Dual Channel EM/Power Attack Using Mutual Information and its Real-time Implementation

Yunkai Bai, Jungmin Park, Mark Tehranipoor and Domenic Forte
Department of ECE, University of Florida, Gainesville, Florida, USA
Email: baiyunkai@ufl.edu, jungminpark@ufl.edu, tehranipoor@ece.ufl.edu, dforte@ece.ufl.edu

***Abstract*—Cryptosystem implementations often leak information about a secret key due to correlation with side channels such as power, timing, EM, etc. Based on this principle, statistical and machine-learning-based side-channel attacks have been investigated, most often using a single channel or modality such as power; however, EM is growing in popularity. Since power and EM channels can leak distinct information, the combination of EM and power channels could increase side-channel attack efficiency. In this paper, we combine EM and power channels in a linear fashion by using mutual information to determine the optimal coefficients for each feature. Mutual information is also systematically applied for lightweight dimensionality reduction. Further, the proposed methodology is implemented onto a platform to simultaneously measure power and EM traces and process them *in real time* to extract AES subkeys. With the proposed dual channel approach, the success rate increases by at least 30% compared to single power/EM channels in the offline mode and over 50% in the real-time mode.**

***Index Terms*—Side-channel attack, linear combination, RDCP, real-time**

I. Introduction

Modern cryptosystems are implemented using semiconductor logic gates, which are constructed out of transistors. These silicon-based transistors generate electromagnetic (EM) radiation and consume power when a voltage is applied between their gate and substrate. Thus, EM and power vary based on gate inputs or, in other words, the data being processed by the silicon chip. Over the past 20 years, this phenomenon has been exploited to non-invasively steal the secret keys of cryptosystems in so-called side-channel attacks (SCAs) using power [9]–[11] and EM [15], [16].

The power SCA was introduced in Paul Kocher's seminal paper [9]. Kocher et al. presented the differential power attack (DPA) against the DES encryption module by utilizing the power leakage collected during cryptographic operations. In 2004, Brier et al [12] proposed the correlation power attack (CPA) approach which is based on the power Hamming distance (HD) model. The CPA attack utilizes the positive linear relationship between power consumption and HD to find the correct secret key from all possible hypotheses. The EM SCA originated in 2001 [15] where Gandolfi et al. [15] successfully retrieved complete key materials from EM leakage in a series of experiments on three different CMOS chips. In [16], Agrawal et al. launched attacks such as simple and differential electromagnetic attacks on straightforward (unprotected) implementations of DES [40], RSA [34], and COMP128 [35] on smart cards, cryptographic tokens, and SSL accelerators. In [20], Hutter et al. reported the first EM attacks on hardware as well as software implementations of AES on RFID tag prototypes. In [21], DING et al. presented the correlation electromagnetic attack (CEMA) on the P89C668 microcomputer.

Although the main principles and algorithms used for power and EM-based side-channel attacks are the same, their performance differs based on the cryptosystem's implementation. For example, some time samples (features) in the EM traces could be used to extract sensitive information but those with the same index from the power channel are useless in the power side-channel attack, and vice versa. Besides fine-tuning the hardware setup to improve signal-to-noise ratio (SNR), another possible way to increase side-channel attack efficiency is to take subkey-correlated features from both EM and power channels [17], [19]. In such dual channel attacks, the combined features contain more information than a single channel and, therefore, increase the SCA success rate with fewer traces. In [19], Standaert et al. concatenated EM and power traces and developed the entropy method to quantify the SCA efficiency. Their experiment shows less entropy is achieved after concatenation to achieve a higher success rate. However, the concatenation method expands the length of the traces, requiring more processing resources. In [17], Souissi et.al try to combine multiple channels by summing up the square of each channel. A mathematical proof is provided to show the improvement in SNR after combination. Compared with the single power channel, this sum of squares method has larger SNR, and the experiments demonstrate that fewer traces are needed to attack the same SBox. However, the sum of the square method only works for correlation-based SCAs, and is not suitable for other algorithms, such as DPA or machine learning. Further, it also takes more time to process traces from two channels and lacks granularity in combination. That is, as features at different time indexes share different SNRs, their combination should be different. Besides the limitations of inapplicability, resource efficiency, and granularity, the methods in [17] and [19] need to be implemented in a lab. The lack of real-time implementation limits their use for intelligence gathering in the field.

In this paper, we present a mutual-information-based approach to create features for dual channel SCA. The high-level diagram is presented in Figure 1. Just like previous works [17] [19], we take both the power and EM features into consideration. Compared with [17] and [19], our proposed methodology focuses on both selection and the combination

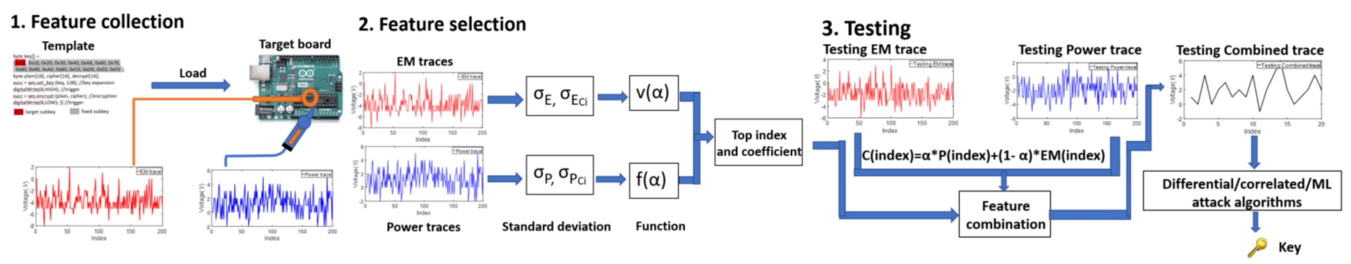


Fig. 1: High-level diagram of proposed methodology. In the feature selection phase, coefficients for combining EM and power traces (red and blue, respectively) into a combined channel (black) during attacks are calculated using mutual information. Further, mutual information also guides dimensionality reduction, making the proposed attacks realizable in real-time.

of features from both channels before implementing a SCA algorithms and then adopts mutual information to quantify the improvement after the combination. Since the combination of power/EM features at different time indices utilizes different coefficients, the proposed methodology offers higher granularity than the others. More importantly, the feature selection could save time and memory resources, allowing for real-time implementation on a resource-limited device. Also, since the proposed approach generates features before implementing the SCA algorithm, the proposed methodology could be integrated with any attack method, e.g., DPA [9] [10] [11], CPA [12] [13], machine learning [38] [39], deep learning [36] [37], etc. Besides, we present the attack capability by implementing the proposed methodology in both offline and real-time modes. In the offline mode, we adopt a traditional side-channel system (oscilloscope and commercial EM probe) to collect traces and attack the target Arduino UNO [32] board. In the real-time mode, we implement the proposed methodology into a custom-designed "real-time dual channel platform" (RDCP for short) to extract the secret key in real-time. Our main contributions are summarized as follows:

- We present a mathematical analysis to quantify the gains made by combining EM and power channels and to identify the optimal combination. The efficiency of the methodology is also improved through lightweight dimensional reduction where mutual information is utilized to select the best time indices of the dual channel traces.
- The applicability of the proposed methodology is presented by comparing the efficiency of multiple attacks. We utilize our selected and combined features in differential attacks, correlation attacks, and machine-learning attacks. Experiments are performed to compare the success rates vs. the number of traces for all 16 subkeys of an AES-128 module. We also implement the dual channel approaches from [17], [19] and show that the proposed approach significantly outperforms them.
- The proposed methodology is successfully implemented into RDCP for real-time processing. The approach scales with number of traces by reusing memory for each trace. Even though the measurements to disclosure (MTD) are worse than the offline version, the results are promising and show improvements over single channel scenarios.

