Journal of Information Security and Applications 93 (2025) 104146

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Switch-T: A novel multi-task deep-learning network for cross-device side-channel attack

Jiale Liao [a], Huanyu Wang [b,*], Junnian Wang [a], Yun Tang [a]

[a] School of Physics and Electronic Science, Hunan University of Science and Technology, Hunan, China
[b] School of Computer Science and Engineering, Hunan University of Science and Technology, Hunan, China

ARTICLE INFO

Keywords:
Side-Channel Analysis
Multi-task attack
Transformer
Elastic weight consolidation
AES

ABSTRACT

Side-Channel Analysis has become a realistic threat to cryptographic implementations, particularly with advances in deep-learning techniques. A well-trained neural network can typically make the attack several orders of magnitude more efficient than conventional signal processing approaches. However, like all profiled methods, most existing deep-learning SCAs frameworks require adversaries to develop dedicated models for the specific target device, which complicates the execution of these attacks. In this paper, we propose a Transformer-based neural network, called Switch-T, for multi-task attacks. By collaboratively employing the Elastic Weight Consolidation (EWC) mechanism with a multi-task structure, the model is feasible to learn sensitive data-dependent features of power and EM traces from devices with different core architectures and PCB layout. We experimentally show that the Switch-T model can effectively compromise different implementations of AES. Furthermore, we investigate to which extent the training order of profiling devices can affect the attack efficiency of the model and discuss the impact of hyper-parameter settings in the EWC mechanism.

1. Introduction

Internet of Things (IoTs) is revolutionizing our society by facilitating connections everywhere and anytime. Meanwhile, their extensive deployment also renders them appealing targets for adversaries' malicious activities [1]. Among various threats, Side-Channel Attacks (SCAs) have emerged as a practical and significant one to information security. Compared with traditional cryptanalysts, SCAs aim to bypass the inherent theoretical strength of algorithm designs and extract the sensitive data by analyzing the unintentional physical leakage [2] generated from implementations during the execution of the algorithm. At present, different types of side channels, such as power consumption and electromagnetic (EM) emissions [3], have been successfully exploited to compromise cryptographic implementations [4,5], reverse neural networks' architecture [6], monitor user-device interaction information [7,8], steal the password from the keystroke [9], track the image stream from camera [10] and recover fingerprints from sensors [11].

Over the past decade, significant advancements in deep-learning techniques have considerably boosted SCAs, as neural network can typically exhibit a better capability in learning correlation between sensitive data and corresponding side-channel measurements [12–14]. For instance, in Far Field SCAs, the conventional Template Attack (TA) [15] is able to use 4K traces to recover the secret key of Bluetooth device implementation of AES at 15 m distance in [16]. In contrast, a proficiently trained Convolutional Neural Network (CNN) only requires around 350 traces at 15 m distance under the same attack condition to recover the key [17], which is four orders of magnitude more efficient than TAs. Besides, different types of neural networks have been demonstrated to mitigate the effect caused by various countermeasures against SCAs. For instance, Convolutional Neural Networks (CNNs) are proved to be effective to overcome random-delay clocks and jitter based countermeasures [18].

However, like all profiled SCA methods, Deep-Learning Side-Channel Attacks (DLSCAs) generally require adversaries to create a specific profile (neural network) for a particular type implementation of a cryptographic algorithm [19–21]. Although there are some works which use deep-learning models as tools for the trace segmentation across different victims, the model is not for the attacking scheme. For example, [22] proposes an automated trace segmentation method based on reinforcement learning applicable to a wide range of common implementation of public-key algorithms. They experimentally proved the transferability of the proposed framework on more than 10 datasets. In a general DLSCA scenario, when targeting alternative implementations with different core architecture or attack conditions, it becomes essential to train new neural networks for each modified

* Corresponding author.
E-mail address: huanyu@hnust.edu.cn (H. Wang).

https://doi.org/10.1016/j.jisa.2025.104146

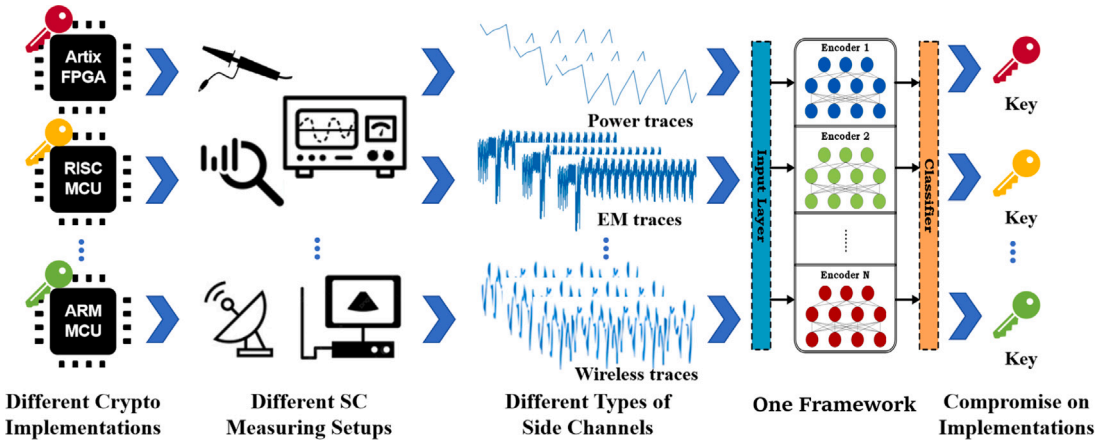


Fig. 1. Illustration of the proposed multi-task DLSCA framework.

attack scenario [23]. This requirement substantially complicates the attack process, as the effort needed to develop and maintain neural networks increases with the number of distinct attack tasks. In practical scenarios, this necessitates substantial resources, both computational power and human expertise, to effectively manage the iterative training and refinement of neural networks for each new target. As such, the flexibility and scalability of DLSCAs are challenged, highlighting a critical trade-off between the adaptability of deep learning models and the operational demands of conducting multiple, varied tasks. Moreover, the dynamic nature of cryptographic implementations and the evolution of countermeasures necessitate updates to attack methodologies. This requires not only multi-task learning capabilities within the neural networks but also a strategic approach to model management and retraining.

In this paper, we propose a Transformer-based multi-task SCA model called Switch-T that employs the Elastic Weight Consolidation (EWC) mechanism to enhance the model's ability to handle multiple tasks simultaneously. This approach allows the Transformer encoder to maintain performance across various cryptographic attack scenarios while mitigating the detrimental effects of catastrophic forgetting. By leveraging the EWC mechanism, our model effectively preserves important learned information from previous tasks when adapting to new tasks, thus ensuring robustness and versatility in dynamic environments. Our contributions can be summarized as follows (see Fig. 1).

1. By introducing a Switch-T model, we present an instance of multi-task deep-learning SCA. We experimentally show that our 3-task model is feasible to effectively compromise three implementations of AES, ATXmega128D4, STM32F3, nRF52832, simultaneously.
2. By leveraging the EWC mechanism, the Switch-T model can overcome catastrophic forgetting and keep memories of features learned from previous tasks, with less than 1% average degradation in classification accuracy for previously learned tasks.
3. We investigate to which extent the training order of different devices and the hyper-parameter settings can affect the attack efficiency of the proposed Switch-T model.

The rest of the paper is organized as follows. Section 2 reviews current states of DLSCAs and introduces the background information related to Transformer and the EWC mechanism. Section 3 illustrates the architecture of our model Switch-T and how does it perform multi-task SCA. Section 4 presents our experimental setup and the corresponding settings to build our Switch-T model. Sections Section 5 shows our datasets of traces collected from experimental devices and how the trace segments for training neural networks are decided. Section 6 summarizes the experimental results and Section 7 concludes the paper.

2. Background

This section begins with an overview of DLSCAs, including a review of the current state of research in this field with a focus on implementations of AES. Afterwards, we introduce relevant background information on Transformer and the EWC mechanism.

2.1. Implementations of AES

The AES [24], standardized by NIST in FIPS 197 and included in ISO/IEC 18033-3, is a symmetric encryption algorithm. It processes a 128-bit plaintext block $\{P\}$ and an n-bit key K, with n being one of the set $\{128, 192, 256\}$, to produce a 128-bit ciphertext block $\{C\}$. This paper focuses on AES with a key size of 128 bits, known as AES-128. AES-128 undergoes 10 rounds of encryption. Each round, except the final one, consists of the following steps: a non-linear substitution step known as SubBytes, a row transposition step known as ShiftRows, a column mixing step known as MixColumns, and a round key addition step known as AddRoundKey, each using a distinct round key RKi, where i ranges from 1 to 10, derived from the original key K. The final round omits the column mixing step.

In practical scenarios, AES implementations can exhibit varied side-channel leakage patterns during operation, influenced by numerous factors, even when running identical firmware versions. For instance, the STM32F3 and ATXmega128D4 microcontroller units (MCUs) may display distinct power consumption signatures while executing the same AES firmware. These differences stem from variations in their core architectures, clock frequencies, instruction sets, and power management capabilities. The STM32F3, with its more sophisticated features, facilitates more efficient and rapid processing, which could result in reduced overall energy expenditure despite occasional higher peak power demands. Conversely, the ATXmega128D4's less complex design leads to lower power consumption per cycle but may incur greater total energy usage due to extended execution times. These discrepancies in the traces' characteristics necessitate that attackers tailor their attack strategies to the specific implementation, potentially requiring different model configurations, trace segment selections, and attack point identifications.

In SCAs, an attack point is an intermediate value within a cryptographic algorithm that characterizes the side-channel measurement. For software-based AES implementations on microcontrollers and microprocessors, the SubBytes step involves replacing an 8-bit symbol using a predefined lookup table, known as the *SBox*. The 8-bit symbol is then retrieved from memory and placed on a data bus, a process that typically demands more power. Consequently, the *SBox* outputs from the initial and final rounds are prominent attack points for software AES implementations. In contrast, for hardware AES implementations

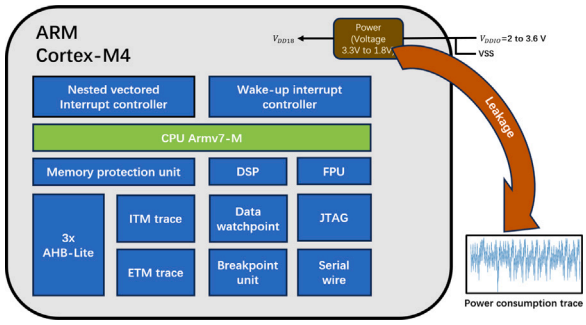


Fig. 2. Power leakage from a 32-bit ARM Cortex-M4 MCU.

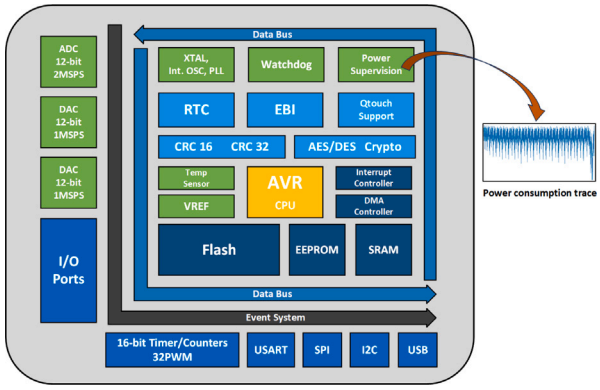


Fig. 3. Power leakage from a 8-bit AVR MCU.

on FPGAs or ASICs [25], where CMOS devices predominantly consume power through switching activity, the XORed result between the last round's input and output is commonly utilized as an attack point.

