Bit-by-Bit : The Side Channel Hackathon

Hackathon Challenge 1

Side-Channel Shenanigans: The RSA Key Recovery Challenge

Objectives

1. Establish communication with the target STM32F303 microcontroller using the simpleserial_rsa firmware.
2. Capture power consumption traces during RSA decryption for multiple random ciphertexts.
3. Analyze the traces using side-channel techniques to recover the 15-bit private key *d*.
4. Verify the recovered key by sending it as a ciphertext and checking for the expected plaintext output of 6267.

Overview

The script controls a Chipwhisperer scope and an attached target microcontroller STM32F3. It programs the target with a firmware that implements a SimpleSerial command and repeatedly sends ciphertext and captured the traces.

- Purpose : Capture power traces from the target running rsa-CW308 STM32F3 firmware
- Requirements : Chipwhisperer + target connected, Installed the software, and the firmware file given
- Expected Output: Verify and derive the 10 bit of secret key of length 15 bits

Methodology

Step 1: Target Programming

- Flashes the RSA firmware onto the target microcontroller and configurable parameters RSA_N = 64507 and the number of traces we captured is 5000 with a sampling points of 5000.

Step 2 : Capture the traces

- Arm the scope, sends the ciphertext c must be less than modulus N, wait for the decryption process and capture the ADC samples, and return false on success. Store the traces (As mentioned in the problem statement)

Step 3: Pre-Processing Traces

- Loaded traces from CSV into NumPy arrays and applied sliding windows (length=20, step=5) to average segments, focusing on operation-specific timing.
- Checked standard deviations to skip invalid correlations.

Step 4: Perform Power Model

Hamming weight power model

- For each bit of exponent (MSB – LSB), simulates the device intermediate value after processing the required bit prefix for both the guesses (1 or 0).
- Convert the simulated to a predicted leakage value.

Step 5: Perform Statistical Method

CPA is performed with the collected power traces.

- Extract a measured feature from each trace
- Compute the Pearson Correlation between the predicted leakage and measured feature across traces
- Picks the bit (0 or 1) and the window that yields the largest absolute correlation – that guess is assumed to be the correct next bit.
- Pearson correlation measures linear relationship between predicted leakage vector and measured feature vector, using this the full key is recovered by considering it as whole 15 bits are unknown instead of hardcoding the known five bits.

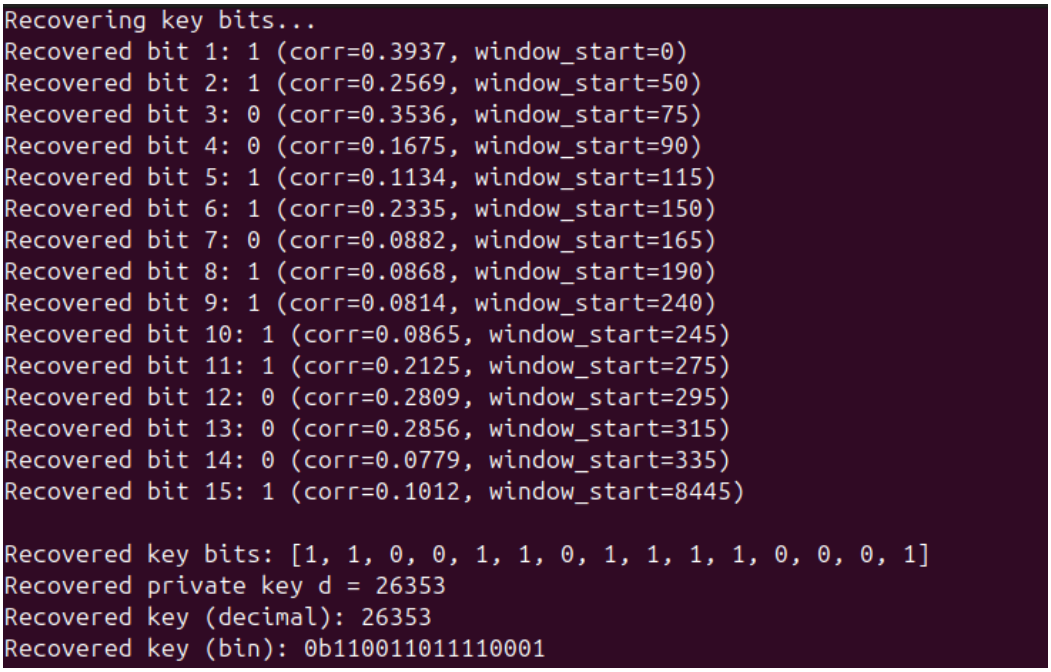

Figure 1 : Recovered the secret key

Step 7 : Verification

- Loads the SimpleSerial RSA firmware so it can accept ciphertexts and return plaintexts.
- Converts the recovered key integer into 2-byte big-endian byte sequence to send as the ciphertext input.
- Waits for the device to perform decryption and then reads back two bytes from the SimpleSerial Command and its been used for success/failure verification.
- Then converts the received bytes to an integer and compares it with the known correct plaintext 6267. If they match verification passes, otherwise it fails.

