Two's Complement: Monitoring Software Control Flow using Both Power and Electromagnetic Side Channels

Michael Amar*
Ben-Gurion University of the Negev
Beer Sheva, Israel
amarmic@post.bgu.ac.il

Lojenaa Navanesan*
Ben-Gurion University of the Negev
Beer Sheva, Israel
lojenaa@bgu.ac.il

Asanka Sayakkara
University of Colombo School of Computing
Colombo, Sri Lanka
asa@ucsc.cmb.ac.lk

Yossi Oren
Ben-Gurion University of the Negev
Beer Sheva, Israel
yos@bgu.ac.il

***Abstract*—Embedded devices leak information about their inner activity through power and EM side channels. A defender who measures this leakage can thus use it to monitor the device and ensure its control-flow integrity. Previous works have investigated the use of power and EM side channels for control-flow monitoring, but they have only used a single side channel at a time. In this paper, we propose an approach that integrates both power and EM side channels to detect deviations from the device's normal behavior. Our model takes inspiration from multimodal machine learning used in image and speech recognition, and uses an intermediate integration design which passes multiple input modalities in parallel through a single self-attention transformer network. We evaluate our model on an off-the-shelf device at multiple noise levels, and show that it outperforms models that use only a single channel as input. In particular, we show how the multimodal approach can improve trace classification and anomaly detection accuracies by up to 18% and 11%, respectively, compared to power/EM-only approaches. Additionally, we show that our approach is superior over the early and late integration approaches currently used in multimodal side channel analysis work. We release our machine-learning architecture, including trained models based on real-world data, as an open-source repository. Our work highlights how advances in the wider field of machine learning can be used to improve the security of embedded systems.**

***Index Terms*—Side channel analysis, Malware detection, Embedded devices, Deep learning, Multi-model architecture**

I. Introduction

Control-flow monitoring is a defensive technique that aims to ensure that a system's real-time execution pattern conforms to a known control-flow graph (CFG) [15]. Cyber-physical systems (CPS), which sense and interact with the environment, are natural candidates for control-flow monitoring defenses, as they tend to execute a small amount of code in a closed loop, making their CFGs relatively easy to determine. While control-flow monitoring was originally carried out by specialized hardware or software components which directly interact with the CPU, recent work has shown how both power and electromagnetic (EM) side channels can be used to implement relatively non-invasive control-flow monitoring defenses [12, 7, 13, 9]. The basic approach employed by these works is to collect a large amount of power/EM traces while the system is in a benign state, and use the traces to train a machine learning (ML) classifier. Then, in the online phase, this classifier is used to determine whether the current power/EM trace conforms to the correct control flow of the program. Since the power and EM traces are collected by external hardware, these defenses do not require any source code modifications to the device under test (DUT), and cannot be trivially disabled by malware which overrides the control-flow monitoring logic. Since the process of adding a power or EM probe to a DUT is relatively simple, a defender can attempt add both power and EM probes to the DUT simultaneously, in an attempt to produce a more effective defense. Unfortunately, state-of-the-art side channel-based control-flow monitoring approaches are limited to using only a single side channel, a limitation we address in this work.

Several works have explored the simultaneous use of multiple side channels for cryptanalytic attacks [18, 3, 17, 4]. These works have shown that using multiple side channels can significantly improve the effectiveness of cryptanalytic attacks. Unfortunately, their approaches are not directly applicable to control-flow monitoring. In particular, most of these works apply an "early integration" model to the multiple side-channel traces [16], combining them into a composite signal before entering the ML pipeline. As we show in this paper, this approach is suboptimal for the control-flow monitoring use case.

In this work we describe a control-flow monitor based around the concept of "intermediate integration", a method

This research is supported by the U.S.-Israel Energy Center managed by the Israel-U.S. Binational Industrial Research and Development(BIRD) Foundation.

*Both authors are considered co-first authors which is currently being used to great success to process multimodal data in the fields of natural language processing (NLP) and computer vision. In an intermediate integration model, multiple input modalities are passed in parallel through a single self-attention transformer network [20]. As we show in this work, the intermediate integration approach is well-suited for the control-flow monitoring use case. More specifically, we introduce a multimodal control-flow monitor based on this approach, and show that it outperforms both of the single-modality sensors it is based on, as well as early- or late-integration multimodal approaches. More specifically, the contributions of this work are as follows:

- We propose a multimodal control-flow monitoring defense that uses a self-attention transformer network to combine EM and power consumption traces through intermediate integration (Section III-D).
- We evaluate our approach on an off-the-shelf device and show that our multimodal approach outperforms single-modality classifiers, providing an increased accuracy rate of up to 18% compared to the EM/Power only approach, depending on the system noise (Section IV-A).
- We compare our intermediate integration approach to early- and late-integration multimodal approaches, and show that our method has better performance (Section IV-B).
- We release our machine-learning architecture, including trained models based on real-world data, as an open-source repository [2].

Our work highlights, yet again, how advances in the wider field of machine learning can be used to improve the security of embedded systems.

