IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–34. DOI:XXXXXXXX

SPA-GPT: General Pulse Tailor for Simple Power Analysis Based on Reinforcement Learning

- Long Paper -

Ziyu Wang, Yaoling Ding, An Wang*, Yuwei Zhang, Congming Wei, Shaofei Sun and Liehuang Zhu

School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China,
wzy23,dyl19,wangan1,ywzhang,weicm,sfsun,liehuangz@bit.edu.cn

Abstract. Power analysis of public-key algorithms is a well-known approach in the community of side-channel analysis. We usually classify operations based on the differences in power traces produced by different basic operations (such as modular exponentiation) to recover secret information like private keys. The more accurate the segmentation of power traces, the higher the efficiency of their classification. There exist two commonly used methods: one is equidistant segmentation, which requires a fixed number of basic operations and similar trace lengths for each type of operation, leading to limited application scenarios; the other is peak-based segmentation, which relies on personal experience to configure parameters, resulting in insufficient flexibility and poor universality.

In this paper, we propose an automated power trace segmentation method based on reinforcement learning algorithms, which is applicable to a wide range of common implementation of public-key algorithms. Reinforcement learning is an unsupervised machine learning technique that eliminates the need for manual label collection. For the first time, this technique is introduced into the field of side-channel analysis for power trace processing. By using prioritized experience replay optimized Deep Q-Network algorithm, we reduce the number of parameters required to achieve accurate segmentation of power traces to only one, i.e. the key length. We also employ various techniques to improve the segmentation effectiveness, such as clustering algorithm, enveloped-based feature enhancement and fine-tuning method. We validate the effectiveness of the new method in nine scenarios involving hardware and software implementations of different public-key algorithms executed on diverse platforms such as microcontrollers, SAKURA-G, and smart cards. Specifically, one of these implementations is protected by time randomization countermeasures. Experimental results show that our method has good robustness on the traces with varying segment lengths and differing peak heights. After employ the clustering algorithm, our method achieves an accuracy of over 99.6% in operations recovery. Besides, power traces collected from these devices have been uploaded as databases, which are available for researchers engaged in public-key algorithms to conduct related experiments or verify our method.

Keywords: Side-channel Analysis · Power Trace Segmentation · Reinforcement Learning · Deep Q-Network

1 Introduction

In 1999, Kocher first proposed side-channel analysis for cryptosystems and successfully achieved key recovery using Timing Analysis and Simple Power Analysis (SPA) [KJJ99].

*An Wang is the corresponding author.

Since then, side-channel analysis in cryptographic algorithms has received widespread attention. Over the course of several decades, various side-channel analysis methods have emerged, including Correlation Power Analysis (CPA) [BCO04], Template Attacks [CRR02], Collision Attacks [Bog07], Mutual Information Analysis (MIA) [GBTP08], and so on.

Public-key algorithms such as RSA and ECC are commonly used for identity recognition and digital signatures. These algorithms exhibit significant power consumption patterns, making them susceptible to SPA [MDS99]. As a result, SPA has become a core step for evaluating the security of cryptographic devices in standards [ISO12, ISO15, ISO16]. Regular methods such as Montgomery ladder [JY02], square-and-multiply always [Cor99], and atomicity-based implementations [GV10] can effectively resist SPA. However, Differential Power Analysis (DPA) [KJJ99], CPA [BCO04], MIA [GBTP08] can still be used to recover secret information. In order to counter these attacks, private key blinding is proposed as a key defense mechanism, although it is worth noting that these methods are limited to performing private key recovery with only a single trace. Correspondingly, researchers have successively proposed several improved SPAs called horizontal attacks by combining machine learning and mathematical analysis to address this type of defense.

Horizontal attacks that target single trace typically involve three steps: trace segmentation, segments classification, and private key recovery. The quality of trace segmentation directly impacts the accuracy of segments classification, which in turn affects private key recovery. Currently, there are three strategies for trace segmentation:

- The first approach assumes that attackers have access to a device of the same model as target device. By setting trigger signal through General-Purpose Input/Output (GPIO), attackers can locate all power segments corresponding to each operation. However, in practical scenarios, it is difficult to obtain such devices.

- The second approach involves connecting special devices [New23, BBGV16] with the cryptographic device. While collecting power traces, trace segmentation is performed using triggering systems based waveform-matching method. However, this method requires establishing templates for each type of trace segments before performing waveform matching. Besides, this segmentation approach is mostly used to locate cryptographic operations in traces of block ciphers. When applied to public-key algorithms, it is often too costly to to obtain usable results.

- The third approach involves directly segmenting traces without any other assisting devices, which includes two methods, i.e. equidistant segmentation [HMH$^+$12] and peak-based segmentation [WHW$^+$22]. Equidistant segmentation divides an entire trace into segments of equal length, assuming that the number of segments are known. Each trace segment corresponds to a basic operation which is related to the private key. This segmentation method is only suitable when the length of each segment corresponding to cryptographic operations is equal. Peak-based segmentation, on the other hand, involves segmenting trace based on peaks. Manual configuration of parameters is required, such as threshold of peak height and minimum distance between peaks. This method is only applicable to traces with periodic peaks which are produced by high-power-consuming operations. It is important to note that both methods require manual determination of segmentation pattern or key parameters (such as fixed lengths and peak heights), which is highly dependent on human experience.

After segmenting a trace, alignment is necessary due to the inevitable timing deviations during the execution of cryptographic algorithm on a device. For horizontal alignment, there are general methods such as Peak Alignment and Dynamic Time Warping (DTW) [vWWB11]. For vertical alignment, all segments can be adjusted to approximate horizontal positions by subtracting their average. Once aligned, leakage analysis and feature extraction are usually performed prior to the analysis to improve efficiency. Common leakage assessment techniques include Test Vector Leakage Assessment (TVLA) [GGJR+11] and Sum-of-Squared T-values (SOST) [PC15]. Common feature extraction methods include Principal Component Analysis (PCA) [SHKS15] and Linear Discriminant Analysis (LDA) [CLH19, BG98]. Afterwards, some classification methods, such as horizontal correlation analysis [CFG+10], cluster-based analysis [HIM+13], and cluster combined with neural network analysis [PCBP21, WHW+22], are applied to original or feature-extracted trace segments to distinguish between operations and recover private keys. In summary, almost all existing literature based on artificial intelligence algorithms focus on automated classification of trace segments, while neglecting trace segmentation.

In this paper, we aim to introduce reinforcement learning to trace segmentation, in order to push foreword fully automatic SPA on public-key algorithms. Reinforcement learning algorithms were initially applied in the field of automatic control [Mat97] and subsequently developed rapidly in stock prediction [AK22] and game competition [SHM+16]. In 2020, this algorithm was introduced to the field of side-channel analysis to test the resilience of cryptographic chips against timing attacks [PSM20]. Later, Ramezanpour et al. used autoencoders to encode trace information and applied reinforcement learning to extract key data, thereby improving the analysis efficiency, even in the absence of knowledge about leakage models [RAD20]. In recent research, Rijsdijk et al. utilized Q-learning algorithm, a type of reinforcement learning method, to adjust hyperparameters in side-channel analysis based on a CNN network, enhancing the efficiency of key recovery [RWPP21]. Based on the above survey and an observation that power traces produced by public-key algorithms exhibit significant periodic characteristics, we propose a reinforcement learning-based trace segmentation method that accurately segments traces of public-key algorithms. The main contributions are as follows:

- We propose a new approach for segmenting traces of public-key algorithms automatically based on reinforcement learning. The new method only require private key lengths processed in the target device to be launched, which significantly reduces reliance on manual expertise compared to existing methods. Experimental results demonstrate that our method improves both accuracy and efficiency of segmentation, and it is applicable to various software and hardware implementations of public-key algorithms, such as SAKURA-G, smart cards, and AISCs.

- Moreover, our approach seamlessly integrates with horizontal attacks, enabling the automation of entire SPA process. In the enhanced version of our method, segments are classified to two sets by clustering algorithms, which improves the accuracy of segmentation significantly. Experimental results show that the new method reaches above 99.6% accuracy for the recovery of cryptographic operations in public-key decryption or signature verification processes which are implemented in smart cards and microcontrollers.

- Specifically, for a trace with less prominent segmentation features, we use envelope detection to obtain envelope curve of the trace and then perform our new method to the envelope curve, of which the segmentation points are mapped back to the original trace. Besides, we propose a fine-tuning technique to make segmentation results more organized. The experimental results demonstrate that equidistant, findpeak, peakdistance segmentation methods fail to accurately segment power traces from devices like smart cards, ASICs, and SAKURA-G. However, our approach enables precise segmentation in these traces.

- Additionally, we organize all the traces involved in this paper and establish a database which include various implementations of public-key algorithms for researchers to validate our new method or conduct other experiments. The database includes five implementations (software/hardware/software-hardware co-design) of RSA and one implementation of ECC.

Paper Organization. In Section 2, we provide a brief introduction of SPA for public-key algorithms and reinforcement learning algorithms. In Section 3, we present the basic framework of SPA-GPT and discuss its details, including environment design, agent design, and interaction between the agent and the environment. In the following Section 4, experiments are conducted on three type of power traces produced by different implementation of public-key algorithms to verify the advantages of SPA-GPT. In Section 5, we enhance SPA-GPT in terms of environment, prepossessing and result fine-tuning to improve segmentation effectiveness. In Section 6, more validation experiments of Enhanced SPA-GPT are carried out on special power traces, followed by its performance and a summary of all experimental results. Finally, we conclude our work and present a future out look in Section 7.

2 Preliminary

2.1 Brief Description of SPA on Public-key Algorithms

Currently, there are two commonly used public-key algorithms: RSA [RSA78] and ECC [Kob87, Mil85]. The sequence of cryptographic operations is directly related to the private key. Taking RSA as an example, when a private key bit is “0”, the algorithm only performs modular square. However, when a private key bit is “1”, the algorithm sequentially performs modular square and modular multiplication. Due to the typically differences in execution time and power consumption (voltage or current) among different cryptographic operations, these variations are often reflected significantly in power traces. As a result, SPA often involves analyzing segments corresponding to cryptographic operations produced by a target device to obtain information about the private keys. Performing SPA on implementations of public-key algorithms mainly involves three steps: the first step is trace segmentation, the second step is segments classification, and the third step is private key recovery.

After AI is introduced into SPA, clustering algorithms are employed to classify segments of power traces, which improves automation of SPA process. However, manual expertise is still required for power trace segmentation. This paper focuses on AI solution of power trace segmentation to achieve full automation of SPA, as illustrated in Figure 1.

In our assumption, the start point of a cryptographic algorithm can be identified when the upper computer (such as a PC) sends a decryption (or signature) request to the lower computer (such as a smart card). But the end point of a cryptographic algorithm on a trace is unknown in our scenario. That is to say, our method is applicable to traces containing redundant segments that do not correspond to the target private key. Besides, Our method is functionable to public-key algorithms with blinding protection. Specifically, it involves analyzing a target trace without any prior knowledge.

2.2 Reinforcement Learning

Literature [SB98] summarizes the two main entities in reinforcement learning as agent and environment. Agent serves as the actor responsible for taking actions, while environment is the entity with which the agent interacts. Beyond the agent and the environment, we identify three main subelements of a reinforcement learning system: policy, reward and value function. Policy is used to map the current state of environment (referred to as a state) to the action to be taken. Reward represents feedback given by the environment to

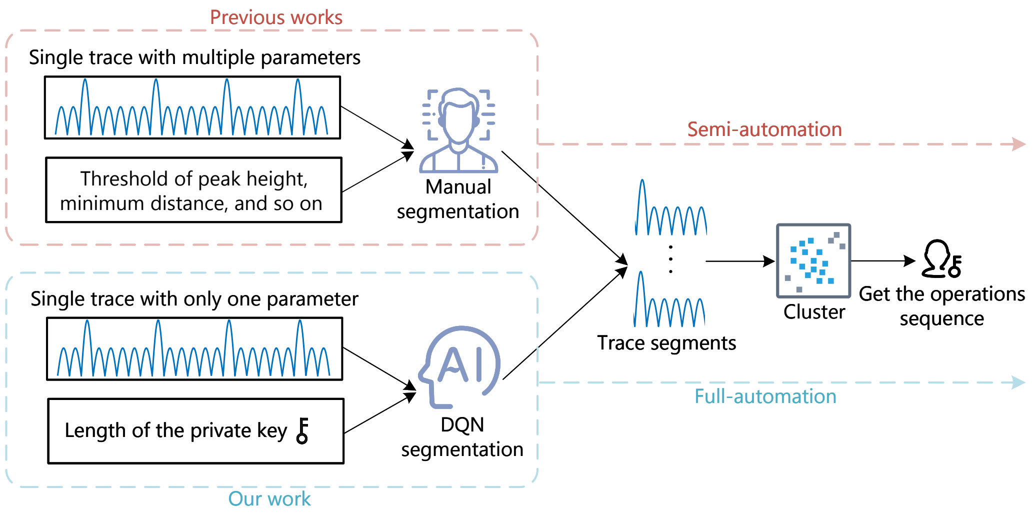

Previous works
Single trace with multiple parameters
Threshold of peak height, minimum distance, and so on
Manual segmentation
Semi-automation
Single trace with only one parameter
Length of the private key
DQN segmentation
Full-automation
Our work
Trace segments
Cluster
Get the operations sequence

Figure 1: Automating segmentation processes combinated AI

the agent, indicating the quality of current action. Value function is used to evaluate the long-term return. The interaction mechanism between the agent and the environment is illustrated in Figure 2. At time t, the agent in state S_t takes an action A_t based on its policy and value function. Subsequently, it receives a reward R_{t+1} from the environment and updates its state to S_{t+1}. The state and reward are used to enhance the decision-making capabilities of the agent. As the agent attempts to complete the task given by the environment, it may experience failures and restarts. The process from the start to the end is defined as an episode, and each execution of an action is defined as a step.

