EURASIP Journal on Information Security

RESEARCH Open Access

Security evaluation of Tree Parity Re-keying Machine implementations utilizing side-channel emissions

Jonathan Martínez Padilla[1,2*], Uwe Meyer-Baese[1,2] and Simon Foo[1,2]

Abstract

In this work, side-channel attacks (SCAs) are considered as a security metric for the implementation of hybrid cryptosystems utilizing the neural network-based Tree Parity Re-Keying Machines (TPM). A virtual study is presented within the MATLAB environment that explores various scenarios in which the TPM may be compromised. Performance metrics are evaluated to model possible embedded system implementations. A new algorithm is proposed and coined as Man-in-the-Middle Power Analysis (MIMPA) as a means to copy the TPM's generated keys. It is shown how the algorithm can identify vulnerabilities in the physical device in which the cryptosystem is implemented by using its power emissions. Finally, a machine learning approach is used to identify the capabilities of neural networks to recognize properties of keys produced in the TPM as they are transferred to an encryption algorithm. The results show that physical exploits of TPM implementations in embedded systems can be identified and accounted for before a final release. The experiments and data acquisition is demonstrated with an implementation of a TPM-AES hybrid cryptosystem in an AVR microcontroller.

Keywords: Tree parity machine, Side channel, Machine learning, Neural networks, Microcontrollers, Security evaluation

1 Introduction

The advent of quantum cryptanalysis has elicited a growing concern for the security of modern cryptographic protocols. For instance, in the mid-1990s, Shor published a quantum algorithm that can factor the prime numbers from any integer with extreme efficiency, thus breaking public-key cryptosystems such as the widely used RSA [1]. When quantum computing technologies rise to practicality, some of the most popular forms of public-key cryptography may be endangered. To mitigate the potential reality, different types of public-key schemes have been proposed as substitutes that deviate from traditional number theory: hash, lattice, and multivariate quadratic equation, are some among other schemes.

The neural key exchange protocol is considered to be a possible alternative that could be resistant to quantum cryptanalysis mostly due to not being established off number theory. Originally proposed in [2], it consists on using the properties of a neural network to generate public or private keys for a cryptosystem with the use of its synaptic weights. But if future cryptosystems where to use the neural key exchange schemes for public-key cryptography, the question arises as to what other forms of attacks may the neural key scheme be vulnerable to. The cryptographic device that hosts the neural key exchange mechanisms could encounter weaknesses that may be exploited for malicious intent. Having the vulnerabilities cataloged and accessible can allow for effective countermeasures to be considered at the time of implementation.

Embedded systems are a crucial asset in many aspects of technology. They are used in various applications including motion estimation in imaging [3, 4], computing optimization [5], secure exchange of information, and among others. Yet every computational device emits a series of leakages coined as side channels. These can be in the form of power dissipation, electromagnetic emissions, execution time of a process, emanated light, and

*Correspondence: jem14m@my.fsu.edu
[1]FAMU-FSU College of Engineering, Florida State University, 2525 Pottsdamer St, 32310 Tallahassee, Florida, USA
[2]Machine Intelligence Laboratory, Florida State University, 2525 Pottsdamer St, 32310 Tallahassee, Florida, USA

among other physical attributes. The methods used to determine a cryptographic device's physical vulnerabilities lie within the realm of side-channel analysis, while the use of it for the purpose of extracting private or secret keys from a cryptographic protocol is described as side-channel attacks (SCAs) which was pioneered by Kocher in [6]. SCAs largely depend on statistical schemes to extract keys and are relatively cheap to mount and perform [7, 8]. Advancements in side-channel cryptanalytics have highlighted the need to account for countermeasures against them especially in the design of secure systems.

This work presents a series of evaluation methods that aim to reveal vulnerabilities in a hardware-based implementation of a neural key architecture, namely the Tree Parity Re-Keying Machine (TPM). First, a virtual model of a TPM-AES hybrid cryptosystem is built to evaluate performance. A case study of various security-compromising scenarios is observed and realized to determine potential weaknesses in a cryptographic device. The model is used to develop an embedded hardware implementation of the TPM hybrid with the ATMega328P microcontroller as the target device. A new algorithm is proposed to effectively synchronize a TPM's generated keys by measuring its power dissipation. Finally, a machine learning approach is used to classify a TPM's generated keys by monitoring the power dissipation of the data transfer between the generated key and an encryption operation in AES. The following sections describe the methodologies used to achieve key extraction and emphasize the need to include additional protection schemes in embedded hardware implementations of TPM cryptosystems.

2 Background and related work

The attack algorithms and numerical foundations for the construction of side-channel attacks are described as preliminaries for execution of SCAs. In the following subsections, a brief description of the procedures used in SCAs are described and generalized. Additionally, the TPM's algebraic structure is formulated and discussed. The unsupervised learning techniques and learning rules are described in more detail to facilitate the incorporation of a neural key exchange protocol.

2.1 Tree Parity Re-Keying Machine

Neural networks as a solution for cryptographic applications originated in [2] with the creation of the TPM. Figure 1 shows the tree-like diagram towards which carried the inspiration for its name. The original paper suggested that synchronized synaptic weights would be anti-parallel to one another and could serve as keys for different cryptosystems. Yet slight modifications to the original algorithm turn the TPM as a generator of equal shared keys.

The structure is comprised of two *machines*, each with an input layer, a hidden layer, and an output layer. The input layer has a total of $n = 1, 2, \ldots, N$ input nodes interconnected to $k = 1, 2, \ldots, K$ hidden nodes in the hidden layer. The inputs ($x_{k,n}$) are a stream of random numbers that are applied to the machines in the TPM. The hidden nodes (y_k) connect to a single output node in the output layer (O_m). The synaptic or learning weights (ω) are wedged between the input and hidden layers. The weights take on bounded integer values $\omega_{k,n} \in [-L, L]$. The hidden nodes have an activation function in the form of Eq. (1). The activation function takes the sum of products between the inputs and weights at each hidden node and determines its sign to be either negative or positive where the sgn(·) function outputs -1 for the former and $+1$ for the latter. The output node also has its own activation function. The trigger is the product of each hidden node's output in the form of Eq. (2). The output of the TPM is then delivered to an identical machine. The possible output values are bounded by $O_m \in \{-1, 1\}$.

$$y_k(x, w) = \operatorname{sgn}\left(\sum_{n=1}^{N} x_{k,n}\omega_{k,n}\right) \tag{1}$$

$$O_m = \prod_{k=1}^{K} y_k \tag{2}$$

A conditional operation is performed to determine whether or not to update the synaptic weights. For the unsupervised training method, the adaptation step requires that the machines output an equal value. Even if both outputs are equal, another condition is set comparing the machine's output to the value of a hidden node's output. This ensures that only the hidden node, whose weights are mismatched to the machine's output, are updated. Additionally, the adaptation step uses a learning rule to modify and update the weights. The result of the learning rule is clipped to satisfy the specified boundaries L of the weights. After complete synchronization of the machines, the weights are used as keys for a cryptosystem. A special case is seen when determining the sign of a zero value. For the machine that initiates the key exchange $y_{A_k} = \operatorname{sgn}(0) = 1$, for the responding machine $y_{B_k} = \operatorname{sgn}(0) = -1$ is held.

The three main learning rules that aid into the synchronization of weights in the adaptation step are *Hebbian*, *anti-Hebbian*, and *Random Walk*. Hebbian learning takes into account both the response received from a machine as well as its input value. Thus, the weight is strengthened in proportion to O_m and $x_{k,n}$ as postulated in Eq. (3) where the new weight is represented by ω^+.

$$\omega^+_{k,n} = \omega_{k,n} + O_m x_{k,n} \tag{3}$$

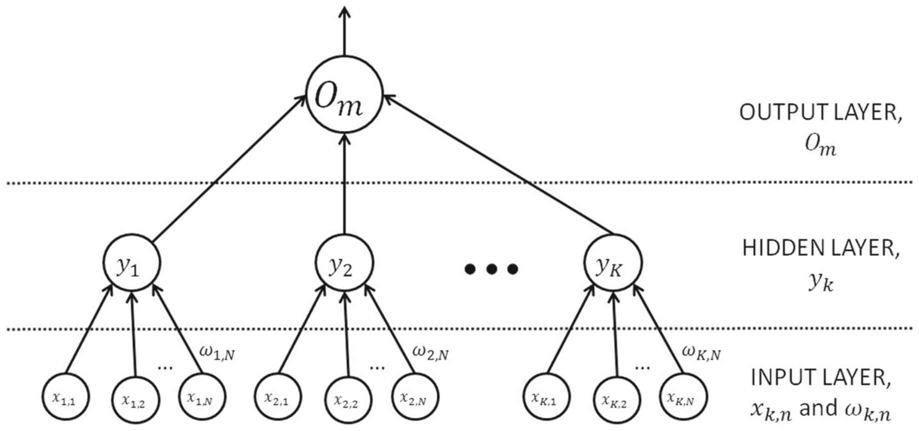

Fig. 1 TPM diagram. The TPM neural network consists of input, hidden, and output layers to generate keys from its weights

Conversely, in anti-Hebbian learning, the weight is weakened in proportion to its output's activation function as seen in Eq. (4).

$$\omega_{k,n}^{+} = \omega_{k,n} - O_m x_{k,n} \tag{4}$$

Finally, Random Walk learning does not take into account the output of a machine, solely leaving the weights to strengthen or weaken based on the values of the inputs to the TPM. Since the input nodes are a constant stream of randomly generated values, the weights are stochastically updated with respect to Eq. (5).

$$\omega_{k,n}^{+} = \omega_{k,n} + x_{k,n} \tag{5}$$

2.2 Side-channel cryptanalysis

Advancements in cryptanalytics allow for the understanding of the security capabilities of a proposed cryptographic protocol. Historically, cryptanalysts largely relied on how to uncover vulnerabilities based on the mathematical foundation of a cipher. But by the 1950s, the US government introduced the TEMPEST [9] program to evaluate the electromagnetic emanations in their computers due to the fear that compromising information was leaking. Although the security of systems via side channels were being studied, it was a largely omitted area in the field of cryptanalytics until the mid-1990s. This was when Kocher introduced a novel paper detailing *timing attacks* against different types of cryptographic implementations in computing devices [6].