Thus, these differences significantly complicates the attack process, as the effort required to develop and maintain neural networks increases with the number of distinct attack tasks.

2.2. Deep-learning side-channel attacks

Side-channel attacks are commonly categorized into two categories: *non-profiled* and *profiled*. Non-profiled attacks, such as Differential Power Analysis (DPA) [26] and Correlation Power Analysis (CPA) [27,28], are designed to exploit cryptographic implementations without the need to create a leakage profile beforehand.

Profiled attacks [29] initiate by acquiring a leakage profile that correlates the side-channel measurements with the sensitive data within a cryptographic algorithm. This profile is subsequently employed to execute the attack. Generally, profiled attacks exhibit greater efficiency than non-profiled attacks under the same conditions, with their efficacy potentially being significantly higher in specific cases [17]. However, it is crucial to note that the preparation for profiled attacks is typically more extensive than for non-profiled ones. To comprehend the leakage profile for all possible values of the sensitive intermediate states, adversaries must gain comprehensive access to one or more instances of the target device, known as profiling devices, to collect a vast array of side-channel traces and corresponding data, which are essential for building the leakage profile.

With a few exceptions [30], in majority cases, deep learning techniques are commonly utilized for SCAs in profiled settings [20,31,32]. Typically, deep-learning SCAs involve the following two steps:

Profiling stage. During this phase, it is typically presupposed that adversaries possess complete control over at least one profiling device, which is analogous to the target device and operates the same version of AES-128. Consequently, the attacker is able to gather a multitude of side-channel traces and compile corresponding data from the profiling device(s), including known plaintexts and keys.

We denote a collection of traces for in-depth analysis as $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_{|\mathcal{T}|}\}$, with each $\mathcal{T}_i \in \mathbb{R}^m$ and $i \in \{1, \ldots, |\mathcal{T}|\}$. Each trace represents the full or partial execution of the AES algorithm. For each trace $\{\mathcal{T}\}_i$, the associated plaintext $\mathcal{P}_i$, secret key $\mathcal{K}_i$, and ciphertext $\mathcal{C}_i$ are elements of $\{0,1\}^{128}$. The notation $\mathcal{P}_{i,j}$ and $\mathcal{C}_{i,j}$ is used to signify the jth byte of the 16-byte plaintext $\mathcal{P}_i$ and ciphertext $\mathcal{C}_i$, respectively, where $j \in \{0, 1, \ldots, 15\}$. Thereafter, the deep-learning model $\mathcal{M}_j$ targeted for the jth subkey is trained to discern the leakage profile that correlates the traces with the jth key-dependent labels. These labels are derived based on the selected attack point and the leakage model, as detailed subsequently.

Attack stage. In this stage, We assume that an adversary is capable of capturing a limited set of side-channel traces from the target device and has access to some known data $\hat{\mathcal{X}}_i$. The traces collected from the victim device are then categorized using the trained deep-learning model $\mathcal{M}_j$. Subsequently, the attacker employs the known data alongside the model's classification outcomes to infer the confidential subkey $\mathcal{K}_j$.

However, as with all profiled SCA methods, adversaries commonly develop specialized neural networks to target specific cryptographic implementations [12,32]. This means that the adversary needs to train and manage various unique models to confront different cryptographic systems, significantly increasing the complexity of the attack process.

2.3. Transformer

Transformers are deep neural networks with an Encoder-Decoder structure, characterized by their self-attention mechanism that enables them to process and capture correlations in sequences of data without relying on recurrent structures [33]. The output of the encoder is linked to the decoder by interactive attention.

With wide applications in Natural Language Processing (NLP), Transformers have been gradually applied to classification tasks. When used in classification, only the encoder part of Transformers is needed, such as BERT [34] model for text classification and ViT [35] model for image classification. In this paper, we desgin a Transformers-based model to perform SCA, which is essentially used for classification tasks, and only the encoder part is used. Therefore, this section only introduces the Transformer encoder.

Transformer encoder generally consists of multiple encoder layers. Each encoder layer contains a self-attention layer and a Feed-Forward Network layer.

2.3.1. Self-attention mechanism

Self-attention mechanism is utilized in Transformers. In a self-attention layer, its input and output are a sequence. Given the input sequence $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, the self-attention mechanism computes the output sequence $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T \in \mathbb{R}^{n \times d_v}$ as follows:

$$\mathbf{y}_i = \sum_{j=1}^{n} softmax(\frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d}})\mathbf{v}_j \tag{1}$$

where $\mathbf{q}_i = W_q \mathbf{x}_i$, $\mathbf{k}_i = W_k \mathbf{x}_i$, $\mathbf{v}_i = W_v \mathbf{x}_i$, for $i = 1, \ldots, n$, W_q, $W_k \in \mathbb{R}^{d_k \times d}$ and $W_v \in \mathbb{R}^{d_v \times d}$. The output of self-attention layer can be finally expressed as $Y' = YW_{out}^T + X$ where $W_{out} \in \mathbb{R}^{d \times d_v}$ projects the d_v-dimensional y_js back into d-dimensional vector space and X is the residual connected to the output sequence. The matrices W_q, W_k, W_v and W_{out} are the weight parameters which can be learned during the training. The two hyper-parameters d_k and d_v are the Key matrix $\mathbf{k}_i$ and Value matrix $\mathbf{v}_i$ dimension respectively. d is the dimension of matrix $\mathbf{q}_i^T \mathbf{k}_j$. The item $softmax(\frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d}})$ can be regarded as the attention that output vector $\mathbf{y}_i$ pays to the input vector $\mathbf{x}_i$ [36]. The self-attention mechanism plays a key role in the Transformer, which gives the model the ability to capture the global information in a sequence. Given two input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ that have interrelations between each other, the output $\mathbf{y}_i' = \mathbf{y}_i W_{out}^T + \mathbf{x}_i$ depends on both $\mathbf{x}_i$ and $\mathbf{x}_j$.

2.3.2. Multi-head self-attention mechanism

The multi-head self-attention mechanism can be regarded as multi-output channels simulating convolution with multi-heads, for which can be used to focus on diverse correlation. In an h-head self-attention layer, given the input sequence $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times d}$, the output $Y^{(l)} \in \mathbb{R}^{n\times d_v}$ of each single-head self-attention layer can be computed as:

$$Y^{(l)} = \sum_{i=1}^{n} softmax(\frac{\mathbf{q}_{(l)}^T \mathbf{k}_i^{(l)}}{\sqrt{d}})\mathbf{v}_i^{(l)} \tag{2}$$

where $\mathbf{q}_{(l)} = W_q^{(l)} X$, $\mathbf{k}_i^{(l)} = W_k^{(l)}\mathbf{x}_i$, $\mathbf{v}_i^{(l)} = W_v^{(l)}\mathbf{x}_i$, and $l = 1, \ldots, h$. Thus, the output Y_h' of h-head self-attention layer is performed as:

$$\begin{aligned} &Y_h = concat(Y^{(1)}, \ldots, Y^{(h)}), \\ &Y_h' = Y_h W_{out}^T + X \end{aligned} \tag{3}$$

where, $concat(Y^{(1)}, \ldots, Y^{(h)})$ is the row-rise concatenation of the matrices $Y^{(1)}, \ldots, Y^{(h)}$, $Y_h \in \mathbb{R}^{n\times hd_v}$ and $W_{out} \in \mathbb{R}^{d\times hd_v}$ [37]. We make an example of computing process of two-head self-attention mechanism.

2.3.3. Feed-forward networks

Feed-Forward Networks (FFNs) are a kind of Multi-Layer Perception (MLP) networks with fully-connection layers in Transformers. FFNs are applied to each position separately and identically after a multi-head self-attention layer. The activation function $ReLU$ is applied between the two MLPs, enhancing the non-linearity and complexity of the classification function. The FFNs perform as shown in Eq. (4) :

$$X_B = FFN(X_A) = F_2(ReLU(F_1(X_A))) \tag{4}$$

where X_A is the input sequence to first layer in FFNs and X_B is the output sequence of FFNs, F_1 and F_2 are two perception layers in FFNs, which can be expressed as a form like $WX + b$, where layer F_1 is used to increase the dimension of X_A and layer F_2 is used to drop the dimension to original dimension. The multi-head self-attention layer and the FFNs layer are alternately utilized, with the original structure being repeated six times [33]. This design allows the model to effectively capture complex dependencies within the data while maintaining a rich representation of features.

2.3.4. Residual connection and layer normalization

The encoder utilizes residual connections after each sub-layer. In Transformers, residual connections are followed by layer normalization [38], both are performed for 2 times in one Transformer encoder. The first connection and normalization are performed after self-attention layer computing, which is depicted in Eq. (5) :

$$X_A = LayerNorm(Attention_h(X) + X) \tag{5}$$

where X is the input sequence to self-attention layer. The second connection and normalization then performed after FFN performance, which is computed as:

$$X_{out} = LayerNorm(F_2(ReLU(F_1(X_A))) + X_A) \tag{6}$$

where $X_A \in \mathbb{R}^{n\times d}$ is the output of the first layer normalization and $X_B \in \mathbb{R}^{n\times d}$ is the output of the FFN.

2.4. Elastic weight consolidation

EWC [39] is a mechanism used in deep learning to overcome catastrophic forgetting by selectively constraining important parameters of a neural network when learning new tasks. EWC introduces a penalty term to the loss function, which constrains the updates of specific parameters deemed important for previously learned tasks. The importance of each weight is quantified using the Fisher Information Matrix (FIM), a measure of the sensitivity of the loss function to changes in the parameters. Given a model M_n that has been optimized on dataset $D_{1:n}$, EWC seeks to train a new model M_{n+1} while maintaining performance on previous n tasks. The loss function in EWC can be expressed as:

$$L_{EWC} = L_{new} + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta_i^*)^2 \tag{7}$$

where L_{new} denotes the loss associated with the current task, and λ represents a hyper-parameter that balances the contributions of the loss for the new task and the preservation of knowledge from previous tasks. The model parameters are denoted by θ_i, where θ_i^* corresponds to the optimized parameters for prior tasks. The term $\theta_i - \theta_i^*$ quantifies the deviation of the ith parameter from its optimal value for previous tasks. Additionally, F_i represents the FIM [40] associated with the ith parameter, which evaluates the sensitivity of the parameter with respect to the data likelihood. The Fisher information for each weight is estimated as the expected value of the squared gradients:

$$F_i = \mathbb{E}_{x\sim D}[(\nabla_{\theta_i}\ log\ p(y|x,\theta))^2] \tag{8}$$

where D is the dataset from the previous tasks, $p(y|x,\theta)$ is the model's prediction probability given the input x and parameters θ. In EWC, only the diagonal elements of the FIM are used, which correspond to the variance of the parameters. These diagonal elements are estimated as follows:

$$F_i \approx \frac{1}{N}\sum_{j=1}^{N}(\nabla_{\theta_i}\ log\ p(y_j|x_j,\theta_i^*))^2 \tag{9}$$

where N is the number of data in the dataset, y_j is the target output (label) of the jth data, x_j is the input features of the jth data, $\nabla_{\theta_i}\ log\ p(y_j|x_j,\theta_i^*)$ is the gradient of the log-likelihood with respect to the ith parameter for the jth data, evaluated at the optimized parameters θ_i^*. As shown in Fig. 4, by penalizing deviations from θ_i^* in proportion to F_i, EWC preserves critical parameters while allowing non-critical parameters to adapt to the new task.

