis*. In the dictionary construction stage, the micro-controller is pre-characterized to construct a dictionary capable of synthesizing EM traces. First, micro-benchmarks that execute

the instruction being characterized in different contexts are generated from program templates under varying values of the parameters assumed to define the architectural state of the micro-controller, such as contents of the internal RAM (data memory), the internal ROM (program memory), the program counter, etc. The micro-benchmarks are then executed on a cycle-level simulator and a physical chip instance of the selected micro-controller to collect prediction features and EM trace measurements, respectively.

EM emanations of the real chip during micro-benchmark executions are sensed using an H-field probe and acquired. The acquired EM traces are subsequently partitioned into sub-traces of cycle length. We apply a transform (L^2-norm of the Fast Fourier Transform (FFT)) to the cycle-length traces to obtain a response variable. This transform allows us to combat any jitter in the acquired traces that may affect the feature selection process. In parallel, a cycle-level simulator is used to identify the architectural state of the CPU before and after the execution of every instruction appearing in the micro-benchmarks. The architectural state for a given cycle forms a *Feature Vector* (vector containing values of the considered parameters), and the *Feature Vectors* and their corresponding response variables are concatenated across cycles to obtain the *Feature Matrix* and the *Response Vector*, respectively.

In the final step of dictionary construction, we apply feature selection to identify the most important features in the *Feature Matrix* that best describe the *Response Vector*. Based on the results of feature selection, the collected cycle-length traces corresponding to the distinct combinations of the identified most important features are stored in the dictionary as individual entries. Finally, a post-processing step is performed to cluster the dictionary entries with similar leakage profiles to allow sharing of cycle dictionaries across instructions. The identification of such similarities allows further reduction of dictionary construction time across spatial locations.

The synthesis stage is then concerned with the synthesis of the EM trace corresponding to a given arbitrary assembly program using the dictionary constructed during pre-characterization. A given program is executed on the cycle-level simulator and the values of the dominant parameters are collected for every cycle based on the instruction appearing at that cycle. These dominant parameters are then used to lookup the dictionary entry for the corresponding stored EM trace. This process is repeated for every instruction in the program and the retrieved traces are concatenated to produce the synthesized trace.

3.1 Dictionary Construction

The power consumption and hence EM emanations of a micro-controller are generally highly correlated with the switching activity of its architectural blocks, which are in turn correlated with the switching activity at the block inputs. The architectural state at a given instant may be defined as a collection of values at the inputs of these architectural blocks at that instant. The architectural state is generally composed of the internal RAM, program memory (ROM), program counter (PC) and other registers, etc. Although each of their switching

activity may contribute to the total power consumed during the execution of an instruction, some may be more dominant than others during different cycles. The dependency of power consumption on switching activity is generally described using *Hamming Weight* (HW) and *Hamming Distance* (HD) models [7]. In the following, we use the word *parameter* to refer to the HW or HD of the inputs of architectural blocks.

A *Baseline* approach to constructing a dictionary would involve collecting traces for all possible combinations of the values assumed by these parameters for all instructions and blocks active in any given cycle. However, as the number of parameters considered increases, the number of traces required to cover all possible combinations may become infeasible to measure experimentally. Instead, we propose using feature selection to identify the dominant parameters during the different cycles of an instruction and hence collect traces corresponding to these cycles. In this paper, we target EM trace synthesis for embedded micro-controllers. In non-pipelined micro-controllers, only one instruction is active in any cycle. Furthermore, in a single-bus architecture, each instruction executes in multiple cycles and only a limited number of blocks is active in each cycle. We propose a feature selection algorithm to identify the dominant blocks and parameters for each instruction and cycle. With this, the number of traces required to characterize a cycle of an instruction is proportional to the number of values assumed by its dominant parameters.

In addition, if these dominant parameters identified for different cycles of an instruction remain consistent spatially, then the cost of feature selection is incurred only for a single spatial location. Smaller dictionaries for other spatial locations can be constructed making use of this dominant feature information.

Feature Selection In this section, we demonstrate the application of feature selection to identify the dominant parameters and show that the results obtained at one spatial location remain consistent across other locations.