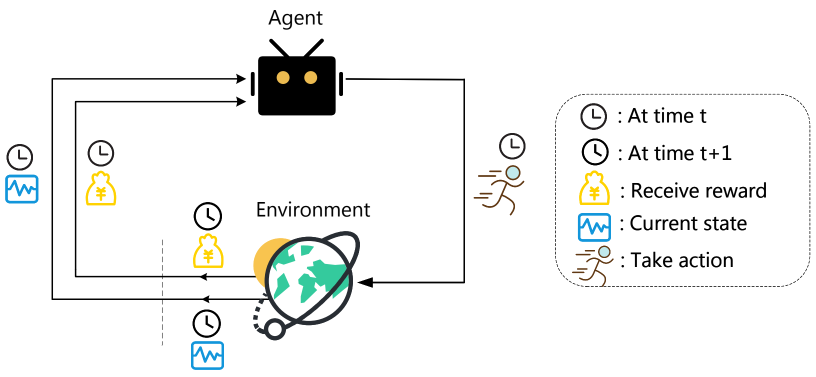


Figure 2: The agent interacts with the environment

2.2.1 Q-learning

The Q-learning algorithm [WD92], characterized by the decision-making process of the agent, is a value-based reinforcement learning algorithm. It utilizes a Q-table to store the expected rewards $Q(S_t, A_t)$ for a given state S_t, as shown in Figure 3. The update rule for the table is given by Equation 1. The term $\gamma \max_a Q(S_{t+1}, a)$ represents the maximum expected reward that the agent can obtain among all possible actions in state S_{t+1}. The discount factor γ is used to adjust the agent's focus on future rewards. When γ is set to 0, the agent only considers the immediate reward in the current state. When γ is set to 1, the agent takes into account the cumulative rewards for all steps until the end of the episode. The parameter α, referred to as the learning rate, dictates the extent to which the current experience should update the existing Q-values. A value of 0 means that this experience will not be learned from, while a value of 1 means that the new Q-value will completely replace the existing value in the table. The learning rate controls the extent to which the agent updates its Q-values based on new information gained from each experience. After multiple iterations, the Q-table will converge to a stable state. At this point, the agent can accomplish its intended goal. For example, in the case of a robot navigating a maze, the robot can use the Q-table to avoid traps and successfully reach the maze exit from the entrance. The converged Q-values in the table provide the necessary guidance for the agent to make optimal decisions and achieve its targets.

$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \alpha\left[R_{t+1} + \gamma \max_a Q\left(S_{t+1}, a\right) - Q\left(S_t, A_t\right)\right] \tag{1}$$

Action State	A_1	A_2	A_3	…
S_1	$Q(S_1, A_1) = -1$	$Q(S_1, A_2) = 5$	$Q(S_1, A_3) = 0$	…
S_2	$Q(S_2, A_1) = 0$	$Q(S_2, A_2) = -10$	$Q(S_2, A_3) = 1$	…
S_3	$Q(S_3, A_1) = 0$	$Q(S_3, A_2) = 0$	$Q(S_3, A_3) = 3$	…
…	…	…	…	…

Figure 3: A simple example of Q-table

2.2.2 Deep Q-Network

In the Q-learning algorithm, the storage cost and search time complexity of the Q-table can be quite high. To address this issue, DeepMind proposed a deep neural network to approximate the Q-table, which is known as the Deep Q-Network (DQN) algorithm [MKS+13]. In DQN algorithm, there are two ways to implement deep neural network: the first approach takes both the current state S_t and the action A_t as inputs and outputs the Q-value. However, when the action space is large, this approach can be computationally expensive due to the nonlinear mapping operations required for each candidate action. The second approach only takes the current state S_t as input and outputs the Q-values of all possible actions. This approach eliminates the candidate action space in the network's input layer, thereby reducing the computational cost of the nonlinear mapping operations. The trace segmentation problem investigated in this paper falls into a large candidate action space. Therefore, the latter approach is employed in our work.

Furthermore, this paper employs a dual-network training approach [MKS+15] to enhance training efficiency. Two Q-value networks, denoted as Q and $\hat{Q}$, are established. Initially, the Q and $\hat{Q}$ networks have the same structure and weights. Then, the Q network weights are iteratively updated using the formula $\alpha\left[R_{t+1} + \gamma \max_a Q\left(S_{t+1}, a\right) - Q\left(S_t, A_t\right)\right]$, while keeping the weights of the $\hat{Q}$ network unchanged. After several iterations of weight updates, weights of the Q network are assigned to the $\hat{Q}$ network. With multiple iterations, the network converges, resulting in an agent capable of achieving the desired targets.

2.2.3 Prioritized Experience Replay

During the interaction between the agent and the environment, the proportion of valuable experiences that are worth learning is relatively low. If only use random sampling to train the network, it would be inefficient. Therefore, we adopt the Prioritized Experience Replay (PER) method [SQAS16] to prioritize and emphasize a small number of valuable samples for learning by the agent. PER method assigns priorities to experiences and samples them based on their priorities during network training, rather than random sampling. Priority is calculated using the TD_{error}, where $TD_{\text{error}} = R_{t+1} + \gamma * max_a Q(S_{t+1}, a) - Q(S_t, A_t)$. A higher TD_{error} indicates that the current network's estimation of the Q-values for this experience is inaccurate and therefore the corresponding experience is more valuable for learning. By assigning higher priorities to experiences with larger TD_{error}, PER method focuses on experiences that provide the most learning potential and improves the learning efficiency of the network. These experiences are stored in an experience pool, and when new experiences are added to the pool, their priorities is initialized to the maximum priority value in the pool. This ensures that these experiences will be sampled in next training iteration. Additionally, a stochastic sampling strategy is introduced to prevent the long-term under-sampling of low-priority experiences, ultimately reducing diversity. This strategy ensures that the sampling frequency increases with priority, but still allows for occasional sampling of low-priority experiences.

3 Basic SPA-GPT

In this section, we present an overview of Basic SPA-GPT, followed by detailed explanations of its key elements.

3.1 Method

In this section, we illustrate our method for automatically segmenting traces of public-key algorithms using reinforcement learning, under the condition of only knowing the key lengths.

The first step involves setting the environment entity as the trace to be segmented. We establish the following correspondences between key concepts in reinforcement learning and the problem investigated in this paper:

- Action A_t: At time t, selecting an integer value from the range $[F, C]$ as the length of next trace segment, which represents the distance between the current segment point and the next segment point. In this paper, we define $F = L_{\text{trace}}/(2 * L_{\text{key}})$ and $C = L_{\text{trace}}/L_{\text{key}}$ to estimate the range of actions. L_{trace} represents the length of the trace, and L_{key} represents the length of the key. The reason why using this estimation for the action range will be further discussed in Section 3.2.4.

- State S_t: In this paper, we denote the trace to be segmented as T, where pos represents the current position of the agent. The trace segment $T[pos : pos + C]$ represents the state at time t, which serves as the input to DQN network. The purpose of setting state length as C is to allow the agent to have a comprehensive understanding of trace segment it can reach.

- Reward R_{t+1}: At time $t+1$, the agent receives a reward after taking an action at time t. Preliminary experiments have shown that the similarity between two complete cryptographic operation segments in a trace is significantly higher than the similarity between two randomly sampled trace segments. After comparing the effectiveness and computational cost of metrics such as manhattan distance, euclidean distance, and pearson correlation coefficient for measuring similarity, we define the reward as the average euclidean distance between the trace segment obtained after executing the current action and the previously obtained one.

The application of the above three concepts in our scenario is illustrated in Figure 4. Firstly, the DQN agent receives a fixed-length trace as its input state. Then, the agent selects a segmentation point on the trace based on a certain strategy. After one segmentation and receiving a reward, the agent moves to the new segmentation point, enters a new state, and repeats this process until the segmentation is complete.

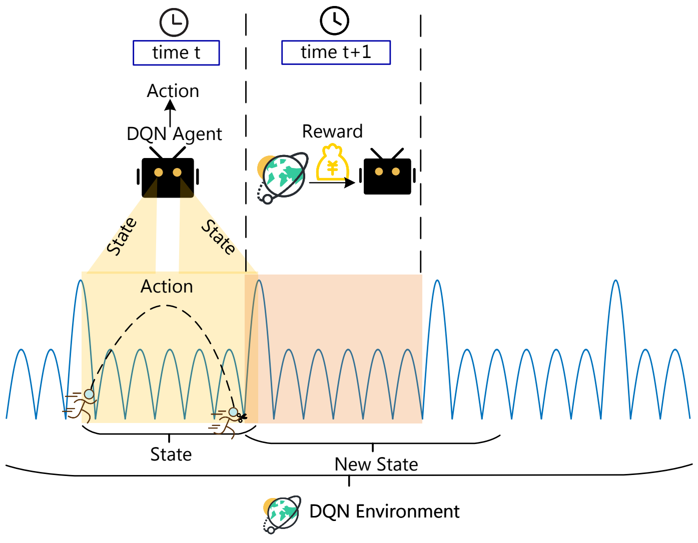


Figure 4: Illustration of three key concepts in reinforcement learning

The flowchart of the trace segmentation method based on reinforcement learning is shown in Figure 5. Please note that the data presented in the figure is for illustrative purposes only. At time t, the agent is located at position pos in a trace. First, the state S_t is inputted into the neural network, and the action with the highest Q-value is selected. In order to learn more diverse segmenting strategies, we employ the ϵ-greedy method, selecting an action with the maximum Q-value with a probability of ϵ, and selecting a random action with a probability of $1 - \epsilon$. Then, based on the reward value, different branches are executed: if the reward R_{t+1} is negative, indicating a failed segmentation, the agent enters the left branch, ends the current episode, and restarts from the beginning by setting $pos \leftarrow 0$ and $episode \leftarrow episode + 1$. Specifically, if there is a redundant part (non-cryptographic algorithm operation) at the end of the trace to be segmented, it can also result in a negative reward. Therefore, upon detecting a negative reward, the number of segmentation points in the current episode will be checked. If it falls within the valid range, this segmentation will be considered a feasible result and stored. Continue executing the aforementioned steps until the maximum number of episodes is reached, at which point the algorithm terminates. It is worth noting that in the traces of public-key algorithms, the segmentation points between two cryptographic operations are often not single points but intervals. Therefore, there can be multiple ways to segment a trace. When the numbers of episode reaches the predefined limit, the algorithm outputs all feasible segmentation results.

Remark 1. M is usually set to a relatively large value. This not only ensures correct segmentation for the majority of traces but also allows the algorithm to offer diverse segmentation options, providing multiple ways to exploit leakage in the subsequent steps of SPA.