3. Switch-T model

In this section, we start with an overview of the architecture of our model Switch-T, shown in Fig. 5, including its preprocessing components for the input traces. Afterwards, we introduce how this architecture with EWC mechanism prevent catastrophic forgetting [41]. Moreover, we describe the model's working process with traces captured from diverse devices with different CPU core architectures.

3.1. Switch-T architecture

The architecture of Switch-T model consists of three parts: dimensional expansion layer, Transformer encoder vector layer and classification layer. The dimensional expansion layer is the model's input layer. It make the dimension of input data elevate to a higher dimension. This elevation in dimensionality facilitates the effective capture and representation of complex patterns and relationships within the data. It aids in enhancing the model's ability to grasp complex features. This layer transformation can be represented as $Z = XW + b$, where $X \in \mathbb{R}^{n\times d_{in}}$ is the input data, n is the number of data points, $Z \in \mathbb{R}^{n\times d_{out}}$ is output data after dimension enhancing, $W \in \mathbb{R}^{d_{in}\times d_{out}}$ is weight matrix, $b \in \mathbb{R}^{1\times d_{out}}$ is the bias vector.

The Transformer encoder vector layer is constructed based on a multi-task structure [42], which consists of multiple parallel Transformer encoders used for its specific task. Therefore, each Transformer encoder can only handle the specific architecture traces according to the task ID that we have labeled for related traces from different devices. The task ID is an unique identifier that guides the Transformer encoder vector layer in selecting the appropriate Transformer encoder for a given task. As presented in Section 2.3, self-attention mechanism dynamically selects and emphasizes certain parts of the input data over others. Multi-head attention enhances this mechanism by applying the

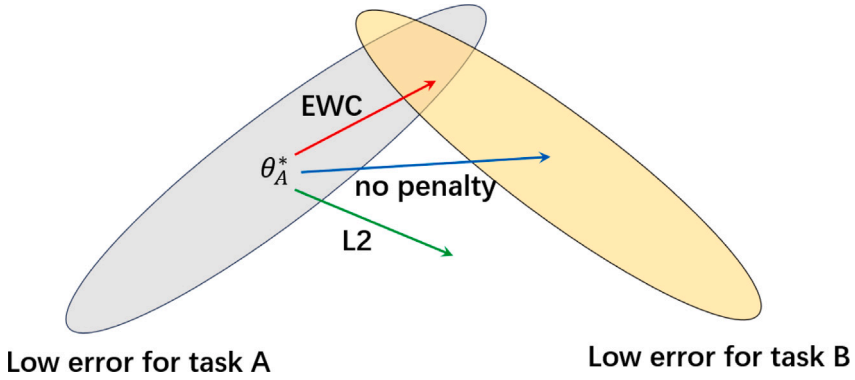


Fig. 4. EWC maintains knowledge of task A during training on task B. Post-task-A training, parameters stabilize at θ_A^*. Solely optimizing for task B (blue arrow) degrades task-A knowledge. Uniform constraints to preserve task A (green arrow) hinder task-B learning. EWC balances this by assessing weight importance for task A, allowing adaptation to task B (red arrow) while retaining task-A performance.

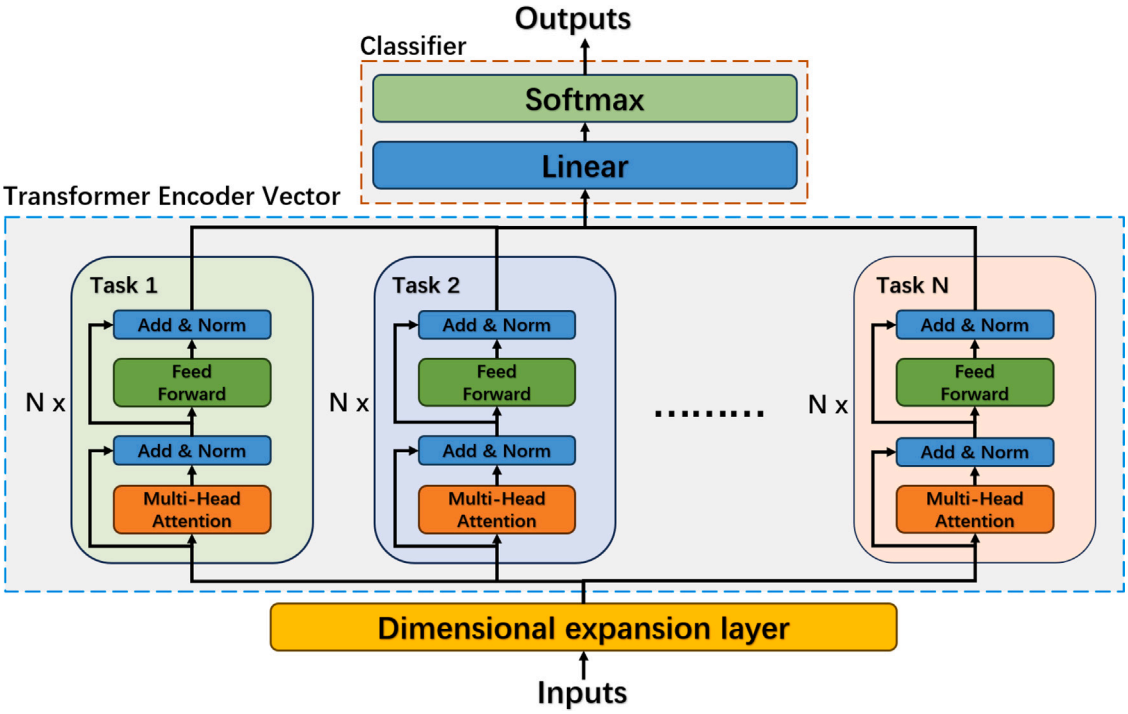


Fig. 5. The illustration of the proposed Switch-T framework.

attention computation multiple times in parallel, with each "head" learning a potentially different aspect of the data. The outputs of these multiple attention processes are then combined and processed further. These multiple operations allow the model to handle complex relationships in the data, including long-range dependencies and various data patterns that are missed when using a single head attention mechanism.

The classification layer is the output layer of the model. We design this layer to generate class predictions by combining a linear transformation and a softmax activation function. The linear transformation maps the high-dimensional output of the Transformer encoder to a vector with dimensions equal to the number of target classes and produces raw scores known as logits. These logits are then passed through the softmax function, which converts them into a probability distribution by normalizing the exponential values of the scores. Each element in the resulting vector represents the probability of the input belonging to a specific class. The class with the highest probability is selected as the model's prediction.

3.2. Multi-task learning with EWC

As described in Section 2.4, deep-learning models often rapidly lose their performance on the old task after being trained on a new task due to the network's parameters being adapted to optimize the new task, which is known as the catastrophic forgetting. To overcome this challenge, we utilize a multi-task structure design with the EWC mechanism. In generous Transformer-based classification model, most parameters are in the Transformer encoder. Therefore, we design a multi-task Transformer encoder structure, in which parameters of each Transformer encoder can only be adapted when learning on its specific-task traces. This structure separates the set of parameters across tasks, which avoids that the optimization process changes parameters for previous tasks when learning a new task and prevents the model from forgetting the old tasks.

Although the multi-task structure above can reduce the disturbance of optimization process to the parameters for previous tasks, there are still two public layers' parameters can be adapted in the new

task learning, which is dimensional expansion layer and output layer. Afterwards, we add EWC mechanism in our model to help preserve the two public layers' performance on old tasks. Let θ be the set of whole parameters in the model. For each task t, we initially train the model to minimize the task-specific loss function $L_t(\theta)$ over the dataset D_t. After training on D_t. We compute the FIM F_s for the public parameters θ_s to quantify their importance for task t:

$$F_s \approx \frac{1}{N_t}\sum_{n=1}^{N_t}(\nabla_{\theta_s}\ log\ p(y_n|x_n,\theta^*_{t,s}))^2 \tag{10}$$

where $\theta^*_{t,s}$ is the sth optimized parameters for task t, and N_t is the number of data in D_t. Then we train the model on a new task dataset D_{t+1}. We define the total loss function $L_{total}(\theta)$ as the sum of the new task loss and a penalty term that discourages changes to the parameters that are important for the previous tasks:

$$L_{total}(\theta) = L_{t+1}(\theta) + \frac{\lambda}{2}\sum_s F_s(\theta_s - \theta^*_{t,s})^2 \tag{11}$$

where λ is a hyper-parameter that controls the strength of the penalty, and $\sum_s F_s(\theta_s - \theta^*_{t,s})^2$ is the penalty term that penalizes deviations from the optimal parameters $\theta^*_{t,s}$ learned for the previous task t. At last, the model parameters θ_{t+1} are updated by minimizing the total loss function $L_{total}(\theta)$:

$$\theta_{t+1} = arg\ min_\theta L_{total}(\theta) \tag{12}$$

The EWC mechanism helps in preserving the performance on old tasks by penalizing the updates to the public parameters θ_s that significantly deviate from their previous task's optimal parameters $\theta^*_{t,s}$. The specific process of multi-task learning with EWC is described in Algorithm 1. We train the model on traces from multiple devices to generalize well to any target device in the cross-device SCA scenario.

Algorithm 1 Multi-Task Learning with EWC

Require: Task sequence $\{T_1, T_2, \ldots, T_n\}$, model $\mathcal{M}$ with parameters θ, regularization hyper-parameter λ_{EWC}
Ensure: Trained model $\mathcal{M}$ preserving performance on all tasks
  Initialize model $\mathcal{M}$ with parameters θ
  Initialize FIM $\mathcal{F} \leftarrow 0$, previous parameters $\theta^*_{\text{prev}} \leftarrow \emptyset$
  for each task T_k in $\{T_1, T_2, \ldots, T_n\}$ **do**
    Train $\mathcal{M}$ on task T_k using loss: $\mathcal{L}_{\text{task}}$
    if $k > 2$ **then**
      previous parameters $\theta^*_{\text{prev}} \leftarrow \theta_{T_{k-1}}$
      Compute Fisher information $\mathcal{F}_s$:

$$\mathcal{F}_s = \mathbb{E}_{x\sim T_k}[(\nabla_{\theta_s}\ log\ p(y|x,\theta^*_{\text{prev},s}))^2]$$

      Add EWC regularization term:

$$\mathcal{L}_{\text{EWC}} = \frac{1}{2}\lambda_{\text{EWC}}\cdot\sum_s \mathcal{F}_s(\theta_s - \theta^*_{\text{prev},s})^2$$

      Total loss: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{EWC}}$
    else
      Total loss: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}}$
    end if
    Update model parameters θ using $\mathcal{L}_{\text{total}}$
    Save current parameters: $\theta_{T_k} \leftarrow \theta$
  end for
  return Trained model $\mathcal{M}$

3.3. Benefits of Switch-T

Compared with conventional methods, like training N separate independent models for a specific device, Switch-T has benefits in cross-device SCAs scenarios as follows.