In this paper, we use *RReliefF* [12] as the feature selection method. To apply feature selection to identify the dominant parameters for different cycles of an instruction, we need the *Response Vector* and the *Feature Matrix*. The feature matrix contains all the values assumed by the considered parameters during the execution of this instruction, while the response vector contains the L^2-norm of the FFT of cycles in the corresponding measured EM trace. The feature matrix and the response vector are then fed to *RReliefF*, which assigns weights to each parameter/feature considered in the feature matrix based on its relevance in predicting the response vector. Features assigned with close to zero or negative weights are considered to be statistically irrelevant and hence can be disregarded from further analysis. To select important features amongst those with positive weights, *RReliefF* employs a threshold τ [14] known as the relevance level. The features/parameters whose weights exceed the relevance level are the dominant parameters. Since we apply feature selection to each cycle separately, the dominant parameters are identified for each cycle of each instruction. Therefore, a dictionary required to synthesize the EM traces for the cycles of an instruction

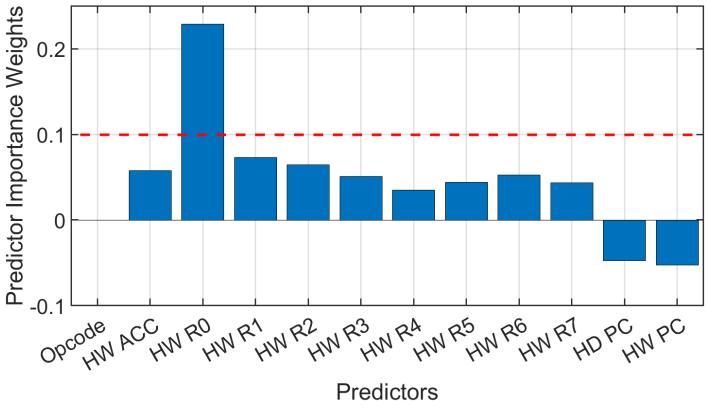

Fig. 2. Ranking of features for the third cycle of *NOP*.

can be constructed by collecting EM traces for distinct values assumed by the dominant parameters in each cycle as opposed to collecting EM traces for all possible values assumed by combinations of all considered parameters.

We now demonstrate the application of feature selection in identifying the dominant parameters for the third cycle of the *NOP* instruction of the micro-controller used in our experiments (see Section 4). The parameters considered for feature selection are the HW of the Accumulator (ACC), the HWs of General Purpose Registers (GPRs) R0-R7, the HW and the HD of the PC, and the HW of instruction-specific operands. The training set required to construct the feature matrix and the response vector are obtained by collecting the EM traces during the execution of the *NOP* instruction under varying values for the considered parameters (encompassing all possible HWs and HDs). The response vector is computed by applying the L^2 norm to the Fast Fourier Transform (FFT) of the sections of acquired traces corresponding to the third cycle, whereas the HW of ACC, HW of GPRs, and HW and HD of the PC in the third cycle form the feature matrix. Fig. 2 shows the weights assigned by *RReliefF* to the considered features and for a chosen relevance level (τ) of 0.1. It is evident that the HW of R0 is the most important feature/dominant parameter, while all other features/parameters can be discarded.

To validate the sufficiency of using a single dominant parameter to build a dictionary for the third cycle of *NOP*, we compare the performances of models trained with

1. Only the highest ranked feature
2. Only the second highest ranked feature
3. A combination of the highest and second highest ranked features
4. No feature, i.e. only using an averaged trace

The performance of a model in this paper is evaluated by computing the mean absolute percentage error (MAPE) between the model forecast trace and