The remainder of this paper is organized as follows. Section II describes the related work and associated results. Section III introduces the background of mutual information. In Section IV, we present the proposed methodology, including the dual channel feature selection and combination, training template, and algorithms. The real-time implementations are described in Section V. Section VI describes the experimental setup, results, and discussion. We conclude and offer directions for future work in the last section.

II. Related work

Mutual Information Analysis. In [14], Gierlichs et al. present a generic differential SCA that is based on an information-theoretic distinguisher. The distinguisher adopts the mutual information between the observed measurements and the values of a hypothetical leakage function. Gierlichs calculates the mutual information for different time indexes and finds the obvious peaks in the plot during the jointly implemented SubBytes and ShiftRows operations as well as during the MixColumn operation of AES. In the experiments, SCA efficiency was improved by selecting features that possess the highest mutual information values.

EM-based Attacks. In [15], Gandolf et al. describe differential EM attacks (DEMA) conducted on three different CMOS chips. Similar to DPA, the result for the DEMA also shows a strong peak, which proves the efficiency of the differential attack in the EM channel. In [16], Agrawal et al. provide a systematic investigation of information leakage via EM emanations from CMOS devices. It is shown that EM signals capture leakage based on not only physical and electrical characteristics but also emanations from other components due to coupling and circuit geometry. The experiments show that DEMA could focus on a small subsection of the EM trace to extract subkeys while DPA could not.

Template Attacks. In [25], Archambeau et al. present the template-based attack against an FPGA implementation of AES Rijndael. The template attacks use the maximum likelihood principle to reveal the secret for a set of traces. Their principal subspace based templates achieve average success rates of 93.3% and 86.7% against RC4 and AES Rijndael, respectively, In [26], Mohamed et al. propose a template-based algebraic SCA. They reduce the information required

TABLE I: Comparison of related work and proposed method. **Legend:** Power = P; MI = Mutual Information; DA = Differential Attack; CA = Correlation Attack; TA = Template Attack; MLA = Machine Learning Attack; SR = Success Rate.

	Non-profiling Attacks				Profiling Attacks					
	[14]	[15]	[16]	[17]	[19]	[22]	[23]	[25]	[26]	Proposed
Modality	P	EM	EM	EM & P	EM & P	P	P	P	P	EM & P
Method	MI	DA	DA	CA	TA	MLA	MLA	TA	TA	MI
Target Algorithm	AES-128	COMP128 DES	COMP128 DES	DES	N/A (Bit transition)	DES	AES	AES	AES	AES-128
Platform	AT90S851	Smartcard	Smartcard	SASEBO	PIC16F877	XCS3E500	ATmega163	PIC16F877	AMD8536	ArduinoUNO
Mode	Offline	Offline	Offline	Offline	Offline	Offline	Offline	Offline	Offline	Real-time

for solving the algebraic system by improving the algebraic representation of AES as well as Hamming weight and conduct a single-trace template attack against AES-128 encryption. They were able to solve the algebraic system in a few seconds and extract all subkeys using a template base of 5,000 measurements and 2,000 testing plaintext/key pairs.

Machine Learning (ML) Attacks. In [22], Lerman et al. present a machine-learning procedure based on dimensionality reduction and model selection to attack the byte of a secret key. They first build classifiers using random forest (RF) and support vector machines (SVMs) using secret bytes and power traces from a DES cryptographic device. Then, they try to predict one target bit of an RSA-512 private key. The experiment result shows that the trained model could achieve high accuracy towards the target bit but low accuracy ($< 20\%$) towards the entire byte for DES encryption. Nevertheless, this method [22] still proved to be faster than template attacks when only a few traces were available.

In [23], Lerman et al. build a ML classifier with power traces and key-related information to attack masked AES encryption modules. The proposed machine-learning attack adopts SVM and RF algorithms, and the success rate could reach 90% success rate with 1,500 traces in the learning set (48 features per trace). However, the normal template and RF attacks only achieved 80% and 70% success rate under the same conditions.

Dual Channel Attacks. In [19], the authors present fair information and security metrics to evaluate the EM and power channels. Their analysis shows that the EM channel has significantly higher information leakage. By concatenating power and EM channels, the experimental results show the conditional entropy of the concatenated trace could be diminished, which could be helpful to an adversary in recovering cryptographic keys. In [17], the author first combines Pearson and Spearman correlation coefficient distinguishers for SCA. Then, EM and power traces were combined using the sum over standard deviation method, which could increase the SNR of the traces. With these combined traces, CPA efficiency increased by 45%.

The related work is summarized and compared to the proposed approach in Table I.

III. Background and Preliminaries

A. Mutual Information

Mutual information is a quantity that measures the mutual dependence between two random variables [5]. In the area of SCAs, mutual information can be adopted to describe the correlation between the features in side-channel traces and secret information, such as a subkey in AES encryption. In this paper, mutual information will be used to quantify the relationship between features combined from power and EM with a target subkey.

Assuming the feature is a discrete random variable X, and the subkey value is a discrete random variable denoted as C, the mutual information between X and C is given by

$$I(X,C) = \sum_{x\in X}\sum_{c\in C} P_{X,C}(x,c)\log\left(\frac{P_{X,C}(x,c)}{P_X(x)P_C(c)}\right), \quad (1)$$

where $P_X(x)$ and $P_C(c)$ is the the marginal probability mass functions of random variable X and C. $P_{X,C}(x,c)$ is the joint distribution and the marginal distributions are P_X and P_C.

B. Mutual Information and Entropy

Based on information theory, the mutual information in Equation (1) could also be expressed in terms of entropy [5]:

$$I(X,C) = h(X) - h(X|C). \quad (2)$$

Here, $h(X)$ stands for the entropy of the target feature X, and $h(X|C)$ means the entropy of the target feature X when given class label C. Considering the combination of EM and power channels, one is likely to normalize all features from EM and power channels into the range $[0,1]$. Thus, the range of X is limited to $[0,1]$ and its probability density function p obeys the Gaussian distribution with the mean μ and the standard deviation σ. The entropy of X could therefore be written as

$$\begin{aligned} h(X) &= -\int_0^1 p(x)\log(p(x))dx \\ &= -\int_0^1 p(x)\left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{x^2}{2\sigma^2}\log(e)\right]dx \\ &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{\sigma^2}{2\sigma^2}\log(e) \\ &= \frac{1}{2}\log(2\pi e\sigma^2). \end{aligned} \quad (3)$$

Notice that Equation (2) can be rewritten using Equation (3) to express mutual information with the standard deviation of the whole dataset and the standard deviation of each class:

$$I = \frac{1}{2}(\log(2\pi e\sigma^2) - \sum_{i=1}^{n} p_{C_i}\log(2\pi e\sigma_{c_i}^2), \quad (4)$$

where σ is the standard deviation of whole dataset, σ_{c_i} denotes the standard deviation of the data from class i, and p_{c_i} represents the probability of class i in whole dataset. In this paper, we refer to mutual information expressed using Equation (4) as the Gaussian approximation format.

IV. Proposed Dual Channel Methodology

This section discusses the methodology and implementation of our dual-channel SCAs. We begin by describing the Gaussian approximation of mutual information (Section IV-A) and its assumptions. Then, we discuss how to determine the optimal coefficients for combining EM and power channels using mutual information (Section IV-B). In Section IV-C, we describe the overall methodology for three algorithms used in this paper: correlation, differential, and ML attacks.