Table 1
Comparison between different methods in cross-device SCAs.

Method	Cross architecture	Cross channel	Overcome forget
X-DeepSCA [43]	No	No	No
AL-PA [44]	No	No	No
MTL-SCA [45]	Yes	Yes	No
Switch-T (Ours)	**Yes**	**Yes**	**Yes**

First, Switch-T's multi-task structure enables the model to share some common layer parameters (such as the dimensional expansion layer and the classification layer). Therefore, this can reduce the overall model parameters and enhance the training and inference efficiency, resulting in a reduction of computational resource consumption. However, training N separate models requires independent training and maintenance of complete model parameters for each model, which leads to a linear increase in computing resources and storage requirements. As the number of devices increases, the efficiency of training N separate TNs gets lower.

Second, Switch-T can transfer knowledge among different tasks. It utilizes the features and patterns learned from previous devices to assist in the learning of new devices' attack tasks. For instance, when attacking devices of different architectures, Switch-T can leverage the learned general side-channel feature patterns to adapt more quickly to the attack tasks of new devices, thereby enhancing the efficiency of the attacks. As for training N separate models, the various models are independent of each other and cannot share knowledge. As a result, each model has to learn the specific attack features of the device from scratch, resulting in low learning efficiency and weak generalization ability.

Third, as a unified multi-task model, Switch-T can respond promptly to attack requests from various devices during inference without frequently switching models or loading different model files. This approach enhances the real-time performance of attacks. If we design a script to call separate N trained models, we need to switch among multiple independent models, which leads to an increase in inference latency in real-time attack scenarios.

Fourth, Switch-T simplifies the model management process by integrating multiple tasks into a unified framework. This design not only reduces the management complexity caused by model fragmentation, but also makes model updates and maintenance more efficient. When attacking new devices, the remaining task modules can be used in the existing framework for training, as long as the preset number of Transformer encoders has not reached the upper limit . This flexibility enables Switch-T to handle multiple devices without redeveloping models for each device. Moreover, the unified framework facilitates the centralized optimization and update of models, reducing the maintenance risks caused by numerous models. However, training N separate models requires training and managing a separate model for each device, which leads to a linear increase in the complexity of model management as the number of devices increases. When a new device is added, a completely new model needs to be developed and maintained, which not only increases the workload but also cause confusion in model management.

As shown in Table 1, compared with other conventional methods like X-DeepSCA [43] and AL-PA [44], Switch-T can perform better in the cross-architecture (ARM vs. AVR) and cross-channel (Far Field EM vs. power) SCAs scenarios. Besides, Switch-T has great performance in overcoming catastrophic forgetting in the cross-device SCAs, which is not considered in MTL-SCA [45].

3.4. Profiling stage

As presented in Fig. 6, the Switch-T model consists of the following three process. (1) **Dimension unification**, in which the lengths of different device's traces intercepted by their PoIs are tantamount to

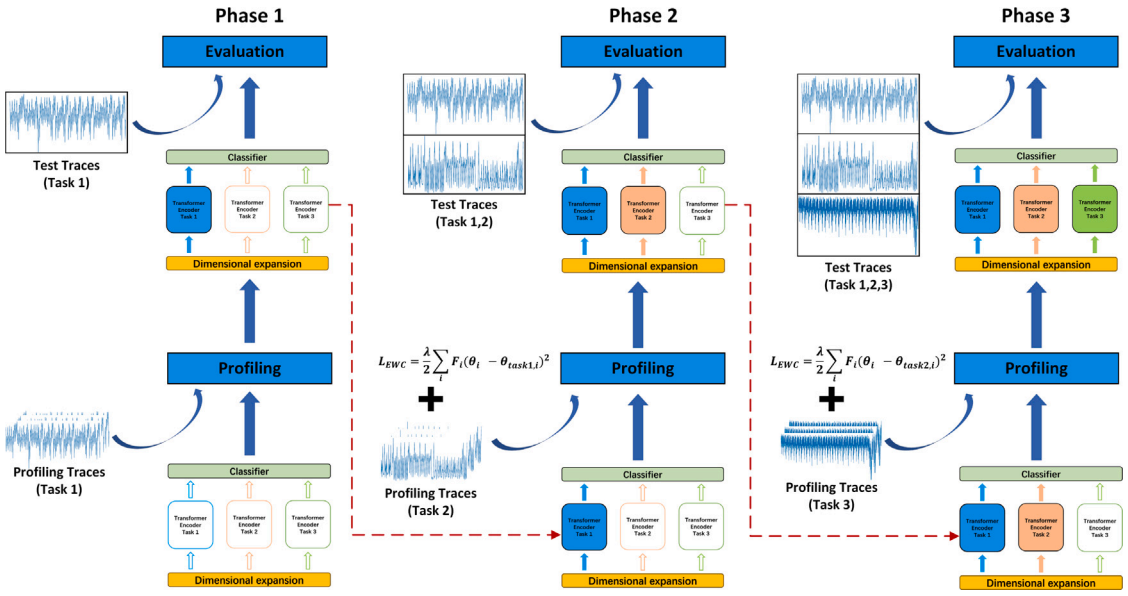


Fig. 6. Illustration of how the proposed Switch-T model can be trained to perform a cross-device attack.

the model's input dimension. (2) **Transformer encoder vector initialization**, in which the number of Transformer encoders for specific task is initialed parallelly according to the number of tasks, (3) **Model training**, in which the deep-learning model is trained task by task. The specific process steps are represented as follows.

Due to inherent differences in the physical structures of devices with varying architectures, the lengths of intercepted traces are inherently inconsistent. To address this challenge, the first step involves unifying the trace lengths to match the model's input dimension. Principal Component Analysis (PCA) is employed to standardize the input dimensions, thereby facilitating seamless integration of data from disparate devices. This preprocessing step is essential for maintaining consistency and ensuring that the model can process traces from different architectures without encountering dimensional mismatches.

Following dimension unification, the model initializes the Transformer encoder vector layer. This layer comprises a vector of Transformer encoders, each dedicated to a specific task. The model's architecture is static, meaning the number of Transformer encoders does not automatically increase with the addition of new tasks. Therefore, a preprocessing step is required to set the number of Transformer encoders based on the number of tasks before training commences. This initialization ensures that each Transformer encoder is appropriately configured to handle the specific architectural traces corresponding to its designated task ID.

The final step in the profiling stage is model training, which is conducted on a task-by-task basis. Prior to training, task IDs are assigned to the corresponding device traces. The model begins by profiling traces from one device, and upon completion, it proceeds to profile traces from subsequent devices. During each profiling session, only the Transformer encoder corresponding to the task ID is trained on the respective traces. This targeted training approach ensures that each encoder learns patterns that are highly relevant to its specific task. Additionally, after the initial profiling, the EWC mechanism computes the FIM to quantify the importance of model parameters for previously learned tasks. This FIM is then incorporated into the loss function with a regularization hyper-parameter λ, which helps in preserving the model's performance on old tasks while adapting to new ones.

3.5. Attack stage

During the attack stage, the Switch-T model is deployed to conduct SCAs across various devices with differing architectures. This stage is designed to demonstrate the model's capability to extract encryption keys by focusing on task-specific traces that correspond to the designated task IDs.

The model is deployed across multiple devices, where we collects side-channel traces. These traces are then preprocessed to align with the model's input requirements. Specifically, the traces are unified in length through PCA, which is crucial given the diverse physical structures of different devices. This preprocessing step ensures that the input data is consistent and compatible with the model's architecture, thereby facilitating accurate feature extraction and subsequent analysis.

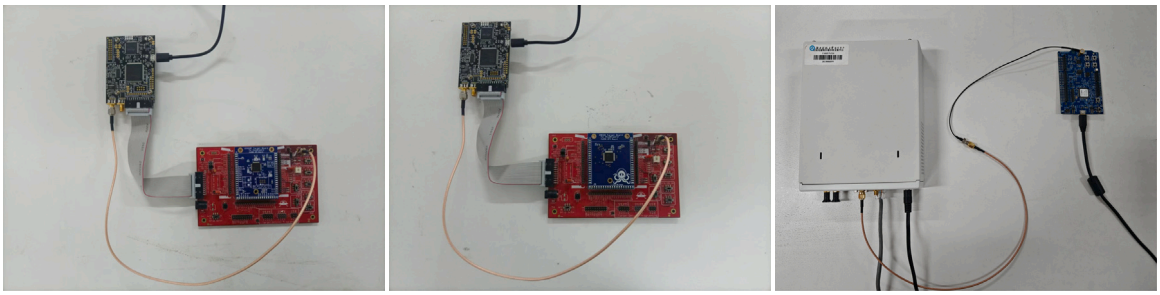
The Switch-T model's multi-task structure enables it to handle multiple tasks simultaneously, with each task-specific encoder focusing on traces that match its task ID. The attack process involves feeding these unified traces into the model, where each trace is processed by the corresponding Transformer encoder. This encoder, having been trained to recognize features indicative of encryption key operations, processes the traces to identify and exploit these features. The model's architecture ensures that the learned patterns are highly relevant to the specific task at hand, thereby enhancing the accuracy and efficiency of the attack.

The Switch-T model's use of the EWC mechanism is instrumental in maintaining its effectiveness over time. As the model adapts to new tasks, it retains the knowledge learned from previous tasks. This capability is crucial for ensuring that the model's performance does not degrade as it encounters new encryption algorithms and device architectures. The EWC mechanism allows the model to continuously improve its attack capabilities while mitigating the risk of catastrophic forgetting, thus ensuring robustness and versatility in dynamic environments.

Once the model successfully reconstructs the encryption keys, it can use them to decrypt data or perform other malicious activities. This demonstrates the model's effectiveness in multi-task SCA and underscores its potential as a powerful tool in the adversarial toolkit. The successful extraction of encryption keys highlights the model's ability to adapt and learn from diverse side-channel information, making it a formidable challenge for cryptographic security protocols.

4. Experimental setup

In this paper, we focus on three implementations of AES-128, which are STM32F3 MCU, ATXmega128D4 MCU, and nRF52832 SoC. We call them STM, Xmega and nRF devices in the following context,



(a) Setup for the STM device (b) Setup for the Xmega device. (c) Setup for the nRF device

Fig. 7. Experimental setup.

respectively. While the STM and nRF devices are both built on the CPU core with 32-bit Arm-Cortex M4 architecture as shown in Fig. 2, the Xmega device is built on the 8-bit AVR architecture as shown in Fig. 3. We call these two architectures as ARM and AVR. Besides, we use power consumption and Far Field EM emanations as side channels to conduct the attack. Specifically, we capture power consumption traces from the STM and Xmega implementations by using the ChipWhisperer-Lite tool and capture Far Field EM traces from the nRF implementation by using the Ettus N210 USRP Software Defined Radio (SDR). By doing this, we aim to show the cross-architecture (ARM vs. AVR) capacity of the Switch-T model within the same side channel (power) and the cross-channel (Far Field EM vs. power) capacity with the same core architecture (ARM).