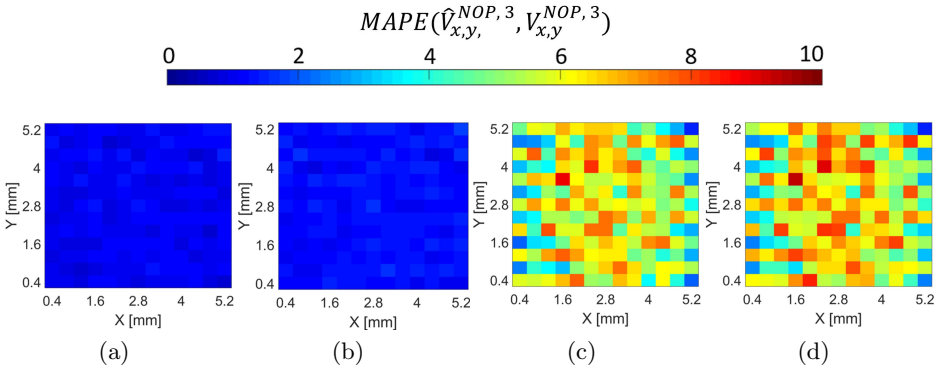

Fig. 3. Spatial variation of MAPE between true and model forecast traces for the third cycle of *NOP* for models trained with different features. The MAPE here has been averaged over 500 traces for each spatial location. (a) HW of R0, (b) HW of R0 and R1, (c) HW of R1, and (d) no feature/averaged, models.

the true trace as,

$$MAPE(\hat{V}^{i,k}_{x,y}, V^{i,k}_{x,y}) = \frac{100}{n} \times \sum_{t=1}^{n} \frac{|v^{i,k}_{x,y,t} - \hat{v}^{i,k}_{x,y,t}|}{|v^{i,k}_{t,x,y}|}, \tag{1}$$

where $\hat{V}^{i,k}_{x,y}$ and $V^{i,k}_{x,y}$ are the model forecast and true measured test trace respectively, for the k^{th} cycle of instruction i at the spatial location (x, y) containing n samples $\hat{v}^{i,k}_{x,y,t}$ and $v^{i,k}_{x,y,t}, t = 1 \ldots n$. From Fig. 3 we see that the model trained with the highest ranked feature (Fig. 3(a)) is comparable in performance to that of a two feature model (Fig. 3(b)). Therefore, since the degradation in accuracy in using a single feature model for the third cycle of *NOP* relative to the true trace is not significant, we can considerably reduce the cost of dictionary construction by using a single feature model. By contrast, models using no feature (Fig. 3(d)) or only the second highest ranked feature (Fig. 3(c)) show poor accuracy. This confirms the need for proper feature selection.

Furthermore, based on the observation that these results hold true across spatial locations, we can offset the cost of feature selection by utilizing the important feature identified at one spatial location to build dictionaries at others.

Redundancy Removal We can further reduce the complexity of dictionary construction by identifying similarities in cycle dictionaries across instructions. If two or more instructions share the same dominant parameters for a particular cycle and their EM traces are similar, then the dictionary for that cycle can be shared. We identify such potential redundancies in cycle dictionaries of two instructions by comparing their respective cycle models after *Feature Selection*. The models for cycles at a specific spatial location, sharing the same dominant parameters are compared using the MAPE metric. A low maximum MAPE

Table 1. MAPE between model forecast traces for instruction-cycles of *NOP* and *MOV A, R0* instructions at spatial location (5mm, 5mm).

Instruction Cycle (k)	Dominant Parameters		Maximum MAPE($\hat{V}^{NOP,k}$, $\hat{V}^{MOV,k}$)
	NOP	MOV A, R0	
1	HD of PC	HD of PC	15.8%
2	HW of ACC	HW of ACC	0.3%
3	HW of R0	HW of R0	1.2%
4	*N/A**	*N/A**	*N/A**
5	*N/A**	HW of R0	*N/A**
6	HW of ACC	HW of ACC	0.5%

* Dominant parameters of cycles for which variation in traces are statistically insignificant with considered parameters are designated as *N/A*.