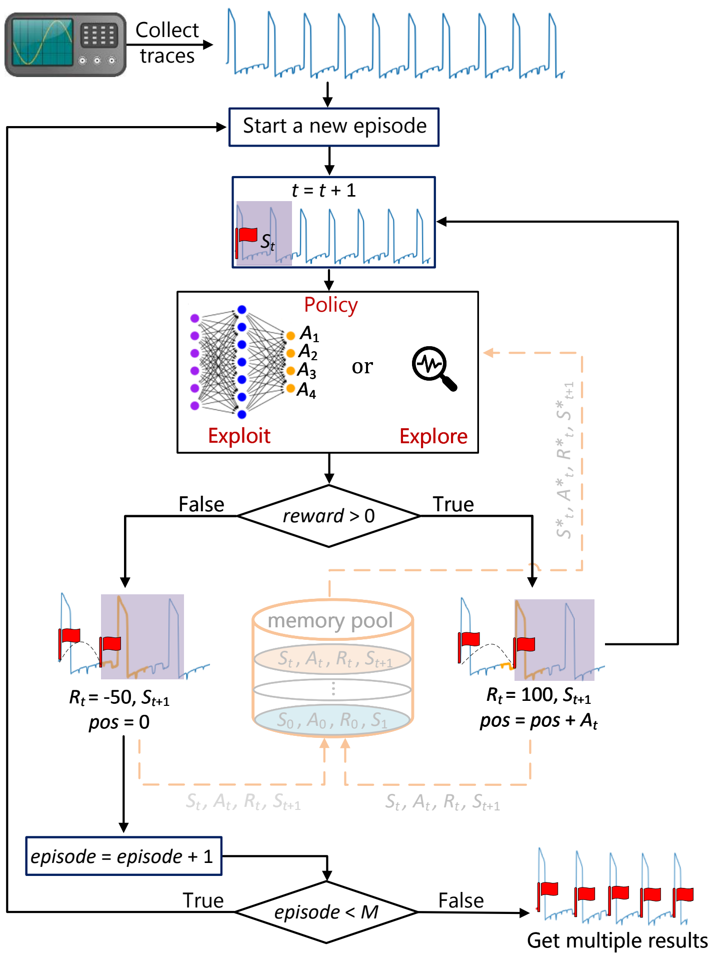


Figure 5: Overall workflow of Basic SPA-GPT based on reinforcement learning

3.2 Elements' Details of DQN

3.2.1 Environment Design

In addition to selecting the entity, the design of an environment also requires formulating reward rules, which include two aspects: distance calculation and the method of converting distance into reward values. Algorithm 1 presents the pseudocode of calculating the distance, denoted as **CalDistance**(s_{now},χ), where s_{now} represents the trace segment obtained after executing the action, and χ represents the set of already obtained trace segments. Due to the variable lengths of trace segments, we employ the second-order B-spline interpolation method [UAE93] to stretch two trace segments because the method requires the inputs have the same length. The target length is set to the longer value between the two trace segments. First of all, the algorithm applies the function **B-splineInterpolation**(s_{now}, s) to stretch each trace segment s in χ to the same length as s_{now}. Then, it uses the function **EuclideanDistance**(s_{now}, s) to calculate the euclidean distance l_k. Finally, it outputs the average of distances.

The sign of the reward value serves as the criterion to determine whether the current episode is terminated. In our work, the reward value of the current action is obtained by setting a threshold and calculating the difference between the threshold and the distance. The pseudocode of reward calculation is shown in Algorithm 2, denoted as

Algorithm 1: Distance calculation

```
Input: Current trace segment $s_{\text{now}}$, trace segments set $\chi$.
Output: Distance $d$.
1  $k \leftarrow 0$;
2  for $s \in \chi$ do
3      B-splineInterpolation($s_{\text{now}}, s$);
4      $l_k \leftarrow$ EuclideanDistance($s_{\text{now}}, s$);
5      $k \leftarrow k + 1$;
6  end
7  $d \leftarrow \sum_0^k l_i / k$;
8  return $d$;
```

GetReward($s_{\text{now}}, \chi, l_{\text{base}}$), where s_{now} represents a trace segment obtained after an action is executed, χ is a set of trace segments, and l_{base} (the determination of its value is explained in Section 3.2.3) is the threshold used to determine whether to terminate the current round and adjust the distribution of the reward. This adjustment ensures a balanced distribution of positive and negative rewards. In Algorithm 2, the first step is to calculate the distance d between the current trace segment and segments in χ using Algorithm 1. Then, the difference between l_{base} and d is computed to obtain the reward R_{t+1}. It is evident that the larger the distance, the smaller the reward. Next, based on the reward's sign, the variable *done* is assigned a value to indicate whether the current episode should be terminated. Finally, the segment is stored in χ for subsequent distance comparison and reward calculation. The output consists of the reward R_{t+1} and the termination signal *done*.

Algorithm 2: Reward calculation

```
Input: Current trace segment $s_{\text{now}}$, trace segments set $\chi$, threshold $l_{\text{base}}$.
Output: Reward $R_{t+1}$, termination signal $done$.
1  if $\chi = \emptyset$ then
2      $\chi$.append($t_{\text{now}}$);
3      return $done \leftarrow$ False, $R_{t+1} \leftarrow 0$;
4  end
5  $d \leftarrow$ CalDistance($t_{\text{now}}, \chi$);
6  $R_{t+1} \leftarrow (l_{\text{base}} - d)$;
7  if $R_{t+1} \geq 0$ then
8      $\chi$.append($t_{\text{now}}$);
9      $done \leftarrow$ False;
10 else
11     $done \leftarrow$ True;
12 end
13 return $done, R_{t+1}$;
```

3.2.2 Agent Design

We adopt a two-layer MLP network for agent. The number of neurons in the input layer is determined by the dimensionality of state s_t. The hidden layers consist of two dense layers with 40 neurons each. The number of neurons in the output layer is equal to the size of the action space, which is the value of $C - F$. Both the network Q and the network $\hat{Q}$ adopt this structure, and their initial weights are the same.

The pseudocode of the agent's action execution, as shown in Algorithm 3, is denoted as $\mathbf{Excute}(A_t, T, pos, s_{\text{now}}, \chi, l_{\text{base}})$. The agent performs an action A_t moving its position to the specified position *pos* on the trace T. Then, the agent receives a reward R_{t+1} with signal *done* and get into a new state S_{t+1}.

Algorithm 3: Agent performs the action

```
Input: Action $A_t$, trace $T$, current position of the agent $pos$, current trace
       segment $s_{\text{now}}$, trace segments set $\chi$, threshold $l_{\text{base}}$.
Output: Reward $R_{t+1}$, new position $pos$, a flag indicating whether to terminate
        the current episode $done$.
1 $s_{\text{now}} \leftarrow T[pos : pos + A_t]$;
2 $pos \leftarrow pos + A_t$;
3 $(R_{t+1}, done) \leftarrow \mathbf{GetReward}(s_{\text{now}}, \chi, l_{\text{base}})$;
4 $S_{t+1} \leftarrow T[pos : pos + C]$;
5 return $S_{t+1}, R_{t+1}, done, pos$;
```

3.2.3 Agent-Environment Interaction Design

The agent segments a trace within the environment, and the environment responds based on reward, presenting the agent with a new state. To initiate the segmentation process, the agent first determines a threshold l_{base} that is used to evaluate the correctness of segmentation. Once an appropriate l_{base} is obtained, the agent proceeds to segment the target trace. After completing multiple episodes of segmentation, the algorithm outputs all results.

Algorithm 4 describes the interaction between the agent and the environment requiring only the input of the target trace and key length. After initializing the variables using $\mathbf{Initial}(D, \theta, \epsilon, l_{\text{base}}, Res, t)$, the value of l_{base} is calculated as the euclidean distance between $T[0 : C]$ and $T[C : 2C]$. However, this estimation of l_{base} often tends to be overestimated. We employ the following mechanism to obtain a more appropriate value for l_{base} (Algorithm 4, lines 13-16): if the agent is able to continuously and randomly divide the trace into $\lceil 0.1 * L_{\text{key}} \rceil$ segments, indicating that a larger l_{base} allows the agent to receive positive rewards even in cases where the segmentation results in errors, we will multiply the current l_{base} by 0.9 to reduce its value and rerun the algorithm.

Once l_{base} is determined, the segmenting process begins at $t = 0$. The experience replay buffer D is cleared, and each new experience has the same priority. The network weights θ are initialized with random values. At each step, after the agent performs an action and receives feedback from the environment, the agent stores the current segmenting point in *Res* for that episode. Additionally, the experience $(S_t, A_t, R_{t+1}, S_{t+1})$ obtained at a step is stored in the experience replay buffer D. After a certain number of steps (in our work, 30000 steps), experience will be used to train network. To improve training efficiency, we set a training frequency of every 15 steps and the priority of experiences will be updated using TD_{error} obtained during training. Additionally, in selection of actions using the ϵ-greedy method, we adopt a dynamic adjustment strategy for ϵ. Initially, ϵ is set to 0, and as the trains, ϵ gradually increases (with a step size of 0.0001). Once ϵ reaches 0.9, it periodically resets to 1 to evaluate the fitting of the agent's network.

Finally, after M episodes of segmentation, the algorithm outputs all segmentation results stored in *Res* and terminates the execution.

3.2.4 Discussion on the Rationality of Action Space

In most implementations, the two operations (such as modular square and modular multiplication) have different execution times, which leads to varying lengths of trace

Algorithm 4: Trace segmentation using DQN

```
Input: Trace $T$, length of the key $L_{\text{key}}$.
Output: Reward $Res$.
1  Initial($D, \theta, \epsilon, l_{\text{base}}, Res, t$);
2  for $episode \in \{0, ..., M\}$ do
3      $pos \leftarrow 0$;
4      $Res[episode]$.append($pos$);
5      $\chi \leftarrow \emptyset$;
6      set $\epsilon \leftarrow 1$ for an episode periodically after $\epsilon$ reaches 0.9;
7      while True do
8          if $\epsilon > 0$ then
9              With probability $\epsilon$ select $A_t \leftarrow \mathbf{argmax}_a Q\left(S_t, A; \theta\right)$;
10         else
11             Select a random action $A_t$;
12         end
13         if $\lceil 0.1 * L_{\text{key}} \rceil$ times consecutive random selection then
14             $l_{\text{base}} \leftarrow l_{\text{base}} * 0.9$;
15             goto line 1;
16         end
17         $(S_{t+1}, R_{t+1}, done, pos) \leftarrow$ excute($A_t, T, pos, s_{\text{now}}, \chi, l_{\text{base}}$);
18         $Res[episode]$.append($pos$);
19         Store experience $(S_t, A_t, R_{t+1}, S_{t+1})$ in $D$ with the highest priority;
20         if $done$ then
21             break;
22         end
23         if $t > 30000$ and every 15 steps then
24             Train the network, update the priority of experiences;
25             if $\epsilon < 0.9$ then
26                 $\epsilon \leftarrow \epsilon + 0.0001$;
27             end
28         end
29         $t \leftarrow t + 1$;
30     end
31 end
32 return $Res$;
```

segments. So, a complete trace contains a number of valid operations in the range of $[L_{\text{key}}, 2L_{\text{key}}]$ (key = 0x000... or 0x111...). The average length of each trace segment falls within the range of $[\frac{L_{\text{trace}}}{2*L_{\text{key}}}, \frac{L_{\text{trace}}}{L_{\text{key}}}]$.

We illustrate four possible scenarios in Figure 6. Here, l_1 represents the segment of a short-duration operation, and l_2 represents the segment of a long-duration operation. The red flags indicate the correct segmentation points under different conditions, and the blue interval represents the available action range $[l_1, l_2]$. However, before executing the segmentation algorithm, the lengths of l_1 and l_2 are unknown. It is necessary to ensure that the action range covers the blue region, which satisfies Equation 2. Here, n_1 and n_2 represent the number of trace segments with lengths l_1 and l_2, respectively, and it should satisfy $L_{\text{key}} \leq n_1 + n_2 \leq 2L_{\text{key}}$, where $L_{\text{trace}} = n_1 l_1 + n_2 l_2$. The derivation leads to $\frac{l_2}{l_1} \leq 2 + \frac{n_1}{n_2}$. To utilize the average trace length for estimating the action range, it is required that the length of the longer trace segment does not exceed twice the length of the shorter one. In public-key algorithms implementations, this condition is easily satisfied. What is more, randomization serves solely as a means of leakage prevention and has minimal impact on the overall power consumption of the entire trace segment. Otherwise, it would adversely affect algorithm efficiency and increase implementation costs. Therefore, it is reasonable for this paper to use $F = \frac{L_{\text{trace}}}{2*L_{\text{key}}}$ and $C = \frac{L_{\text{trace}}}{L_{\text{key}}}$ to estimate the range of the action interval.