Fig. 7(a) and (b) show the experimental setups for measuring power traces from the STM and Xmega devices, respectively. The sampling frequency of the Chipwhisperer board in this experiment is set to 40 MHz for both cases.

Fig. 7(c) shows the experimental setups for capturing Far Field EM traces from the nRF implementation of AES during the execution. In this case, we use the nRF device to encrypt plaintext and send the data continuously. Meanwhile, we use the USRP N210 SNR at the receiver side to capture the Far Field EM traces, which is as the same setting as in [17,32,46]. By following the training approach in [17], the training set used in the profiling stage contains 'clean' traces that are captured by coaxial cable and each trace represents the average of 100 measurements with the same encryption. A coaxial cable is capable to transmit signals that oscillate at high RF frequencies without radiating them outside. When the transmitter and receiver are connected by such a cable, the receiver receives the EM emission directly, in the form of the RF signal sent from the RF block on the chip. The central receiving frequency is configured to 2.528 GHz as derived in [47]. This frequency corresponds to the Bluetooth channel's center frequency $f_{\text{Bluetooth}} = 2.4$ GHz shifted by twice the target CPU's clock frequency $f_{clock} = 64$ MHz. In our experiments, we set the sampling frequency to 5 MHz, a value validated as sufficient for conducting the attack.

In both the training and testing datasets, all traces are pre-processed through the Max-Min normalization scaling method. This approach is employed to map the trace amplitude values into the [0, 1] interval. All datasets for experiment are conducted on a system equipped with an Intel i9-13900H processor, 16 GB RAM and Windows 11 operation system. The deep learning model is accelerated by NVIDIA GeForce RTX 4060 Laptop GPU. The proposed Switch-T model consists of a dimensional expansion layer (expand the input dimension to 128), three task-specific Transformer encoder blocks, where each block has 6 encoder layers with 32 heads in each encoder layer's multi-head self-attention sub-layer. The input dimension of Switch-T is 32. In the FFN sub-layer, the inner-layer has a dimensionality $d_{ff} = 2048$. The final classification model encompasses 256 distinct categories. The model is trained for 30 epochs with a batch size of 64 for datasets of STM and Xmega, 32 for datasets of nRF.

5. Dataset

In this section, we provide a detailed description of the dataset used in our experiments to evaluate the performance of the Switch-T model in conducting SCAs across different cryptographic implementations and side-channel types. In total, we evaluate the model on three datasets, built as described in Section 4. These three sets contain 40K power traces from the STM device, 40K power traces from the Xmega device, and 60K Far Field EM traces from the nRF device, respectively. We allocate the trace into training, validation, and test sets in a ratio of 6:2:2. For the STM and Xmega device, the selected trace segments represent the first round of AES-128. For the nRF device, the segment is for the last round.

For labeling traces in three datasets, we record the corresponding information during the measuring process. More specifically, we store the corresponding key and plaintext values for each trace for further process. The labeling process is a two-step procedure involving bitwise XOR operations followed by a transformation using the AES S-box. For each byte position i (where i ranges from 0 to 15), the $labels_i[j]$ for each sample j (where j ranges from 0 to $n-1$) is computed using Eq. (13) :

$$labels_i[j] = S_{box}[roundkey[j][i] \oplus pt[j][i]] \tag{13}$$

where $roundkey[j][i]$ and $pt[j][i]$ denote the ith byte of the round key and plaintext, respectively, for the jth trace in a specific dataset. The XOR operation is applied to these two bytes, and the result serves as an index to retrieve the corresponding value from the AES S-box S_{box}.

In this paper, we use CPA to identify PoIs with traces that exhibit a high degree of correlation with hypothesized intermediate values during the encryption process. The initial step involves standardizing the trace data, denoted as $\mathbf{X}$, and the vector of hypothesized intermediate values, $\mathbf{Y}$, to mitigate the effects of scale and mean bias. This standardization is achieved through the application of Eq. (14) :

$$\mathbf{X}_{std} = \frac{\mathbf{X} - \mu_{\mathbf{X}}}{\sigma_{\mathbf{X}}}, \mathbf{Y}_{std} = \frac{\mathbf{Y} - \mu_{\mathbf{Y}}}{\sigma_{\mathbf{Y}}} \tag{14}$$

where μ and σ represent the mean and standard deviation, respectively. Subsequently, the correlation coefficient matrix $\mathbf{C}$ is computed via the matrix multiplication:

$$\mathbf{C} = \mathbf{X}_{std}^T \mathbf{Y}_{std} \tag{15}$$

Elevated values within the correlation coefficient matrix serve as indicators of the PoIs, which are instrumental in extraction of key information and are particularly susceptible to SCAs. Before profiling, our datasets are preprocessed using the *min–max scaling* technique, whereby the amplitudes are normalized to the range [0, 1]. This preprocessing step ensures consistency and compatibility across the datasets. Detailed information on our datasets is presented in Table 2. Before training models, we use PCA to unify the dimension of input traces to 32 dimensions.

Table 2
Statistics of the dataset.

Device	# Total traces	# Train traces	# Valid traces	# Test traces	Trace length	PoIs range	Side channel	Arch
STM	40K	24K	8K	8K	400	[135, 170]	Power	ARM
Xmega	40K	24K	8K	8K	1700	[80, 160]	Power	AVR
nRF	60K	36K	12K	12K	400	[119, 151]	Far Field EM	ARM

Table 3
The experimental results of cross-architecture attacks in a STM-Xmega order.

Method	STM	Xmega	STM (Return)	STM (Adjust)	Adjust epochs
MLP	48.3%	28.5%	0.4%	47.7%	15
MLP-EWC	48.6%	1.7%	17.5%	48.9%	1
Switch-M	48.5%	79.3%	48.1%	–	–
Transformer	63.9%	98.7%	0.3%	63.4%	7
Transformer-EWC	64.3%	46%	1.2%	63.9%	1
Switch-T	64.2%	98.7%	64.5%	–	–

Table 4
The experimental results of cross-architecture attacks in a Xmega-STM order.

Method	Xmega	STM	Xmega (Return)	Xmega (Adjust)	Adjust epochs
MLP	85.7%	15.3%	0.4%	85.1%	14
MLP-EWC	83.8%	1.7%	35.6%	83.6%	1
Switch-M	81.8%	43.5%	82.8%	–	–
Transformer	98.7%	63.1%	0.2%	98.2%	5
Transformer-EWC	98.6%	61.5%	0.2%	98.4%	1
Switch-T	99%	62.5%	98.9%	–	–

Table 5
The experimental results of cross-channel attacks in a STM-nRF order.

Method	STM	nRF	STM (Return)	STM (Adjust)	Adjust epochs
MLP	45.1%	1.4%	0.6%	45.3%	6
MLP-EWC	44.6%	0.5%	31.4%	44.9%	1
Switch-M	43.7%	2.3%	44.6%	–	–
Transformer	64%	3.2%	0.9%	63.9%	3
Transformer-EWC	64.1%	1.5%	52.6%	64.1%	1
Switch-T	64.3%	3.1%	64%	–	–

6. Experimental results

This section presents a comprehensive analysis of the experimental results of using the proposed Switch-T model for classifying traces from different implementations. The evaluations are designed to assess the model's performance across various dimensions, such as its cross-architecture and cross-channel capabilities, as well as its robustness against catastrophic forgetting.

In the following subsections, we evaluate 6 DLSCA models for the comparison, including MLP, MLP with EWC (denoted as MLP-EWC), Switch-M (MLP with multi-task architecture), Transformer, Transformer with EWC (denoted as Transformer-EWC), and the proposed Switch-T model. The MLP based methods are derived from [43].

All of our experiments share the same workflow, which can be summarized as follows.

1. Initially, train and test the proposed model on a specific dataset (base).
2. Furthermore, train and test the profiled model sequentially on the remaining one or two datasets, while re-testing the model on the base after each training process.
3. Repeat the evaluation with different orders of the datasets.

In Sections 6.1–6.3, we set the critical hyper-parameter λ in EWC at 5×10^4, which plays a pivotal role in keeping memories for previous learned tasks. In the following experiments, we use the classification accuracy (single-trace attack rate) as the evaluation metrics. We obtain each attack result by averaging out classification accuracies of 10 tests.

6.1. Cross-architecture SCA

In this experiment, we compare the attack accuracy of the models on AES implementations built on different CPU core architectures (ARM and AVR) by using the same side channel (power consumption). We first train and test the models in a STM-Xmega order, followed by performing the same process in reverse order. Our experiments are structured to simulate a real-world scenario where an attacker must quickly adapt to different targets, thus necessitating a robust evaluation of the models' adaptation capabilities.

Tables 3 and 4 show the experimental results of the models in a STM-Xmega order and Xmega-STM order, respectively. In these two tables, the results in tables refer to single-trace recovery accuracy. Notably, the Transformer-based models, particularly the proposed Switch-T, demonstrate superior performance in cross-architectural attacks. For instance, in Table 3, Switch-T achieves a single-trace attack accuracy of 64.3% on the STM device and 98.7% on the Xmega device, significantly outperforming the MLP-based models. This superiority is attributed to the Transformer's self-attention mechanism, which excels at capturing intricate patterns in side-channel data, a critical capability for effective SCAs across diverse architectures.

Besides, we can also find that the EWC mechanism is instrumental in mitigating catastrophic forgetting, as evidenced by the fewer adjustment epochs required by models equipped with EWC, such as MLP-EWC, Switch-M, Transformer-EWC and Switch-T. As presented in the 5th and 6th columns in Tables 3 and 4, after profiling on the Xmega device, with the EWC's penalty to keep the knowledge learning from previous tasks, the MLP-EWC and Transformer-EWC require only 1 epoch to adjust back to the STM device, achieving a degradation less than 1%, extremely close to their initial attack accuracy. This rapid recovery underscores the effectiveness of EWC in preserving the model's knowledge of initial tasks.

Furthermore, the multi-task structure models, particularly the Switch-T model, demonstrate a significant enhancement in attack effectiveness on cross-architecture SCA when compared to traditional models. As illustrated in Table 3, after profiling STM and Xmega, the Transformer-EWC model's attack accuracy on the Xmega traces is 46%, which is notably lower than the 98.7375% achieved by the Switch-T model. This gap can be attributed to the EWC mechanism. When the EWC value is set to a large magnitude and the model has a significant number of parameters that need to be adjusted, It can limit the model's ability to change its parameters as it learns new tasks. However, the multi-task structure facilitates the segregation of certain critical parameters designated for specific tasks. Consequently, EWC is limited in its capacity to inhibit the adjustment of parameters that are less critical for feature extraction. The integration of EWC with a multi-task structure, such as Switch-M and Switch-T, results in an attack accuracy degradation of less than 1% on the base device, achieved without any parameter adjustment. This finding implies that the multi-task structure offers a more resilient framework for conducting SCAs across cryptographic implementations in diverse architectures. It is capable of adapting more effectively to novel tasks while simultaneously preserving the knowledge acquired from previous tasks.

In summary, Switch-T can perform better in cross-architecture SCA and overcoming catastrophic forgetting than other traditional DLSCA methods.

Table 6
The experimental results of cross-channel attacks in a nRF-STM order.