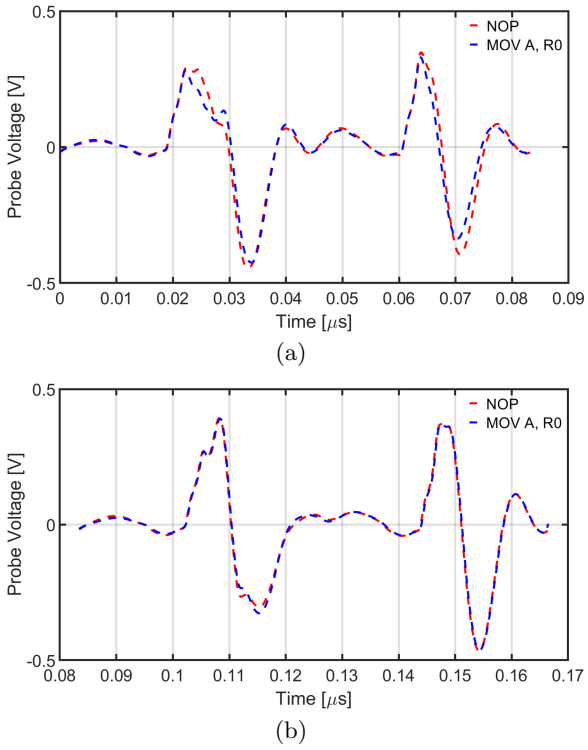

Fig. 4. EM traces for the first and second cycles of *NOP* and *MOV A, R0* at the spatial location (x,y)=(5mm,5mm): (a) EM traces for the first cycle show a MAPE($\hat{V}^{NOP,1}$, $\hat{V}^{MOV,1}$) of 10.6% for a HD value of the PC of 2, (b) EM traces for the second cycle show a MAPE($\hat{V}^{NOP,2}$, $\hat{V}^{MOV,2}$) of 0.01% for a HW value of the ACC of 0.

across all values of all dominant parameters implies that the models are similar and hence can be used as an indicator for potential sharing.

We demonstrate this optimization on the instructions *NOP* and *MOV A, R0* in our micro-controller. The dominant parameters for different cycles of *NOP* and *MOV A, R0* are given in Table 1. We see that the first, second, third and sixth cycles share the same dominant parameters between the two instructions. Therefore, the dictionaries for these cycles can potentially be shared. From the computed maximum MAPE metric between cycle models of *NOP* and *MOV A, R0* across all possible HWs and/or HDs of all dominant parameters, it is evident that dictionaries for the second, third and sixth cycles can be shared between the two instructions. By contrast, since the first cycle shows a large MAPE, separate dictionaries have to be built for this cycle for the two instructions. This result is visualized in Fig. 4, where the traces for the first cycle of the two instructions show a large difference as shown in Fig. 4(a), while traces for the second cycle agree with each other (Fig. 4(b)). Therefore, for this example, we can reduce the dictionary cost for one of the instructions by 50% provided we have the dictionary for the other.

3.2 Synthesis

We finally discuss the procedure for using this dictionary to synthesize complete traces of an assembly program. Given an assembly program, we execute it using a cycle-level simulator. During the execution of an instruction, we collect the values assumed by the dominant parameters (identified during the feature selection process) for all the cycles of that instruction. We then look up the dictionary for the EM traces corresponding to these values of the dominant parameters for each cycle of the instruction. These EM traces are then concatenated to synthesize the EM trace for that instruction. This procedure is repeated till the end of execution of the program to synthesize the complete trace.

4 Experiments

We evaluate our approach using an AT89S51 belonging to the 8051 family of micro-controllers built around the Intel C-51 ISA. The device operates at a clock frequency of 2MHz and is programmed serially via an ISP programmer. The EM field emanated from the device is measured using a 1mm radius H-field probe RF-U 2.5-2 from Langer. Riscure's EM Probe Station is used to vary the spatial location of the probe over the micro-controller's area (10mm $\times$ 10mm). A PA303 30dB pre-amplifier is used to amplify the measured EM traces. We sample traces at 5GS/s using a Keysight DSOS054A oscilloscope with a bandwidth of 500MHz. Each acquired trace is averaged 200 times before being processed. Our experimental setup is shown in Fig. 5.

The 8051 micro-controller architecture includes a register file with 8 GPRs numbered R0-R7 and an Accumulator register ACC. Each of these registers are 8 bits wide. The GPRs along with ACC are addressed as part of the internal RAM,

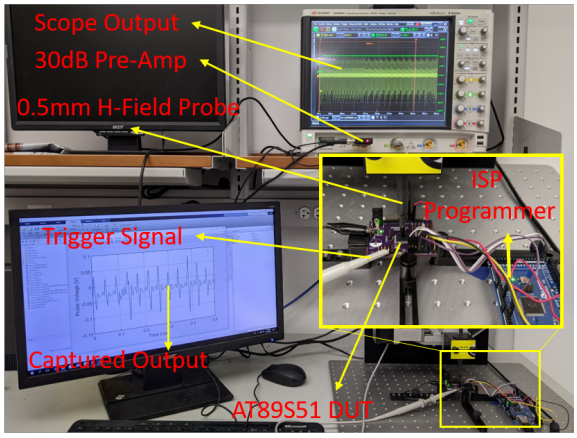

Fig. 5. Measurement setup used for acquisition of traces.