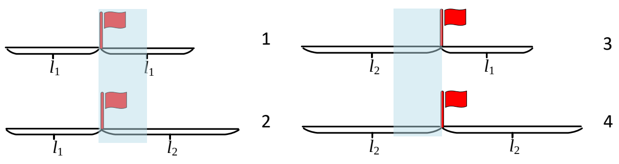


Figure 6: All possible combinations of two different length segments

We also provide the theoretically acceptable length of redundant part. Let L_R represent the length of the redundant part. Based on the above analysis, it is necessary to satisfy Equation 3 when the trace contains redundancy. We then derived Equation 4, which represents the valid proportion relationship between the redundant part and the decryption (or signature) part. If the proportion of the redundant part exceeds the upper limit specified in Equation 4, the action range will not include feasible segmentation points.

$$F = \frac{n_1 l_1 + n_2 l_2}{2 * L_{\text{key}}} \le l_1, \; C = \frac{n_1 l_1 + n_2 l_2}{L_{\text{key}}} \ge l_2 \tag{2}$$

$$F = \frac{n_1 l_1 + n_2 l_2 + L_R}{2 * L_{\text{key}}} \le l_1, \; C = \frac{n_1 l_1 + n_2 l_2 + L_R}{L_{\text{key}}} \ge l_2 \tag{3}$$

$$0 \le \frac{L_R}{L_{\text{trace}}} \le 1 - \frac{2}{\frac{n_1 l_1}{n_2 l_2} + 1} \tag{4}$$

4 Evaluation of Basic SPA-GPT

We have established an open-source power trace database[1] of public-key algorithms to validate our method. To avoid commercial disputes, we have only disclosed six traces, and we have not publicly released some traces from third-party products. Table 1 presents the key information about the traces in the database. For specific details regarding the experimental setup, please refer to the documentation in the database. Some details of the experimental configuration are omitted here. Part of the traces in the database were collected from a black box perspective. All traces have undergone low-pass filtering. For traces with excessive data points, we performed resampling to improve computational efficiency. Furthermore, some cryptographic devices execute special operations, such as precomputations, at the start of cryptographic calculations. As a result, there can be significant differences between the initial unstable portion and the stable portion that follow. We have addressed this problem by truncating the unstable portion at the beginning. Alternatively, it is also possible to make the trace become stable through specialized processing.

[1]Available at https://github.com/pilipili520/SPA-GPT

Table 1: The power trace database of public-key algorithms

Algorithm	L_{key}	Operations Number	Device	Implementation	Disclosure
RSA	1024	1562	smart card	co-design	Yes
RSA	1024	1536	ASIC X	hardware	Yes
RSA	1024	1531	SAKURA-G	hardware	Yes
RSA-RD[a]	1024	1531	SAKURA-G	hardware	Yes
RSA	1024	1517	ASIC Y	co-design	No
RSA	1024	1535	STM32F429	software	Yes
RSA-CRT	1024	1496	USB Key	co-design	No
ECC[b]	128	192	AT89S52	software	Yes
ECC	256	372	smart card	co-design	No

[a]We have incorporated random delay method in RSA and call it RSA-RD.
[b]This is a toy implementation due to limited memory and computational resources on the AT89S52.

4.1 Evaluation on Different Platforms

We applied our method to all traces in the power trace database, but did not achieve satisfactory results for all of them. Basic SPA-GPT demonstrated good segmentation performance on the traces of RSA on SAKURA-G, RSA-CRT on USB Key, and RSA on ASIC X, which we present in this section (The last one has been included in the appendix A). The remaining traces were processed using Enhanced SPA-GPT to improve the segmentation results, as described in Section 6.

The parameter configuration of the DQN used in our experiment is shown in Table 2. To ensure compatibility with most traces, we set the parameters to relatively large values.

Table 2: The key parameter settings of DQN in experiments

ID	Parameter	Value
1	learning rate	0.01
2	batch size	128
3	reward decay	0.01
4	memory pool size	5000
5	maximum episodes M	1000000

4.1.1 Hardware RSA on SAKURA-G

We implemented RSA on the SAKURA-G, and the power trace generated during the execution of the algorithm are shown in Figure 7. For this trace, we are aware of its characteristics and distinguishing features. The significant power consumption in the depicted portion corresponds to RSA decryption, while the less significant power consumption represents redundancy. After zooming in on the trace during the decryption phase, we observed periodic spikes with regular patterns. Each cryptographic operation corresponds to the interval between two prominent peaks.

We applied our method to segment this trace, and the results are shown in Figure 8. The red dots indicate the segmentation positions. When we zoom in on the decrypted traces, it is evident that each cryptographic operation has been accurately segmented and preserves the significant "peaks" between them (which can be used to distinguish different operations). We have marked the "Termination Point" on this trace, indicating the position where the segmentation end. According to the termination condition, it is known that the last segment is not valid. Therefore, unless otherwise specified in subsequent experiments,

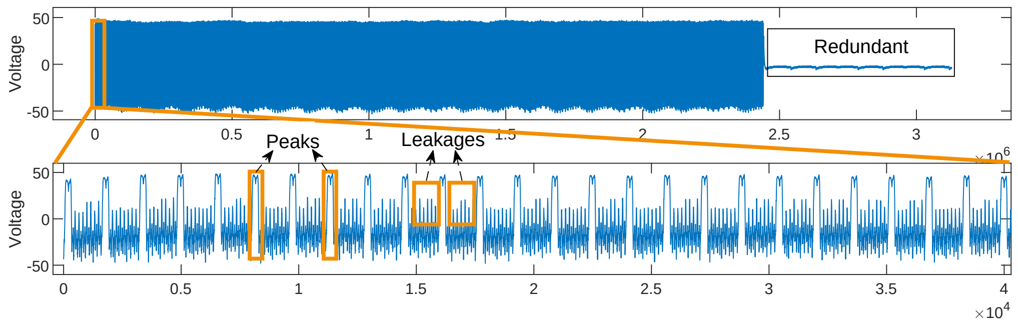


Figure 7: Power trace of RSA decryption on SAKURA-G

we will hide these points in other traces. Using horizontal analysis, we can fully recover the correct private key based on this segmentation.

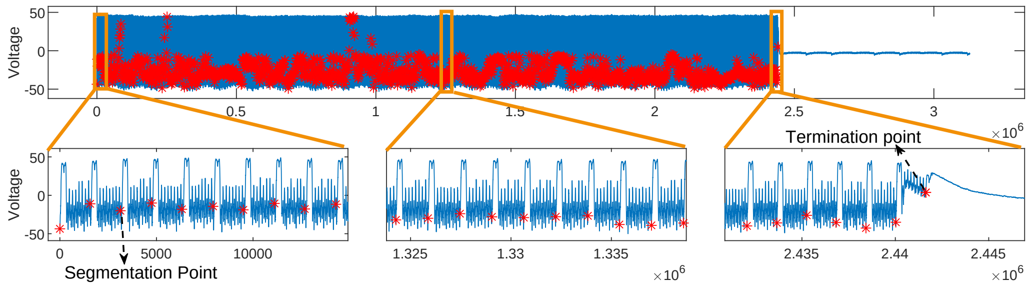


Figure 8: Power trace segmentation result of RSA decryption on SAKURA-G

In multiple experiments, we found that some segmentation results mistakenly identified the redundant portion as decryption subtrace and performed segmentation on them. In Section 5, we will present improvements to address this issue.

As a comparative experiment, we conducted equidistant segmentation on the same trace. Figure 9 illustrates the results of equidistant segmentation, highlighting the method's inability to accurately segment the trace. This failure can be attributed to slight variations in the lengths of different operations, which accumulate over time, causing significant deviations in the later segments of the trace from their correct positions.

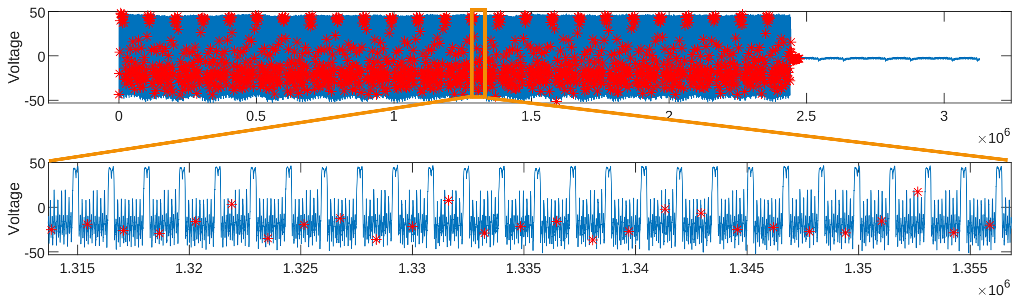


Figure 9: Erroneous result from equidistant segmentation method

In addition, we have heightened the segmentation difficulty in this experiment by introducing random delay protection into the decryption algorithm's implementation.

Even with the inclusion of random delays, maintaining time efficiency remains paramount. Therefore, the length of the trace segment typically does not exceed the action range discussed in Section 3.2.4. Our experimental results demonstrate good robustness, even when dealing with traces containing random delays, as depicted in Figure 10.

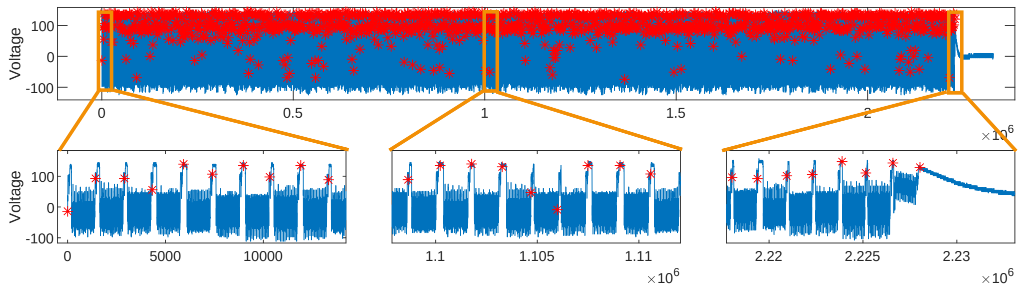


Figure 10: Power trace segmentation result of RSA decryption on SAKURA-G containing random delays

4.1.2 Co-design RSA-CRT on USB Key

This trace was collected while running RSA-CRT on a USB Key developed by a company and we do not know its segmentation features. We use the subtrace during the calculation of d_p as an example for segmentation, as shown in Figure 11. The trace exhibits several repetitive segments, but the exact repetition pattern is uncertain. Therefore, all the repetitive segments could potentially represent a single cryptographic operation. For example, in Figure 11, the highlighted yellow boxes indicate two possible patterns.

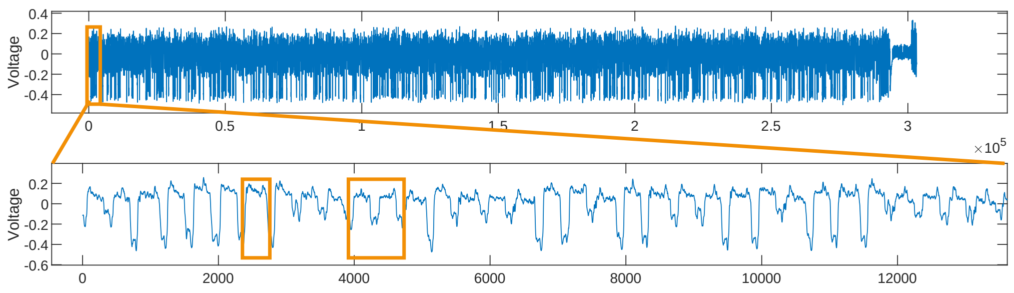


Figure 11: Power trace of RSA-CRT signature on USB Key

After applying our method to segment this trace, we obtained the segmentation results shown in Figure 12. The number of segments falls within a reasonable range. After analyzing the results, we observed variations in the lengths of these trace segments, indicating the timing leakage between different operations. As shown in Figure 12, “S” represents the modular square operation, and “M” represents modular multiplication operation. Utilizing this leakage information, we can fully recover the sequence of operations.

The peaks on this trace and the differences in distance between them are noticeable, making it challenging to identify specific peaks or distance as reliable segmentation criteria. We attempted to use specified peak height and minimum distance between them (peakdistance) for segmentation. Despite multiple parameter adjustments, some operations could not be correctly segmented, and redundant portion were also segmented, as shown in Figure 13.