Side-channel cryptanalysis has expanded as a crucial asset to evaluate the security in physical implementations of the different types of cryptographic algorithms. Statistical and analytical techniques are used to correlate these leakages into processed data, so its content can be extracted. A variety of methods exist which are chosen based on availability of equipment to monitor side channels, familiarity with a target cryptographic protocol, computational resources of a method, known architectural weaknesses of a computational device, and among other reasons. The most often sought out side channel is the power consumption of a target device. Some of the prevalent SCA methods are simple power analysis (SPA), differential power analysis (DPA), correlation power analysis (CPA), and template attacks (TA).

SPA comprises of acquiring power traces from a cryptographic device and looking at visible patterns that can be related to functions of known cryptosystems or to actual private keys. The attacker can use the traces to reverse engineer instruction sequences when sufficient knowledge of a processor's architecture is known, thus giving clues about what the cryptosystem is processing [10, 11]. DPA utilizes statistical methods to indicate if a key bit is dependent to the power consumption of the device. To execute DPA, an attacker must record numerous power traces while the implemented cryptosystem operates on input data. After collecting the power traces, a statistical analysis is performed to identify how samples in a power trace correlate to changes of a possible key [12]. CPA is an extension of DPA that adds improvements to the statistical methods used. The main difference is in the use of Pearson's correlation coefficient to determine if there's a proportional association between the target key and the power consumption of a cryptographic device as described in [13]. The Hamming weight (HW) and Hamming distance (HD) power consumption models are widely used to describe the output behavior of an intermediate function in this attack. Finally, a TA is based on acquiring a private key by means of multivariate statistics and probability methods [14]. An attacker essentially builds a set of templates from a clone device that has the expected power trace for a key-dependent operation when a possible key is used. All the templates form a tuple that are used to compare with the original power trace of the target device. A probabilistic estimator, e.g., maximum likelihood estimator, is typically applied for the comparisons to the possible key templates where the largest value is said to be the targeted key.

2.3 TPM-AES hybrid cryptosystem

Neural networks can be utilized as an asset to the construction of cryptosystems. An application is found in the context of key exchange and generation. For the sake of keeping data confidential, cryptosystems use either a single shared key or a public-private key to achieve encryption and decryption. By the 1990s, Günther suggested *forward secrecy* unto key agreement protocols [15]. It entails the generation of a unique private key for each session initiated for station-to-station communications. That way, if the secure communication is compromised, only the data from that specific session would be affected. In neural networks, a stochastic pattern of synaptic weights is generated after a learning session is completed which can be used as keys. Reinitializing a neural network creates a new stream of synaptic weights fitted for a cryptosystem that refreshes its keys for every session. The mechanisms to perform a successful neural key exchange protocol rely on the key generation, the key management procedures, and the neural network architecture to achieve both.

A practical scenario where a TPM key generation mechanism may be implemented is with the use of a cryptographic protocol such as the Advanced Encryption Standard (AES). AES is of particular interest because its usage spans many applications. Originally named Rijndael, it is one of the most widely used symmetric algorithms in use today [16]. Some of the most notable users of it are governmental agencies such as the National Institute of Standards and Technology and the National Security Agency. The former classified AES as a US Standard for federal information handling while the latter utilizes AES to encrypt SECRET and TOP SECRET levels of classified information [17, 18]. These factors serve to magnify the importance of AES in modern communications.

2.3.1 *Advanced Encryption Standard*

AES is a symmetric cryptographic protocol structured by means of three processes: encryption, decryption, and key scheduling. The encryption process aims to hide the meaning of the message while the decryption process interprets and recovers the original message with a shared key. The secret key is expanded into *round keys* by means of a KSA which is used in both the encryption and decryption processes. Three modes of operation are available: 128-bit, 192-bit, and 256-bit key sizes. The key sizes only change the amount of resources used by AES and are an option for applications where higher security against a brute force attack is required.

The encryption operation can be described as a series of interconnected modules ordered as: `KeyAddition()`, `SBOX()`, `ShiftRows()`, `MixColumn()`, and `Ciphertext()`. The following lists are a brief description of its operation.

- `KeyAddition()`: A round key is added to the original plaintext by means of a modulo-two addition.
- `SBOX()`: Also known as a substitution box, it has series of generated numbers in a lookup table that are indexed by the resulting values in `KeyAddition()`.
- `ShiftRows()`: After mapping the SBOX output values, the resulting matrix has its rows shifted to the right.
- `MixColumn()`: The resulting matrix is further shuffled by multiplying it to a constant matrix.
- `Ciphertext()`: After all modules loop exhausting the total number of round keys available, an output 16-byte block of ciphertext is generated.

In the decryption operation, an `ISBOX()` (inverse substitution box) module is used instead. It has a different set of generated numbers from the encryption operation's version of the lookup table. The process is done in reverse order from the encryption operation to recover the original plaintext. More information about the internal structure of AES can be found in [16, 19].

2.3.2 *Incorporation of TPM with AES*

A possible implementation of TPM utilizing the AES cryptographic protocol is shown in Fig. 2. To add the properties of a neural key exchange mechanism, the cryptosystem must be able to publicly share information that relates to the message in an unsecure channel without compromising its content. When both machines synchronize with one another, a shared key is generated to aid in the transmission and recovery of an input plaintext. PRNGs utilized upon implementation have the option of being either public or private. The generated inputs $x_{k,n}$ can act as an implicit authentication method if privately generated by both sender and receiving machines [20]. Some authors have suggested that a higher number of input and hidden nodes may improve security against eavesdropping for TPM implementations [21], but for AES, the three modes of operation have limited key sizes. Thus, at a minimum, the learning weights must produce 128-bit keys for usage.

3 Experimental setup and methodology

Two distinct setups are built to implement the TPM-AES cryptosystem. First, a virtual model is created for analysis of possible security-compromising scenarios. Then, an experimental setup is constructed based on low-cost equipment and available measurement instrumentation to evaluate how SCAs can affect its security. The methodology to acquire and pre-process data is explained and

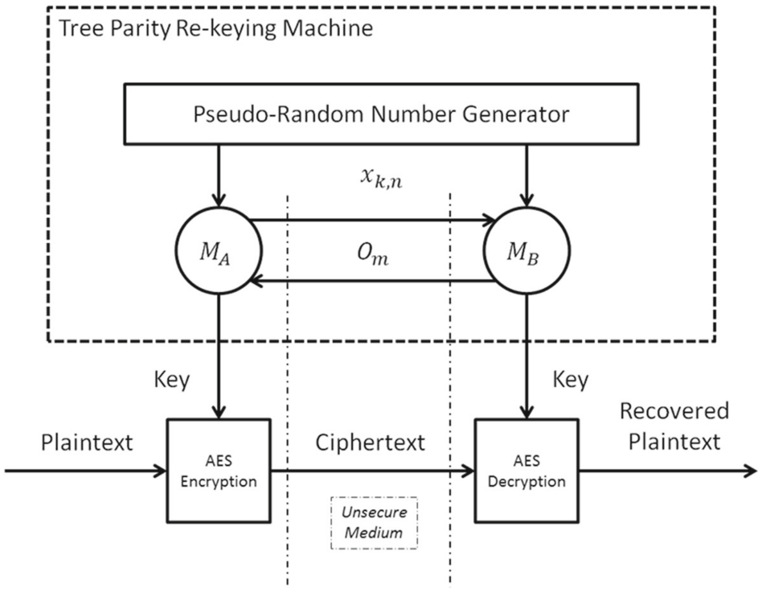

Fig. 2 TPM-AES cryptosystem. An implementation of a TPM and AES hybrid cryptosystem is shown. The PRNG can be generated publicly or privately for each machine

usage of the AVR ATMega328P microcontroller for SCAs is discussed.