A. Preconditions

Before deriving the condition for a better combination than the single channel, we first emphasize three preconditions for the SCA and explain them in sequence. When target feature X is selected:

1) The probability of the selected feature of the EM/power trace ($p(x)$) obeys a Gaussian distribution.
2) Given class c_i of n classes, the probability of the selected feature of the EM/power trace ($p(x|c_i)$) obeys the Gaussian distribution.
3) The probability of all classes are equal to each other, i.e., $p(c_i = 1) = p(c_i = 2) = \ldots p(c_i = n)$).

For the first precondition, the operations should be the same in every iteration of the AES encryption, and the only variation for the encryption is the value processed in on-chip registers. As well known, the power/EM features are mainly decided by operating instructions inside the cryptosystem. Thus, the power/EM features should obey the Gaussian distribution for a specified time index. In this case, for calculating mutual information between power/EM features and the target subkey, we can use the Gaussian approximation (Equation (4)) for mutual information.

For the second precondition, class c_i means the target subkey. Since the subkey is one byte long, there are $n = 256$ classes. The given class c_i means the target subkey is fixed. The operating instructions at the specified time index also remain the same. Thus, it is plausible that we still take the Gaussian approximation for the second precondition.

The third precondition assumes the chance of each value of the target subkey is the same, which is valid. In the training, we send the same plaintexts (same amount and information) to the cryptosystem to keep the presence of each value of the target subkey equal to each other.

Under these three preconditions and Equation (4), the mutual information between target feature X and the class label C could be transformed into

$$I = \frac{1}{2}\log\left(\frac{\sigma^2}{\prod_{\forall i}\sigma_{c_i}^{\frac{2}{n}}}\right) \tag{5}$$

where n stands for the total number of classes (256 for attacking a subkey), σ is the standard deviation of the target feature, and σ_{c_i} is the standard deviation of the target feature in class i. Figure 2 shows the mutual information of EM and power traces with class labels using two different approaches: "Gauss approximation" uses Equation (5) and the "General formula" uses Equation (1). The calculated mutual information value under the two calculation methods is almost identical. In other words, the three preconditions about Gaussian distribution hold in practice.

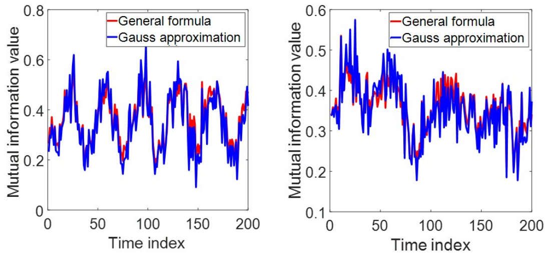


Fig. 2: Mutual information between (a) EM and class label; (b) power and class label.

B. Optimal Combination of Features

Based on information theory [5], mutual information (I) is larger or equal to 0. Thus, we can get the following inequality from Equation (5):

$$\frac{\sigma^2}{\prod_{\forall i}\sigma_{c_i}^{\frac{2}{n}}} \geq 1. \tag{6}$$

Further, since $\log(x)$ is a positive correlation function, the mutual information will increase if Equation (6) becomes larger after linear combination. We use the symbol $\dot{o}$ to describe the nth power of Equation (6):

$$\dot{o} = \frac{\sigma^n}{\prod_{\forall i}\sigma_{c_i}}. \tag{7}$$

If $\dot{o}$ becomes larger, the mutual information should also be larger.

After this simplification, we finally move on to derive the conditions for achieving higher mutual information after the combination of traces from two channels. The features from the first and second channels at the same time index are denoted by random variables H_1 and H_2, and both are in the range $[0, 1]$. We adopt α to describe the linear combination coefficient and Z to describe the combined feature:

$$Z = \alpha H_1 + (1 - \alpha)H_2. \tag{8}$$

Note that $Z \in [0, 1]$ as long as $\alpha \in [0, 1]$.

Without loss of generality, assume the mutual information of channel 1 is equal or larger to that of channel 2, i.e.,

$$\dot{o}_{H_1} \geq \dot{o}_{H_2}. \tag{9}$$

After the linear combination, the value of mutual information of the combined feature Z could be described with $\dot{o}_Z$ if the distribution of combined features still obeys a Gaussian distribution:

$$\dot{o}_Z = \frac{\sigma_Z^n}{\prod_{\forall i}\sigma_{Z_{C_i}}}. \tag{10}$$

This is possible since the mean value of the sum of the two Gaussian distributions equals the sum of their mean values, and the standard deviation could also be calculated from the standard deviation of two Gauss distributions. Their relationship is listed in Equation (11):

$$\begin{aligned}&H_1 \sim \mathcal{N}(\mu_{H_1}, \sigma_{H_1}^2), H_2 \sim \mathcal{N}(\mu_{H_2}, \sigma_{H_2}^2),\\ &Z \sim \mathcal{N}\left(\alpha\mu_{H_1} + (1-\alpha)\mu_{H_2}, \alpha^2\sigma_{H_1}^2 + (1-\alpha)^2\sigma_{H_2}^2\right),\end{aligned} \tag{11}$$

where $\mathcal{N}(\mu, \sigma)$ represents the normal distribution function with mean μ and standard deviation σ. Equation (10) could be rewritten in terms of $\sigma_{H_1}^n$ as

$$\dot{o}_Z = \frac{\sigma_{H_1}^n}{\left(\frac{\sigma_{H_1}}{\sigma_Z}\right)^n \prod_{\forall i} \sigma_{Z_{C_i}}}. \tag{12}$$

Finally, the condition that supports the improvements gained from a combined feature is

$$\left(\frac{\sigma_{H_1}}{\sigma_Z}\right)^n \prod_{\forall i} \sigma_{Z_{C_i}} \geq \prod_{\forall i} \sigma_{H_{1_{C_i}}}. \tag{13}$$

Empirically, Equation (13) is satisfied by some choice of α, and the mutual information will therefore increase in many cases. For example, in Figure 3, the maximum of mutual information of the combined feature is around $\alpha = 0.7$. Assuming H_1 and H_2 are the power and EM channels, this value is larger than the single channels ($\alpha = 0$ and $\alpha = 1$, respectively). Nevertheless, by choosing $\alpha = 0$, the mutual information of the combined feature could never be worse than the first channel.

In order to determine the value of α that makes the mutual information of the combined channel larger than individual EM and power channels, we shall find $\alpha^* = \arg\max_\alpha(\dot{o}_Z)$. Further, since $\dot{o}_Z \geq 0$, this is equivalent to $\arg\max_\alpha(\dot{o}_Z^2)$. The analytical derivation is provided as follows:

$$\dot{o}_Z^2 = f(\alpha) = \prod_{\forall i} f_i(\alpha) \tag{14}$$

where

$$f_i(\alpha) = \frac{{\sigma_Z}^2}{\sigma_{Z_{c_i}}^2}. \tag{15}$$

Therefore,

$$f(\alpha) = \frac{{\sigma_Z}^{2n}}{\prod_{\forall i} \sigma_{Z_{c_i}}^2} = \prod_{\forall i} \frac{{\sigma_Z}^2}{\sigma_{Z_{c_i}}^2} \tag{16}$$