A. Threat Model

This work assumes a DUT that is running a closed-loop code. The DUT may consist of multiple interfaces for network communication as well as for controlling peripherals. The network interfaces may include either or both wired and wireless communication.

We assume that the defender has attached both EM and power probes to the field-deployed DUT, and is able to estimate the signal-to-noise ratio (SNR) of the data from each probe while the system is in operation. Furthermore, the defender of the system has access to an identical copy of DUT, which can be used to profile the DUT and create a baseline model of its behaviour in a safe state. Comparing to this baseline model, the defender can actively monitor the field-deployed DUT in real time using data being acquired through EM and power probes.

We assume that an attacker may have access to the field-deployed DUT through its network and peripheral interfaces. The objective of the attacker is to modify the system's behaviour in a malicious way without being detected by the defender. Consequently, it is not possible for the attacker to physically damage or replace EM and power probes mounted on the field-deployed DUT for monitoring purpose since it would be noticeable to the defender.

II. Related Work

A. Control-Flow Monitoring Using Side Channels

The idea of using side-channel for detecting malicious code execution was first introduced by Aguayo Gonzalez and Reed in 2010 for the power domain [6], and by Stone and Temple for the EM domain [19]. These early works assumed a single valid "golden execution" exists for the program, an assumption that does not hold if the program has multiple code paths. Liu et al. later showed how power signals can be used to monitor this more general class of programs [12]. They defined each basic block of the program CFG as a state, and used a Hidden Markov Model (HMM) to infer the most probable sequence of states, where the power consumption signals served as the observations of the HMM. An improved Viterbi algorithm was used to infer the most probable sequence of states. Using their method, the recognized instruction sequence accuracy was reported to be 97%. Han et al. proposed a control-flow monitor based on the EM side channel [7]. Their passive EM-based monitor was designed to detect the currently-executing control flow of a commercial PLC. They did so by creating a set of test cases for each control flow by using symbolic execution, generating input vectors that will cause the program to follow a desired path, and finally recording the EM radiation of every feasible execution path of the PLC program. To deal with the low signal to noise Ratio (SNR) of their signals, they applied techniques from the speech recognition domain and used the frequency representation of the signals. Finally, they trained an LSTM based network to classify signal to the correct control flow and declared an anomaly in case of a low confidence of the model for all control flows. In this paper we adapt their anomaly detection technique to the multi-modal setting, considering only the time-domain signals, rather than the frequency-domain signals.

Nazari et al. [13] identified spikes in the frequency spectrum of the received EM signal which conforms with periodic events such as loops and functions execution. Then, they used statistical tests to decide whether the code is malicious or benign. The preprocessing stage included instrumentation before and after every loop to help identify tested code regions. Then, their monitor ran the application multiple times with different inputs to achieve better coverage. During the monitoring stage the original program is executed, and the monitor identifies peaks in the EM spectrum and compares them to the peaks learned in the preprocessing stage. Finally, they used a nonparametric test, which does not assume distribution type of the data, to compute the probability the execution belongs to some code region.

B. Multimodal Learning

Multimodal learning methods can be broadly defined as machine learning models that fuse together multiple input sources that measure different aspects of the same event [16].

Research in multimodal fusion has applications in several fields, from natural language processing to image processing, and computer vision. Snoek et al. introduce a taxonomy of fusion schemes, based on the point in the analysis in which the data from multiple sources is combined [16]. *Early integration* is defined by Snoek et al. as a *"Fusion scheme that integrates unimodal features before learning concepts"*. This integration is usually done on raw data, or on features directly derived from the data. *Late integration* is performed by processing each input separately, passing each one through a separate machine learning model, and ultimately aggregating the outputs of the models, or as more formally defined by Snoek et al. [16] a *"Fusion scheme that first reduces unimodal features to separately learned concept scores, then these scores are integrated to learn concepts"*. Between these two extremes are *intermediate integration* schemes, for example the scheme of Ngiam et al. [14], where an intermediate representation of the data is learned for each modality, and then the representations are combined and further processed by a single machine learning model.

Transformer networks are a deep learning architecture introduced by Vaswani et al., which was shown to work well in the multimodal learning setting [20]. This emerging architecture has proved itself in the fields of natural language processing [21] and computer vision [11], and has a major role in the generative AI revolution. Transformer networks achieve their effectiveness by using a self-attention mechanism which enables the deep neural network to weigh the significance of different input elements relative to each other. For example, in the context of NLP, the attention model first weighs the importance of each word compared to other words in the sequence, then the importance of pairs of words compared to pairs of words, then pairs of pairs, and so on [5]. One of the main benefits of self-attention is that it is applied to the entire sequence simultaneously, making it much faster than traditional RNNs. Another factor that assists transformers in the field of NLP is their ability to preserve long-term dependencies. This proves to be valuable also in the side channel use case, given that power consumption (and EM signal as consequence) also depends on the previously-executed instructions, rather than the current executed instruction solely [12].