- The figure 2 shows the recovered key d = 26353 is writing back the plaintext as 6267, so the verification is passed.

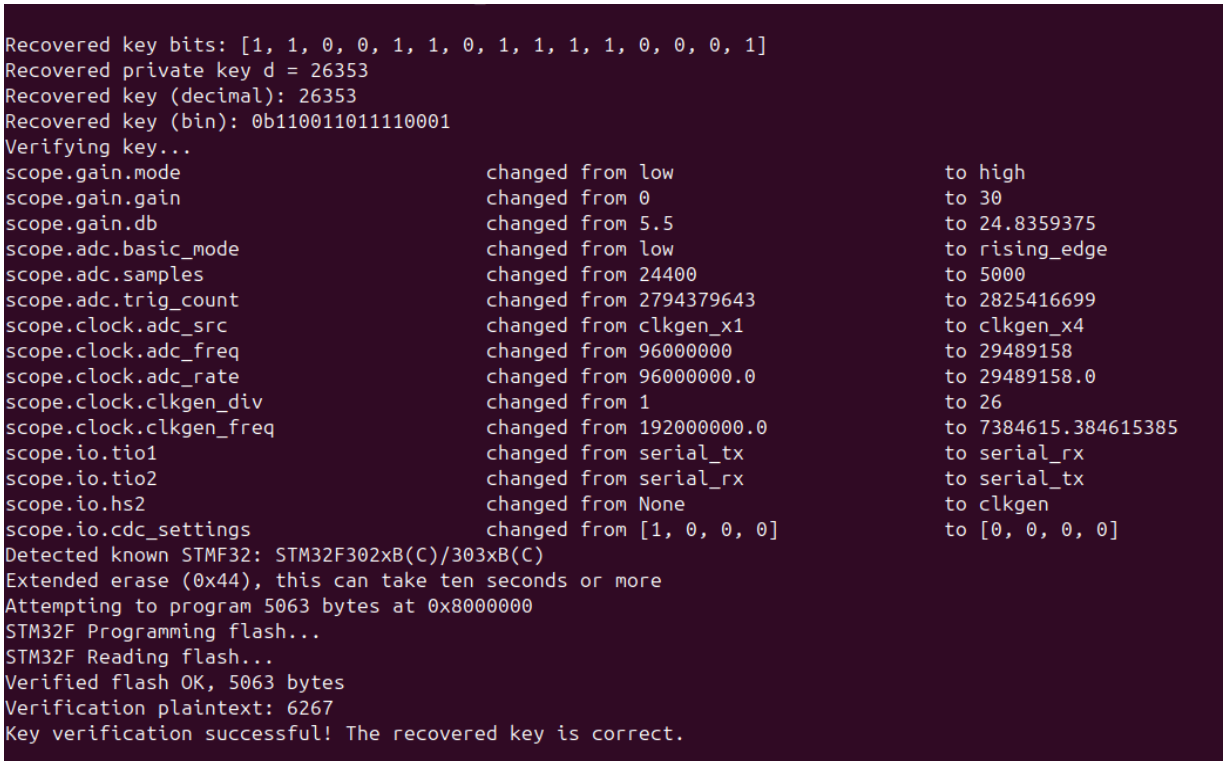

Figure 2: Verification result

Challenges Faced

- Code corrections: Replaced simulated traces with real ADC captures; fixed scope connection.
- Trace quality: done trail and error method for trace collection with different size of real power matrix. Noise and misalignment; mitigated by window averaging and high trace count.
- Statistical model : Tried Simple power analysis and differential power analysis with machine learning algorithms.
- Optimization: Adjusted window parameters (length/step) for better correlations; handled errors in CPA.

Conclusion

We successfully established communication with the STM32F303 microcontroller using the simpleserial_rsa firmware, capturing 5000 power traces during RSA decryption of random ciphertexts. Through Correlation Power Analysis and the Hamming Weight model, we

recovered the 15-bit private key d (decimal: 26353, binary: 110011011110001), aligning with the known MSB=1 and LSBs=0001. Verification confirmed the key's correctness, as sending it as a ciphertext yielded the expected plaintext 6267. Despite challenges like trace noise, adaptive windowing and parallel processing ensured success. The scripts are available at https://github.com/suryaarasu11/bitbybit/ .