# Project Title: Side-Channel Attack Evaluation Platform

---

### Short Project Description:

A comprehensive framework to design, implement, and evaluate a side-channel attack evaluation platform. The framework captures physical signals (power/EM), analyzes cryptographic key leakage, tests countermeasures, and generates actionable reports to assess hardware security.

---

| Component | Role |
|---|---|
| Signal Acquisition Module | Collects power traces or EM emissions |
| Side-Channel Leakage Analyzer | Analyzes traces for information leaks |
| Key Recovery Engine | Attempts to recover cryptographic keys |
| Countermeasure Evaluation Module | Tests effectiveness of side-channel defenses |
| Reporting Engine | Summarizes leakage and risk findings |

---

### Component Details:

1. **Signal Acquisition Module**:
    - Collects:
        - Power usage fluctuations
        - Electromagnetic radiation signals
    - Tools:
        - ChipWhisperer
        - Oscilloscopes
        - Etc
2. **Side-Channel Leakage Analyzer**:
    - Applies:
        - Differential Power Analysis (DPA)
        - Correlation Power Analysis (CPA)
        - Simple Power Analysis (SPA)
3. **Key Recovery Engine**:
    - Recovers:
        - AES keys
        - RSA private keys
    - Based on leaked information.
4. **Countermeasure Evaluation Module**:
    - Evaluates:
        - Noise injection defenses
        - Randomized instruction execution
5. **Reporting Engine**:
    - Summarizes:

            - Attack success
            - Key leakage probability
            - Suggestions for stronger countermeasures

---

### Overall System Flow:

- Input: Physical device during cryptographic operations
- Output: Risk evaluation report
- Focus: **Side-channel cryptanalysis** of hardware.

---

### Internal Functioning of Each Module:

### 1. Signal Acquisition Module

- **Data capture**:
    - **Power Analysis**:
        - Shunt resistor method: place a tiny resistor and measure voltage drop (related to current).
    - **Electromagnetic Analysis**:
        - Use EM probes to detect emissions near cryptographic operations.
- **Devices**:
    - ChipWhisperer capture platforms.
    - High-speed oscilloscopes (>100 MS/s).

---

### 2. Side-Channel Leakage Analyzer

- **Leakage detection methods**:
    - **Simple Power Analysis (SPA)**:
        - Directly observe patterns (e.g., visible key-dependent operations).
    - **Differential Power Analysis (DPA)**:
        - Statistical attack:
            - Collect hundreds or thousands of traces.
            - Use mathematical correlations to deduce key bits.
    - **Correlation Power Analysis (CPA)**:
        - Calculate correlation between hypothetical intermediate values and measured power consumption.

---

### 3. Key Recovery Engine

- **Attacks**:
    - AES:
        - Correlate subkey guesses during S-box operations.

- RSA:
    - Recover private exponents based on modular exponentiation timing.
- **Tools**:
    - ScaLib
    - ChipWhisperer Analyzer
    - Etc

---

## 4. Countermeasure Evaluation Module

- **Testing**:
    - Evaluate effectiveness of countermeasures like:
        - Randomized dummy operations
        - Noise injection
        - Balanced circuit designs
    - Score based on increased number of traces needed to break security.

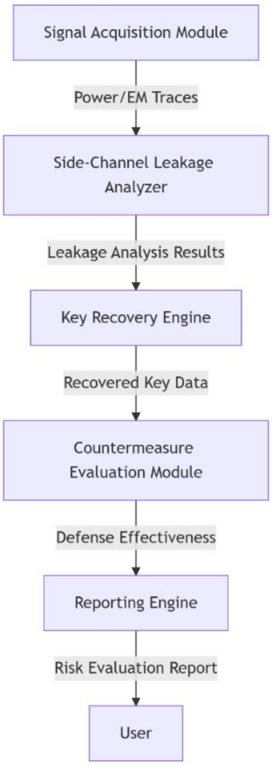
---

## 5. Reporting Engine

- **Final report**:
    - Leakage levels (high, medium, low).
    - Key recovery success/failure.
    - Number of traces required.
    - Recommendations:
        - Add blinding
        - Rework power line filtering

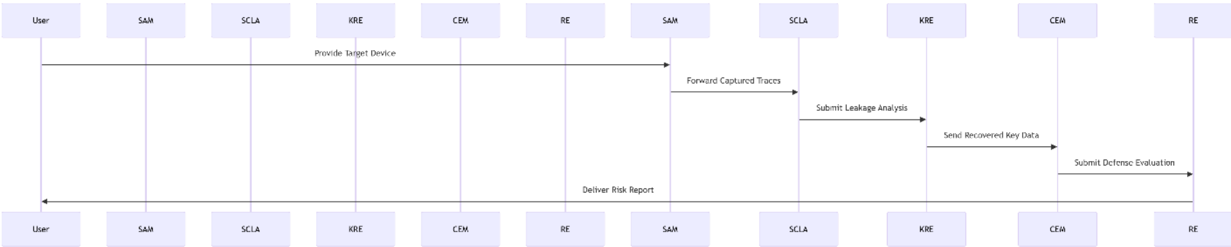
---

### Component Diagram




- **Signal Acquisition Module**: Captures power/EM traces during cryptographic operations (e.g., using ChipWhisperer, etc).
- **Side-Channel Leakage Analyzer**: Detects leaks via methods like Differential Power Analysis (DPA) or Correlation Power Analysis (CPA).
- **Key Recovery Engine**: Attempts to recover cryptographic keys (AES/RSA) using leakage data.
- **Countermeasure Evaluation Module**: Tests defenses (e.g., noise injection, etc) for effectiveness against attacks.
- **Reporting Engine**: Generates a report detailing leakage risks, key recovery success, and defense recommendations.

