Learning Timing Side-channel in Embedded Systems Design Core Course

Tauhidur Rahman, Florida International University

Full points: 100 (see rubric)

Objectives: *The objective of this lab module is to learn time side-channel while the students learn the core design concepts.*

Pre-lab Reading: *Read class lectures and tutorials.*

Problem 1: *You have an embedded system with a password-based authentication mechanism. The system takes 1 second to process an incorrect password attempt and 2 seconds for a correct one.*

Task for Students:

- Write a simple program to measure the time it takes for the system to respond to password attempts. **[design]**
- Use the timing data to determine if it's possible to distinguish between correct and incorrect password attempts without knowing the actual password. **[time side-channel]**
- Discuss potential countermeasures to prevent timing side-channel attacks in this scenario. **[countermeasures]**

Problem 2: Imagine an embedded system that performs cryptographic operations. It employs different algorithms, and their execution times vary significantly based on input data and keys. The attacker has access to the system and can observe the execution time of cryptographic operations.

Task for Students:

- Develop a program or setup that can measure and record the execution time of cryptographic operations on the embedded system. **[design]**
- Create a dataset of execution times for different inputs and keys. **[preparation phase]**
- Analyze the dataset to identify patterns or correlations between execution times and cryptographic parameters (e.g., key length, input data size). **[time side-channel]**
- Demonstrate how an attacker could potentially use this timing information to infer information about cryptographic operations or secret keys. **[attack]**
- Explore advanced countermeasures to mitigate timing side-channel attacks in this context, such as implementing constant-time cryptographic algorithms or adding random delays. **[countermeasures]**

Problem 3: *In a secure embedded system, a cryptographic module performs elliptic curve cryptography (ECC) operations to establish secure connections. The ECC operations are implemented using different algorithms, and their execution times depend on the specific algorithm and input data.*

Task for Advanced Students:

- Build a setup to capture precise execution times of ECC operations on the embedded system without modifying its code or hardware. **[design]**
- Collect a comprehensive dataset of execution times for various ECC operations with different algorithms, key lengths, and input data. **[preparation phase]**
- Develop advanced statistical techniques (e.g., machine learning) to analyze the timing data and identify subtle patterns and correlations that can reveal information about the algorithms or keys. **[Attack]**
- Demonstrate how an attacker, armed with the timing data, can distinguish between ECC algorithms and potentially deduce sensitive information about the cryptographic keys. **[Attack]**
- Explore advanced countermeasures to protect against timing side-channel attacks, such as implementing constant-time algorithms, incorporating noise in execution times, or using hardware-level protections like blinding or masking. **[countermeasures]**