C. Multimodal Cryptanalytic Side-Channel Attacks

One of the early works that explored the idea of using several channels to improve the effectiveness of a side channel attack is Agrawal et al. [3]. In their work, Agrawal et al. demonstrate how an adversary interested in the internal state of a device, such as the LSB of the data bus during LOAD instruction, can use multiple channels to reduce the error rate of the attack. First, the attacker invokes the target instruction multiple times from any relevant state while monitoring the EM and power consumption of the device. Then, the attacker uses maximum likelihood estimation to find the most probable state of the device. They show that by concatenating the EM and power signals, the error rate of the attack is reduced by up to 13.5% compared to using a single channel. Standaert and Archambeau [18] continue the approach of Agrawal et al. [3] and show how the concatenation of the EM and power signals reduces the entropy when the attacker performs a template attack for key extraction. Concatenating the signals, however, results in longer traces and increases the computational resources needed. Souissi et al. [17] performed a theoretical analysis of the multimodal task, and proposed to combine the Pearson and Spearman correlation coefficients of the measurements. They concatenated two EM traces provided by probes located in different positions on the device, and used CPA to attack the concatenated trace by computing the Pearson correlation coefficient of the key hypothesis for each trace out of the two. Then, the combination was done by summing up the calculated coefficients. This method showed a gain of up to 44.86% in the number of traces needed for the attack. Most recently, Bai et al. [4] showed how an attacker can use both the EM and power side channels to extract AES sub-keys in a real-time setting. They use a linear combination of the coefficients of the features for each trace type to create a combined trace. The combined trace is then fed into an SVM classifier than predicts the target bit. This method requires on average 58% less traces compared to the power only method, and 40% less compared to EM only method in offline settings. Applying the taxonomy of Snoek et al. to the above works, we see that most of them follow an early integration approach, where the signals are combined before the ML model.

III. Methodology

A. Overview

In this work, our general approach is to capture power and EM traces from a real device, use them to train both unimodal (only power, or only EM) and multimodal (both power and EM) machine learning models, and then evaluate the performance of the models on both control-flow classification and anomaly detection tasks. We compare the three approaches for multimodal fusion (early, intermediate, and late), to discover which approach is optimal for our use case. To examine the robustness of our models, we artificially add a controlled amount of noise to the traces, and evaluate the performance of the models on these increasingly-noisy traces.

B. Experimental Setup

Our device under test was the Nordic nRF52833 DK, a microcontroller board equipped with an ARM cortex M4 processor running at 64 MHz. This board comes with built-in current measurement pins, making it possible to profile power consumption with no hardware modifications. The device was powered a by Keysight B2962A low-noise power source. We measured voltage drops across a 10 Ω resistor using a Keysight MSOS604A oscilloscope at a sampling rate of 1 GSa/sec. To measure EM signals, we used a Langer LF-U5 probe connected through a Langer PA 303 amplifier to a Tektronix RSA306 Spectrum Analyzer at a center frequency of 975 KHz and a sampling rate of 56 MSa/sec. The probe's location above the DUT and the spectrum analyzer's frequency parameters were identified through a cartography process which is described in our repository [2]. Figure 1 shows the tested device and the experimental environment.