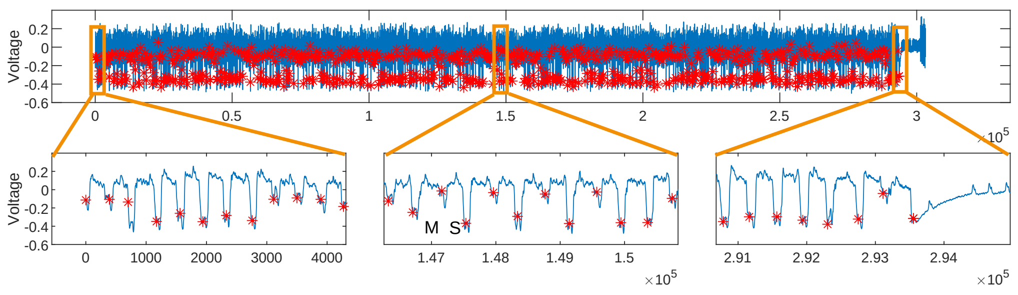


Figure 12: Power trace segmentation result of RSA-CRT signature on USB Key

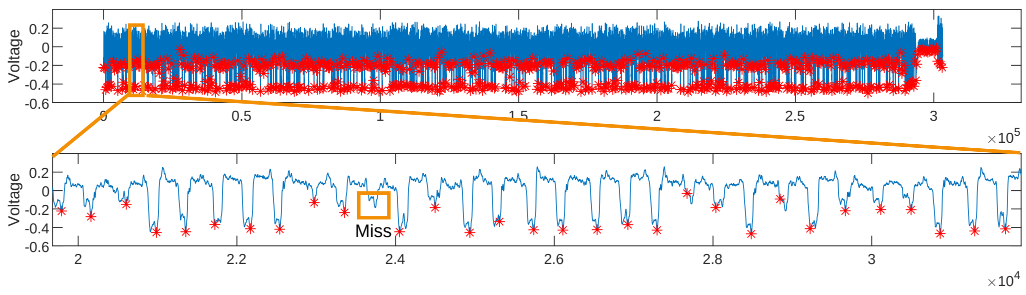


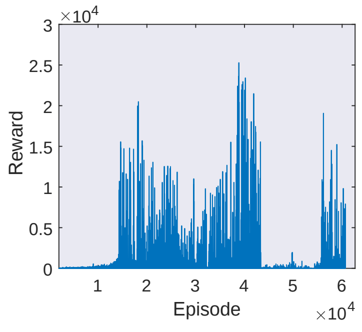
Figure 13: Error encountered in segmentation using peakdistance method

4.2 Learning Process of The Agent

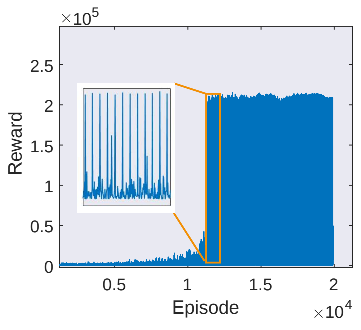
We collected the reward of the agent during the learning process in previous section. Due to the different characteristics of each trace, the number of episodes required for the agent to mature varies as depicted in the Figure 14. During the early episodes, the agent obtained low cumulative rewards because it was in the phase of gathering experience and executing many random actions. Some of these actions may have resulted in premature termination of the segmentation process. However, as the network is trained, it gradually achieved higher cumulative rewards in the later episodes. But the total reward obtained by the agent did not stabilize in a higher range and exhibited significant fluctuations. Even after network convergence, the agent could still receive low rewards. This is because the maximum value of ϵ was set to 0.9, allowing the agent to continue taking random actions to explore multiple segmentation approaches.

It is worth noting that the episode with the highest total reward in Figure 14(a) (38000-40000) does not correspond to the best segmentation result. This is because the agent discovered a way to include redundant portion in the segmentation process. What is more, due to the randomness introduced by ϵ, the rewards obtained by the agent in later episodes did not stabilize within a specific range, as shown in Figure 14(b). However, the highest reward values remained stable at higher levels, and these episodes demonstrated accurate segmentation outcomes.

We also collected the loss during the learning process of the agent, as shown in Figure 15. Since reinforcement learning is not supervised learning, the experiences fed into the agent's network are autonomously collected. As the learning progresses, new experiences are gradually accumulated. Therefore, the loss curve does not decrease smoothly. From Figure 15(a), it can be observed that the agent has difficulty learning the segmentation features (the learning curve shows significant fluctuations), which can be attributed to the incorrect experiences obtained during the segmentation of redundancies. In Figure 15(b), it is evident that the agent quickly learns the segmentation features (rapid decrease in loss), which could be attributed to the fewer number of operations and distinct characteristics of the trace segments.

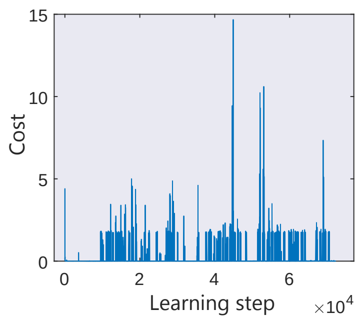


(a) RSA on SAKURA-G

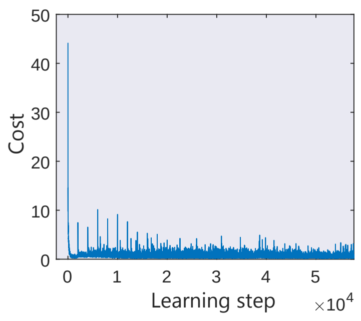


(b) RSA-CRT on USB Key

Figure 14: The total reward obtained by the agent of each episode



(a) RSA on SAKURA-G



(b) RSA-CRT on USB Key

Figure 15: The Neural Network cost of each learning step

5 Enhanced SPA-GPT

Basic SPA-GPT demonstrates good performance on many traces, indicating a certain level of versatility. Nevertheless, it does come with its limitations. To address these shortcomings, we upgrade it to enhance segmentation performance, as follows:

- Assigning all trace segments into the same set can waste features. We optimize the trace segments allocation method using clustering algorithms, which enhances segmentation efficiency and allows the recovery of operation sequences for some traces.
- Using only the euclidean distance threshold sometimes fails to accurately determine the end position of cryptographic. We propose a new termination criterion by integrating trace variance features to address this issue.
- Some traces are heavily affected by strong noise interference, and conventional filtering techniques fail to yield satisfactory results. Therefore, we extract the envelopes before performing segmentation.

- The segmentation results exhibit certain biases in some cases. We use fine-tuning method to optimize the outcomes.

5.1 Enhanced Environment

The basic method described in Section 3 places all operations into a single set and then calculates the average distance. This calculation is essentially an “average denoising” applied to different parts of the trace segments. It utilizes only the common features among the trace segments for identification. However, most public-key algorithms consist of at least two operations. For trace segments that exhibit significant differences between these different operations, it can affect the segmentation accuracy. Therefore, we divided the trace segments into two separate sets based on the characteristics of the cryptographic algorithm.

We utilize a state machine, as depicted in Figure 16, to illustrate the process of assigning segments to two sets. In the state machine, E_0 represents the state where both sets are empty. E_1 represents the state where set χ_1 is non-empty while χ_2 is empty. E_2 represents the state where both sets are non-empty. d_1 and d_2 represent the average distances between the trace segment to be assigned and the trace segments in the two sets, respectively. Firstly, both d_1 and d_2 are set to 0, and both sets are empty. When the agent performs its first segmentation and obtains an initial trace segment, the environment will directly place it into set χ_1 (i.e., setting d_1 to 0). At this point, the state of the sets becomes E_1. Subsequently, the agent carries out further segmentations. If a trace segment that satisfies $d_1 \leq l_{\text{base}}$, the environment will place it into set χ_1 and stay at E_1. However, if a trace segment that satisfies $d_1 > l_{\text{base}}$, the environment will assign this segment to set χ_2, and the state transitions to E_2. In E_2 state, the environment assigns the trace segment to the set corresponding to $\mathbf{min}\,(d_1, d_2)$. When a trace segment that satisfies $d_1 > l_{\text{base}}$ and $d_2 > l_{\text{base}}$, the environment terminates the current episode and resets both sets to empty, returning to the E_0 state.

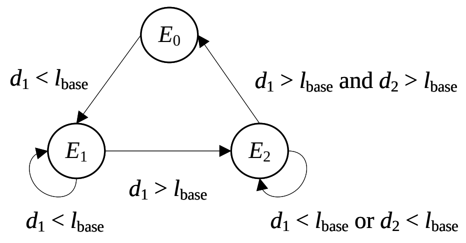

E_0
E_1
E_2
$d_1 < l_{\text{base}}$
$d_1 > l_{\text{base}}$ and $d_2 > l_{\text{base}}$
$d_1 > l_{\text{base}}$
$d_1 < l_{\text{base}}$
$d_1 < l_{\text{base}}$ or $d_2 < l_{\text{base}}$

Figure 16: The state machine of assigning the trace segments into two sets

However, in order to ensure that the initially immature agent gradually accumulates sufficient experience, the threshold value l_{base} is set to be lenient. Consequently, the environment assigns all trace segments to set χ_1 and maintains the E_1 state throughout, resulting in suboptimal segmentation. To address this issue, we introduce a clustering algorithm to assist with segment classification. After 30000 steps, we use the clustering algorithm to reassign the trace segments between the two sets. The pseudocode of this process is presented in Algorithm 5.

For cases where there are significant differences in the trace segments corresponding to cryptographic operations, the process of assigning segments to different sets is equivalent to clustering of segments. The different sets correspond to labels, and based on these labels, we can obtain the operation sequences corresponding to the private key, then recovering it. To improve the quality of trace segmentation and the effectiveness segment classification for cases where leakage intervals are not clearly visible, we utilize the SOST method [PC15]

Algorithm 5: Refined reward calculation and determination

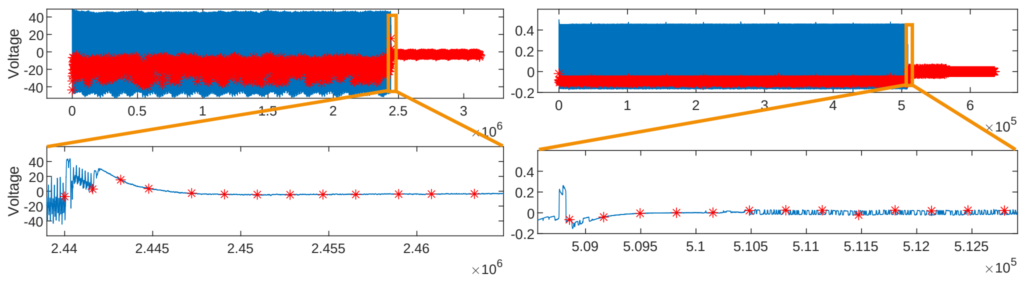
```
Input: Current trace segment $s_{\text{now}}$, two trace segment sets $\chi_1$ and $\chi_2$, threshold $l_{\text{base}}$.
Output: Reward $R_{t+1}$, terminate signal $done$.
1  if $\chi_1 = \emptyset$ and $\chi_2 = \emptyset$ then
2      $\chi_1$.append($t_{\text{now}}$);
3      return $R_{t+1} \leftarrow 0, done \leftarrow$ False;
4  end
5  $d_1 =$ CalDistance($t_{\text{now}}, \chi_1$);
6  if $\chi_2 = \emptyset$ and $l_{\text{base}} \geq d_1$ then
7      $\chi_1$.append($t_{\text{now}}$);
8  else if $\chi_2 = \emptyset$ and $l_{base} < d_1$ then
9      $\chi_2$.append($t_{\text{now}}$);
10 else
11     $d_2 \leftarrow$ CalDistance($t_{\text{now}}, \chi_2$);
12     if $d_1 < d_2$ and $d_1 \leq l_{\text{base}}$ then
13         $R_{t+1} \leftarrow l_{\text{base}} - d_1$;
14         $\chi_1$.append($t_{\text{now}}$);
15     else if $d_1 > d_2$ and $l_{\text{base}} \geq d_2$ then
16         $R_{t+1} \leftarrow l_{\text{base}} - d_2$;
17         $\chi_2$.append($t_{\text{now}}$);
18     else
19         $R_{t+1} \leftarrow l_{\text{base}} - (d_1 + d_2)$;
20         $done \leftarrow$ True;
21     end
22 end
23 return $R_{t+1} \leftarrow 0, done \leftarrow$ False;
```

to locate the leakage intervals. We only perform distance calculations and trace segments reassignment based on these identified intervals.

When using the threshold obtained solely from euclidean distance calculation, there are cases, as shown in Figure 17, where the agent segments the redundant portions in order to obtain more rewards. However, this leads to the agent learning incorrect experiences. Upon thorough observation of the majority of power traces, a clear discrepancy in variance between the cryptographic operation intervals and the redundant portions has been identified. Since this work assumes that the starting position of the public-key algorithm on the trace is known, the initial state S_0 can reflect the variance during the cryptographic operation. We calculate the variance of S_0 and set it as the threshold. Subsequently, we calculate the variance of each step's state S_t by the agent. If the variance is in a different order of magnitude than the threshold, the reward for this step is set to negative, and the current episode is terminated. This ensures that the segmenting termination point aligns with a completion of the cryptographic algorithm.