which is 128 bytes in size. An assembly instruction in the C-51 ISA can take any of the GPRs, the ACC or any RAM address as an operand. Consequently, their values can be manipulated by the execution of an instruction hence contributing to switching activity. Therefore, we consider these as the parameters during the micro-benchmark generation sub-stage. As stated before, the micro-benchmarks are used for collecting training data for the feature selection sub-stage. They are generated from a standard template and consist of two sections, an initialization section where random values are assigned to the considered parameters and the execution section that contains the instruction being characterized. Listing 1 shows an example micro-benchmark for collecting training data to identify the dominant parameters for the cycles of the *NOP* instruction. Traces of the instruction being characterized for varying values of the HWs and the HDs of the PC are collected by including multiple executions of the instruction in the template (for brevity, only two such *NOP* instructions are shown in Listing 1).

The accuracy of our EM simulator is evaluated by comparing the synthesized and measured traces for a set of standard benchmark programs from the Dalton benchmark suite [2]. The Dalton benchmarks were originally developed to optimize the power consumption of 8051 cores.

4.1 Accuracy and Dictionary Cost

We first evaluate the degradation in accuracy and the reduction in cost of dictionary construction due to the *Feature Selection* optimization alone. Dictionaries capable of synthesizing traces for all cycles of all the instructions in the ISA were constructed using just the feature selection optimization, where we applied *RReliefF* with 10 nearest neighbors and a relevance factor of $\tau = 0.1$.

To evaluate accuracy at instruction and cycle granularity, we generate 100 assembly programs, each containing all the instructions in the ISA in random

```
PRECHAR:    CPL P1.3        ; For Triggering the Oscilloscope
            CPL P1.3

            MOV A , #1d     ; Random Values for considered parameters
            MOV R0, #40d
            MOV R1, #15d
            CLR C
            MOV R2, #159d
            MOV R3, #255d
            MOV R4, #0d
            MOV R5, #119d
            MOV R6, #64d
            MOV R7, #50d

            NOP             ; Instruction being characterized
            NOP
            ...
```

Listing 1: Micro-benchmark to collect training data for the *NOP* instruction.

order and with random values for all the considered parameters. EM traces for each program were synthesized using our flow and measured on the device in our lab setup. Each of the 100 synthesized and test traces corresponding to the same assembly programs are then divided into instruction- and cycle-level traces indexed using the i, k scheme described earlier. Finally, the synthesized and measured traces are compared at the instruction and cycle level using MAPE to compute accuracy. The MAPE results corresponding to the same instruction-cycle are then averaged across cycles of the same type in the 100 assembly programs. A maximum MAPE is similarly computed. Table 2 summarizes the average and maximum MAPE results for the first six instruction-cycles of all instructions. We see from the tabulated results that the average number of dominant parameters for any given cycle across all instructions is a small subset of the total considered parameters, which is consistent with our argument for feature selection. Furthermore, the degradation in accuracy between the synthesized and measured test traces is not substantial. However, compared to a baseline dictionary that would require 10^{12} entries, the total number of traces required to construct a dictionary for the whole ISA is reduced by eight orders of magnitude.

We next perform a similar evaluation of our approach when the *Redundancy Removal* optimization is used in addition to *Feature Selection*. Table 3 summarizes the result for the first six instruction-cycles of all instructions in the ISA. The table shows the number of distinct instructions along with accuracy and dictionary size results. Results show that dictionaries created for a small subset of instructions are sufficient to synthesize traces for all the other instructions in the ISA. We observe that the degradation in accuracy is slightly higher than when using the *Feature Selection* optimization alone, but is still not significant. However, the reduction in the number of dictionary entries is substantial. We only need to measure and record on the order of 10^3 traces to construct the

Table 2. Feature selection summary.