α^* can be calculated by setting the first derivative of $f(\alpha) = 0$. Using the relationship shown in Equation (11), this derivative can be written as

$$\frac{\mathrm{d}f(\alpha)}{\mathrm{d}\alpha} = \left(\prod_{\forall i} f_i(\alpha)\right)\left(\frac{2\alpha(1-\alpha)}{\alpha^2\sigma_{H_1}^2 + (1-\alpha^2)\sigma_{H_2}^2}\right) v(\alpha) \tag{17}$$

where

$$v(\alpha) = \sum_{\forall i} \frac{\sigma_{H_1}^2\sigma_{H_{2_{C_i}}}^2 - \sigma_{H_2}^2\sigma_{H_{1_{C_i}}}^2}{\alpha^2\sigma_{H_{1_{C_i}}}^2 + (1-\alpha)^2\sigma_{H_{2_{C_i}}}^2}. \tag{18}$$

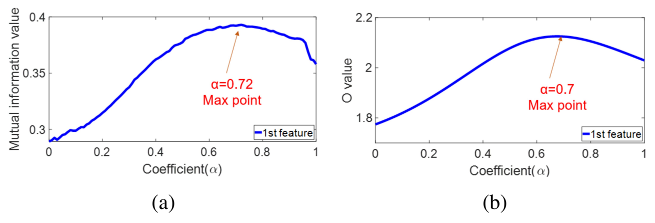


Fig. 3: Combined 1st feature's linear coefficient α vs. (a) mutual information (MI) and (b) $\dot{o}$.

Examining the middle term in Equation (17) yields two trivial zero points that occur at $\alpha = 0$ and $\alpha = 1$. Note again that 1 and 0 correspond to single power and EM channels, respectively. However, another optimal point also occurs when $v(\alpha) = 0$. Thus, we define

$$\alpha^* = \alpha|_{v(\alpha)=0} \tag{19}$$

Empirically, for the 1st feature coefficient, the calculated maximum point using this formulation is 0.69, which is very close to the experimental result of 0.7 in Figure 3(b).

C. Step-by-Step Attack Workflows

The overall methodology consists of three steps or phases: feature collection, feature selection, and testing/attack. The feature collection phase (Section IV-C1) simultaneously collects EM and power traces during AES encryption. In the feature selection phase (Section IV-C2), we select a subset of feature indices from traces and compute their associated coefficients (α) for the dual channel combination. In the testing (attack) phase (Section IV-C3), we collect power and EM traces, generate combined traces with the indices and coefficients from the feature selection phase, and execute one of three attack algorithms to recover target subkeys or key bits.

1) Feature collection phase:

Correlation Attack. At the start of the feature collection phase for the correlation attack, we create training templates for each class label and load the first one onto the target board (i.e., 0x00 for subkey label #1). After loading a sample template, we generate s random plaintexts and send each to the target's AES. For each plaintext, we simultaneously collect power ($P_{t,C_i=0}$) and EM ($E_{t,C_i=0}$) traces for the first round of AES encryption [28]. Here, $1 \leq t \leq s$ and thus t represents the trace number. c_i is the value of the target subkey and is in the range 0x00 to 0xFF while i stands for the index of target subkey ($1 \leq i \leq 256$). This entire process is completed for every target subkey value ($c_i, 1 \leq i \leq 256$). In the end, we get a dataset that has 256 classes. For each class c_i, the dataset contains s power traces and s EM traces. We refer to each sampling point of a trace as a feature. Considering the difficulty of processing all these features, we normalize each power and EM feature into the range [0, 1]. Further, we emprically select a w-size feature window for the target subkey. Thus, from this point forward, the index of each trace is in the range [1, w].

Differential & Machine-learning Attacks. The differential and machine-learning attacks target a single bit of the target subkey related SBox output for each subkey and maximize the difference between different logic levels of the target bit. Thus, we implement an appropriate template for the target board and simultaneously collect s number of EM and power traces with random plaintexts and target bit values during AES encryption. Like the correlation attack, we also normalize the power and EM features into the range of [0, 1].

2) Feature-selection phase: The purpose of the feature selection phase is dimensional reduction, which is critical to reducing computational complexity of the attacks, especially for a real-time implementation. Note that the feature-selection phase for the three attacks is similar. The only difference is the number of classes. The correlation attack targets a whole byte, and thus it has 256 classes. The differential and machine-learning attacks, on the other hand, target a single bit of Sbox output, and thus only have two classes.

For our dual channel attacks where EM and power features are combined according to Equation (8), we begin by calculating the combination coefficients (α) for each feature index k using the approach described in Section IV-B. That is, we calculate all the standard deviations required by Equations (18). Using them, we determine the optimal combination coefficients (α_k^*) according to Equation (18) for each feature index, $1 \leq k \leq w$. Each α_k^* is used in Equation (16) and the resulting value is denoted as $\dot{o}_{Z_k}^2$. This value is an estimate of the mutual information for the kth combined (or dual channel) feature, i.e., $Z_k = \alpha_k^* P_k + (1 - \alpha_k^*) EM_k$. These features are ranked according to $\dot{o}_{Z_k}^2$ and we select a subset consisting of w^* dual channel features ($1 \leq w^* \leq w$).

3) Testing phase: In the testing phase, we collect T_N testing EM (EM) and power (P) traces, generate combined dual channel traces (Z) for the w^* features using the feature selection approach described above, and the combined traces (Z) and their associated plaintexts are used by the attacks described below. Algorithm 1 is used to explain the correlation and differential attacks (combined for brevity) while Algorithm 2 illustrates the ML attack.

Correlation attack. The details of the correlation attack are explained in Algorithm 1 (black and red lines). First, the possible guess subkeys (j) and the kth plaintexts ($p(k,i)$) are used to calculate the hypothesis data (g_1) for different key guess indices (i). The key guess index (i) here means the position of the subkey in the whole key. The 128-bit key can be divided into 16 subkeys, and each subkey has 8 bits. In this case, each subkey has 256 possible values and the hypothesis (g_1) should be calculated 256 times, i.e., once for each possibility. The hypothesis model that we use is the Hamming weight (HW) model (**line 3**). The HW model is based on how many bits with a value of '1' exist in the result. The guessing value of the subkey needs to be XORed with its corresponding 8 plaintext bits, and the result is sent to its related SBox. After getting the hypothesis value ($g_1(k,j)$), the correlation ($r(k,j)$) can be calculated between the combined trace sample points/features (Z) and the hypothesis value (g_1)

Algorithm 1 Algorithm for correlation and differential attack.

Input: **Z**: Testing combined trace, **P**: Plaintext, **N1**: Total trace number in set1, **N0**: Total trace number in set0, **Set1**: Sum of trace for set1, **Set0**: Sum of trace for set0, **i**: Index or position of target subkey, T_N: Number of collected traces, **Red**: Correlation attack, **Brown**: Differential attack