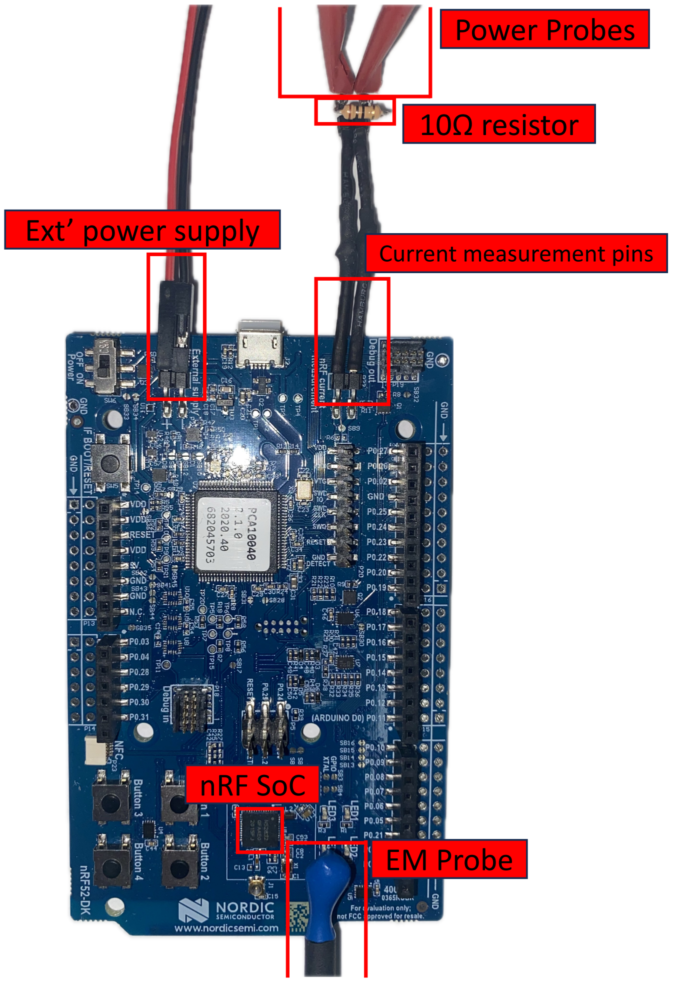


Fig. 1: The experimental environment

The software running on the DUT was the Siemens Traffic Collision Avoidance System (TCAS) [10]. Han et al. recently analyzed TCAS using a symbolic execution framework, producing a set of input vectors covering the entire control-flow graph, which we also use in our work [8]. We modified the program so that it set up a rising edge signal on a GPIO pin at the program's start and a falling edge signal at its end. The GPIO pins were connected to the oscilloscope and the spectrum analyzer, serving as the trigger.

Machine learning training was performed on an NVIDIA GeForce GTX 1080 cluster with 57 GPUs, orchestrated through the Slurm job scheduling system. The nodes were running Linux Centos with Python version 3.8, TensorFlow version 2.6 and CUDA version 11.1.

C. Data Collection and Processing

The TCAS program receives an input vector which determines the path of execution through the control-flow graph. As documented by Han et al. [8], there exists a set of 24 input vectors that covers all possible paths through the program. We ran the program with each one of these input vectors 1000 times, collecting both power and EM traces, resulting in a total of 48,000 benign traces.

We simulated a code injection attack using three different attack scenarios, illustrating different amounts of injected malicious code. In the first attack scenario, we manually inserted 5 NOP instructions into the binary. In the second attack scenario, we inserted 10 NOP instructions, and in the third attack scenario, we inserted code that transmits a single byte through the UART pins of our test device. Each one of these three variants was applied to the 24 input vectors.

Since each flow of the program executes a different set of instructions, there might be differences among the execution time of the different flows. Those differences will be expressed in the length of the captured signals, and might cause the model to learn the length of the captured signal rather than the signal itself. To overcome this issue, and other shape mismatch issues during the training process, we resized each signal to match the size of the longest signal in the training set using linear interpolation and decimation.

We post-processed the signals and normalized them to a mean of 0 and a standard deviation of 1, and split the benign traces into training and testing groups using a 80/20 split. To evaluate the performance of our classifier under noisy conditions, we artificially generated 20 additional datasets by adding a varying amount of noise to the normalized signals, with noise level (i.e. standard deviation) σ varying from 1 to 20. Note that the standard deviation of the signals is 1 since we applied normalization.

D. Machine Learning

Our machine learning classifier receives as input either a power trace, an EM trace, or both power and EM traces simultaneously, and outputs a confidence vector matching the trace to each of the 24 possible control flows of the program. Anomaly is declared in case the classifiers' confidence in all classes (control flows) is lower than a predefined threshold. This method which was also presented by Han et al. [7] gives a single model the ability to classify signals to the correct flow and monitor code execution, while also serving as an anomaly detector. It also does not require any anomalous samples for the training process.

To evaluate the model's ability to classify a given trace to the correct control flow, we feed the model with the benign samples, and measure its accuracy. We tested this anomaly detector against the malicious trace dataset, and measured the area under the ROC curve (AUC) of the classifier as we varied the threshold value.

1) Unimodal Classifiers: After collecting the training samples, we trained two independent transformer-based classifiers. Each classifier was trained on a single modality, either EM or power signals. The architecture of our models is based the Keras transformer model [1]. It consists of one encoder block, composed of a multi-head attention layer followed by dropout and normalization layers, and a feed forward part that is composed of 1-dimensional convolutional layer followed by dropout and normalization layers. Finally, the output of the feed-forward part is residually connected to its input. Next, the output of the block is fed into a global average pooling layer, followed by a dense layer with sigmoid activation for classification. We also used an L2 regularizer on the output layer with a regularization factor of 0.0005. The architecture of the models is identical between the modalities, and it is trained with cross-entropy loss and Adam optimizer for 10 epochs and a batch size of 8. We explore our machine learning parameter choices further in Section IV-C. We repeated the training process 21 times, accounting for the different noise levels we used in our experiments. We denote the EM based model that was trained with a noise level of e as EM_e, and the power based model that was trained with a noise level of p as $Power_p$.

2) Multimodal Classifiers: The multimodal classifier accepts both EM and power signals as input. We evaluated three possible multimodal fusion approaches: early integration, intermediate integration, and late integration.

To implement the *early integration* approach, we concatenated the signals before feeding them to a single model, as in Standaert and Archambeau [18], Agrawal et al. [3]. To implement the *late integration* approach, we applied the signal-specific models as trained previously, and then used the mean of their outputs as the output of the integrated model, noting that since we used the sigmoid layer as the output layer, the mean value of the outputs is still a probability for each class. We now describe the *intermediate integration* approach, which is the main focus of this work.