Instruction Cycle	Average No. of Dominant Param.	MAPE Avg	MAPE Max	No. of Dict. Entries
1	2	1.0%	3.4%	2,560
2	1	1.1%	3.9%	2,304
3	1	1.1%	3.8%	2,304
4	0.9	1.6%	3.6%	8,820
5	0.7	0.3%	2.6%	2,187
6	1.1	1.6%	3.3%	3,278

Table 3. Redundancy removal summary.

Instruction Cycle	No. of Instructions	MAPE Avg	MAPE Max	No. of Dict. Entries	Reduction rel. to feature sel.
1	9	1.1%	4%	90	96.5%
2	1	2.3%	4.9%	9	99.6%
3	1	1.3%	4.2%	9	99.6%
4	7	1.2%	3.8%	541	92.9%
5	6	1.7%	4.2%	54	97.5%
6	24	2.6%	4.8%	353	89.2%

entire dictionary using this optimization. This represents an over 90% reduction in dictionary size compared to feature selection alone, and a nine order of magnitude reduction compared to the baseline approach.

With the effect on accuracy and cost due to the optimizations studied, we now proceed to validate our synthesis approach on real-world benchmark programs. Accuracy is computed using the MAPE between the synthesized trace and test trace across all samples in each benchmark. Table 4 shows that the synthesized traces agree well with the test traces across the benchmark programs. Overall, synthesized traces agree with measurements with less than 5% MAPE.

Table 5 compares the computational resources required by each of the considered approaches. To ensure a fair comparison, only one trace required for building the dictionary is recorded for each micro-benchmark. All traces are stored as double precision arrays with each trace stored in the dictionaries requiring ≈3KB of disk space. As discussed above, the total number of traces required to build a dictionary for the entire ISA using the *Redundancy Removal* optimization is 9 orders of magnitude lower than the *Baseline* approach. Consequently the improvement in memory and time required is substantial.

4.2 Case Study: Compile-Time Control Flow Prediction

We further performed a case study to show the applicability of the synthesis flow in predicting the vulnerability/information leakage of a software program. We use the side-channel vulnerability factor (SVF) metric from [3] to measure the control flow leakage of the *GCD* benchmark program from [2], a snippet of

Table 4. MAPE between test and synthesized traces for benchmark programs.

Benchmarks	Size (Bytes)	MAPE
GCD	55	3%
FIB	303	3%
SORT	572	4.2%
SQRT	1167	2.8%
MATRIX	490	2.7%

Table 5. Resources for a single spatial location (5GHz sampling frequency).

Construction Method	Traces	Memory	Time
Baseline	10^{12}*	3 PB*	$\approx 5\times10^{9}$ h*
Feature Selection	10^{4}	40 MB	60 h
Redundancy Removal	10^{3}	4 MB	6 h

*Projected

which is shown in Listing 2. Control-flow leakage here refers to the certainty with which the *If-Else* block executions can be predicted using the information available through the EM side-channel. Computation of SVF requires two inputs, the *Oracle Distance Vector* and the *Side-Channel Distance Vector*. An *oracle* trace is the ground-truth pattern of *If-Else* block executions. Corresponding synthesized or measured EM traces are the *side-channel* traces. The *Oracle Distance Vector* is computed by applying a distance function to the elements of the oracle trace in a pairwise manner [1], where we choose XOR as the distance function in this work. The *Side-Channel Distance Vector* is computed in a similar manner from the side-channel trace elements, where we use the Euclidean distance between the L^2-norm of the FFT of elements as the distance function. Finally, the information leakage is estimated by computing the Pearson Correlation between the oracle and side-channel distance vectors. We computed cycle-level SVF using both synthesized and measured EM traces for six variations of the GCD benchmark using register or direct addressing and replacing the instruction at the *IF* label in Listing 2 with variants with different leakage. Results in Table 6 show that for both register and direct addressing, the measured and synthesized SVF agree with at least 87% accuracy.

5 Summary and Conclusions

In this paper, we presented a methodology to allow synthesis of security-relevant EM traces at instruction and cycle granularity for the class of micro-controllers possessing a single bus architecture. Different EMSCAs require varying levels of granularity in the EM traces to pose a threat to the running software. Our proposed synthesis flow allows EM traces to be synthesized at a granularity similar to that observed using measurement. Thus, we can equip software design

```
LOOP:   MOV  A, R6
        ...
        SETB C
        SUBB A, R7
        JC   ELSE
IF:     CLR  C      ;;;If-Block;;;
        MOV  A, R6
        SUBB A, R7
        MOV  R6, A
        SJMP LOOP
ELSE:   CLR  C     ;;;Else-Block;;;
        MOV  A, R7
        SUBB A, R6
        MOV  R7, A
        SJMP LOOP
```

Listing 2: Disassembled GCD program.