Output: $key_i^{(1)}$, $key_i^{(2)}$: the i^{th} guess subkey for differential attack

```
1:  for $k = 0$ to $T_N - 1$ do
2:      for $j = 0 : 255$ do
3:          $g_1(k,j) = \text{HW}(\text{SBox}(p(k,i) \bigoplus j))$
4:          $g_2(k,j) = \text{BIT}(\text{SBox}(p(k,i) \bigoplus j), 8)$
5:      end for
6:  end for
7:  for $k = 0 : T_N - 1$ do
8:      for $j = 0 : 255$ do
9:          $r(k,j) = \text{Correcoef}(\mathbf{Z}, \mathbf{g_1})$
10:         if $g_2(k,j) = 1$ then
11:             $Set1_{i_j} = Z + Set1_{i_j}$
12:             $N1_{i_j} = N1_{i_j} + 1$
13:         else
14:             $Set0_{i_j} = Z + Set0_{i_j}$
15:             $N0_{i_j} = N0_{i_j} + 1$
16:         end if
17:     end for
18: end for
19: $key_i^{(1)} = \arg\max_j (r(k,j))$
20: $key_i^{(2)} = \arg\max_j (|Set1_{i_j}/N1_{i_j} - Set0_{i_j}/N0_{i_j}|)$
```

for every possibility with the Pearson correlation equation (**line 9**). After finishing the calculation, the index value of the largest peak in the correlation coefficients is chosen as the guessed subkey (**line 19**). In our later experiments, we repeat this process N_{iter} times, verify the outcome of the guessed subkey, and record the number of times that the algorithm's subkey guess is correct (N_{correct}). The success rate (SR) represents the accuracy of the attack and is expressed as

$$SR = \frac{N_{\text{correct}}}{N_{\text{iter}}} \times 100\% \tag{20}$$

Differential attack. The details of the differential attack are presented in Algorithm 1 (black and brown lines). The hypothesis data (g_2) stands for the target bit of the corresponding SBox outcome (**line 4**). j is the guess subkey, i is the position of the target subkey in the whole key, and $p(k,i)$ is the outcome of the initial round. If the target bit ($g_2(k,j)$) equals 1, the power traces are divided into $Set1_{ij}$ (**line 10**), and the total number of traces in set 1 ($N1_{ij}$) in incremented by 1 (**lines 11 to 12**). Otherwise, the power traces are divided into $Set0_{ij}$ (**line 13**), and the total number of traces in set 0 ($N0_{ij}$) is incremented by 1 (**line 14**). In the end, the guessed key is the one with the maximum absolute difference between the mean values of set 1 and set 0 (**line 20**). Success rate (SR) is also used to evaluate this algorithm's effectiveness.

Machine-learning (ML) attack. The details of the machine-learning attack are presented in Algorithm 2. When the machine-learning module receives the combined testing trace (Z), it obtains the result of the target bit (g) from an SVM classifier (**line 3**). The target bit predicted by the classifier

Algorithm 2 Algorithm for machine-learning attack.

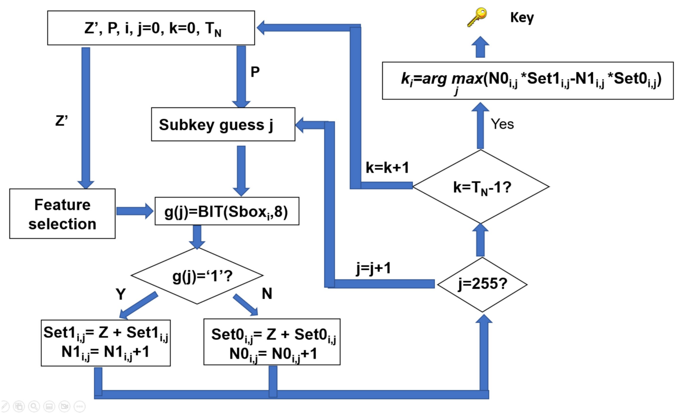
```
Input: Z: Testing combined trace, i: bit position of target
    subkey, Classifier: SVM classifier
Output: $key_i$ : the $i^{th}$ guess subkey
1: for $k = 0 : T_N - 1$ do
2:     for $j = 0 : 255$ do
3:         $g_{\text{classifier}}(k) = \text{Classifier}(Z)$
4:         $g(k, j) = \text{BIT}(\text{SBox}(p(k, i) \bigoplus j), i)$
5:     end for
6: end for
7: $key_i = \arg\min_j \left(\sum_{\forall k} |g_{\text{classifier}}(k) - g(k, j)|\right)$
```

($g_{classifier}$) is compared with the guess target bit ($g(k)$) from 256 guess subkeys (**line 4**). Ideally, the match rate between the wrong guess bit (g) and the subkey bit from the classifier ($g_{classifier}$) should be around 0.5 since they are not related. However, the match rate between the guess target bit from the classifier ($g_{classifier}$) and the correct guess bit (g) should be close to 0 as the total number of testing traces (T_N) becomes larger. In the end, we take the index of guess with the highest match rate as the correct subkey (k_i) for target bit index i (**line 7**) by minimizing the absolute sum of differences between the guess and classifier results across traces. The success rate (SR) is also used to evaluate the effectiveness of this algorithm.

V. Real-time Implementations

In the section, we explain how to implement the DPA and SVM-based SCAs inside RDCP for the real-time mode. Real-time implementations are important for intelligence-related applications which demand in situ and stealthy approaches. SCAs originated during the Cold War, e.g., TEMPEST [1] which is now mitigated by RF shielding. However, if a small module like RDCP (see Figure 5) could be inserted into a system, it can bypass such countermeasures. In contrast, modern SCAs require the attacker to acquire the target under attack, bring it into a lab setting, and use an expensive oscilloscope and PC to extract the key. Some target systems are too large and moving them into the lab is not feasible. Further, even for smaller targets that are easy to remove, stealing the target alone might alert the system's owner that it is under attack. This is non-ideal for intelligence-gathering applications where the point is to recover information without being detected.

The FPGA on RDCP is a Spartan 3e500 [27]. Even though it is inexpensive and tiny in size, it still has limitations in data transfer speed and memory size (30kB). Thus, the most challenging aspects of implementing the algorithms in real-time are the time-consuming and memory-costly calculations. Thus, we cannot directly load the correlation attack algorithm presented in Section IV-C since it needs too much space. The differential and machine learning attacks can be implemented, however, with careful memory management. Below, we describe how we implemented the testing (attack) phase of the differential and ML attack classifiers into RDCP.



Z', P, i, j=0, k=0, T_N
P
Z'
Subkey guess j
Feature selection
g(j)=BIT(Sbox_i,8)
g(j)='1'?
Y
N
Set1_{i,j}= Z + Set1_{i,j} N1_{i,j}= N1_{i,j}+1
Set0_{i,j}= Z + Set0_{i,j} N0_{i,j}= N0_{i,j}+1
j=j+1
j=255?
k=k+1
k=T_N-1?
Yes
k_i=arg max_j(N0_{i,j} *Set1_{i,j}-N1_{i,j} *Set0_{i,j})
Key

Fig. 4: Flow diagram for differential attack in RDCP.

A. Differential Attack's RDCP Implementation

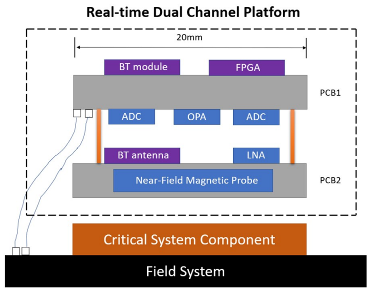
In short, memory is reused by processing traces for each plaintext one at a time and also saved by only saving dual-channel features. Complicated calculations inside the FPGA are bypassed by transforming the division into multiplication. With this approach, no matter how many traces RDCP uses in the differential attack, memory usage remains the same. This makes the real-time implementation scalable.

