CSAW ESC 2025 Qualification Report

TRX Technical Labs - Sapienza

Kristjan Tarantelli
Sapienza University of Rome
tarantelli.kristjan@gmail.com

Francesco Bianchi
Sapienza University of Rome
f.bianchi202@gmail.com

Simone Di Maria
University of Verona
simonedimaria2004@gmail.com

Federico Angelilli
Sapienza University of Rome
fedeangelilli@gmail.com

Emilio Coppa
Luiss University of Rome
ecoppa@luiss.it

I. Introduction

Side-channel attacks exploit unintended information leaks in hardware and software implementations of cryptographic or authentication systems. Even when an algorithm is mathematically secure, its physical implementation may reveal sensitive information through timing, power consumption, electromagnetic emanations, or fault behavior.

In this challenge, teams are provided with the source code of a vulnerable password verification routine. The implementation includes conditional branches and timing variations that create measurable side-channel leakage. Additionally, power traces are available, enabling further analysis of the underlying operations. The task is to describe how one would exploit these vulnerabilities and to demonstrate preliminary analysis using the provided traces.

II. Vulnerable Implementation

The vulnerable function `verify()` performs a character-by-character comparison between the input password and the secret. When a character matches, it increments a counter and introduces an artificial delay loop. When a mismatch is detected, the comparison terminates early.

```
1:  procedure VERIFY(data, dlen)
2:      correct ← 1
3:      matched_chars ← 0
4:      for i = 0 to password_len − 1 do
5:          if data[i] = password[i] then
6:              matched_chars ← matched_chars + 1
7:              for k = 0 to 5000 do
8:                  nop
9:          else
10:             break
11:     if matched_chars ≠ password_len then
12:         correct ← 0
13:     return correct
```

This behavior leaks the number of consecutive correct characters via both timing and power consumption:

- The presence of the delay loop increases the runtime for each correct character, making the system susceptible to timing attacks.
- Power consumption patterns differ depending on whether the comparison continues or exits early, providing exploitable side-channel traces.

Such designs allow adversaries to gradually recover the password by testing inputs one character at a time and measuring the response.

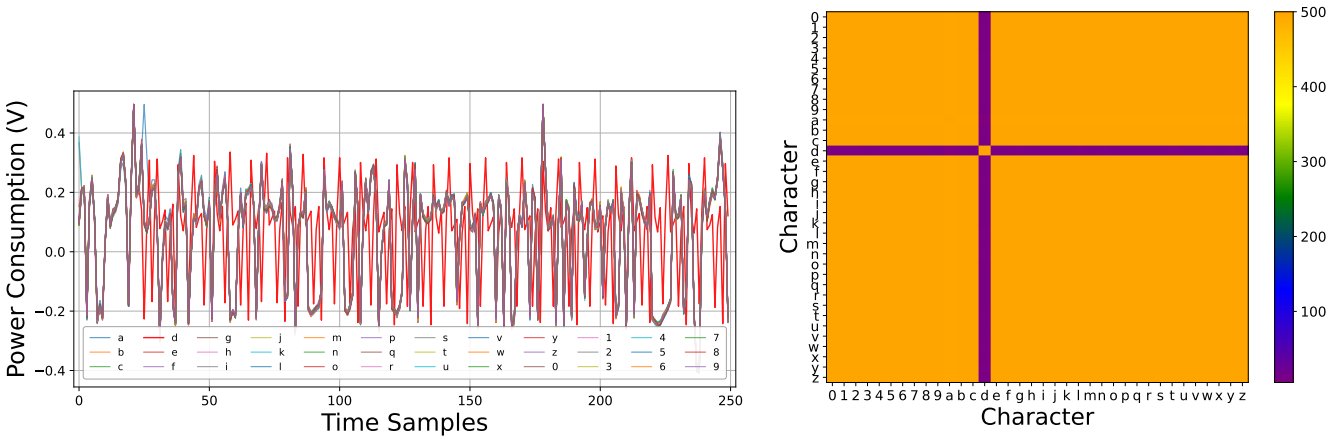
III. Methodology

The general strategy for exploiting this vulnerability combines classical side-channel techniques with modern machine learning tools:

1) **Trace Acquisition.** Measure timing and power consumption while supplying chosen plaintext inputs.
2) **Preprocessing.** Align traces, remove noise, and normalize signals to enable consistent analysis.
3) **Correlation Analysis.** Use statistical methods such as correlation or normalized cross-correlation (NCC) to identify differences between traces associated with correct and incorrect characters.
4) **Deep Learning Automation.** Apply convolutional neural networks (CNNs) or recurrent neural networks (RNNs) to classify traces and automate leakage exploitation. Large Language Models (LLMs) can be used to generate attack scripts, adapt methodologies, and even propose fault injection experiments.
5) **Fault Injection.** Optionally, introduce voltage glitches or electromagnetic pulses to induce errors, observing how incorrect states may leak further information.