3.1 Virtual model security setup

The topology of the TPM and the AES cryptographic protocol is constructed and tested within the MATLAB® environment. The hybrid cryptosystem's interface relies on having the machines in the TPM fully synchronize and utilizing the final weights as input keys to AES. The weights reset their values for each run yielding a fresh set of keys for each session. The user specifies both the amount of hidden nodes and the amount of input nodes desired. For AES, a 16-byte key (128 bits) is required, thus it is fitted accordingly with $K = 4$ and $N = 4$. The bounded integer values for the weights is also user-defined, with an arbitrary number of $L = 4$ utilized for the experiments. The internal inputs to the machines are public vectors generated by MATLAB's PRNG function `rand()`.

The experimental setup aims to test and simulate different scenarios involving the security and the performance of the TPM in MATLAB. Security is evaluated on a case-by-case basis involving an attacker coined as the Eve machine. The Eve machine represents an untrusted source with full knowledge of the cryptosystem and with access to the TPM's output nodes. The performance metric is split into various sets of parameters including execution time, synchronization score, and stability. Figure 3 depicts a system-view flowchart of how each module is interfaced. Initially, the TPM starts and checks whether its machines are not synced. To simulate the security-compromising cases, the Eve machine is implanted to try and sync with the Alice and Bob machines. If the Eve machine is not enabled, the program execution will evaluate performance metrics. Finally, a key is generated and used as input for the AES. The generated key is expanded into round keys and used to create a ciphertext after the encryption process. The original message is recovered after the decryption process.

3.2 Testbed for SCA experiments

Security evaluations of side channels require a measurement station and a testing setup. The measurement station has to provide a means to evaluate multiple side channels as well as capability of logging data to a computing platform for statistical analysis required in most cases of SCAs. Utilizing the *Machine Intelligence Laboratory* at the FAMU-FSU College of Engineering, a testing platform is built to achieve a means for further study and exploration into vulnerabilities of cryptographic devices via side channels.

The main idea of developing an experimental setup for SCAs revolves around a large amount of stored measurements from a leakage source in the target device. From this, a testbed is constructed utilizing the following instrumentation: Tektronix™ DPO4054B oscilloscope, ATmega328P microcontroller, 10 Ω resistor, and a PC station. The resistor is utilized as a measurement point to connect the oscilloscope's probes and read the power consumption of the microcontroller along its GND rail. The microcontroller hosts an implementation of a TPM-AES cryptosystem with added triggers for leakage

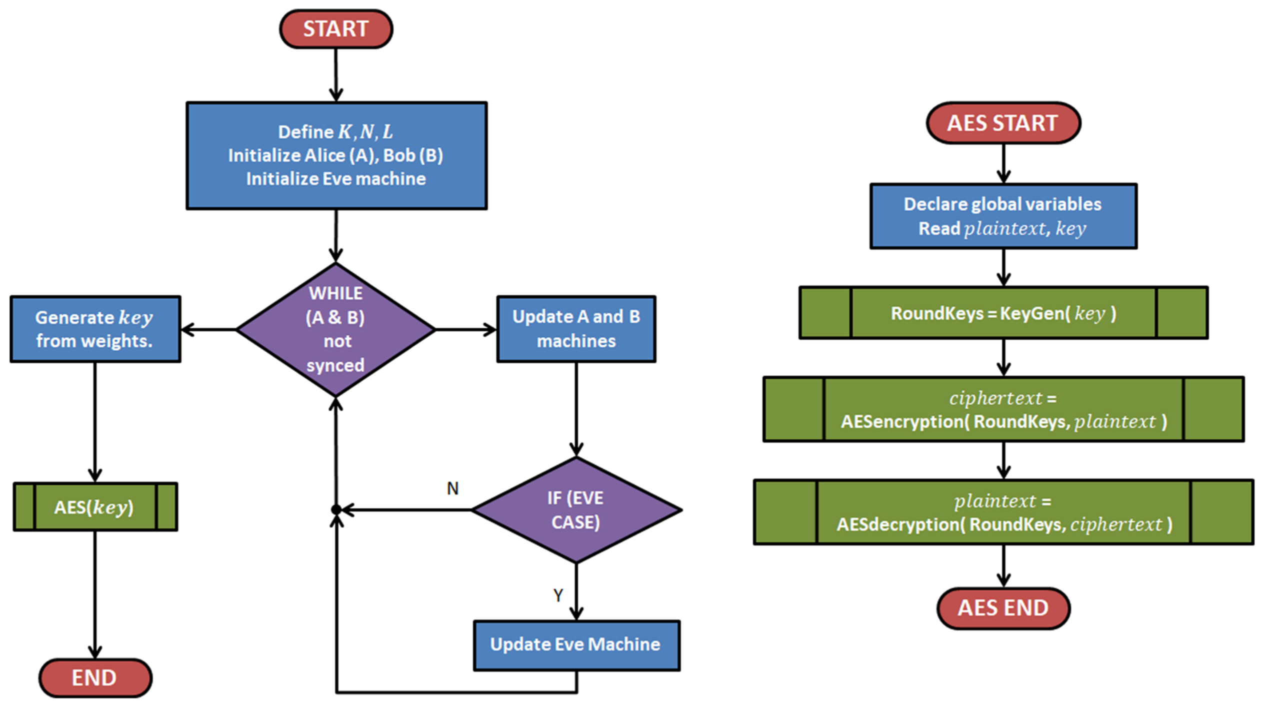

Fig. 3 Virtual model flowchart. The flowchart described how the performance metrics and security evaluation is performed on the cryptosystem

evaluation of certain intermediate operations: memory reads, machine outputs inside the TPM, and key transfers to AES. The triggers are sent to the oscilloscope to capture the time frames of the intermediate functions chosen. This is possible due to the availability of a customizable environment within the Arduino® platform. Since it compiles code via a C/C++ wrapper, implementations can be examined to insert triggers in specific locations that hold special significance. The PC station acts as the master controller of the measuring system to read and store oscilloscope data via a custom interface.

The PC oscilloscope interface is realized via serial communication as shown in Fig. 4. The communication protocol TekVISA [22] is implemented utilizing the Python programming language and integrated development environment (IDE). The written Python code reads and decodes RPB binary formatted output measurements from the oscilloscope and adds data-logging capabilities. The measurements are written to comma-separated value (CSV) file for general usage in statistical software. The oscilloscope's output buffer is cleared before a new measurement takes place to avoid possible overflow flags to be triggered. Additionally, the interface allows programming the microcontroller and tests for interrupted communication. The Arduino Duemilanove™ shield is utilized to provide the voltage supply, crystal oscillator, and programming interface to the microcontroller.

Although external triggers provide a means to capture a specific time frame, it often is not enough to have reliable side-channel traces. A main issue arises with the synchronicity of the PC, oscilloscope, and the microcontroller when extracting data. To combat the problem, the microcontroller is programmed to poll for a message from the PC before a new operation is commenced. The polling allows the Python script to decide when to commence a next round of an encryption operation, continue a function, or halt the system. The microcontroller waits for the PC to extract the measurements from the oscilloscope before being asked to perform another task. To counter noise in measurements, the functions of interest are set to iterate over a period of time. The oscilloscope is configured to average the repeated side-channel measurements. This method provides a robust signal that can be used for statistical analysis and reduces data overhead. Yet, in a practical implementation, devices may not be able to be fully controllable, thus requiring a larger amount of measurements and data to implement SCAs.

For the experiments, the inputs are selected to hold a relationship with a statistical distribution. The input plaintext to a cryptosystem is selected to hold a uniform distribution with respect to the Hamming weight of its integer values. It allows for a realistic scenario where the plaintext can be a set of random values. The symmetric keys used, however, hold a uniform distribution between

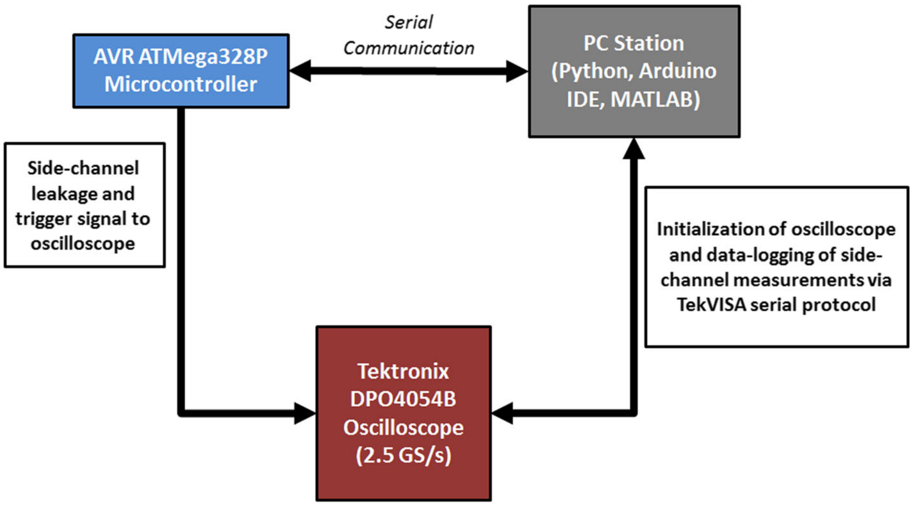
AVR ATMega328P Microcontroller
Serial Communication
PC Station (Python, Arduino IDE, MATLAB)
Side-channel leakage and trigger signal to oscilloscope
Initialization of oscilloscope and data-logging of side-channel measurements via TekVISA serial protocol
Tektronix DPO4054B Oscilloscope (2.5 GS/s)

Fig. 4 Diagram of the experimental setup. PC oscilloscope serial communication interface to extract side channels from target device

its integer values. It is purposely selected to magnify a potential relationship within HW or HD leakage models.