The step-by-step diagram is shown in Figure 4. RDCP first receives the plaintext P, and and combine testing traces (Z). The indices with the highest mutual information contents and their combinational coefficients are determined in the feature-selection phase[1]. RDCP stores a subset of these useful features (e.g., $w^* = 5$ in our results section) of the upcoming power/EM traces, and uses the associated coeffcients to shorten the combined traces (Z). At the same time, RDCP calculates the target bit (g_j) from the associated bit of the correlated Sbox (i) output. The correlated Sbox is decided by the position of the target subkey (i), and the hypothesis value j. For AES-128, $i \in [1, 16]$ and $j \in [0, 255]$. If the target bit is equal to logic 1 (0), the combined feature is added into $set1_{i,j}$ ($set0_{i,j}$), and the number of traces in set 1 (set 0) denoted by $N1_{i,j}$ ($N0_{i,j}$) is increased by 1. Once the hypothesis (j) reaches 255, RDCP moves on to process the next trace and associated plaintext. Then once RDCP has processed enough traces as specified by the user (T_N), it adopts a bubble sorting method [24] to find the index of the max absolute value of the difference between the "mean value" of corresponding $set1$ and $set0$. Note that since the division operation is hard to achieve in an FPGA, we covert the division into multiplication as follows: the mean difference $\left(\frac{Set1_{i,j}}{N1_{i,j}} - \frac{Set0_{i,j}}{N0_{i,j}}\right)$ is transformed to $(Set1_{i,j} \times N0_{i,j}) - (Set0_{i,j} \times N1_{i,j})$.

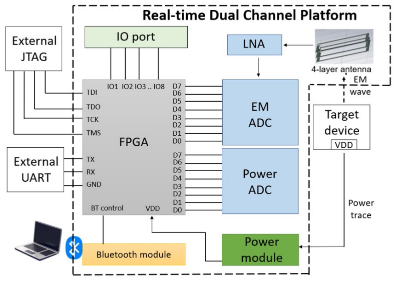
B. ML Attack's RDCP Implementation

In short, real-time ML attack implementation exploits the same memory reuse/saving techniques as the differential attack. The detailed flow diagram is not shown since it is similar to the differential attack. The main difference is the classifier.

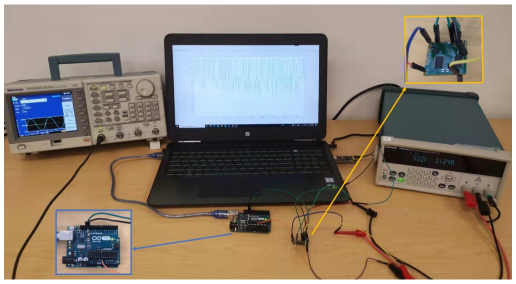
[1]This is done offline using the approach described in Section IV-C2


Real-time Dual Channel Platform
20mm
BT module
FPGA
PCB1
ADC
OPA
ADC
BT antenna
LNA
Near-Field Magnetic Probe
PCB2
Critical System Component
Field System

(a)



(b)



(c)

Fig. 5: (a) Depiction of RDCP in the field. RDCP is a 20 mm by 20 mm PCB board with chips for signal processing, communication, and memory. RDCP also measures the target's power/EM traces; (b) Depiction of the RDCP schematic. RDCP is powered by the V_{dd} pin of the target. The traces are digitalized by ADC chips on RDCP. JTAG and UART modules are used to program the FPGA and transmit data; (c) Experimental setup for collecting power traces from ATmega328P.

Considering the complexity of the ML algorithm, we adopt the linear-SVM classifier

$$f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \mathbf{b}. \tag{21}$$

Here $\mathbf{x}$ stands for the input feature vector, $\boldsymbol{\beta}$ refers to the SVM coefficient and $\mathbf{b}$ is the bias for the classifier. $\boldsymbol{\beta}$ and $\mathbf{b}$ are obtained offline using the approach described in Section IV-C3. To implement SVM classifier on the FPGA, we bypass the floating point calculations by taking their 10 times integer approximation.

VI. Experimental Results and Discussions

In this section, we validate our methodology by simultaneously collecting power and EM traces from a target AES implementation and extracting the key using two different modes: offline and real-time. The proposed dual channel approach is compared using single channel variants as well as other existing dual channel approaches (concatenation [19] and sum [17]).

A. Experimental Setup

Power/EM trace measurements. For collecting EM/Power traces in offline mode, we use the MDO3102 oscilloscope [29] and a commercial EM probe (LF1 Set from LANGER EMV Technik [30]). For collecting EM/power traces in real-time modes, we use our custom-designed 20 mm by 20 mm RDCP. Figure 5(a-b) depicts the RDCP deployment scenario and schematic. RDCP contains two ADCs [31] for digitalizing power/EM traces at the same time, a Bluetooth module [33] for remote communication, and a Xilinx Spartan-3E (XC3S500E) FPGA for data processing. We also use 8 I/O ports on the board and connect them to an external UART module to transmit trace data for offline mode.

The experimental setup is shown in Figure 5(c). The target device is Arduino UNO, and its original core frequency is 16 MHz. The sampling speed of the RDCP and oscilloscope (MDO3102) for real-time and offline modes are 128MS/s and 100MS/s, respectively. The power/EM traces are collected by RDCP and transmitted to the PC via UART whenever offline processing is performed. In the real-time mode, RDCP collects power/EM traces and processes them internally. A guess subkey is sent via Bluetooth module or external UART modules to the laptop.

Methodology parameters. In our experiments, the window size of the traces (w) is set to 200 for offline experiments, and number of traces collected for each class to compute coefficients and mutual information (s) is 1,000. The number of selected features (w^*) in the real-time mode is 5, and in the offline-mode is 20.

AES target benchmark. The main function of the target benchmark is to encrypt plaintext with the AES-128 algorithm. While training our classifiers (specifically, finding the relationship between all features and the target subkey), we need to send the same plaintexts to the Arduino and only make modifications to the target subkey. In other words, the target subkey is the only variable in the feature-selection phase.

Attack model. We assume the attacker can access the encryption device. This means the attacker can get the information of plaintext and ciphertext, and access the power/EM channels of the encryption device with oscilloscope/RDCP and/or a commercial EM probe. However, the attacker does not know the hidden key inside the encryption module.

All experiments are performed on two different Arduino UNOs. Training and calculation of optimal coefficients/features are only performed using data from board 1 while testing is performed on both boards 1 and 2.

B. Offline Mode Results

In this subsection, we use offline mode and the three attacks mentioned in the methodology section. Table II shows the number of measurements to disclosure (MTD) to achieve a 100% success rate with the differential attack for all 16 AES-128 subkeys on both Arduino UNO boards. Here, the "Combined" results combine EM and power channel using the proposed dual channel methodology while power/EM channel refer to only using power/EM features. The single EM and the power channels need above 400 and 600 traces for different boards, respectively. In the offline experiments,

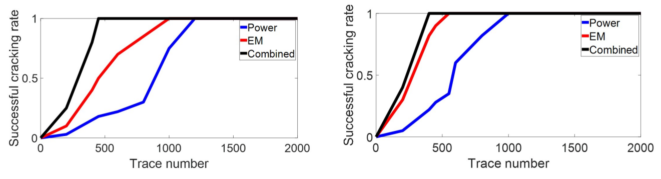
TABLE II: MTD for AES-128 encryption module in offline mode using differential attack.

Subkey	Power		EM		Combined	
	Board 1	Board 2	Board 1	Board 2	Board 1	Board 2
1	150	150	100	100	70	85
2	400	350	250	250	160	200
3	950	900	750	600	420	400
4	800	800	400	400	350	350
5	800	800	380	400	320	300
6	200	170	80	100	40	50
7	1200	1150	1000	950	450	450
8	1000	1000	700	750	400	420
9	800	800	400	380	300	300
10	200	250	150	150	60	50
11	300	320	250	280	150	180
12	1000	900	650	600	450	430
13	1000	950	550	550	400	400
14	300	320	200	200	140	150
15	700	650	400	450	350	350
16	400	400	250	250	200	250
Avg.	637	619	394	400	267	272

interference/white noise is inevitably generated, and they could directly impact the collected power traces from PCB boards. However, for collecting EM waves, the selected position is more close to the signal source and these interference signals/white noise attenuate through the air. Thus, the power channel has lower SNR ratio than the EM channel, and needs more traces to achieve full extraction rate than the EM channel. The experiment result shows the combined feature needs less traces than power/EM channel to achieve 100% success rate. That is, the proposed approach only needs 267 traces on average for extracting subkeys from board 1, and 272 traces for board 2.