## Sequence Diagram




1. **User** provides a target device to the **Signal Acquisition Module** for data capture.
2. **Signal Acquisition Module** collects traces and forwards them to the **Side-Channel Leakage Analyzer**.
3. **Leakage Analyzer** identifies vulnerabilities (e.g., key-dependent patterns, etc) and submits results to the **Key Recovery Engine**.
4. **Key Recovery Engine** attempts to extract keys and sends findings to the **Countermeasure Evaluation Module**.
5. **Countermeasure Module** evaluates defense robustness and passes results to the **Reporting Engine**.
6. **Reporting Engine** delivers a final risk report to the **User**, including mitigation strategies.

## Detailed Project Description: Side-Channel Attack Evaluation Platform

A side-channel attack evaluation platform. This platform captures physical signals (power/EM), analyzes cryptographic key leakage, tests countermeasures, and generates actionable reports to assess hardware security.

---

### 1. System Components and Roles

#### 1.1 Signal Acquisition Module

**Purpose**: Capture power consumption or electromagnetic (EM) emissions during cryptographic operations.
**Implementation Details (e.g.)**:

- **Tools**:
    - **ChipWhisperer**:
      ```
      import chipwhisperer as cw
      scope = cw.scope()  # Initialize hardware
      target = cw.target(scope)
      trace = cw.capture_trace(scope, target, b'plaintext')  # Capture power trace
      ```
    - **Oscilloscope**: Configure for high-speed sampling (>100 MS/s) via Python APIs (e.g., `pyvisa`).
- **Setup**:
    - **Power Analysis**: Use a shunt resistor (1–10 Ω) in the device's power line to measure voltage fluctuations.
    - **EM Analysis**: Position an EM probe near the target's cryptographic component (e.g., CPU).

#### 1.2 Side-Channel Leakage Analyzer

**Purpose**: Detect key-dependent leakage using statistical methods.
**Implementation Details (e.g.)**:

- **Methods**:

- **Correlation Power Analysis (CPA)**:
  - Hypothesize intermediate values (e.g., AES S-box outputs) and correlate with traces.

```
from scalib.attacks import CPA
cpa = CPA(n=1000, d=256, p=1)  # 1000 traces, 256 possible S-box outputs
cpa.fit(traces, plaintexts)
key_guess = cpa.get_scores().argmax()  # Recover subkey
```

- **Differential Power Analysis (DPA)**: Average traces grouped by key bit predictions.

- **Tools**:
  - **ChipWhisperer Analyzer**: GUI for visualizing leakage.
  - **ScaLib**: Python library for side-channel analysis.
  - Etc

### 1.3 Key Recovery Engine

**Purpose**: Recover cryptographic keys (AES, RSA) from leakage data.

**Implementation Details (e.g.)**:

- **AES Key Recovery**:
  - Attack the first-round S-box operations using CPA.

```
# Recover 16-byte AES key iteratively
for byte in range(16):
    cpa = CPA(n=5000, d=256, p=1)
    cpa.fit(traces, plaintexts[:, byte])
    key[byte] = cpa.get_scores().argmax()
```

- **RSA Key Recovery**:
  - Use timing analysis during modular exponentiation.

```
from rsasim import recover_rsa_key
private_key = recover_rsa_key(timing_data, public_exponent)
```

### 1.4 Countermeasure Evaluation Module

**Purpose**: Test defenses like noise injection or masking.

**Implementation Details (e.g.)**:

- **Techniques**:

    - **Noise Injection**: Add Gaussian noise to traces and measure attack success rate.
    - **Randomized Delays**: Insert dummy operations and evaluate trace misalignment.
- **Metrics**:
    - **Traces-to-Disclosure (TTD)**: Number of traces required to recover a key pre/post-countermeasure.

#### 1.5 Reporting Engine

**Purpose**: Summarize leakage risks and countermeasure effectiveness.
**Implementation Details (e.g.)**:

- **Tools**:
    - **Python + Pandas**: Aggregate results into CSV/JSON.
    - **Jupyter Notebook**: Visualize TTD metrics and leakage heatmaps.
- **Report Contents**:
    - Key recovery success rate.
    - TTD improvement after countermeasures.
    - Recommendations (e.g., "Implement masking for AES S-boxes").

---

### 2. System Integration and Component Interaction

1. **Signal Capture**:
    - **Signal Acquisition Module** collects 1,000+ traces → saves as `traces.npy`.
2. **Leakage Analysis**:
    - **Leakage Analyzer** runs CPA/DPA → identifies vulnerable operations.
3. **Key Recovery**:
    - **Key Recovery Engine** extracts AES/RSA keys → outputs `key.txt`.
4. **Countermeasure Testing**:
    - Apply noise to traces → rerun CPA to measure TTD increase.
5. **Reporting**:

- **Reporting Engine** generates PDF/HTML report with findings.

---

## 3. Evaluation Criteria

1. **Key Recovery Rate**: Percentage of successful key extractions (e.g., 90% for 1,000 traces).
2. **TTD Improvement**: Increase in traces needed post-countermeasure (e.g., 5x more).
3. **Noise Robustness**: Signal-to-noise ratio (SNR) reduction after defenses.

---

## 4. Ethical and Safety Considerations

- **Authorization**: Only test devices you own or have explicit permission to analyze.
- **Safety**: Use ESD-safe equipment to avoid damaging hardware.

---

## 5. Tools and Resources (e.g.)

- **Acquisition**: ChipWhisperer, Picoscope, etc.
- **Analysis**: ScaLib, ChipWhisperer Analyzer, etc.
- **Reporting**: Jupyter, Pandas, etc.

---