Our model follows the general structure described in Fig. 2. The architecture of each stream is identical to the architecture of the signal-specific models until it reaches the output layer. At this point, instead of feeding the embeddings of each modality to the output layer, we fuse them using a cross attention layer, then feed the fused embeddings to the output layer. We note that since the noise levels of the power and EM traces are not necessarily identical, there is a quadratic number of possible noise level combinations for the multimodal model, resulting in 441 models trained in total. We denote the multimodal model that was trained with EM noise level of e and power noise level of p as $MM_{e,p}$.

IV. Results

A. Proposed Multimodal Classifier

As described in Section IV-B, the intermediate integration approach is the main focus of this work. In Fig. 3, we compare the performance of this type of multimodal model to the performance of the unimodal EM and power based models for the classification task at different noise levels. As mentioned in Section III-C, there is a quadratic number of possible noise level combinations for the multimodal model. For efficiency of presentation, we plot cases where the noise level of both EM and power samples is identical, i.e., $MM_{i,i}$ for $i \in 0 \cdots 20$. Additional results for all noise combinations are available in the artifact repository [2].

As the figure indicates, the multimodal model outperforms the single-channel models at all noise levels. It is also worth noting that the EM-only model outperforms the power-only model when noise levels are low, while as the noise level increases the power model becomes superior over the EM model. Additionally, the power model's accuracy drops at a slower rate compared to the EM model as the noise level increases.

To better understand the benefit of the multimodal model across all possible noise level combinations, we define the *Accuracy Gain* and *AUC Gain* as the difference between the performance of the multimodal model to that of the best of the two single-modality models at the same noise levels in the classification and anomaly detection tasks, respectively. For example, the accuracy gain at an EM noise level of 20 and a power noise level of 10 is defined as $Gain_{20,10} = MM_{20,10} - max(EM_{20}, Power_{10})$

Fig. 4a shows the accuracy gain, and Fig. 4b shows the AUC gain, for our proposed multimodal classifier. As the figure shows, at low noise levels the multimodal model has no significant advantage over the unimodal models, possibly due to the high accuracy of the unimodal classifier at these noise levels. As the noise level increases, however, the multimodal model shows a significant improvement in both accuracy and AUC of up to 18% and 11%, respectively, over the corresponding unimodal models. We also note that when the noise values are low there is a possible minor drop in accuracy and AUC of up to 2% and 4%, respectively. Overall, Fig. 4 shows that our proposed approach improves both the accuracy and the AUC of the models in most realistic settings.

B. Comparison With Early- and Late-Integration Multimodal Classifier

While we proposed an intermediate integration approach in this paper, other approaches exist as well. For example, both Standaert and Archambeau [18] and Agrawal et al. [3] used the early integration approach and concatenated the signals before using them for classification. We compare our proposed approach to both early and late integration approaches at different noise levels in Fig. 5.

As the figure shows, in the absence of noise both the late and intermediate integration models perform similarly, and better than the early integration model. As the noise level increases, however, the intermediate integration model significantly outperforms both early and late integration models. Comparing these results with the results for unimodal models shown in Fig. 3, we observe that the performance of the intermediate integration is strictly superior to the performance of the unimodal models, the performance of the early integration model is similar to that of the power-only model. This emphasizes the advantage of our proposed integration technique over the other approaches.

C. Choosing the Output Layer

As previously explained, our model serves two purposes, a classifier and an anomaly detector. When creating a multiclass deep-learning classifier, a typical approach is to use a softmax layer as the final activation function. Since our model serves