4 Security and performance evaluation of TPM-AES hybrid cryptosystem

A total of three different cases are studied in which the Eve machine tries to fully synchronize with the Alice and Bob machines: case 1 deals with the mutual syncing of the Alice and Bob machine outputs, case 2 focuses on syncing solely with Alice machine's outputs, and case 3 is a direct intercept of the communication channel to sync acting as Alice or Bob. For all three cases, 1000 different iterations of synchronizations run between Alice and Bob using three different learning rules as depicted in Eqs. (3)–(5): Hebbian, anti-Hebbian, and random walk.

The vulnerability of the system is measured by defining a synchronization score s_{Eve} as the percentage of weights that match between the Eve and Alice or Bob machines. Equation 6 depicts the relationship where n^E is the number of matched weights from the total available weights $n^{A|B}$ for a single session key in the TPM. If Eve manages to achieve a perfect synchronization score, the key would be rendered compromised and the cryptosystem unsecure.

$$s_{\text{Eve\%}} = 100 \cdot \frac{n^E_{\text{match}}}{n^{A|B}_{\text{total}}} \tag{6}$$

Histograms of the Eve machine's synchronization score are obtained and plotted in Fig. 5. In the first case, Fig. 5a, Eve applies each of the specified learning rules to update its weights only when all the three machines have the same outputs. The resulting histogram shows that Eve could not synchronize with Alice or Bob. The most frequent synchronization score is tabbed at $s_{\text{Eve}} = 60\%$ meaning that the failure to synchronize is because the training is slower than the Alice and Bob machines. Although for some runs, Eve manage to achieve up to 88% sync score. For the other learning rules, similar behavior is seen with no full synchronization. In the second case, Fig. 5b, Eve focused on training its weights by observing solely Alice's output. The results indicate that full synchronization is achieved in 30 of the 1000 runs using the Hebbian learning rule. Lastly, in Fig. 5c, an active attack is simulated where Eve intercepts the channel and overrides Alice and Bob's output with its own output. The attack achieved full synchronization across all learning methods and runs signifying a notable weakness.

To test the synchronization performance of the TPM, only the Alice and Bob machines are utilized to sync and measured with respect to execution time. The surface plot of Fig. 6a is generated by sweeping through the parameters of hidden nodes K and the number of input nodes N. The synchronization time is seen to increase proportionally to the number of hidden nodes and input nodes. Figure 6b shows that by increasing the number of hidden nodes, the execution time for full synchronization increases when holding a fixed amount of input nodes. Likewise for increasing amounts of input nodes utilized, performance slows down for a fixed amount of hidden nodes. The results imply that larger key lengths will affect performance since weights are distributed with respect to the number of input nodes to the hidden layer.

The synchronization errors between Alice and Bob can be seen by revisiting Fig. 5 and reading the synchronization failures. Table 1 summarizes the error rates with respect to each learning method. Anti-Hebbian learning produces the highest amount of synchronization errors as compared with the other utilized learning methods. The case where Eve intercepts the channel shows a 25.8%

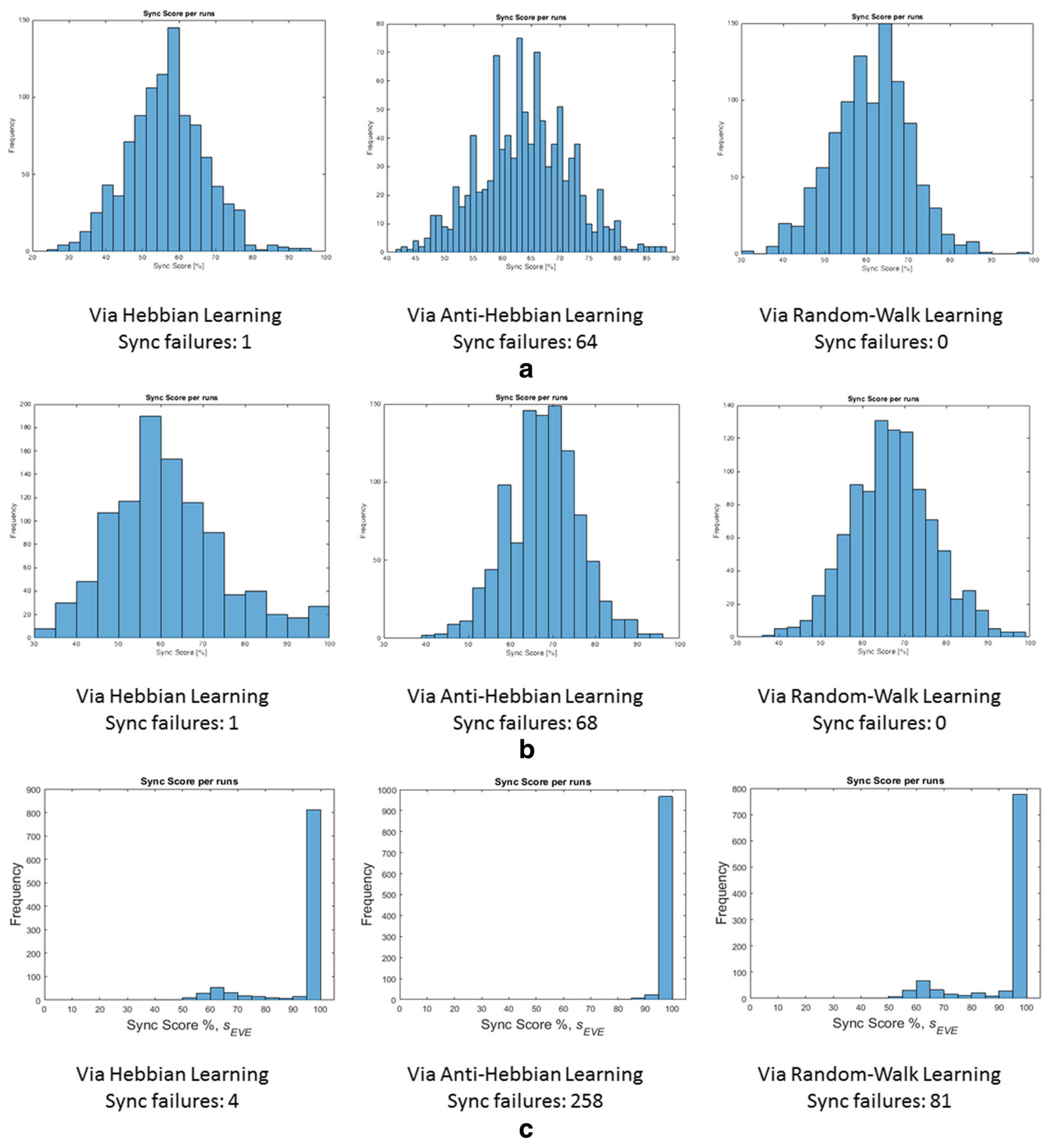

Fig. 5 Security evaluation cases. **a** Case 1: Eve syncs with Alice and Bob. **b** Case 2: Eve syncs only with Alice. **c** Case 3: Eve intercepts channel and acts as Alice or Bob

error rate. The drastic increases in synchronization errors for case 3 can aid in determining whether the channel is being intercepted. The number of errors can be used as a mechanism to detect intrusion or a compromised communication channel since synchronization is more likely achieved in the other cases.