Method	nRF	STM	nRF (Return)	nRF (Adjust)	Adjust epochs
MLP	2.8%	17.5%	0.4%	2.1%	15
MLP-EWC	2.4%	2.3%	0.8%	2.1%	1
Switch-M	2.8%	44.4%	3.2%	–	–
Transformer	3.4%	64.6%	0.4%	3.4%	2
Transformer-EWC	3.4%	18.1%	2%	3.5%	1
Switch-T	3.7%	61.6%	3.5%	–	–

6.2. Cross-channel SCA

In this experiment, we extend our analysis to evaluate the Cross-channel SCA capabilities of DLSCA models, focusing on their adaptability across different types of side-channel information, specifically power consumption traces and Far Field EM radiations. Thus, we use the STM and nRF device as the targets, in which they share with the same CPU core. This dimension of our study is crucial for assessing the models' generalization capabilities when faced with varying physical leakages.

Tables 5 and 6 show the classification results of the models in a STM-nRF order and nRF-STM order, respectively. The Transformer-based models, such as Transformer, Transformer-EWC and Switch-T, once again demonstrate their superior performance in cross-channel attacks. For instance, in Tables 5 and 6, Switch-T achieves an initial attack accuracy of 64.3% on power consumption traces from the STM device and 3.7% on Far Field EM traces from the nRF device, significantly outperforming the MLP-based models. This performance indicates the Transformer's ability to capture and generalize subtle patterns across different side-channel types. Notice that the SOTA attack results [17,48] on nRF52 implementations of AES by using the Far Field EM emissions as the side channel requires more than 10 traces to recover a subkey under the same condition. This indicates that the SOTA single-trace attack result on this victim should be $\leq 4\%$.

The EWC mechanism is again crucial in mitigating catastrophic forgetting, as models with EWC require fewer adjustment epochs to regain their initial attack accuracy after switching side-channel types. For example, in Table 5, MLP-EWC and Transformer-EWC require only 1 epoch to adjust back to previous device traces after profiling on new device traces, achieving an accuracy close to its initial performance. However, MLP and Transformer need more than 3 epochs for the optimal adjustment.

Furthermore, the multi-task structure models, especially Switch-T, also show better attack accuracy on new side-channel types compared to traditional models. Compared with methods without multi-task structure, Switch-M and Switch-T can perform well in cross-channel attack and overcome catastrophic forgetting simultaneously. For example, in Table 6, Switch-T can achieve attack accuracy of 61.6% to STM after profiling nRF and STM. However, Transformer's accuracy to STM is only 18.1%. When finishing profiling on traces captured from STM device, Switch-T's re-test accuracy to nRF (base) is 3.5%, which has approximately few degradation with the initial accuracy of 3.7%. This indicates that the multi-task structure combined with the EWC mechanism is not only beneficial for cross-architecture attacks, but also for cross-channel attacks, providing a robust framework for conducting SCAs.

6.3. Attack multiple devices

In this experiment, we conduct a comprehensive assessment of the efficacy of DLSCA models in performing cross-architecture and cross-channel SCAs, with a particular focus on their ability to leverage EWC to overcome catastrophic forgetting. The experimental models include MLP-EWC, Switch-M, Transformer-EWC, and Switch-T, all of which are trained with the EWC mechanism. This configuration is essential for evaluating how well these models can maintain knowledge of previous tasks while adapting to new ones.

Table 7
The experimental results of models in a STM-Xmega-nRF order.

Method	STM	Xmega	nRF	STM (Return)
MLP-EWC	48.1%	1.7%	0.8%	11.3%
Switch-M	45.8%	78.4%	2.2%	45.9%
Transformer-EWC	64%	46%	0.9%	1.8%
Switch-T	64.9%	98.5%	4%	64.7%

Table 8
The experimental results of models in a STM-nRF-Xmega order.

Method	STM	nRF	Xmega	STM (Return)
MLP-EWC	48.9%	0.2%	8.3%	2.3%
Switch-M	45.8%	78.4%	2.2%	45.9%
Transformer-EWC	64.5%	1.4%	90.6%	0.5%
Switch-T	62.6%	3.7%	98.7%	64.1%

Table 9
The experimental results of models in a nRF-STM-Xmega order.

Method	nRF	STM	Xmega	nRF (Return)
MLP-EWC	2.7%	2.1%	6.8%	0.6%
Switch-M	2.6%	42.4%	85.4%	2.5%
Transformer-EWC	3.5%	19.5%	17.5%	1.7%
Switch-T	3.6%	60.5%	98.7%	3.4%

Table 10
The experimental results of models in a nRF-Xmega-STM order.

Method	nRF	Xmega	STM	nRF (Return)
MLP-EWC	2.8%	3.9%	5.3%	0.5%
Switch-M	2.6%	80.5%	46.6%	2.6%
Transformer-EWC	3.7%	37.1%	5.8%	1.2%
Switch-T	3.2%	98.8%	63.6%	3.4%

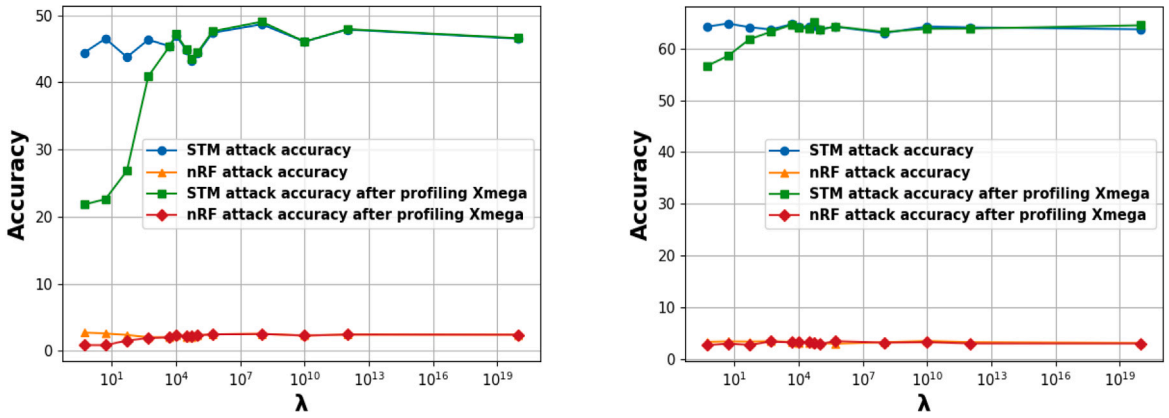
Table 11
The experimental results of models in a Xmega-STM-nRF order.

Method	Xmega	STM	nRF	Xmega (Return)
MLP-EWC	82.9%	1.4%	0.7%	36.6%
Switch-M	84.6%	42.8%	2.1%	84.6%
Transformer-EWC	98.8%	61.7%	0.4%	0.3%
Switch-T	98.7%	61.6%	3.2%	98.5%

Table 12
The experimental results of models in a Xmega-nRF-STM order.

Method	Xmega	nRF	STM	Xmega (Return)
MLP-EWC	81.3%	0.5%	1.6%	19%
Switch-M	87.7%	1.8%	46.3%	87.6%
Transformer-EWC	98.6%	3.1%	2%	8.4%
Switch-T	98.7%	3.3%	62.3%	98.7%

The results, as presented in Table 7 to Table 12, indicate that the Switch-T model excels in terms of attack accuracy. For instance as shown in Table 7, for the STM-Xmega-nRF order, Switch-T achieves an attack accuracy of 64.9% on STM, 98.5% on Xmega, and 4% on nRF. Impressively, after profiling nRF, the model's re-testing accuracy on STM, denoted as STM (Return), is 64.7%, demonstrating a minimal performance drop. Switch-T's resilience against catastrophic forgetting is evident in its sequential task learning performance. It shows only a slight decrease in accuracy when switching between devices, highlighting its capability to generalize across different conditions while retaining insights from previous tasks. This is particularly notable in the last column of the tables, which shows the model's testing accuracy when re-test on the initial device after profiling on subsequent ones (see Tables 7–12).



(a) Switch-M (b) Switch-T

Fig. 8. Comparison of attack accuracies impact by λ on Switch-M and Switch-T models in cross-device SCAs with a STM-nRF-Xmega order.

The superior performance of Switch-T can be attributed to its multi-task learning structure combined with the EWC mechanism. The multi-task structure allows the model to maintain task-specific parameters, reducing interference between tasks and enhancing the model's ability to adapt to new tasks without forgetting old ones. The EWC mechanism further consolidates this by penalizing updates to parameters that are crucial for performance on previous tasks, thus preserving the model's knowledge over time. Overall, the Switch-T model stands out as the most accurate and robust option for attacks across multiple devices, effectively mitigating the challenges posed by catastrophic forgetting. Its performance underscores the potential of combining multi-task learning structure with EWC in enhancing the adaptability and robustness of deep-learning models in the cross-device SCAs.

6.4. Effect of the EWC penalty hyper-parameter

In this experiment, we explore the impact of the EWC penalty hyper-parameter λ on the ability of overcoming catastrophic forgetting. We set Switch-T and Switch-M as test models for this experiment. The primary goal is to understand how the variation of λ affects the model's ability to balance the retention of previously learned tasks with the acquisition of new knowledge. This hyper-parameter plays a crucial role in the EWC mechanism, influencing the degree to which the model preserves the parameters associated with well-learned tasks while adapting to new tasks.

To investigate this, the Switch-T model undergoes sequential profiling across three distinct datasets representing the STM, nRF and Xmega devices. The λ is systematically varied across a range from 0.5 to 10^{20}, allowing us to observe how the model's performance fluctuates with increasing constraints on weight consolidation. After each full cycle of training, the model's accuracy on the previous devices' traces is evaluated to measure the degree of forgetting that occurs as a result of learning subsequent tasks.

The experimental setup involves training models on traces from each device in a sequential manner. For each value of λ, we train models on the traces from one device to the next devices sequentially. When profiling traces of one device is completed, we record the attack accuracy to this device. After accomplishing profiling and attacking the last device, we re-attack the previous devices and record the attack accuracy. This process above is repeated for each λ value to gather comprehensive data on how the hyper-parameter affects the models' ability to retain knowledge across tasks.

The experimental results, as shown in Figs. 8 to 13, reveal a nuanced relationship between λ and the model's capacity of overcoming catastrophic forgetting. At $\lambda = 0.5$, the model exhibits significant forgetting, as evidenced by a substantial drop in accuracy on the previous traces. This indicates that with low penalty or without any penalty, the model's parameters are freely updated, leading to catastrophic forgetting of previously learned tasks. As λ increases, the model's accuracy degradation initially lessens, suggesting that higher constraints on weight changes indeed assist in preserving knowledge of the initial task. This is because the EWC mechanism penalizes deviations from the optimal parameters of previous tasks, thereby stabilizing the model's performance on those tasks.