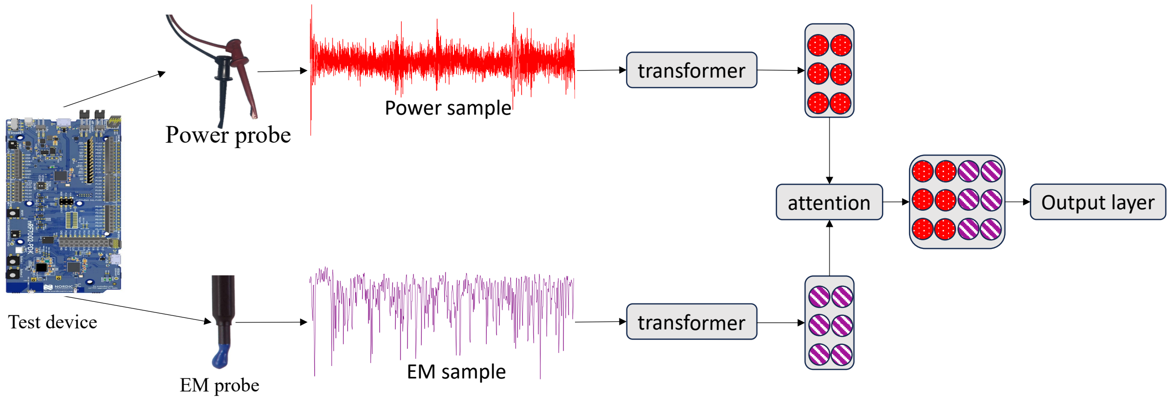


Fig. 2: The architecture of the suggested bimodal fusion transformer model

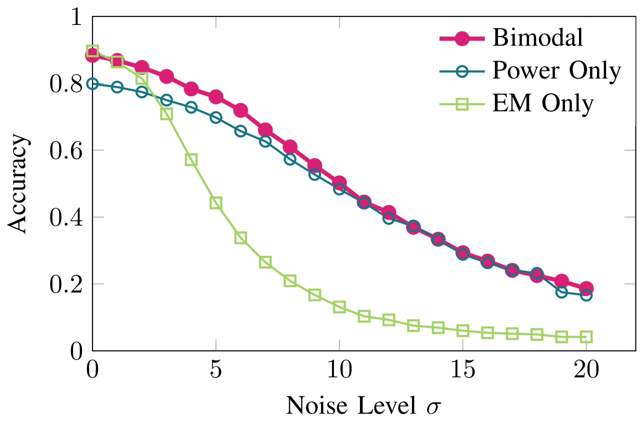


Fig. 3: Accuracy vs. noise of multimodal and single-modality models

an additional purpose, we need to consider the effect of the output layer on anomaly detection.

The softmax layer which is used to predict probabilities with categorical distributions is defined as:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \tag{1}$$

One attribute of this function is that it outputs a distribution function, meaning that the sum of the output vector is always 1. As a consequence, an increase in the confidence of one class will necessarily result a decrease in another. This attribute is not wanted in our use case for detecting anomalies, since a low confidence of the model in all classes is actually used to detect anomalies, and is thus a desirable outcome. Instead of using the classic softmax activation, we used the sigmoid function which is more typically used for binary classification problems. The sigmoid function is defined as:

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Each value in the output vector is independent of the other values, and can be interpreted as the probability of the sample to belong to the corresponding class. Yet, the sum of the output vector is not necessarily 1, and the output vector is not necessarily a distribution function. Table I shows the importance of choosing the right output layer for our task. We compared the accuracy and the AUC of the models when using several output functions. Namely, softmax, sigmoid, and softplus activations. The model was trained for the noiseless scenario (i.e. $MM_{0,0}$). The table shows that, while the accuracy of the model is not greatly affected by the choice of the output layer, the AUC is significantly affected by this choice. The AUC of the model using the sigmoid activation reaches 99%, which is 6.1% higher than when using the softmax activation. Surprisingly, the softplus activation, which is a smooth version of relu and more typically used in hidden layers, outperforms the softmax activation in AUC as well. However, since the output values of the softplus function do not represent a probability for the class we used the sigmoid activation for more interpretable results.

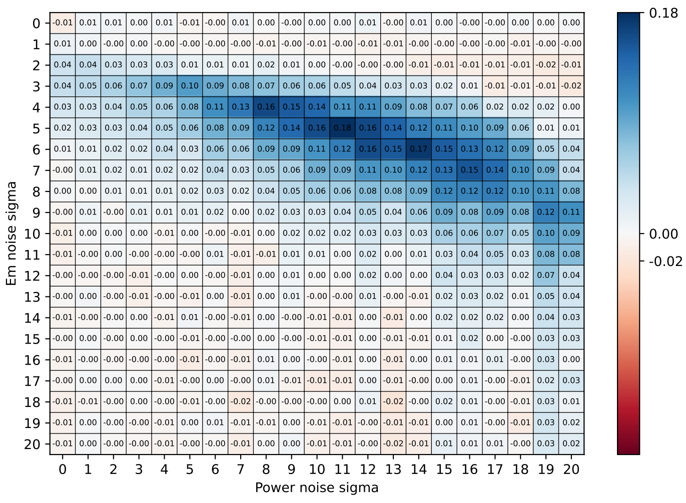
TABLE I: Comparing the output layer activation function

	sigmoid	softmax	softplus
Accuracy	0.883	0.903	0.897
AUC	0.990	0.929	0.969

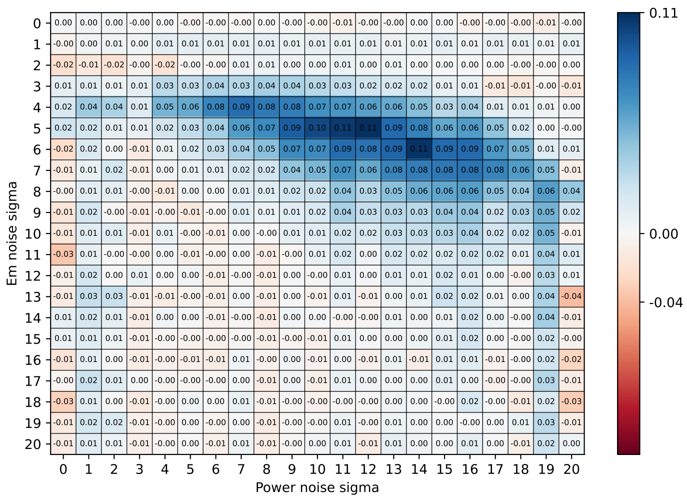
V. Discussion

A. Overhead analysis

Using a ML pipeline with two modalities may increase the inference time of the model, leading to anomaly detection solution that is not acceptable in time-critical environments. The transformer network proposed in this paper better fits the requirements of a real-time environment than previously proposed RNN models [7] as there is no recurrent leading to faster inference. As deep learning models are a natural choice for signal classification, classic ML models can also be considered as they tend to be faster, however, they require a feature engineering step. To estimate the overhead of our multimodal approach, we measured the inference time of the multimodal model and compared it with the single modality models. We also implemented a RandomForest (RF) model,