5 Man-in-the-middle power analysis

Taking advantage of the properties of the SPA, DPA, CPA, and TA attack algorithms, a new MIM attack is designed and developed taking into account a target device's side-channel emissions. The attack algorithm is developed as a set of steps and is directed towards uncovering

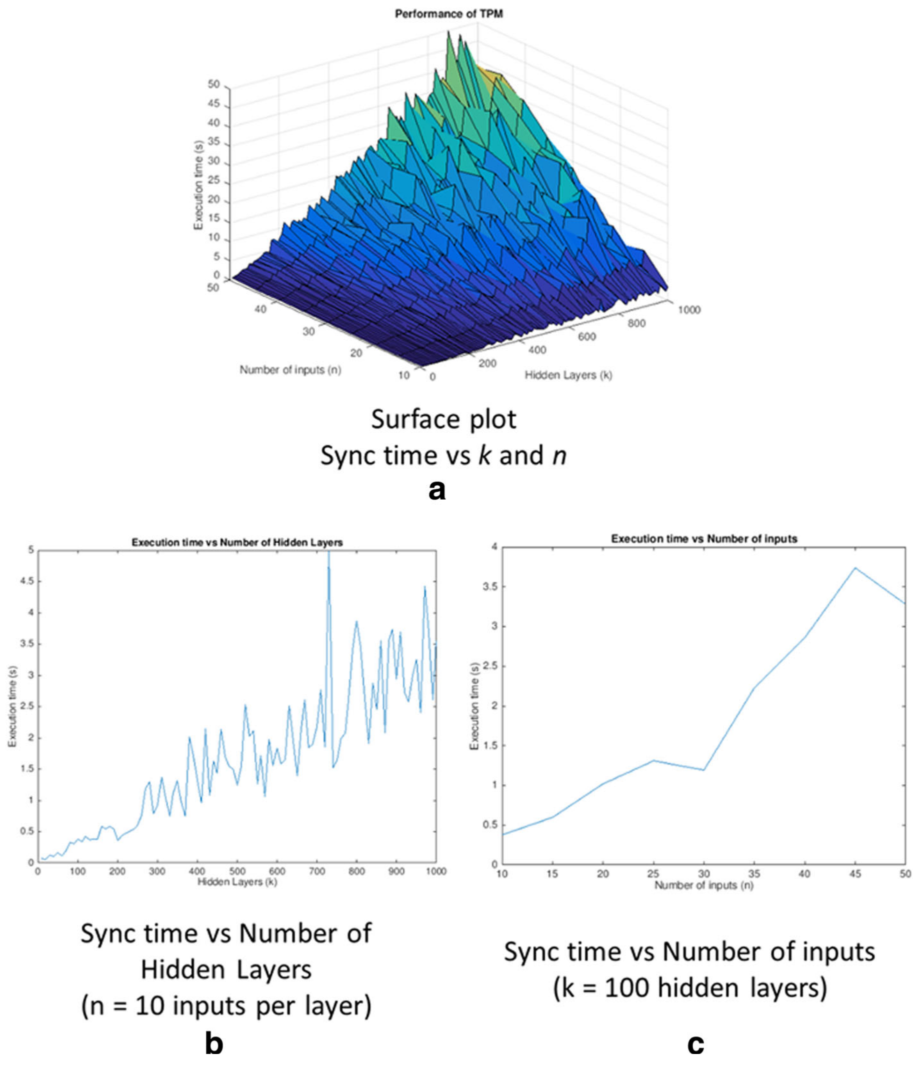

Fig. 6 TPM performance evaluation. **a** Surface plot with varying hidden layers and varying number of inputs. **b** Execution time with varying number of hidden layers and **c** with varying number of inputs

the synchronized synaptic weights, or the final generated keys, of the TPM by observing its machine outputs. The modified MIM attack presented is inherently another form of profiling attack that takes advantage of the power consumption of the target device. Additionally, an identical device is required to perform offline processing of the profiling's results. The attack is coined as man-in-the-middle via power analysis attack (MIMPA) for simplicity.

Table 1 TPM synchronization errors

Learning rules	Error %		
	Case 1	Case 2	Case 3
Hebbian	0.1	0.1	0.4
Anti-Hebbian	6.4	6.8	25.8
Random walk	0	0	8.1

The following subsections elaborate on the attack and analysis of the results.

5.1 MIMPA attack algorithm

MIMPA is developed utilizing a modified version of the DPA, CPA, and TA attack algorithms. The method aims to utilize the power emissions from a cryptographic device utilizing a neural key exchange protocol for the generation of session keys. The TPM is the target cryptosystem as applied to an observable cryptographic device. It is a non-invasive attack and cataloged as a profiling-based attack. Full knowledge of the neural key implementation is required to land a successful extraction of keys. The approach to develop a MIMPA attack is described by the following steps:

- *Actively record traces from a key session*: Power dissipation traces are recorded from an active neural key session. The traces are labeled with respect to the learning steps performed by the neural network. After full synchronization of the machines' respective weights, the recorded traces are stored in a matrix **T** indexed by $t_{i,j}$ as described in Eq. (7). The rows $1 \leq i \leq I$ represent the recorded learning step's power consumption while the columns $1 \leq j \leq J$ represent the number of samples in the measured trace.

$$\mathbf{T} = \begin{bmatrix} t_{1,1} & t_{1,2} & \dots & t_{I,J} \\ t_{2,1} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ t_{I,1} & \dots & \dots & t_{I,J} \end{bmatrix} \tag{7}$$

- *Pick a single point of interest*: A point of interest is chosen from the recorded traces from the results of an applied discriminant. The discriminant test is formulated in Eqs. (8) and (9). The calculated Δ determines the distance between differences in a sample point and its mean value per learning step. A single arbitrary sample is taken as a point of interest (POI) by evaluation of the highest peak in the sum of absolute differences between the traces and mean. The peak indicates the highest distance encountered from samples to the mean and can be an indicator of data-dependent power consumption.

$$\mu_j = \frac{1}{I}\sum_{i=1}^{I} t_{i,j} \tag{8}$$

$$\Delta_j = \sum_{i=1}^{I} \left| t_{i,j} - \mu_j \right| \tag{9}$$

$$POI = \left(j, \operatorname{argmax}\left(\Delta_j\right)\right) \tag{10}$$

- *Characterize the machine outputs*: To characterize the outputs, threshold estimations are created from the average value between the highest and lowest peak from all the traces at the POI as shown in Eq. (11). Afterwards, a conditional operator is utilized to determine the value of O_m and mapped to a matrix **V** of size $1 \times I$ as formulated in Eq. (12). **V** represents the machine outputs from either the Alice of Bob machines. In the case where $t_{i,POI} = T_{th}$, the value is adjusted with respect to the machine output observed. This means that $v_i = 1$ if the traces observed are from the Alice machine and $v_i = -1$ if they are from the Bob machine.

$$T_{th} = \frac{1}{2}\left(\operatorname{argmax}\left(t_{i,POI}\right) + \operatorname{argmin}\left(t_{i,POI}\right)\right) \tag{11}$$

$$\mathbf{V} = \begin{cases} v_i = 1 & \text{if } t_{i,POI} > T_{th} \\ v_i = -1 & \text{if } t_{i,POI} < T_{th} \end{cases} \tag{12}$$

- *Verify validity of mapped machine outputs*: CPA is executed to verify if the mapped machine outputs are consistent with its power dissipation. The hypothetical machine outputs in **V** are used as the intermediate function. A power model of choice (such as the HW power model) is selected to generate a matrix **H** with respect to **V**. Finally, Pearson's correlation coefficient is performed to determine the relationships existing between the generated machine outputs and its traces. Matrix **H** is indexed by $h_{i,k'}$ where its rows are the number of traces gathered $1 \leq i \leq I$ and its columns are the possible keys in a key space of a cryptosystem $0 \leq \text{key} \leq K' - 1$ indexed as $k' = \text{key} + 1$. Equations 13 and (14) help formulate the parameters.

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,K'} \\ h_{2,1} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ h_{I,1} & \dots & \dots & h_{I,K'} \end{bmatrix} \tag{13}$$

$$r_{k',j} = \frac{\sum_{i=1}^{I}\left(\left(h_{i,k'} - \overline{h_{k'}}\right)\cdot\left(t_{i,j} - \overline{t_j}\right)\right)}{\sqrt{\sum_{i=1}^{I}\left(h_{i,k'} - \overline{h_{k'}}\right)^2 \cdot \sum_{i=1}^{I}\left(t_{i,j} - \overline{t_j}\right)^2}} \tag{14}$$

- *Feed outputs to identical device*: After verifying the machine outputs, they are ready to be implemented in a cryptographic device. The testing device must be identical to the original target device. However, the implementation should utilize a single Eve machine that receives the calculated machine outputs in matrix **V**. The internal input nodes $x_{k,n}$ are publicly available in the standard architecture of the TPM, which can be used in conjunction with the machine outputs to emulate the learning steps. Selecting a learning method for the Eve machine is entirely reliant on knowledge of the target device. The Eve machine is trained posing as either the Alice machine or the Bob machine. Its final synaptic weights represent the target key for a single session of the key exchange. It is important to note that the initial weights play a large factor in determining an accurate key.