Besides the difference between single and dual channels, it is clear that different subkeys require different number of traces to achieve 100% success rate. For example, for the combined case in Table II, subkeys 1, 6, and 10 need less than 100 traces. However, subkeys 3, 7, 8, 12, and 13 need more than 400 traces. This behavior is consistent regardless of the channel used. The reason is caused by the encryption module on Arduino UNO, and the position where we collect traces. When the Arduino encrypts data, it process the plaintext with different subkeys at different positions. If this position is closer to where we collect power/EM traces, the SNR is higher and we need less traces to achieve full extraction rate. Nevertheless, the proposed dual channel methodology (combined) has various improvements against power/EM channel for different subkeys. In Figure 6, we plot success rate versus trace number for subkeys 7 and 13 for the differential attack algorithm. For subkey 7, the combined channel requires 40% and 65% fewer traces compared to single channel EM and power. However, as shown in Figure 6(b), the combined channel only increase in efficiency by 8% than the EM channel for subkey 13.

We note a similar improvement for machine-learning attack results. Figure 7 shows the success rate versus trace number for subkeys 12 and 16. In Figure 7(a), the combined channel for subkey 12 increases efficiency by 50% and 80% from single EM and power channels. However, in Figure 7(b), the combined channel only reduces number of traces by 10% compared to EM channel. The improvement in efficiency is



(a) (b)

Fig. 6: Success rates vs. number of traces for differential attack on subkeys (a) 7 and (b) 13.

decided by SNR and the mutual information of features from power/EM channels at different time indices. Empirically, if the mutual information of the EM channel is much better than the power channel, the combined channel will chose $\alpha \approx 1$ and won't achieve as much improvement as the case where EM and power channels have similar mutual information.

In Table III, we also compare the average MTD for achieving 100% success rate with our three methods and with existing dual channel methods. Compared with the average traces used for extracting subkeys with [17] and [19], we save over 50% and 70% traces with our dual-channel ML variant.

The results in Tables II and III also prove the robustness of our methodology when a different board is used for training and testing. Even though there exists some process variations among boards, we need nearly the same amount of traces for 100% success rate in board 2 using feature indices and coefficients obtained from analysis of board 1 data. Due to space limitations, we only present results for two boards. However, we note that a third board produced similar results.

In Table III, we also have tried PCA for feature extraction in our machine-learning attack in offline mode. In PCA, 1000-sample traces are used and we take 20 principal components. Compared to the machine-learning attack without PCA, the single EM channel needs 179 traces on average, the single power channel needs 257 traces and the combined channel needs 125 traces. It is better than the machine-learning attack result with a single/combined channel before (280 for power, 202 for EM, 143 for combined). It must be noted, however, that PCA is not lightweight enough for implementation on a resource-limited device and for real-time. PCA's dimensionality reduction only reduces the effort needed to perform classification. Implementing PCA on the device performing the attack needs *extra memory* to store all power and EM trace sampling points as well as a large PCA coefficient matrix. In addition, to generate each principal component, *all the samples (1,000) of each trace need to be linearly combined* using the PCA decimal coefficient matrix. This is time consuming and will make it difficult to meet real-time constraints without larger computational resources. In contrast, the proposed approach only needs to store the useful trace samples (e.g., 5 for each channel) to create 5 features.

C. Real-time Mode Results

In the real-time mode, we implement DPA and machine-learning algorithms onto RDCP and let it extract the subkey

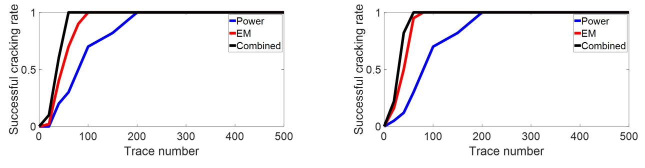


Fig. 7: Offline success rate vs. number of traces for machine learning attack on subkeys (a) 12 and (b) 16.

TABLE III: Comparison of average MTD for 100% success rate on AES-128 in offline mode.

Method	Channel	Trace	
		Board 1	Board 2
Correlation attack	Power	5597	5802
	EM	4135	4322
	Combined	3216	3455
Differential attack	Power	690	702
	EM	406	423
	Combined	267	272
Machine-learning attack w/out PCA	Power	280	285
	EM	202	207
	Combined	143	150
Machine-learning attack w/ PCA	Power	257	266
	EM	179	185
	Combined	125	129
Concatenated trace profiling [19]	Combined	399	413
Sum over std non-profiling [17]	Combined	3623	3850

bits internally. Since feature selection coefficents and indices can be determine offline, we can eliminate less useful features and then combine the useful ones just as in the offline mode. The details of how we implement the proposed methodology in real-time was presented in Section V.

The number of MTD to achieve 100% success rate for different channels in real-time is presented in Table IV. Compared with the offline mode results, RDCP needs more traces to achieve 100% success rate. The average MTD for power/EM/Combined channel of board 1 is 3171, 5637, and 2450. We also show the success rate vs. number of traces for extracting subkey 6 in Figure 8. For example, with 200 traces for the DPA algorithm, the combinational channel has 70% success rate, the power channel has 55% successful extraction rate and the EM channel has 25% successful extraction rate. However, in the offline mode, the extraction rate for subkey 6 could achieve 100% with 200 traces. This also happens with the ML attack.

Compared with the offline mode results, RDCP needs more traces to achieve 100% success rate. The likely culprit is the approximations needed for the real-time approach as discussed in Section V. For bypassing division and decimal calculation inside FPGA, we approximate all decimals with integers, and this reduces the attack accuracy. Another thing to clarify is why the single power channel requires less traces than the single EM channel in the real-time mode.[2] This is due to the design of the internal EM antenna inside RDCP. Different from the mature design of the commercial EM probe in the offline mode, the internal EM antenna inside RDCP is distributed

[2]This is the opposite trend compared to offline mode (see Table II).

TABLE IV: MTD for AES-128 encryption module in real-time using differential attack.

Subkey	Power		EM		Combined	
	Board 1	Board 2	Board 1	Board 2	Board 1	Board 2
1	350	500	1200	1200	300	400
2	2500	3000	6000	6500	2000	2200
3	5000	5500	7000	7500	4000	4500
4	4000	5000	7500	7800	3500	4000
5	3000	5000	5000	6000	2500	3500
6	150	200	500	650	100	150
7	5500	6000	10000	10000	4000	4500
8	2500	3500	5500	6500	2000	3000
9	3000	3500	8000	8000	2000	3000
10	550	900	3000	4000	500	700
11	3000	4000	7500	8000	2500	3000
12	6000	6500	9000	9000	4000	4200
13	6000	6500	9500	9500	4500	4700
14	3000	3000	7000	7000	2500	2500
15	5000	5000	1000	10000	4000	4000
16	1200	1500	2500	2500	800	900
Avg.	3171	3725	5637	6509	2450	2828

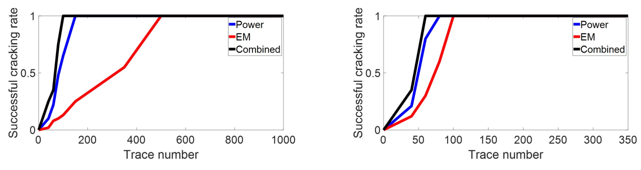


Fig. 8: Real-time success rate vs. number of traces on subkey 6 for (a) differential attack and (b) machine-learning attack.

in four inner layers of RDCP, and the bad interconnection between different antenna layers undermines the SNR of collected EM traces in the real-time mode. Nevertheless, the real-time experiments still demonstrate the efficiency of the proposed dual-channel methodology. That is, it achieves a higher success rate compared to a single EM/power channel with the same amount of traces and has lower MTD.