5.2 Envelope-based Feature Enhancement

Due to the noise in the original power traces, it is common to apply filtering before analysis. Figure 18 shows the result of applying a low-pass filter to the power trace of ASIC Y. However, some segmentation features of the trace, such as sharp spikes, may overlap in frequency with the noise signal. If the cutoff frequency of the filter is lowered further, it will weaken the segmentation features, as depicted by the red curve in Figure 19. If the cutoff frequency of the filter is not reduced, it will lead to the trace still being influenced



(a) RSA on SAKURA-G **(b)** RSA on smart card

Figure 17: Performing segmentation without variance threshold

by noise, making it difficult to determine l_{base} and causing the state space to increase. Consequently, the agent would need to expend a considerable amount of time gathering experience. To remove noise signals while preserving segmentation features and enhancing the learning speed of the agent, we employ a peak detection method to extract the lower envelope of the trace. This involves specifying a window size and sliding it forward with a fixed step size. The local minima within the window are selected as target points on the envelope, and the curve between these points is interpolated using a third-order B-spline interpolation method to make it as long as the filtered trace. The resulting envelope is depicted by the blue curve in Figure 19.

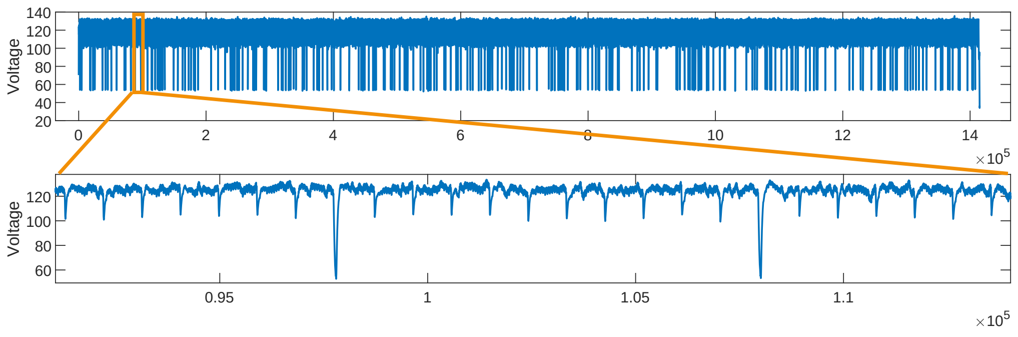


Figure 18: Power trace of RSA decryption on ASIC Y

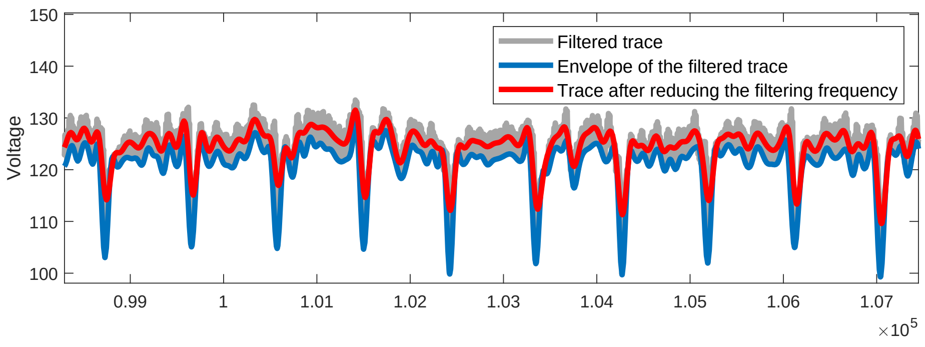


Figure 19: Comparison of envelope and low pass filtering frequencies

5.3 Fine-tuning Method

In order to provide more accurate trace segment for subsequent analysis, we fine tune the segmentation results. We use a specific segmentation point as a “Pole”, and allow the remaining segmentation points to slide within a certain time range to align their vertical coordinates with the “Pole” as closely as possible. If it is not possible to adjust the points to the same voltage within an appropriate range, the segmentation points are kept in their original positions, as illustrated in Figure 20.

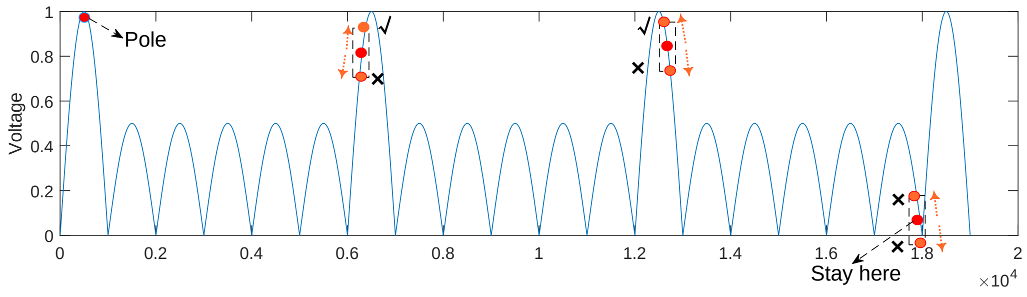


Figure 20: Adjustment rules for segmentation points during fine-tuning process

After preprocessing, some traces may lose leakage information, which can impact feature extraction. Therefore, it is necessary to map the segmented results of the preprocessed traces back to the original traces. However, the envelope method can expand the segmentable range, leading to deviations in the positions of the mapped segmentation points on its original trace. Similarly, fine-tuning method are required to rectify these deviations.

6 Evaluation of Enhanced SPA-GPT

In this section, we evaluate Enhanced SPA-GPT on the traces of ECC on smart card, RSA on smart card, RSA on ASIC Y, STM32F429, and ECC on AT89S52. The detailed processes of the last two experiments are presented in the appendix B and C.

6.1 Experimental Validation of the Enhanced Environment

We applied the assigning the segments into two sets method from the enhanced environment to the power traces of ECC on smart card, ECC on AT89S52, and RSA on STM32F429. Additionally, we applied the whole improvements from the enhanced environment to segment the power trace of RSA on smart card.

6.1.1 Software ECC on smart card

The ECC decryption is running on a smart card, and the power trace we collected are shown in Figure 21. After zooming in on a local region of this trace, we initially speculate that the downward spikes serve as segmentation feature between two operations. Moreover, segmenting it based on these feature reveals two distinct categories of trace segments with noticeable differences.

Figure 22(a) illustrates the experimental results without using clustering algorithm for reassignment. In this case, the environment predominantly assigns most of the trace segments to the same set, resulting in suboptimal segmentation. Figure 22(b) presents the results after incorporating the clustering reassignment mechanism. To attain higher rewards, the agent selects segmentation points within the region of the spikes, thereby

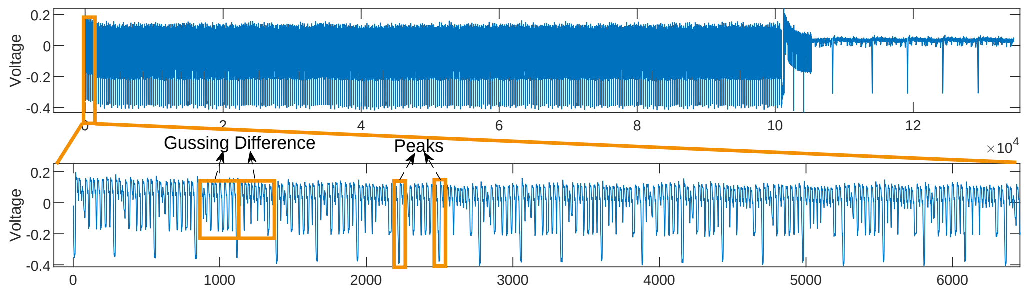
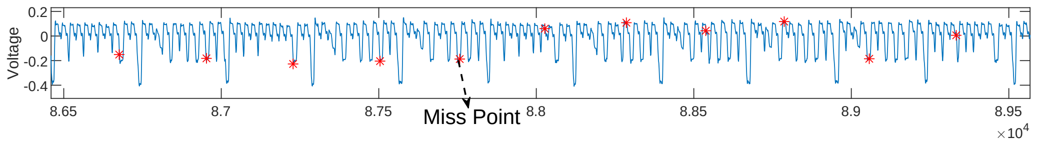
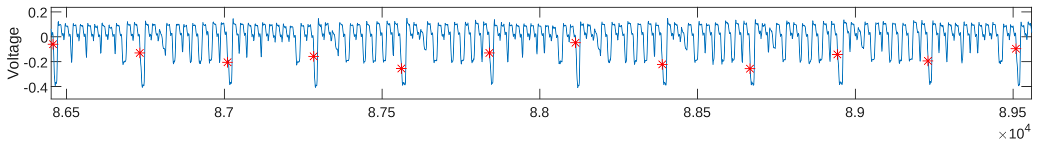


Figure 21: Power trace of ECC decryption on smart card

reducing the euclidean distance between trace segments within the same set and increasing the differentiation between different sets.



(a) Without segments reassignment method



(b) With segments reassignment method

Figure 22: Comparison of segmentation results with and without segments reassignment method

The complete segmentation and operations recovery result for this trace are shown in Figure 23. The black segments represent doubling point operations, the blue ones indicate point addition operations, and the gray portions represent redundant parts. The number of trace segments falls within a reasonable range, and the entire operation sequence is correctly recovered.

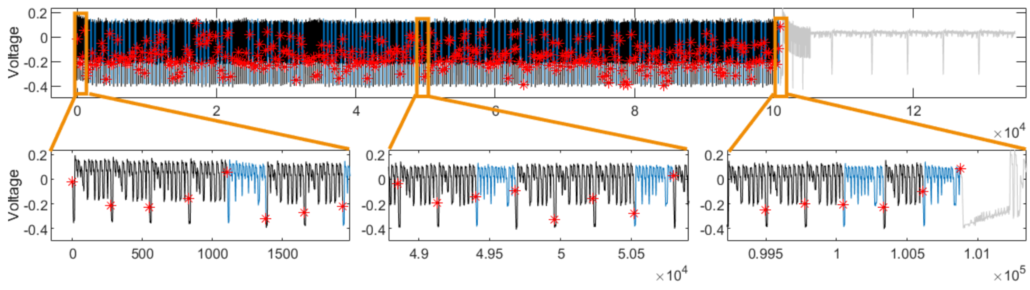


Figure 23: Power trace segmentation and operations recovery result of ECC on smart card

6.1.2 Co-design RSA on smart card

We have implemented the RSA decryption based on Montgomery modular exponentiation on the smart card platform, and the power trace generated during its execution are shown in Figure 24. Upon zooming in, we observed that the large upward spikes can serve as indicators for segmenting, and the features within the yellow boxes in the figure can be utilized to distinguish between the two types of operations. However, due to the redundancy in this trace and the inability to use euclidean distance threshold to accurately terminate the segmentation, we used the variance threshold to identify redundant portion at the end.

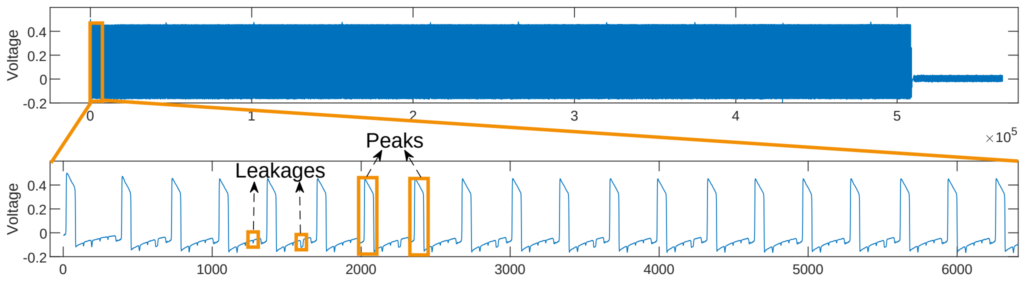


Figure 24: Power trace of RSA decryption on smart card

During the segmentation process, the leakage positions identified using the SOST method are shown in Figure 25. Based on this leakage interval, we perform the reassignment of trace segments.

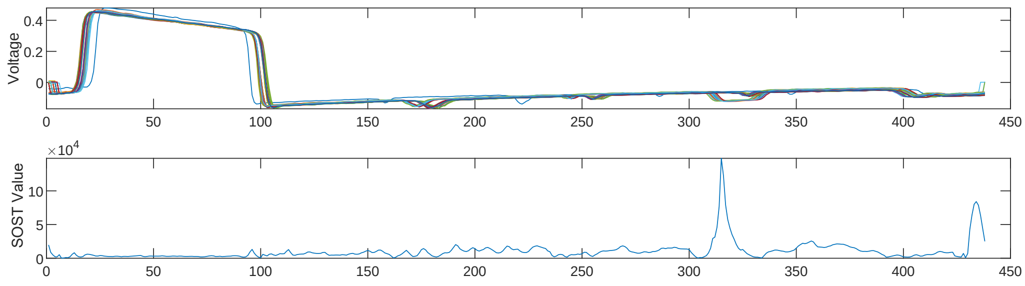


Figure 25: Result of SOST method obtained during the segmentation process

We obtained the segmentation and operation sequence recovery results as shown in Figure 26. The black segments represent the segments corresponding to square operations, the blue ones represent the segments corresponding to multiplication operations, the red ones represent correctly segmented but incorrectly recovered operations, the gray portion represents redundancy. Through segmentation, we achieved an operations recovery accuracy of 99.6%, excluding the last two operations caused by Montgomery reduction, which can be handled as a special case during the key recovery phase. From this figure, it can be observed that the leakage positions of the incorrectly recovered trace segments are different from the others. We speculate that this discrepancy may be due to the electrical characteristics of the smart card.