5.2 MIMPA results and analysis

A MIMPA attack is performed against a TPM implementation in the ATMega328P microcontroller. Figure 7a shows how the absolute difference between the samples in the recorded traces and its mean. The plot indicates that the maximum distance from the mean is seen between samples $840 \leq j \leq 860$ while other local maximums of interest peak at $j = 538$ and $j = 145$. Via this method, the POI chosen is at $j = 845$. Other possible POIs can be selected by the next highest peaks. The samples at the POI for every learning step

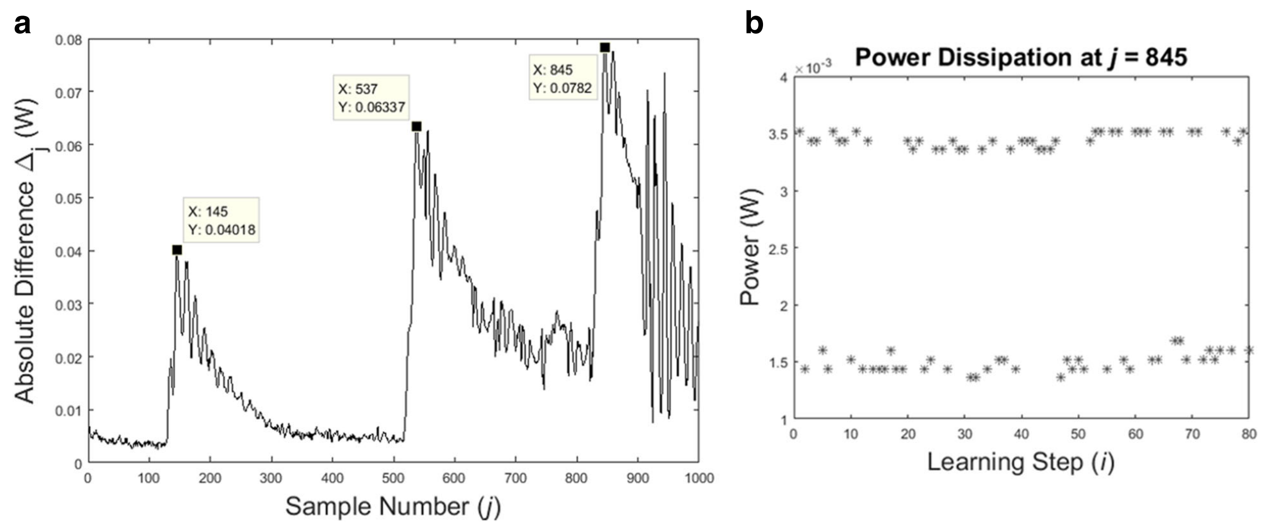

Fig. 7 Classifying POI and O_m. **a** Absolute difference of sample and mean. Another form of a standard deviation calculation. **b** Recorded power at $j = 845$. For each learning step, a noticeable gap exists

are shown in Fig. 7b. It is seen that the power consumption exhibits distant values based on the possible machine output. The dataset essentially holds a linearly separable pattern that can be exploited to classify the actual machine output that happened in a respective learning step.

The distance between power consumption values for a machine's output allows for its mapping. Figure 8 demonstrates the threshold values for the M samples in the N learning steps. At the POI, $T_{\text{th}} = -58\mu\text{W}$. A snippet of the recorded traces adjusted to the mean are also shown. Samples around $730 \leq j \leq 745$ would not be useful as classifiers due to the threshold being too close to the mean. With the T_{th} selected, the machine outputs are mapped.

The extracted machine outputs are tested for validity by using CPA. HW power model is selected to create a hypothetical model with respect to the machine output data. The correlation plots are displayed in Fig. 9a. It indicates the correlation between the samples in the observed traces and the hypothetical power model. The correlation coefficient is $r \geq 0.80$ for $550 \leq j \leq 900$. The high correlation indicates that the power model proportionally matches the hypothetical model. It holds significance in that the machine outputs classified bear a high likelihood on being the actual outputs as produced by the target device. At the POI, the correlation is $r = 0.997$, guaranteeing that the behavior matches the actual use of the classified output values. Figure 9b presents how r varies with respect to the learning steps. It holds that as more learning steps are needed to synchronize the final weights, the more accurate the correlation calculation is. This result also indicates that faster synchronization in the TPM can be beneficial to countering a CPA attack. Yet, if more learning steps are required to synchronize, the resulting correlation calculation can reach a steady state indicating higher accuracy for interpretation.

The machine outputs are fed to an identical TPM implementation and device. A synchronization score is plotted in Fig. 10 outlining how similar are the resulting Eve machine's synaptic weights are with the target machine. The plot highlights the dynamic learning that Eve machine invokes until reaching synchronization. Full synchronization is interpreted as the identical keys utilized by the original session of the target device operating the TPM. Initially, the synchronization sinks from 50% of matched weights to 30%, but as more data is fed, the synchronization scores rapidly increases.

6 Machine learning for TPM key transfer classification utilizing side channels

Neural networks can aid as a tool to create secure cryptosystems as seen in the form of the TPM. They can also be applied to evaluate cryptosystems by utilizing some of their capabilities: distinguishing information, classifying linearly separable data, and information-based prediction. Multiple learning models exist that can be used to expand side-channel cryptanalysis, in particular, profiling-based attacks. In the following subsections, a machine learning model is proposed as an alternative to a template attack targeting a cryptosystem utilizing TPM and parameters for evaluation of the model are discussed.

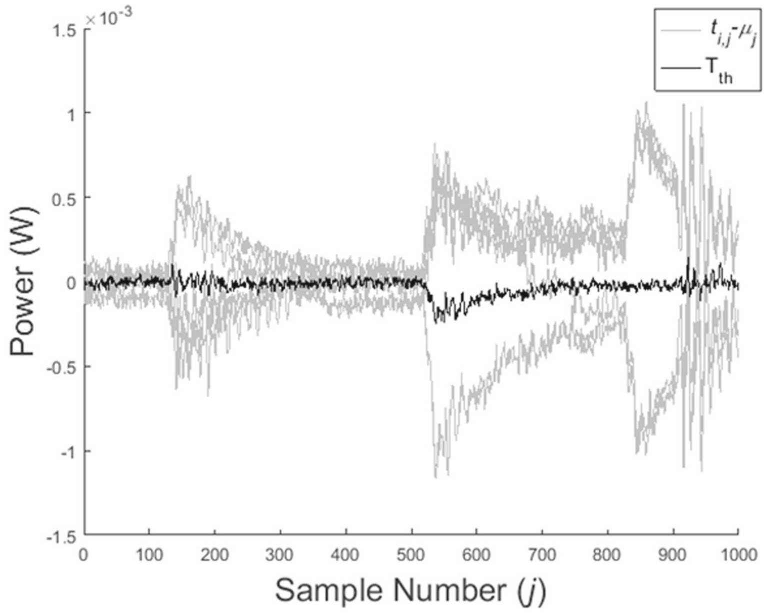

Fig. 8 Threshold values. T_{th} for every sample in recorded traces

6.1 Profiling attack via machine learning model

Execution of a profiling attack with a machine learning model is similar to performing a template attack. Full knowledge of the target device is needed and an additional identical device. The identical device provides the side-channel traces required to train the neural network (NN). Afterwards, features are selected to reduce the dimensionality of the side-channel traces and maximize the NN model's ability to classify patterns. Feature selection is performed via a discriminant test. These range from principal component analysis, sum of absolute differences, correlation coefficient, rank, and among others. This allows for a more narrowed classification process, less errors in the training phase of the NN, and reduction in computational effort.

Training algorithms are utilized to achieve optimum classification, some of these are perceptron learning, Levenberg-Marquardt (LM), scaled conjugate gradient, and gradient-descent algorithms. These algorithms generally differ in computational cost and optimization speed. Overall, LM algorithms produce faster speeds but with higher memory usage. Perceptron learning requires linearly separable data to operate and suffers from being a rather slow algorithm in practice. Gradient-descent

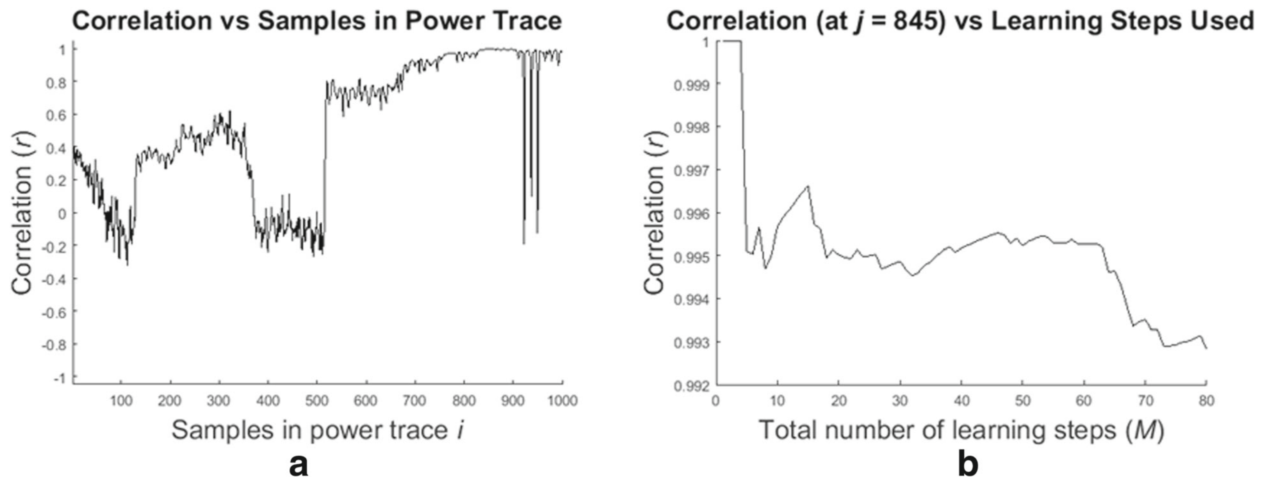

Fig. 9 CPA in MIMPA. Correlation coefficient plots with respect to **a** the samples and **b** the learning steps