(a) Accuracy Gain



(b) AUC Gain

Fig. 4: Performance gains of the multimodal model compared to the single-modality models

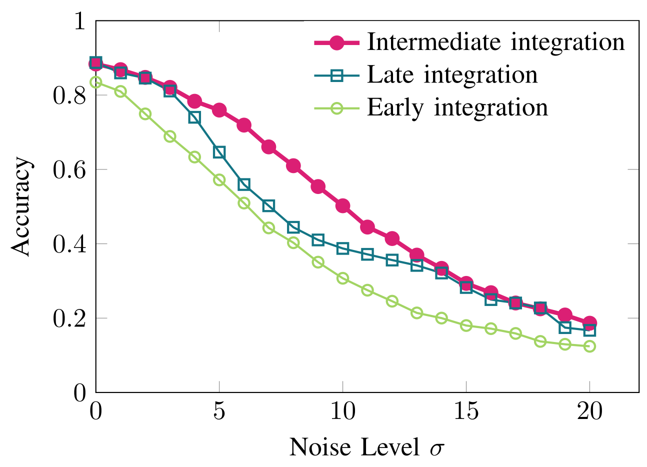


Fig. 5: Accuracy vs. noise of different integration approaches

TABLE II: Comparing the inference time and accuracy of the various models

	Inference(ms)	Accuracy $@\sigma = 0$	Accuracy $@\sigma = 5$
$Transformer(EM)$	4.04±0.60	0.896	0.442
$Transformer(Power)$	4.58±0.55	0.798	0.697
$Transformer(Multimodal)$	5.15±0.77	0.883	0.759
$LSTM(EM)$	4.89±0.65	0.846	0.202
$LSTM(Power)$	7.41±0.57	0.678	0.380
$LSTM(Multimodal)$	8.09±0.65	0.875	0.610
$RF(EM)$	0.8±0.008	0.423	0.041
$RF(Power)$	2.4±0.006	0.211	0.20
$RF(Multimodal)$	3.23±0.006	0.387	0.22

and three LSTM networks - one for each modality and a multimodal model with an LSTM backbone, We followed [7] to implement the LSTM architecture. For the RandomForest classifier we extracted classic features from the signals like the average power, maximum power, number of peaks, standard deviation, skewness, kurthosis, and percentiles. Table II shows the results of this experiment, it shows the mean inference time of a single sample for each one of the models, and the accuracy the models reached. Note that models of the same type have different inference time based on the modality the model was trained on. This is due to the significantly higher sampling rate that the power samples were measured, leading to longer samples than the EM samples. By comparing the transformer and LSTM models that were trained on the same modality (e.g. Transofmer(EM) vs LSTM(EM)), we can learn that the transformer based models are faster, more accurate and their accuracy rate drops slower as the noise level increase. Furthermore, on average the slowest transformer model - the multimodal model, is only 5% slower than the fastest LSTM based model. Alternatively, the RF models suggest a significantly faster models, however they are much less accurate than the deep learning models, probably due to the limited amount of knowledge that can be extracted from the suggested features.

B. Limitations

The primary limitation of our work is the fundamental question of whether side channels are the right tool to be used for monitoring the control flow of a device. As recently shown by Han et al., an adversary capable of accessing the classifier's confidence score can use an iterative search process to craft malware which is both semantically correct and difficult to detect by a power-based side channel monitor[8]. We hypothesize that the higher dimensionality of the multimodal detector would make it more difficult to evade than single-modality detectors, but this remains to be tested. Another vulnerability in the defense is its reliance on external power and EM sensors. An attacker with physical access to the devices can potentially tamper with the sensors, an act which might be simpler than tampering with the device itself, and thus either disable protection (by replaying traces of valid executions, irrespective of what the device is actually doing), or otherwise expose the device to potentially harmful false positives.

Additionally, in this paper we used random Gaussian noise as our noise source for both EM and power traces, we note that in reality the source of the noise may be different for each channel. Moreover, in real-life scenarios the noise levels may be higher than presented in this paper, this remains to be explored.