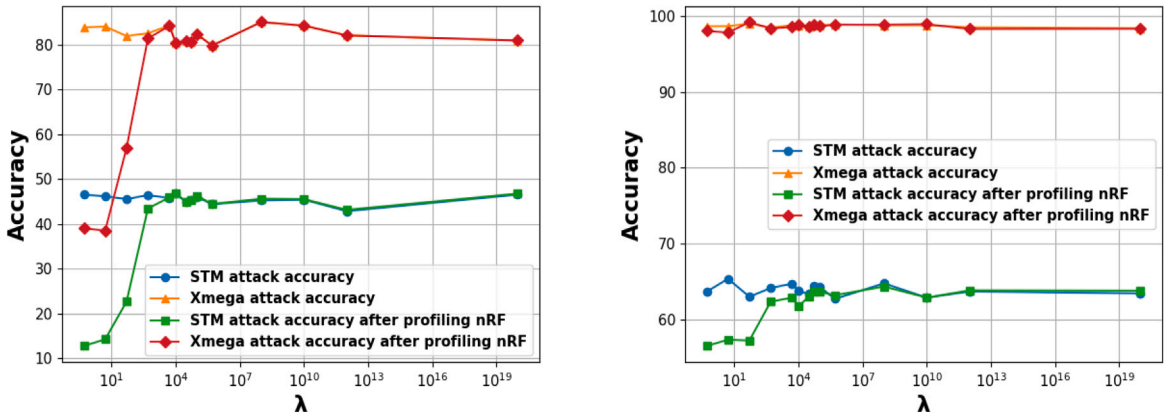
As λ continues to rise beyond a certain threshold, the accuracy degradation begins to be less than 1%, and interestingly, the Switch-T model's performance even slightly improves, as shown in Fig. 8(b). This suggests that while an optimal λ value can mitigate forgetting, an excessively high value leads to over-constraining the model's ability to adapt, thus incurring the cost of reduced flexibility in learning new tasks. The experimental findings underscore the delicate balance required to set the hyper-parameter λ. While an increase in λ can initially help in preserving previous knowledge, there is a point of diminishing returns where further increases lead to a cost in terms of adaptability. This trade-off between retention and adaptability provides insights into the careful calibration of λ necessary for reducing catastrophic forgetting in the scenario of cross-device SCAs. The optimal value of λ depends on the specific tasks and datasets, highlighting the importance of empirical tuning to achieve the best performance in dynamic environments.

In comparing Switch-T and Switch-M's ability to overcome catastrophic forgetting, Figs. 8 to 13 provide valuable insights through their illustrations of (a) Switch-M and (b) Switch-T performance in various cross-device SCAs scenarios. These figures depict the models' accuracy on initial tasks after being trained on subsequent tasks, highlighting how effectively each model retains previously learned information.

Switch-T consistently demonstrates superior performance in retaining knowledge across tasks, as evidenced by its higher accuracy levels in the (b) figures compared to Switch-M's in the (a) figures. This advantage is attributed to Switch-T's Transformer-based architecture, which excels at capturing complex patterns and relationships in side-channel data. The self-attention mechanism allows Switch-T to dynamically allocate attention to relevant features, ensuring that critical information from previous tasks is not overshadowed by new data. Consequently, Switch-T can maintain a robust understanding of the initial tasks even after exposure to additional tasks.

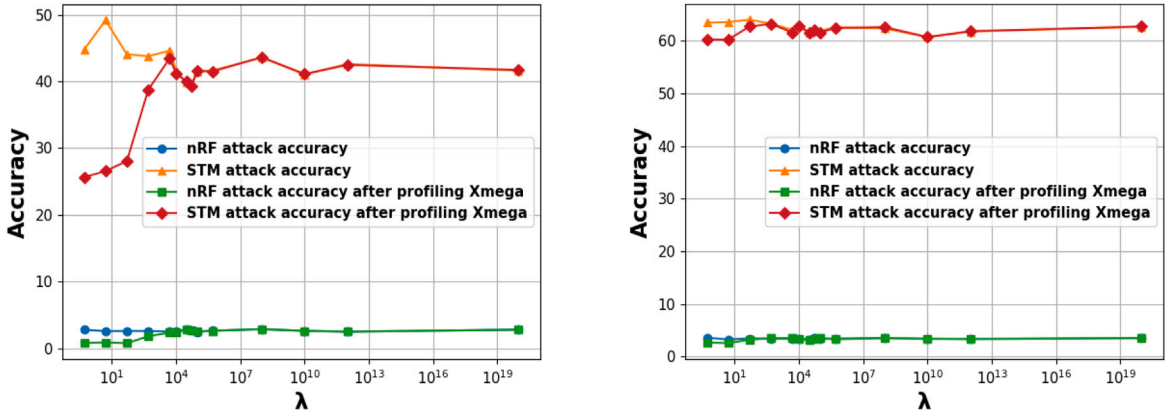
Moreover, Switch-T's multi-head attention component enables it to process information from multiple perspectives, further enhancing its ability to discern and retain task-specific features. This multifaceted approach allows Switch-T to create a more comprehensive and nuanced representation of the data, making it less susceptible to catastrophic forgetting as new tasks are introduced.

In contrast, Switch-M's reliance on traditional neural network models, such as MLPs, limits its capacity for dynamic feature selection and



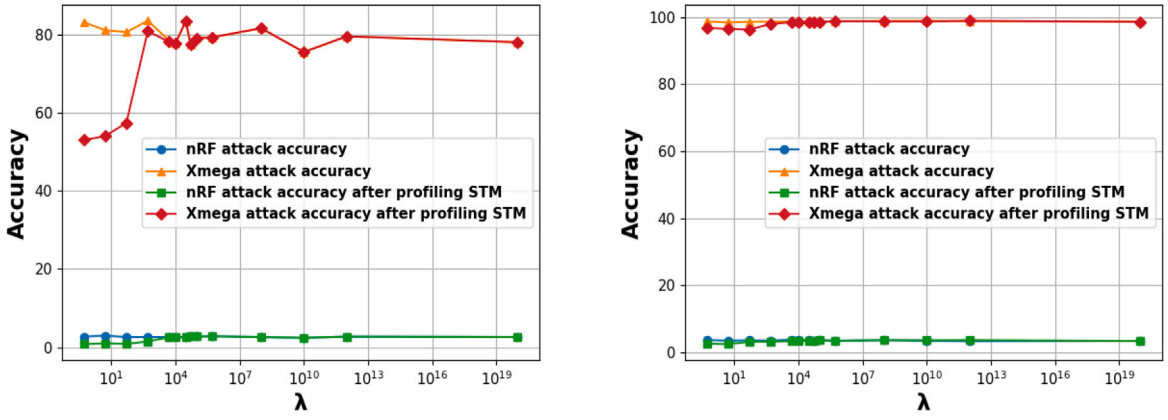
(a) Swtich-M (b) Switch-T

Fig. 9. Comparison of attack accuracies impact by λ on Switch-M and Switch-T models in cross-device SCAs with a STM-Xmega-nRF order.



(a) Switch-M (b) Switch-T

Fig. 10. Comparison of attack accuracies impact by λ on Switch-M and Switch-T models in cross-device SCAs with a nRF-STM-Xmega order.

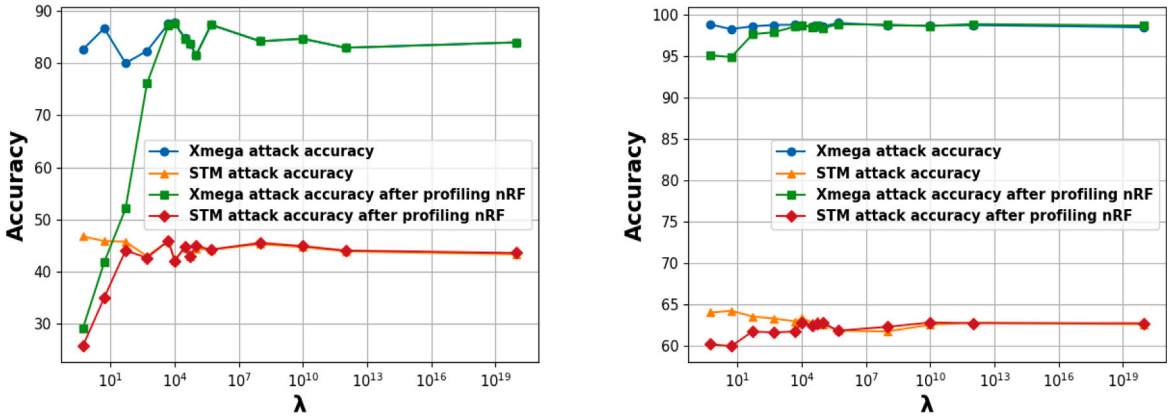


(a) Switch-M (b) Switch-T

Fig. 11. Comparison of attack accuracies impact by λ on Switch-M and Switch-T models in cross-device SCAs with a nRF-Xmega-STM order.

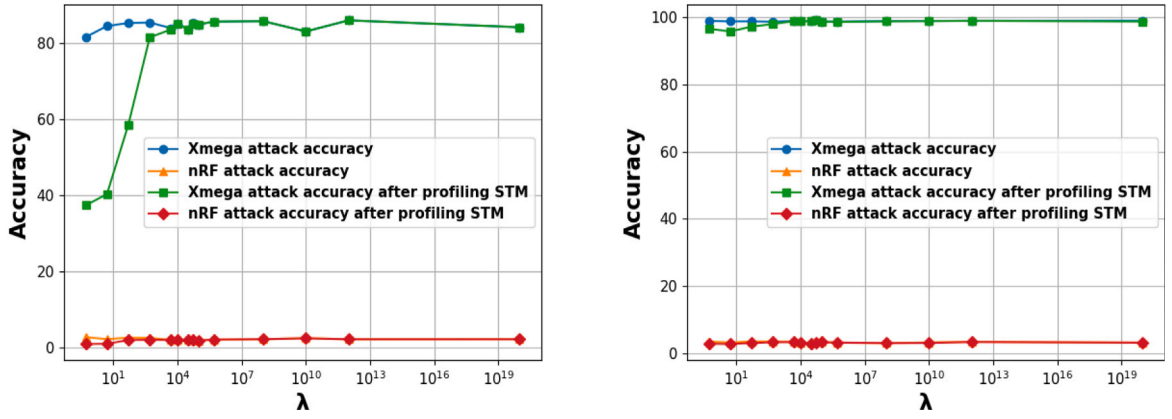
emphasis. MLPs tend to process data in a more uniform manner, which can lead to the dilution of important features from initial tasks when the model is trained on subsequent tasks. As a result, Switch-M's accuracy on initial tasks tends to decline more significantly than Switch-T's when λ is set as numeric values less than 10^4, as shown in the (a) figures, indicating a greater susceptibility to catastrophic forgetting.

The EWC mechanism, while beneficial for both models in stabilizing performance on previous tasks, is more effectively leveraged by Switch-T due to its advanced architecture. Switch-T's ability to maintain task-specific parameters and its sophisticated feature extraction capabilities allow the EWC penalty to be applied more precisely, preserving crucial knowledge without hindering adaptation to new tasks.



(a) Switch-M (b) Switch-T

Fig. 12. Comparison of attack accuracies impact by λ on Switch-M and Switch-T models in cross-device SCAs with a Xmega-STM-nRF order.



(a) Switch-M (b) Switch-T

Fig. 13. Comparison of attack accuracies impact by λ on Switch-M and Switch-T models in cross-device SCAs with a Xmega-nRF-STM order.

In essence, Switch-T's combination of multi-task structure innovations and the EWC mechanism results in a more resilient model that can adeptly balance retention of past knowledge with the acquisition of new information.