IV. Experimental Analysis

The provided traces capture the chip's power consumption (through voltage drop) when tested with different input characters. We implemented a proof-of-concept correlation analysis in Python, truncating the traces to the first 250 samples and normalizing with a z-score transformation. For each pair of



(a) Overlay of the first 250 samples of traces corresponding to different characters. Subtle differences are visible in the correct character's trace.

(b) Normalized cross-correlation matrix between character traces. The character `d` is clearly an outlier.

Fig. 1: Side-channel analysis results: (a) comparison of traces and (b) compiled NCC matrix.

characters, we computed the normalized cross-correlation at zero lag:

$$\mathrm{NCC}(a, b) = \frac{(a - \mu_a) \cdot (b - \mu_b)}{\sigma_a \sigma_b}.$$

The results were compiled into an $n \times n$ matrix where n is the number of distinct characters tested, shown in Figure 1b.

V. Results and Discussion

Two visualizations highlight the leakage. Overlaying traces reveals subtle differences between characters (Figure 1a), while the NCC matrix shows that most characters correlate strongly, with `d` emerging as an outlier. These differences suggests `d` is the first correct character.

These results confirm that the vulnerable design leaks exploitable information. Beyond basic correlation analysis, more advanced techniques can be applied. Deep learning models such as CNNs can automatically extract leakage features from noisy traces, while large language models (LLMs) can support automation of the exploitation pipeline by generating attack scripts, issuing candidate guesses to the chip, and assisting with interpretation of results. Fault injection represents another vector, where induced glitches may bypass comparisons or reveal length information.

VI. Mitigation Strategies

The leakage in the `verify()` routine comes from data-dependent branches and timing variations. A constant-time implementation, as shown in Algorithm 1, avoids early exits and delay loops by processing all characters and folding differences into a single accumulator. This ensures that execution time and power use are independent of the input.

Additional protections include masking (randomizing intermediate values), hiding (balancing operations or adding noise), and countermeasures against fault injection such as redundant checks or error detection. In practice, combining constant-time code with these techniques provides the strongest defence.

Algorithm 1 Constant-time/power and comparison

procedure VERIFY_SAFE($data, dlen$)
  $diff \leftarrow 0$
  $len\_diff \leftarrow (dlen \oplus passwd\_len)$
  pad $data$, $password$ **to** MAX_LEN
  for $i = 0$ **to** $MAX\_LEN - 1$ **do**
    $a \leftarrow data[i]$
    $b \leftarrow password[i]$
    $diff \leftarrow diff \mid (a \oplus b)$
  $diff \leftarrow diff \mid len\_diff$
  return not $diff$

VII. Related Work

Kocher first demonstrated that cryptographic implementations could leak secrets through timing variations [1], and later introduced Differential Power Analysis (DPA), which established power side-channels as a practical threat [2]. Since then, correlation-based methods such as CPA have been widely adopted. More recently, deep learning approaches have achieved strong results in noisy environments, learning discriminative features directly from traces without manual preprocessing [3], [4]. Fault injection has also emerged as another powerful technique, showing that induced glitches can compromise embedded systems and cryptographic devices [5].

VIII. Conclusion

The provided vulnerable password verification function leaks sensitive information through both timing and power side channels. By computing normalized cross-correlation between traces, we demonstrated the attack's practicality. While this phase only required description and initial analysis, the results confirm that a determined adversary could fully recover the password. Combining classical correlation methods with modern deep learning and LLM-based automation would make such attacks even more efficient and scalable.

REFERENCES

[1] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual international cryptology conference*. Springer, 1996, pp. 104–113.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[3] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

[4] S. Karayalcin, M. Krcek, and S. Picek, "A practical tutorial on deep learning-based side-channel analysis," *Cryptology ePrint Archive*, 2025.

[5] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.