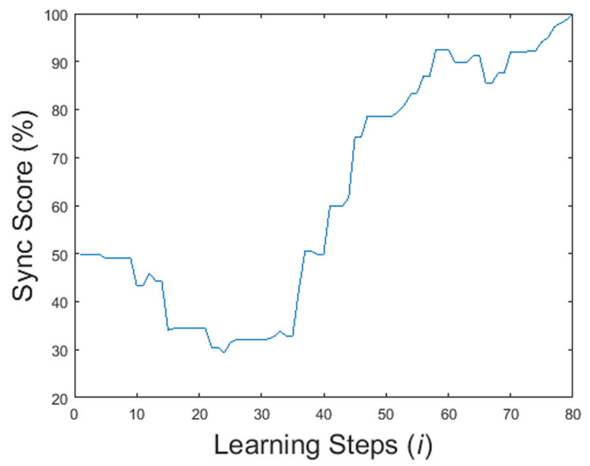

Fig. 10 Final synchronization score. Synchronization score determining matching weights from session key generation

algorithms and scaled conjugate gradient offer a balanced trade-off between computational cost and speed for training neural networks.

After a successful training session, the NN is ready to classify the possible keys. The classification depends on how distinct the power consumption is for each of the keys utilized. For instance, the NN may be trained to correctly classify the Hamming weight of the keys based on the recorded traces. Yet, validation is needed to ensure that the training is accurate. Cross-validation algorithms aid in determining the progression of accuracy of the key classification model and are typically taken from the input data set. Unknown testing data is utilized for the final model before heading for a practical use. It is then where a trace from the actual target device is presented to classify the transferred key.

Some work has been done developing machine learning as prediction and classification tools in SCAs. For instance, acoustic signals from a keyboard are used as the side-channel source to eavesdrop on what the user is typing in [23]. A backpropagation-based neural network was built to receive the frequency spectrum of a keyboard press. Its output equaled to the number of keys present in the studied keyboard. Training the algorithm involved pressing each key individually hundreds of times to give the neural network a baseline into what to determine. They concluded that the shape, make, and force applied to a mechanical keyboard can reveal its content with machine learning techniques. Additionally, [24] explores the utilization of least squares support vector machine (LS-SVM) as their neural network model. The method is found to be affected by the choice of regularization and kernel parameters where tuning is needed for minimization of training errors. The approach utilized simulated power traces of an AES SBOX lookup operation rather than a hardware implementation with physical emissions.

6.2 Approach for classification of TPM keys upon transfer

Similar to other forms of SCAs, side-channel measurements are required from a target device. The experiment utilizes the TPM-AES hybrid and aims to intercept the power signals emanating from the transfer of generated keys unto the AES modules. The intermediate operation of choice is the lookup of the session keys into the AES KSA. The scenario being targeted and implemented uses a range of weights with $L = 2$ in the TPM that generated 16-byte keys for the AES KSA. The steps are summarized below. For the experiments realized, the scaled conjugate gradient method is selected as the training algorithm.

- *Acquire traces from identical target device*: Side-channel traces are gathered from an identical target device to profile the power characteristics and serve as training data. The data needs to reflect the intended intermediate operation. Before undergoing further analysis, SPA can be used to identify data-dependent operations (visual inspection).
- *Build a training and target dataset*: Supervised training algorithms require a training and target dataset for the expected outcomes. The training dataset is generated by selecting *features*, or samples, that exhibit data dependency to the keys. A discriminant test is applied in the form of the variance seen in Eq. (15) to select the features. The target dataset should be a predetermined set of expected session key values. In the TPM, this range is $\omega = \{-L, L\}$. The classifier aims to identify the following properties of the targeted keys: actual values, sign, and Hamming weight.

$$\sigma_j^2 = \frac{1}{n-1} \sum_{i=1}^{M} \left(t_{i,j} - \overline{t_j}\right)^2 \tag{15}$$

- *Build a neural network classifier*: The approach implemented utilizes the MATLAB Neural Network Toolbox™. The training and target data sets are applied to generate a neural network classifier. The training algorithm is selected, and error metrics are evaluated to study the efficiency of classification of keys.
- *Acquire traces from actual target device*: At least one side-channel trace must exist from an actual TPM session key operation of the target device. This trace must be measured and stored for use in the generated neural network. Pruning the target traces is

sometimes required to align the traces with the training data.

- *Input session key traces to neural network*: The session key traces are used as inputs to the trained network. Its outputs determine the classification of the keys. Comparison of the target results and actual results are used to evaluate how accurate the classification is performed.

6.3 Classification results and analysis

The traces acquired are of the form shown in Fig. 11. They indicate a clear distinction in power consumption between samples $170 \leq j \leq 220$. It shows that the key transfer exhibits dependency between the data handled and the power consumed. At $j = 181$, the maximum variance is seen, which is used as a characteristic feature selected for use in training data generation. The targeted properties exhibit differing performance ratios after training.

Applying the neural network to classify TPM session keys yields the results in Table 2. A total of 1000 neural networks are created and trained with the side-channel traces acquired from the microcontroller. The attempted recovery of the exact key being transferred yielded an average of 54% true positive rates (TPR). The classification of Hamming weights in the keys exhibited a higher average TPR of 69%. Finally, a 100% TPR is seen for identification of signed values in the keys. For key transfers that exhibit signed values, the key space can be lessened to identify the session key in use. The machine learning tool can essentially reduce a brute force attack on the key transfers by half if it feeds it the correct sign classification. For the exact key and Hamming weights to be retrieved, other learning methods may need to be employed. The TPR rate is not sufficiently high enough to accurately classify its values. An interesting thing to note is that for all properties, key $= -L = -2$ yields a TPR of 84%. It indicates that there is enough separation to distinguish the number from key $= -1$ on the traces. This also means that for non-negative keys, the power measurements are highly correlated. It is observed in the 98% TPR of non-negative HW classifications.

7 Conclusions

A security evaluation is applied to a TPM-AES hybrid cryptosystem utilizing side-channel cryptanalytics. Two

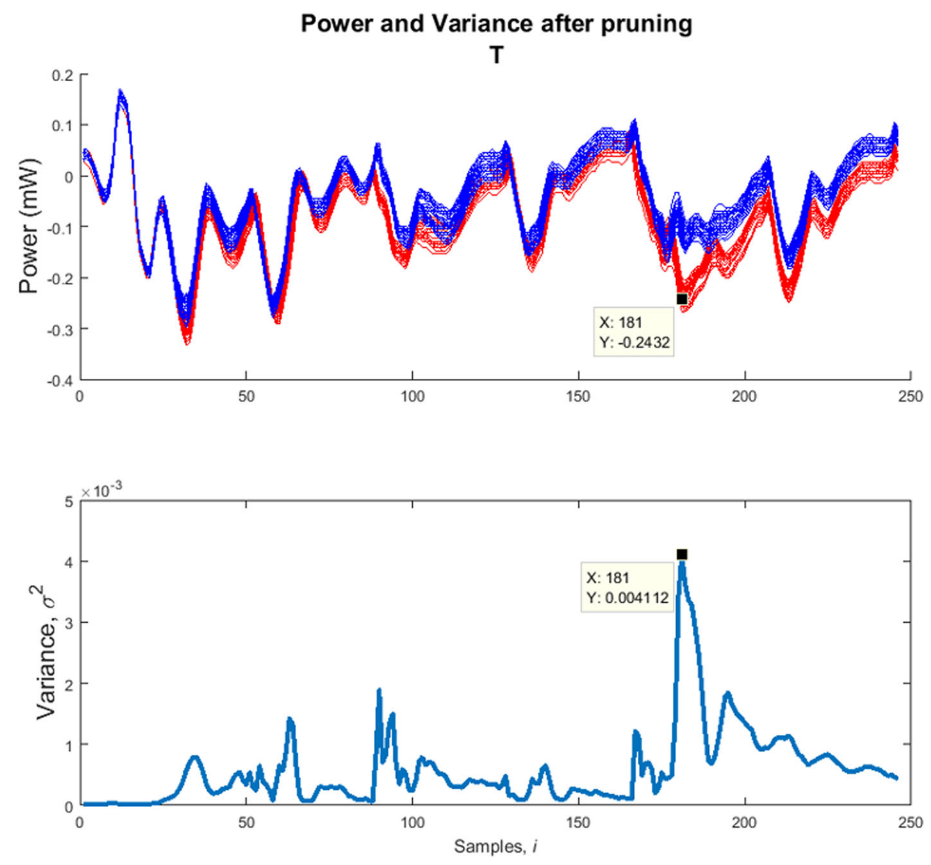

Fig. 11 Power dissipation in key transfer. The power dissipation varies with respect to the key being transferred to the AES KSA