VII. Conclusion and Future Work

In this paper, we proposed a mutual-information-based feature selection method to find high-correlation features and the best combination of power and EM traces for SCAs. Then, we successfully implemented the dual channel DPA and SVM algorithms in real-time. In future work, we will upgrade RDCP to a new version that consists of a larger memory FPGA, higher accuracy ADCs, and improved EM measurement capabilities. The larger memory FPGA could contain more complicated algorithms (e.g., deep learning) and the 12-bit ADC can collect EM and power traces at higher accuracy and faster speed. With the improved version of RDCP and more advanced algorithms, we also hope to test the framework on a RISC-V implementation on an FPGA as well as on a masked implementation of AES and demonstrate real-time instruction disassembly.

Acknowledgment

This work has been supported in part by the US Army Research Office (ARO) under award number W911NF-19-1-0102.

References

[1] Anderson, R. *Security engineering: a guide to building dependable distributed systems.* John Wiley & Sons, 2020.

[2] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[3] Peng, H., Long, F. & Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **27**, 1226-1238 (2005)

[4] Ding, C. & Peng, H. Minimum redundancy feature selection from microarray gene expression data. *Journal Of Bioinformatics And Computational Biology*. **3**, 185-205 (2005)

[5] Cover, T. Elements of information theory. (John Wiley & Sons,1999)

[6] Park, J., Xu, X., Jin, Y., Forte, D. & Tehranipoor, M. Power-based Side-Channel Instruction-level Disassembler. *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. pp. 1-6 (2018)

[7] Entropy and Mutual Information (Continuous Random Variables), https://gtas.unican.es/files/docencia/TICC/apuntes/

[8] Information Theory and Predictability Lecture 7: Gaussian Case, https://www.math.nyu.edu/ kleeman/infolect7.pdf.

[9] Kocher, P., Jaffe, J. & Jun, B. Differential power analysis. *Annual International Cryptology Conference*. pp. 388-397 (1999)

[10] Kocher, P., Jaffe, J., Jun, B. & Rohatgi, P. Introduction to differential power analysis. *Journal Of Cryptographic Engineering*. **1**, 5-27 (2011)

[11] Kocher, P., Jaffe, J., Jun, B. & Others Introduction to differential power analysis and related attacks. (1998)

[12] Brier, E., Clavier, C. & Olivier, F. Correlation power analysis with a leakage model. *International Workshop On Cryptographic Hardware And Embedded Systems*. pp. 16-29 (2004)

[13] Benhadjyoussef, N., Mestiri, H., Machhout, M. & Tourki, R. Implementation of CPA analysis against AES design on FPGA. *2012 International Conference On Communications And Information Technology (ICCIT)*. pp. 124-128 (2012)

[14] Gierlichs, B., Batina, L., Tuyls, P. & Preneel, B. Mutual information analysis. *International Workshop On Cryptographic Hardware And Embedded Systems*. pp. 426-442 (2008)

[15] Gandolfi, K., Mourtel, C. & Olivier, F. Electromagnetic analysis: Concrete results. *International Workshop On Cryptographic Hardware And Embedded Systems*. pp. 251-261 (2001)

[16] Agrawal, D., Archambeault, B., Rao, J. & Rohatgi, P. The EM side—channel (s). *International Workshop On Cryptographic Hardware And Embedded Systems*. pp. 29-45 (2002)

[17] Souissi, Y., Bhasin, S., Guilley, S., Nassar, M. & Danger, J. Towards different flavors of combined side channel attacks. *Cryptographers' Track At The RSA Conference*. pp. 245-259 (2012)

[18] Matlab: fitcsvm. https://www.mathworks.com/help/stats/fitcsvm.html

[19] Standaert, F. & Archambeau, C. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. *International Workshop On Cryptographic Hardware And Embedded Systems*. pp. 411-425 (2008)

[20] Hutter, M., Mangard, S. & Feldhofer, M. Power and EM attacks on passive 13.56 MHz RFID devices. *CHES*. **7** pp. 320-333 (2007)

[21] Ding, G., Chu, J., Yuan, L. & Zhao, Q. Correlation Electromagnetic Analysis for Cryptographic Device. *2009 Pacific-Asia Conference On Circuits, Communications And Systems*. pp. 388-391 (2009)

[22] Lerman, L., Bontempi, G., Markowitch, O. & Others Power analysis attack: an approach based on machine learning.. *Int. J. Appl. Cryptogr.*. **3**, 97-115 (2014)

[23] Lerman, L., Bontempi, G. & Markowitch, O. A machine learning approach against a masked AES. *Journal Of Cryptographic Engineering*. **5**, 123-139 (2015)

[24] Astrachan, O. Bubble sort: an archaeological algorithmic analysis. *ACM Sigcse Bulletin*. **35**, 1-5 (2003)

[25] Archambeau, C., Peeters, E., Standaert, F. & Quisquater, J. Template attacks in principal subspaces. *International Workshop On Cryptographic Hardware And Embedded Systems*. pp. 1-14 (2006)

[26] Mohamed, M., Bulygin, S., Zohner, M., Heuser, A., Walter, M. & Buchmann, J. Improved algebraic side-channel attack on AES. *2012 IEEE International Symposium On Hardware-Oriented Security And Trust*. pp. 146-151 (2012)

[27] Spartan3e500, https://docs.xilinx.com/v/u/en-US/ds312

[28] AES, https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[29] MDO,https://www.tek.com/en/oscilloscope/mdo3000-mixed-domain-oscilloscope-manual/mdo3000-series-3

[30] EMV, https://www.langer-emv.de/en/index

[31] ADC08200, https://www.ti.com/lit/ds/symlink/adc08200.pdf

[32] Arduino, https://en.wikipedia.org/wiki/Arduino_Uno

[33] SESUB, https://product.tdk.com/system/files/dam/doc/product/rf/rf/module/-catalog/sesub-pan-d14580_en.pdf

[34] Milanov, E. The RSA algorithm. *RSA Laboratories*. pp. 1-11 (2009)

[35] Brumley, B. A3/A8 & COMP128. *T-79.514 Special Course On Cryptology*. pp. 1-18 (2004)

[36] Maghrebi, H. Deep learning based side channel attacks in practice. *Cryptology EPrint Archive*. (2019)

[37] Kubota, T., Yoshida, K., Shiozaki, M. & Fujino, T. Deep learning side-channel attack against hardware implementations of AES. *Microprocessors And Microsystems*. **87** pp. 103383 (2021)

[38] Lerman, L., Bontempi, G. & Markowitch, O. Side channel attack: an approach based on machine learning. *Center For Advanced Security Research Darmstadt*. **29** (2011)

[39] Picek, S., Heuser, A., Jovic, A., Ludwig, S., Guilley, S., Jakobovic, D. & Mentens, N. Side-channel analysis and machine learning: A practical perspective. *2017 International Joint Conference On Neural Networks (IJCNN)*. pp. 4095-4102 (2017)

[40] Nie, T. & Zhang, T. A study of DES and Blowfish encryption algorithm. *Tencon 2009-2009 IEEE Region 10 Conference*. pp. 1-4 (2009)