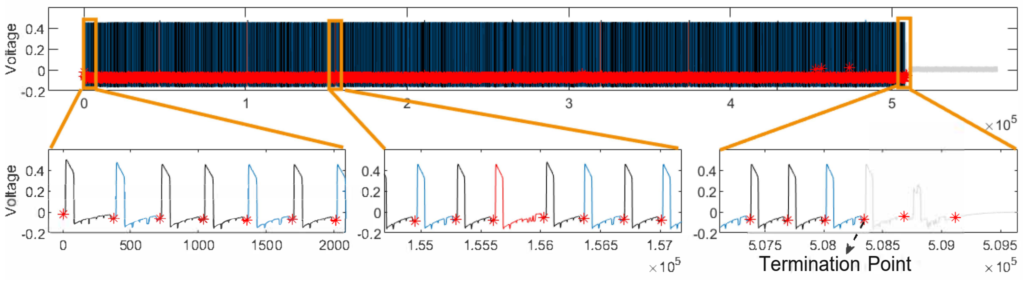


Figure 26: Power trace segmentation and operations recovery result of RSA on smart card

6.2 Experimental Validation of the Envelope-based Feature Enhancement

We collected a power trace of ASIC Y running the RSA and used the envelope-based feature enhancement method to perform the segmentation. In this case, we specified that all trace segments were assigned to the same set. After obtaining the segmentation points of the operations, we mapped these points back to the original power trace, resulting in Figure 27. The number of trace segments obtained after running this method is valid. But, this trace does not exhibit obvious leakage, making it difficult to effectively distinguish between operation categories. Nonetheless, the individual operations differentiated using this method can still be useful for assisting subsequent analysis.

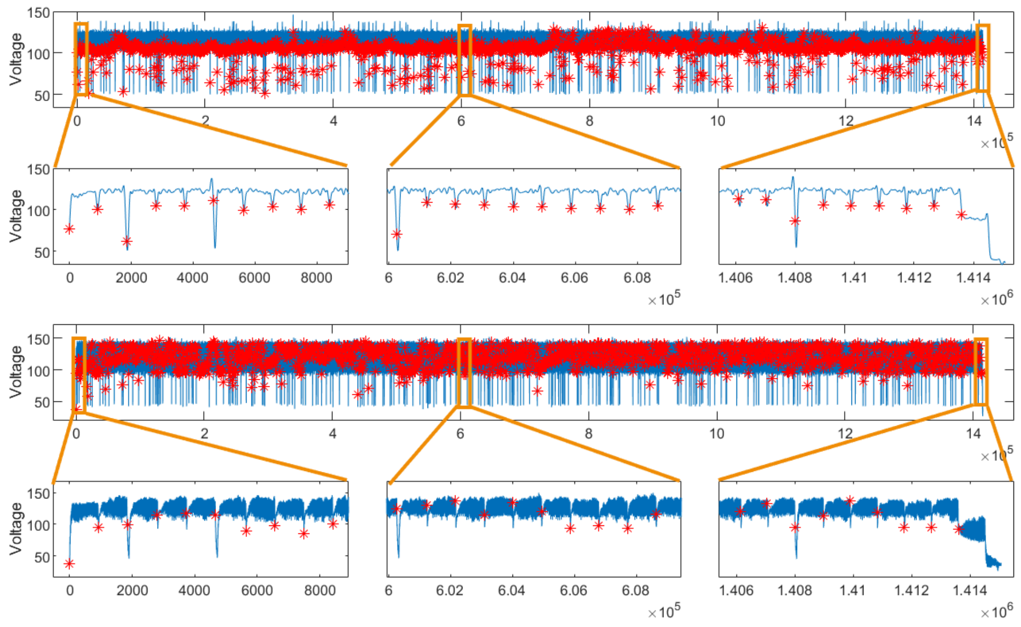
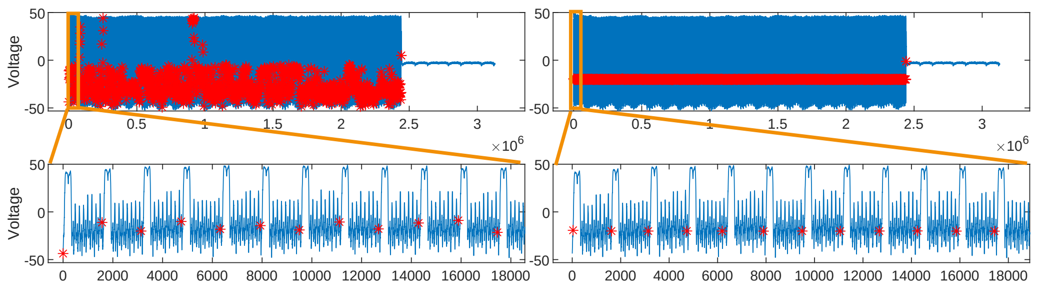


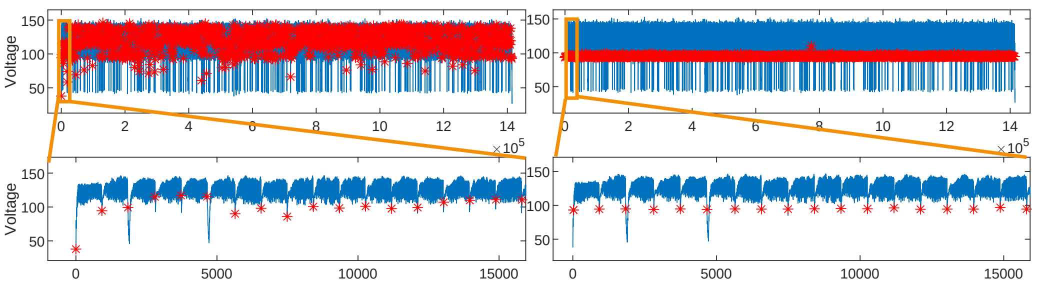
Figure 27: Power trace segmentation and mapping result of RSA on ASIC Y

6.3 Experimental Validation of the Fine Tuning

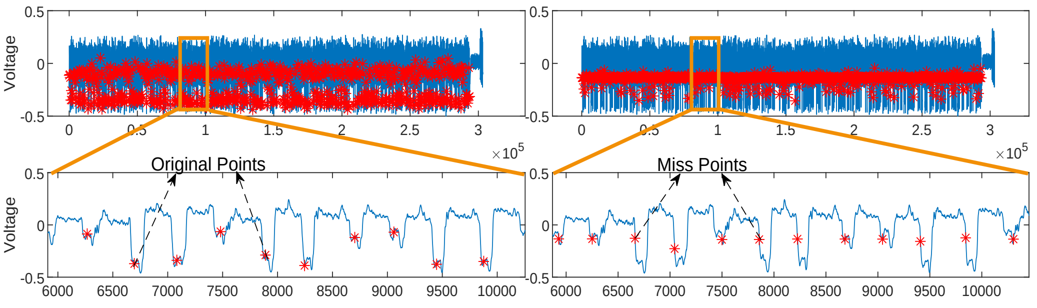
We applied the fine-tuning method to the segmentation results of all power traces, achieving similar results as shown in Figure 28(a). The left figure displays the original segmentation, while the right figure shows the results after fine-tuning. As shown in Figure 28(b), fine-tuning method effectively improves the issue of segmentation point position offset that arises when mapping the segmentation points back to the original trace. It can be observed that after fine-tuning, the noise parts outside of the leakage are reduced, which is beneficial for subsequent analysis. However, it is worth noting that for the trace of RSA-CRT on USB Key, which have significant amplitude variations, fine-tuning is not suitable. This can result in the issue depicted in Figure 28(c), where the segmentation points fail to get a complete trace segment.



(a) RSA on SAKURA-G



(b) RSA on ASIC Y



(c) RSA-CRT on USB Key

Figure 28: Segmentation results after processing with fine-tuning method

6.4 Summary of Experimental Results

We summarized all experimental results in Table 3. When there is clear periodicity and minimal variation between different trace segments, the segmentation task can be effectively accomplished using Basic SPA-GPT. However, when leakage is present and operations can be divided into two categories, employing the enhancement methods in Enhanced SPA-GPT can yield higher-quality segmentation results. This allows for both the completion of the segmentation task and the recovery of the private key operation sequence.

Table 3: Experimental Results

Trace	Equidistant	Findpeak	Peakdistance	Basic SPA-GPT	Enhanced SPA-GPT [a]
RSA on smart card	Fail	Success	Success	Success	Success (Acc_{op} : 99.6%)
RSA on AISC X	Fail	Success	Success	Success	NULL
RSA on SAKURA-G	Fail	Fail	Success	Success	NULL
RSA (random delay) on SAKURA-G	Fail	Fail	Success	Success	NULL
RSA on ASIC Y	Fail	Fail	Success	Fail	Success
RSA on STM32F429	Fail	Fail	Fail	Fail	Success (Acc_{op} : 100%)
RSA-CRT on USB Key	Fail	Fail	Fail	Success	NULL
ECC on AT89S52	Fail	Fail	Fail	Fail	Success (Acc_{op} : 100%)
ECC on smart card	Fail	Fail	Success	Fail	Success (Acc_{op} : 100%)

[a] Acc_{op} represents the average accuracy for recovering operations using assigning the segments into two sets method from the enhanced environment in Enhanced SPA-GPT.

7 Conclusion and Discussion

When collecting power traces of an unknown device, if the key length is known, our method can be used for preliminary analysis. Thanks to advancements in artificial intelligence algorithms and the improvement of computational resources, the agent in reinforcement learning can discover more diverse segmentation approaches than manual experience. This provides additional insights for subsequent analysis. In combination with horizontal analysis, our method can directly yield the operation sequence corresponding to the private key. This approach demonstrates strong robustness and performs well on many traces, making it applicable to most of power traces from general cryptographic devices.

Currently, this work is only applied to power traces of public-key algorithms, but it is also applicable to traces of block ciphers. Researchers in this community can customize the environment to better adapt to the target traces. This can be done by modifying rewards, adding state features, or selecting clustering algorithms that better suit the dataset distribution. This method calculates the action range and threshold automatically. However, if the sizes of individual segments are known, one can manually calculate the action range and threshold to further improve the segmentation efficiency. For some traces, the segmentable range may be larger, and the agent may choose points within the action range that are not suitable as segmentation points. But, it can still receive high rewards. To improve the segmentation results for such traces, better preprocessing methods are required.

In the future, we plan to develop a visualization tool based on this method to demonstrate the agent's segmentation process. This tool will greatly enhance the efficiency of analyzing public-key algorithms.

References

[AK22] Anil Berk Altuner and Zeynep Hilal Kilimci. A novel deep reinforcement learning based stock price prediction using knowledge graph and community aware sentiments. *Turkish J. Electr. Eng. Comput. Sci.*, 30(4):1506–1524, 2022.

[BBGV16] Arthur Beckers, Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. Design and implementation of a waveform-matching based triggering system. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, volume 9689 of *Lecture Notes in Computer Science*, pages 184–198. Springer, 2016.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BG98] Suresh Balakrishnama and Aravind Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*, 18(1998):1–8, 1998.

[Bog07] Andrey Bogdanov. Improved side-channel collision attacks on AES. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2007.

[CFG$^+$10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In Miguel Soriano, Sihan Qing, and Javier López, editors, *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, volume 6476 of *Lecture Notes in Computer Science*, pages 46–61. Springer, 2010.

[CLH19] Valence Cristiani, Maxime Lecomte, and Thomas Hiscock. A bit-level approach to side channel based disassembling. In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications - 18th International Conference, CARDIS 2019, Prague, Czech Republic, November 11-13, 2019, Revised Selected Papers*, volume 11833 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2019.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International*

Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[GGJR+11] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST non-invasive attack testing workshop*, volume 7, pages 115–136, 2011.

[GV10] Christophe Giraud and Vincent Verneuil. Atomicity improvement for elliptic curve scalar multiplication. *CoRR*, abs/1002.4569, 2010.

[HIM+13] Johann Heyszl, Andreas Ibing, Stefan Mangard, Fabrizio De Santis, and Georg Sigl. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 79–93. Springer, 2013.

[HMH+12] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In Orr Dunkelman, editor, *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, volume 7178 of *Lecture Notes in Computer Science*, pages 231–244. Springer, 2012.