7. Conclusion

In this paper, we propose a novel multi-task deep learning SCA model called Switch-T, which utilizes a Transformer encoder and the EWC mechanism to perform multi-task SCAs. Our experimental results demonstrate the effectiveness of the Switch-T model in compromising different implementations of AES, including Xmega, STM, and nRF, respectively. Furthermore, we investigate the impact of the training order of devices on the attack efficiency of the model and discuss the impact of hyper-parameter settings in the EWC mechanism.

The experimental results show that the proposed Switch-T model outperforms other baseline models in cross-architecture attacks, achieving best attack accuracies of 64.5% on STM and 99% on Xmega. In cross-channel attacks, the proposed model also demonstrates superior performance, with best attack accuracies of 64.3% on power consumption traces from STM and 3.7% on Far Field EM traces from nRF.

The Switch-T model's ability to maintain its memory for knowledge learned from previous tasks in attacks across multiple devices highlights its robustness and adaptability. The EWC mechanism plays a crucial role in balancing the retention of previously learned tasks with the acquisition of new knowledge. Our experiments show that an appropriate EWC hyper-parameter λ is essential for deep-learning model to mitigate catastrophic forgetting after profiling on a new task's dataset. However, an excessively high value can limit the model's flexibility in learning new tasks.

Despite the Switch-T model's significant performance in cross-architecture and cross-channel SCAs, there are several potential directions for future work. These include further optimizing the model structure to enhance profiling flexibility and attack accuracy, extending the model to handle multiple tasks such as attacking various cryptographic algorithms or different side-channel leakages simultaneously, developing more effective defense mechanisms to counter such multi-task deep learning SCAs, and evaluating the model's performance in real-world applications involving IoT devices and embedded systems.

Overall, the Switch-T model provides a new and effective approach to multi-task deep learning in the field of SCAs, offering important theoretical and practical implications. Its capabilities in overcoming catastrophic forgetting and multi-task processing make it a promising tool for future cryptographic security research.

CRediT authorship contribution statement

Jiale Liao: Writing – original draft, Methodology, Investigation, Formal analysis, Data curation. **Huanyu Wang:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Conceptualization. **Junnian Wang:** Writing – review & editing, Validation, Supervision, Resources, Conceptualization. **Yun Tang:** Writing – review & editing, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work received support from the Department of Education of Hunan Province for the Distinguished Young Scholars (Grant No. 24B0446).

Data availability

Data will be made available on request.

References

[1] He D, Wang H, Deng T, Liu J, Wang J. Improving iIoT security: Unveiling threats through advanced side-channel analysis. Comput Secur 2025;148:104135.
[2] Bokharaie VS, Jahanian A. Side-channel leakage assessment metrics and methodologies at design cycle: A case study for a cryptosystem. J Inf Secur Appl (JISA) 2020.
[3] Zhang J, Liang B, Zhang H, Zhang W, Ling Z, Yang M. Mobile applications identification using autoencoder based electromagnetic side channel analysis. J Inf Secur Appl (JISA) 2023.
[4] Qu S, Wang Y, Yu J, Zhang C, Gu D. Trace copilot: Automatically locating cryptographic operations in side-channel traces by firmware binary instrumenting. IACR Trans Cryptogr Hardw Embed Systems (TCHES) 2025;2025(1):128–59.
[5] Nassi B, Vayner O, Iluz E, Nassi D, Jancar J, Genkin D, et al. Optical cryptanalysis: Recovering cryptographic keys from power LED light fluctuations. In: ACM SIGSAC conference on computer and communications security. 2023, p. 268–80.
[6] Gao Y, Qiu H, Zhang Z, Wang B, Ma H, Abuadbba A, et al. Deeptheft: Stealing DNN model architectures through power side channel. In: IEEE symposium on security and privacy. IEEE; 2024, p. 3311–26.
[7] Oberhuber M, Unterguggenberger M, Maar L, Kogler A, Mangard S. Power-related side-channel attacks using the android sensor framework. In: Network and distributed system security symposium. 2025.
[8] Ni T, Zhang X, Zuo C, Li J, Yan Z, Wang W, et al. Uncovering user interactions on smartphones via contactless wireless charging side channels. In: 2023 IEEE symposium on security and privacy. IEEE; 2023, p. 3399–415.
[9] Hu J, Wang H, Zheng T, Hu J, Chen Z, Jiang H, et al. Password-stealing without hacking: Wi-fi enabled practical keystroke eavesdropping. In: Proceedings of the 2023 ACM SIGSAC conference on computer and communications security. 2023, p. 239–52.
[10] Long Y, Jiang Q, Yan C, Alam T, Ji X, Xu W, et al. EM eye: Characterizing electromagnetic side-channel eavesdropping on embedded cameras. In: Proceedings of ACM NDSS. 2024.
[11] Ni T, Zhang X, Zhao Q. Recovering fingerprints from in-display fingerprint sensors via electromagnetic side channel. In: Proceedings of the 2023 ACM SIGSAC conference on computer and communications security. 2023, p. 253–67.
[12] Hu F, Shen J, Vijayakumar P. Side-channel attacks based on multi-loss regularized denoising AutoEncoder. IEEE Trans Inf Forensics Secur 2023.
[13] Cao P, Zhang C, Lu X, Gu D, Xu S. Improving deep learning based second-order side-channel analysis with bilinear CNN. IEEE Trans Inf Forensics Secur (TIFS) 2022;17:3863–76.
[14] Do N-T, Hoang V-P, Doan VS. A novel non-profiled side channel attack based on multi-output regression neural network. J Cryptogr Eng 2023;1–13.
[15] Chari S, Rao JR, Rohatgi P. Template attacks. In: International workshop on cryptographic hardware and embedded systems. 2002, p. 13–28.
[16] Camurati G, Francillon A, Standaert F-X. Understanding screaming channels: From a detailed analysis to improved attacks. IACR Trans Cryptogr Hardw Embed Systems (TCHES) 2020;358–401.
[17] Wang R, Wang H, Dubrova E, Brisfors M. Advanced far field EM side-channel attack on AES. In: ACM on cyber-physical system security workshop. 2021, p. 29–39.
[18] Cagli E, Dumas C, Prouff E. Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without pre-processing. In: Cryptographic hardware and embedded systems. Springer; 2017, p. 45–68.
[19] Dubrova E, Ngo K, Gärtner J, Wang R. Breaking a fifth-order masked implementation of CRYSTALS-kyber by copy-paste. In: ACM Asia public-key cryptography workshop. 2023, p. 10–20.
[20] Ji Y, Dubrova E. A side-channel attack on a masked hardware implementation of CRYSTALS-kyber. In: Proceedings of the 2023 workshop on attacks and solutions in hardware security. 2023, p. 27–37.
[21] Benadjila R, Prouff E, Strullu R, Cagli E, Dumas C. Deep learning for side-channel analysis and introduction to ASCAD database. J Cryptogr Eng 2020;10(2):163–88.
[22] Wang Z, Ding Y, Wang A, Zhang Y, Wei C, Sun S, et al. SPA-gpt: general pulse tailor for simple power analysis based on reinforcement learning. IACR Trans Cryptogr Hardw Embed Systems (TCHES) 2024;2024(4):40–83.
[23] Wang H, Forsmark S, Brisfors M, Dubrova E. Multi-source training deep-learning side-channel attacks. In: IEEE International symposium on multiple-valued logic. 2020, p. 58–63.
[24] Rijmen V, Daemen J. Advanced encryption standard. Proceedings of federal information processing standards publications, vol. 19, 2001, p. 22.
[25] Sravani M, Durai S. Attacks on cryptosystems implemented via VLSI: A review. J Inf Secur Appl (JISA) 2021.
[26] Kocher P, Jaffe J, Jun B. Differential power analysis. In: Advances in cryptology. 1999, p. 388–97.
[27] Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model. In: International workshop on cryptographic hardware and embedded systems. Springer; 2004, p. 16–29.
[28] Guo P, Yan Y, Zhang F, Zhu C, Zhang L, Dai Z. Extending the classical side-channel analysis framework to access-driven cache attacks. Comput Secur 2023;129:103255.
[29] Barenghi A, Carrera D, Mella S, Pace A, Pelosi G, Susella R. Profiled side channel attacks against the RSA cryptosystem using neural networks. J Inf Secur Appl 2022.
[30] Timon B. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Trans Cryptogr Hardw Embed Systems (TCHES) 2019;107–31.
[31] Ngo K, Wang R, Dubrova E. Higher-order boolean masking does not prevent side-channel attacks on LWE/LWR-based PKE/KEMs. In: International symposium on multiple-valued logic. 2023, p. 190–5.
[32] Wang H. Amplitude-modulated EM side-channel attack on provably secure masked AES. J Cryptogr Eng 2024;1–13.
[33] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, N.Gomez A, et al. Attention is all you need. In: Neural information processing systems. 2017.
[34] Delvin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. 2019, arXiv:1810.04805v2.
[35] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An image is worth 16x16 words: transformers for image recognition at scale. In: International conference on learning representations. 2021.
[36] Hajra S, Alam M, Saha S, Picek S, Mukhopadhyay D. On the instability of softmax attention-based deep learning models in side-channel analysis. IEEE Trans Inf Forensics Secur 2023.
[37] Hajra S, Chowdhury S, Mukhopadhyay D. EstraNet: An efficient shift-invariant transformer network for side-channel analysis. 2024.
[38] Müller M, Vlaar T, Rolnick D, Hein M. Normalization layers are all that sharpness-aware minimization needs. In: Conference on neural information processing systems. 2023.
[39] Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, A.Rusu A, et al. Overcoming catastrophic forgetting in neural networks. 2017, arXiv:1612.00796v2.
[40] Heng A, Soh H. Selective amnesia: A continual learning approach to forgetting in deep generative models. In: Conference on neural information processing systems. 2023.
[41] Song Z, Nie B, Huang S. Pulse transfer learning: Multi-area river ammonia nitrogen prediction with limited data. In: Expert systems with applications. 2025.
[42] Weissbart L, Picek S. Lightweight but not easy: Side-channel analysis of the ascon authenticated cipher on a 32-bit microcontroller. In: IACR cryptol. ePrint arch. 2023.
[43] Das D, Golder A, Danial J, Ghosh S, Raychowdhury A, Sen S. X-Deepsca: Cross-device deep learning side channel attack. In: Proceedings of the 56th annual design automation conference 2019. 2019, p. 1–6.
[44] Cao P, Zhang H, Gu D, Lu Y, Yuan Y. AL-pa: Cross-device profiled side-channel attack using adversarial learning. In: 2022 59th ACM/IEEE design automation conference. IEEE; 2022, p. 691–6.
[45] Yu H, Shan H, Panoff M, Jin Y. Cross-device profiled side-channel attacks using meta-transfer learning. In: 2021 58th ACM/IEEE design automation conference. IEEE; 2021, p. 703–8.
[46] Wang R, Wang H, Dubrova E. Far field EM side-channel attack on AES using deep learning. In: ACM workshop on attacks and solutions in hardware security. 2020, p. 35–44.
[47] Camurati G, Poeplau S, Muench M, Hayes T, Francillon A. Screaming channels: When electromagnetic side channels meet radio transceivers. In: ACM SIGSAC conference on computer and communications security. 2018, p. 163–77.
[48] Wang H. deep learning side-channel attacks on advanced encryption standard [Ph.D. thesis], KTH Royal Institute of Technology; 2023.