Table 2 Average TPR of session key classification

Key values	Average TPR (%)		
$\{-L, L\}$	Exact key	Sign	Hamming weight
– 2	84	100	85
– 1	10	100	05
0	01	100	01
1	41	100	98
2	41	100	98
Avg. session key prediction (%)	54	100	69

methods are applied for the evaluation: virtual and physical. In the virtual method, it is found that the TPM-AES systems need extra layers of protection to implement due to their susceptibility to traditional MIM attacks. This means that countermeasures must be included for viability of a practical implementation. These can be of the form of digital signature schemes to identify the actual sender and receiver of a machine's outputs. Additionally, the findings indicate that synchronization errors increase when the TPM is directly intercepted. The errors can be used as a form of warning of an attempt to compromise the TPM. It is also found that execution time is higher with growing amount of input nodes and hidden nodes. The limits in a practical implementation arise in the size of the TPM for usage with the cryptographic protocols as well as in total speed of an encryption operation. A higher number of weights correlate to better resistance against a brute force attack, yet the trade-off lies in poor overall performance.

In the physical method, a new attack algorithm is described to evaluate cryptosystems utilizing TPM. The MIMPA is constructed as an alternative SCA where the results show that in optimal conditions, the TPM is vulnerable to have its keys compromised. The results indicate that a potential attacker can recreate session keys offline by creating a hypothetical power model with respect to the mapped machine outputs. The MIMPA possesses countermeasures that can be accounted for. First, faster synchronization of session keys can thwart the attack due to its accuracy depending on the number of traces gathered. Since CPA is utilized as part of the algorithm, it shares its vulnerability of random pre-charge masking and noisy traces.

Another physical method is utilized to evaluate the security of the TPM via a machine learning approach. A supervised training algorithm is used to develop a neural network to classify session keys. The algorithm allows the network to be trained with respect to captured traces from a target cryptographic device. The classifier is created by using a clone device to create the training and target data sets. Several neural networks are created to classify several properties of keys. The classification results indicate that the exact key and Hamming weight TPR need a data set with more features for more accurate representations. Yet the data set had enough features to properly distinguish the signed value of a session key transferred. Thus, running a signed value classifier can reduce the key space necessary to determine the actual session key.

The MIMPA may be further improved to be applied in a real-time scenario where power traces can be scanned and keys immediately identified during a communication session. More tests can be implemented to determine its effectiveness while running a TPM-based cryptosystem that utilizes private input node generation. Countermeasures to general SCAs can also be implemented to test their resilience to MIMPA. Another possible expansion relies on the creation of an ASIC that performs MIMPA to evaluate any TPM implementation.

The machine learning method has room for improvement and exploration as well. Extracting a higher number of features from the measured traces can aid in the classification of the various properties in a key. Other intermediate operations can be selected such that the key dependency is magnified and distinct when creating the training data. Another room for investigation lies in examining the types of operations that emit side channels with distinctions based on the key utilized. Increasing the dimensionality of the features can also aid a training set by utilizing other properties of a trace, e.g., using power measurements along with frequency measurements.

Abbreviations

AES: Advanced Encryption Standard; CPA: Correlation Power Analysis; CSV: Comma-separated value; DPA: Differential power analysis; HW: Hamming weight; HD: Hamming distance; IDE: Integrated development environment; KSA: Key scheduling algorithm; NN: Neural network; MIMPA: Man-in-the-middle power analysis; POI: Point of interest; PRNG: Pseudo-random number generator; RSA: Rivest-Shamir-Adleman; SCA: Side-channel attack; TA: Template attack; TPM: Tree Parity Re-Keying Machine; TPR: True positive results

Acknowledgements

I would like to thank the ECE Department at Florida State University for giving me the opportunity to research this topic.

Funding

Funding is provided in parts by the National GEM Consortium and sponsor GTRI.

Availability of data and materials

The datasets supporting the conclusions of this article are available in the Github repository [25].

Authors' contributions

JMP developed, performed, and implemented all the main portions of the theory and experiments. UMB and SF helped craft ideas and approve the final manuscript to be submitted.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 13 December 2017 Accepted: 20 March 2018
Published online: 13 April 2018

References

1. PW Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
2. I Kanter, W Kinzel, E Kanter, Secure exchange of information by synchronization of neural networks. EPL Europhys. Lett. **57**(1), 141 (2002)
3. G Botella, U Meyer-Baese, A García, M Rodríguez, Quantization analysis and enhancement of a VLSI gradient-based motion estimation architecture. Digit. Signal Proc. **22**(6), 1174–1187 (2012)
4. G Botella, A García, M Rodríguez-Álvarez, E Ros, U Meyer-Baese, MC Molina, Robust bioinspired architecture for optical-flow computation. IEEE Trans. Very Large Scale Integr.(VLSI) Syst. **18**(4), 616–629 (2010)
5. U Meyer-Baese, G Botella, DE Romero, M Kumm, in *Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering X*. Optimization of high speed pipelining in FPGA-based FIR filter design using genetic algorithm, vol. 8401 (SPIE, Maryland, 2012), p. 84010R
6. PC Kocher, in *Annual International Cryptology Conference*. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems (Springer, Berlin, 1996), pp. 104–113
7. J-J Quisquater, D Samyde, in *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security. E-SMART '01*. Electromagnetic analysis (EMA): measures and counter-measures for smart cards (Springer, London, 2001), pp. 200–210
8. A Matthews, Low cost attacks on smart cards: the electromagnetic sidechannel. Next Generation Security Software, 1–15 (2006). Manchester
9. J Friedman, Tempest: a signal problem. NSA Cryptologic Spectr. **35**, 76 (1972)
10. E Brier, M Joye, in *International Workshop on Public Key Cryptography*. Weierstraß elliptic curves and side-channel attacks (Springer, Berlin, 2002), pp. 335–345
11. K Okeya, K Sakurai, ed. by L Batten, J Seberry. On insecurity of the side channel attack countermeasure using addition-subtraction chains under distinguishability between addition and doubling (Springer, Berlin, Heidelberg, 2002), pp. 420–435
12. P Kocher, J Jaffe, B Jun, *Differential power analysis*. (M Wiener, ed.) (Springer, Berlin, Heidelberg, 1999), pp. 388–397
13. E Brier, C Clavier, F Olivier, *Correlation power analysis with a leakage model*. (M Joye, J-J Quisquater, eds.) (Springer, Berlin, Heidelberg, 2004), pp. 16–29
14. S Chari, JR Rao, P Rohatgi, in *International Workshop on Cryptographic Hardware and Embedded Systems*. Template attacks (Springer, Berlin, 2002), pp. 13–28
15. CG Günther, *An identity-based key-exchange protocol*. (J-J Quisquater, J Vandewalle, eds.) (Springer, Berlin, Heidelberg, 1990), pp. 29–37
16. C Paar, J Pelzl, *Understanding cryptography: a textbook for students and practitioners. 1st edn*. (Springer, New York, 2009)
17. NIST, in *Proc. FIPS PUB*. Federal information processing standards publication 197: Advanced Encryption Standard (National Insitute of Standards and Technology, United States, 2001), pp. 46–53
18. NIST, Cnss policy no. 15, fact sheet no. 1: National policy on the use of the advanced encryption standard (AES) to protect national security systems and national security information. Technical report, National Insitute of Standards and Technology, United States (2003)
19. J Daemen, V Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. (Springer, Secaucus, 2013)
20. M Volkmer, Entity authentication and authenticated key exchange with tree parity machines. IACR Cryptol. ePrint Arch. **2006**, 112 (2006)
21. R Mislovaty, Y Perchenok, I Kanter, W Kinzel, Secure key-exchange protocol with an absence of injective functions. Phys. Rev. E. **66**(6), 0661021–0661025 (2002). APS
22. Inc TekVISA, *TekVISA Version 1.1 Programmer Manual*, 1.1 edn. (TekVISA, Beaverton
23. L Zhuang, F Zhou, JD Tygar, in *Proceedings of the 12th ACM Conference on Computer and Communications Security. CCS '05*. Keyboard acoustic emanations revisited ACM, New York, 2005), pp. 373–382
24. G Hospodar, B Gierlichs, E De Mulder, I Verbauwhede, J Vandewalle, Machine learning in side-channel analysis: a first study. J Cryptographic Eng. **1**(4), 293 (2011)
25. J Martínez Padilla, *NN Side channel classifier dataset*. (GitHub, 2017). https://github.com/dbossnirvana/NN_SideChannelClassifier. Accessed 10 Dec 2017