[ISO12] ISO. ISO/IEC 19790: Information Security Management Systems – Requirements. ISO/IEC Standard 19790, International Organization for Standardization, 2012.

[ISO15] ISO. ISO/IEC 30104: Information technology — Security techniques — Security requirements for cryptographic modules. ISO/IEC Standard 30104, International Organization for Standardization, 2015.

[ISO16] ISO. ISO/IEC 17825: Information technology — Security techniques — Testing methods for the mitigation of non-invasive attack classes against cryptographic modulese. ISO/IEC Standard 1782590, International Organization for Standardization, 2016.

[JY02] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[Mat97] Maja J. Mataric. Reinforcement learning in the multi-robot domain. *Auton. Robots*, 4(1):73–83, 1997.

[MDS99] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Power analysis attacks of modular exponentiation in smartcards. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 1999.

[Mil85] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.

[MKS⁺13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[MKS⁺15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

[New23] NewAE. Tutorial P1 Using a Custom Trigger. https://wiki.newae.com/Tutorial_P1_Using_a_Custom_Trigger, 2023. Accessed: 2023-05-29.

[PC15] Guilherme Perin and Lukasz Chmielewski. A semi-parametric approach for side-channel attacks on protected RSA implementations. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 34–53. Springer, 2015.

[PCBP21] Guilherme Perin, Lukasz Chmielewski, Lejla Batina, and Stjepan Picek. Keep it unsupervised: Horizontal attacks meet deep learning. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):343–372, 2021.

[PSM20] Zhixin Pan, Jennifer Sheldon, and Prabhat Mishra. Test generation using reinforcement learning for delay-based side-channel analysis. In *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*, pages 109:1–109:7. IEEE, 2020.

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. SCARL: side-channel analysis with reinforcement learning on the ascon authenticated cipher. *CoRR*, abs/2006.03995, 2020.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[RWPP21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):677–707, 2021.

[SB98] Richard S. Sutton and Andrew G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9(5):1054–1054, 1998.

[SHKS15] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2015.

[SHM+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

[SQAS16] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[UAE93] Michael Unser, Akram Aldroubi, and Murray Eden. B-spline signal processing. i. theory. *IEEE Trans. Signal Process.*, 41(2):821–833, 1993.

[vWWB11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2011.

[WD92] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[WHW+22] An Wang, Shulin He, Congming Wei, Shaofei Sun, Yaoling Ding, and Jiayao Wang. Using convolutional neural network to redress outliers in clustering based side-channel analysis on cryptosystem. In Meikang Qiu, Zhihui Lu, and Cheng Zhang, editors, *Smart Computing and Communication - 7th International Conference, SmartCom 2022, New York City, NY, USA, November 18-20, 2022, Proceedings*, volume 13828 of *Lecture Notes in Computer Science*, pages 360–370. Springer, 2022.

A Using Basic SPA-GPT to segment the power trace of RSA on AISC X

We collected the power traces during the execution of RSA on ASIC X, which was designed in a certain laboratory. After preprocessing steps such as alignment and averaging (detailed processing methods are described in trace database), we obtained the trace shown in Figure 29. Upon zooming in on the trace, we can observe regular variations of sharp peaks. However, the amplitude of these peaks gradually decreases. The yellow boxes indicate the speculated operation distinguishing features.

After applying our method to segment the trace, we obtained the result shown in Figure 30. Upon zooming in on the segmented regions, we can observe that the segmentation of each trace segment is accurate. Furthermore, we can distinguish the difference between

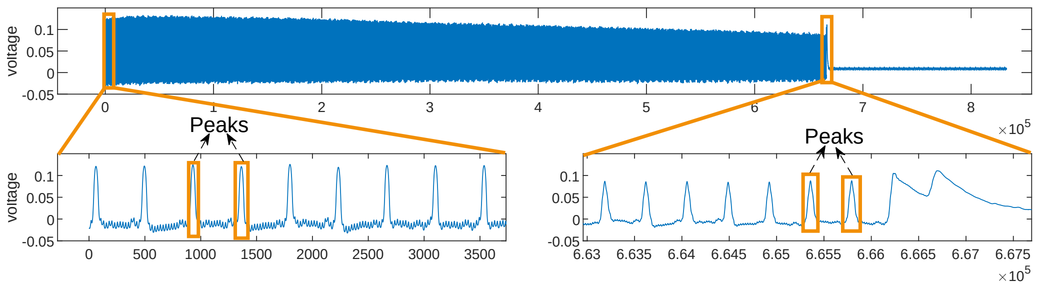


Figure 29: Power trace of RSA decryption on ASIC X

the trace segments corresponding to “S” and “M”. Through horizontal analysis, we can fully recover the correct private key.

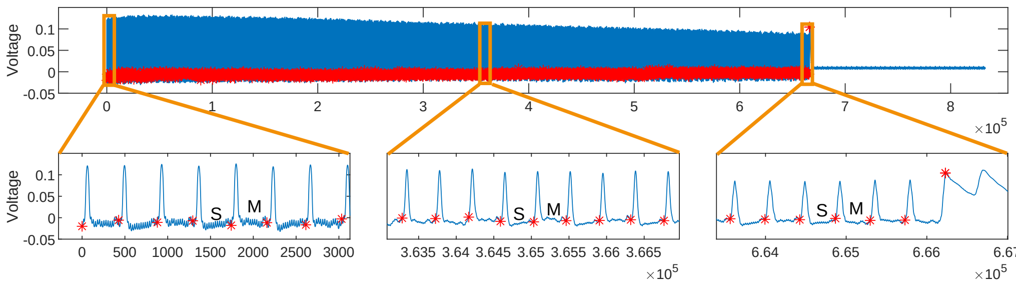


Figure 30: Power trace segmentation result of RSA on ASIC X

B Using Enhanced SPA-GPT to segment the power trace of ECC on AT89S52

We have implemented ECC decryption on the AT89S52 platform, and the power trace generated during the algorithm execution are shown in Figure 31. After zooming in on the power trace, we can observe periodic segments and a clear termination point. Among them, the double operation exhibits a downward spike (represented by “D”), while the add operation displays two downward spikes (represented by “A”). Additionally, the length of the trace segments corresponding to these two operations differs. The segmentation and operations recovery results are shown in Figure 32. The black trace segments represent double operations, the blue ones represent add operations, and the gray portion represent redundant parts. The segmentation result align with the individual operations, and the segments classification is completely accurate. This demonstrates that our method can correctly segment traces with differences in segment lengths.

C Using Enhanced SPA-GPT to segment the power trace of RSA on STM32F429

We have implemented RSA decryption based on Montgomery modular exponentiation on the STM32F429 target board, and the power trace generated during the algorithm execution is shown in Figure 33. After zooming in on the power trace, it is evident that there are noticeable differences between the modular square operation and the modular

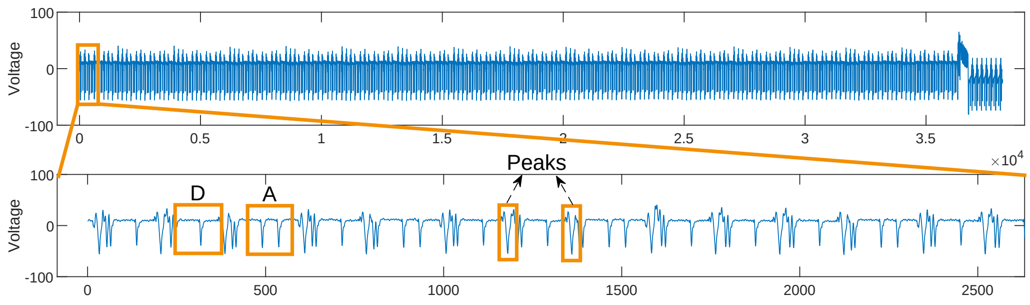


Figure 31: Power trace of ECC decryption on AT89S52

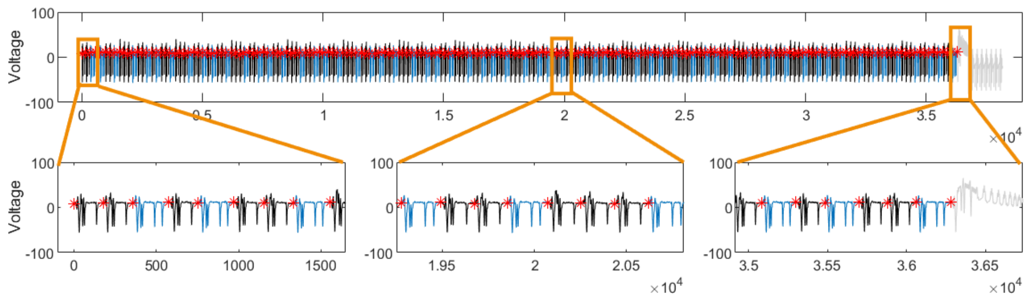


Figure 32: Power trace segmentation and operations recovery result of ECC on AT89S52

multiplication operation. The modular square operation exhibits lower spikes, while the modular multiplication operation displays higher spikes.

After applying our method, the segmentation and operations recovery results are depicted in Figure 34. Each operation corresponds to the interval between two segmentation points. The black segments represent modular square operations, the blue ones represent modular multiplication operations, and the gray regions indicate redundant parts. Except for the last segment caused by Montgomery reduction operation (after the termination point), we have correctly recovered all operations. However, due to the significant amplitude variation in this trace, during fine-tuning, the segmentation point crosses an operation as it adjusts towards the "pole", as illustrated in the Figure 35. So, fine-tuning is not suitable in this scenario.

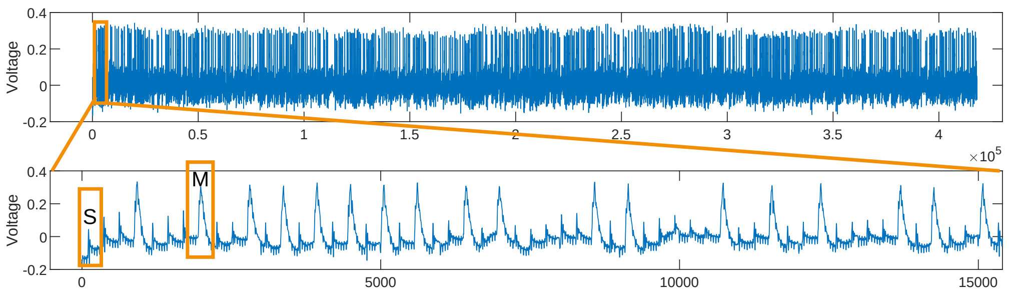


Figure 33: Power trace of RSA decryption on STM32F429

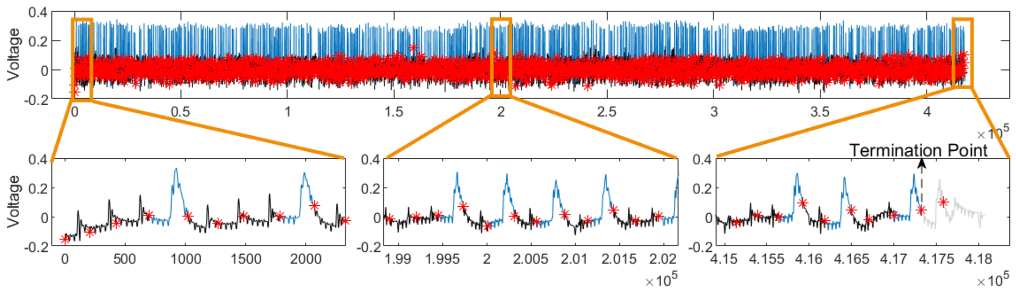


Figure 34: Power trace segmentation and operations recovery result of RSA on STM32F429

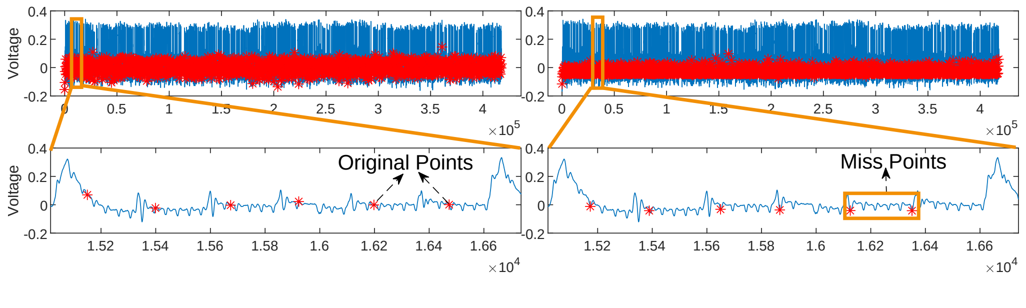


Figure 35: Segmentation results of power trace of RSA on STM32F429 after processing with fine-tuning method