C. Future Work

The most obvious direction for future work is to test the proposed approach on additional real devices, and in particular to refine the noise profile of badly-positioned sensors beyond the white Gaussian noise we used in this work. The multimodal approach we proposed is not limited to a pair of EM and power channels. It would be interesting to evaluate it with arbitrary combinations of power channels, EM channels, or even more exotic side channels such as acoustic or thermal. In addition, since the traces are only integrated after feature extraction, our approach may be more resilient to misalignment between traces, both within a single modality and both between modalities. It would be interesting to test this robustness in future work. It would also be interesting to consider how multimodal cryptanalytic side-channel attacks would benefit from the intermediate integration approach.

Finally, while self-attention has been shown to be effective in this problem domain, it is not the only possible approach to multimodal integration. It would be interesting to find other hyperparameter combinations, or other architectures entirely, which are more effective than the one we proposed.

D. Conclusion

In this paper we presented a passive approach to detect control-flow deviations during an attack. We leveraged a multimodal approach that used both power consumption and EM side channels to monitor the control flow of a device. Our proposed approach is superior to approaches that only use a single type of signal, improving the accuracy by up to 18% and the AUC by up to 11%. We also compare our models' performance to traditional RNN, and show how it is both faster and more accurate. Finally, Our results show the benefit of using concepts from the field of multimodal learning to improve side-channel defenses.

References

[1] Timeseries Classification with a Transformer Model. URL https://keras.io/examples/timeseries/timeseries_classification_transformer.

[2] Two's Complement Artifact Repository. URL https://github.com/Michael-amar/Twos_complement.git.

[3] Dakshi Agrawal, Josyula R. Rao, and Pankaj Rohatgi. Multi-channel Attacks. In *CHES*, volume 2779 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2003.

[4] Yunkai Bai, Jungmin Park, Mark M. Tehranipoor, and Domenic Forte. Dual Channel EM/Power Attack Using Mutual Information and its Real-time Implementation. In *HOST*, pages 133–143. IEEE, 2023.

[5] Benyamin Ghojogh and Ali Ghodsi. Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey. 2020.

[6] Carlos R Aguayo Gonzalez and Jeffrey H Reed. Detecting Unauthorized Software Execution In SDR Using Power Fingerprinting. In *2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*, pages 2211–2216. IEEE, 2010.

[7] Yi Han, Sriharsha Etigowni, Hua Liu, Saman A. Zonouz, and Athina P. Petropulu. Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. In *CCS*, pages 1095–1108. ACM, 2017.

[8] Yi Han, Matthew Chan, Zahra Aref, Nils Ole Tippenhauer, and Saman A. Zonouz. Hiding in Plain Sight? On the Efficacy of Power Side Channel-Based Control Flow Monitoring. In *USENIX Security Symposium*, pages 661–678. USENIX Association, 2022.

[9] Marius Herget, Faezeh Sadat Saadatmand, Martin Bor, Ignacio González Alonso, Todor Stefanov, Benny Akesson, and Andy D Pimentel. Design Space Exploration For Distributed Cyber-Physical Systems: State-of-the-art, Challenges, and Directions. In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pages 632–640. IEEE, 2022.

[10] Monica Hutchins, Herbert Foster, Tarak Goradia, and Thomas J. Ostrand. Experiments of the Effectiveness of Dataflow- and Controlflow-Based Test Adequacy Criteria. In *ICSE*, pages 191–200. IEEE Computer Society / ACM Press, 1994.

[11] Salman H. Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in Vision: A Survey. *ACM Comput. Surv.*, 54(10s):200:1–200:41, 2022.

[12] Yannan Liu, Lingxiao Wei, Zhe Zhou, Kehuan Zhang, Wenyuan Xu, and Qiang Xu. On Code Execution Tracking via Power Side-Channel. In *CCS*, pages 1019–1031. ACM, 2016.

[13] Alireza Nazari, Nader Sehatbakhsh, Monjur Alam, Alenka G. Zajic, and Milos Prvulovic. EDDIE: EM-Based Detection of Deviations in Program Execution. In *ISCA*, pages 333–346. ACM, 2017.

[14] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal Deep Learning. In *ICML*, pages 689–696. Omnipress, 2011.

[15] Michael A. Schuette and John Paul Shen. Processor Control Flow Monitoring Using Signatured Instruction Streams. *IEEE Trans. Computers*, 36(3):264–276, 1987.

[16] Cees Snoek, Marcel Worring, and Arnold W. M. Smeulders. Early Versus Late Fusion In Semantic Video Analysis. In *ACM Multimedia*, pages 399–402. ACM, 2005.

[17] Youssef Souissi, Shivam Bhasin, Sylvain Guilley, Maxime Nassar, and Jean-Luc Danger. Towards Different Flavors of Combined Side Channel Attacks. In *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*,

pages 245–259. Springer, 2012.

[18] François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[19] Samuel J. Stone and Michael A. Temple. Radio-Frequency-Based Anomaly Detection for Programmable Logic Controllers in the Critical Infrastructure. *Int. J. Crit. Infrastructure Prot.*, 5(2):66–73, 2012.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NIPS*, pages 5998–6008, 2017.

[21] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP (Demos)*, pages 38–45. Association for Computational Linguistics, 2020.