Table 6. Correlation of cycle-level SVF between measured and synthesized traces.

Addressing mode	IF-Instruction	SVF correlation
Register	CLR C	87%
Register	NOP	87%
Register	MOV A, R6	87%
Direct	CLR C	93%
Direct	NOP	92%
Direct	MOV A, 0x08	90%

environment with EM side-channel awareness to predict and hence fortify the software against EMSCAs during the design cycle of the software. The cost of pre-characterization was reduced through the application of feature selection to identify dominant features and redundancy removal to share dictionaries across instructions. The cost associated with collecting the training data set for feature selection was justified by demonstrating that the identified dominant features remain consistent spatially and hence the information obtained at one spatial location can be utilized to construct dictionaries at others. Traces synthesized using the proposed approach were shown to be accurate with less than 5% MAPE compared to measurements, and a case study of control-flow leakage prediction showed an agreement with measured test traces with better than 87% accuracy. In future work, we plan on extending our approach to pipelined micro-controllers.

Acknowledgments

This work was supported in part by NSF grant CCF-1901446.

Bibliography

[1] Arsath K F, M., Ganesan, V., Bodduna, R., Rebeiro, C.: PARAM: A Microprocessor Hardened for Power Side-Channel Attack Resistance. In: (HOST) (2020)

[2] Dalton-Project: Benchmark Applications for Synthesizeable VHDL Model, University of California Riverside http://www.ann.ece.ufl.edu/i8051/i8051benchmarks/index.html

[3] Demme, J., Martin, R., Waksman, A., Sethumadhavan, S.: Side-channel vulnerability factor: A metric for measuring information leakage. In: (ISCA) (2012)

[4] Genkin, D., Pachmanov, L., Pipman, I., Tromer, E.: ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs. In: (CT-RSA) (2016)

[5] Getz, R., Moeckel, B.: Understanding and eliminating EMI in microcontroller applications (1996)

[6] Han, Y., Etigowni, S., Liu, H., Zonouz, S., Petropulu, A.: Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. In: (CCCS) (2017)

[7] Iyer, V.V., Yilmaz, A.E.: Using the ANOVA F-Statistic to Isolate Information-Revealing Near-Field Measurement Configurations for Embedded Systems. In: (EMC+SIPI) (2021)

[8] Kumar, A., Scarborough, C., Yilmaz, A., Orshansky, M.: Efficient simulation of EM side-channel attack resilience. In: (ICCAD) (2017)

[9] Li, H., Markettos, A., Moore, S.: Security evaluation against electromagnetic analysis at design time. In: (HLDVT) (2005)

[10] McCann, D., Oswald, E., Whitnall, C.: Towards Practical Tools for Side Channel Aware Software Engineering: 'Grey Box' Modelling for Instruction Leakages. In: (USENIX Security) (2017)

[11] Menichelli, F., Menicocci, R., Olivieri, M., Trifiletti, A.: High-level side-channel attack modeling and simulation for security-critical systems on chips. (IEEE TDSC) **5**(3), 164–176 (2008)

[12] Robnik-Sikonja, M., Kononenko, I.: An Adaptation of Relief for Attribute Estimation in Regression. In: (ICML) (1997)

[13] Thuillet, C., Andouard, P., Ly, O.: A Smart Card Power Analysis Simulator. In: (CSE) (2009)

[14] Urbanowicz, R.J., Meeker, M., La Cava, W., Olson, R.S., Moore, J.H.: Relief-based feature selection: Introduction and review. Journal of Biomedical Informatics **85**, 189–203 (2018)

[15] Veshchikov, N.: SILK: High Level of Abstraction Leakage Simulator for Side Channel Analysis. In: (PPREW) (2014)

[16] Yoshikawa, M., Asai, T.: Platform for Verification of Electromagnetic Analysis Attacks against Cryptographic Circuits. In: (ITNG) (2013)