Hands-on – SSPREW 2018

Damien Couroussé – CEA¹

December 3, 2018
Outline

Hands-on session
  Requirements
  Setup
  Practice
mylittlepwny is a little tool to cover the first practical needs and steps in side-channel analysis. You can either:

▶ run the tool in a docker container
▶ or build it with the stack tool for Haskell
Install mylittlepwny

- Grab the docker image from the SSPREW website, or the source code from github: https://github.com/cogito-cea/mylittlepwny
- To install the docker image:
  
  ```bash
  $ docker load -i mylittlepwny-<GITCOMMIT>.tar.gz
  ```
  
- the latest version of the README is on github’s repository.
AES zoo: a bestiary of AES implementations with various levels of protections against side-channel attacks.  

*Caution note: these implementations are only provided for educational purposes. They should be considered as weak w.r.t. side-channel analysis.*

1. AES-128, 8-bit version, unprotected
2. AES-128, execution of the AddRoundkey and SubBytes loops in random order
   
Materials – list of files

- **Secret key:**
  
  traces-0-starter
  
  key.txt

- **Input plaintexts:**
  
  traces-0-starter/
  
  key.txt
  
  plaintexts-1000000.txt
  
  plaintexts.txt

- **Description of the two population of plaintext files for the non-specific t-test:**
  
  traces-0-starter/
  
  plaintexts-ttest-NS.txt
  
  separate-ttest-NS.txt

- **Set of traces:**
  
  traces-0-starter/
  
  0-unprotected-spa
  
  1-unprotected
  
  traces-1-shuffling/
  
  2-shuffling
  
  2-shuffling-no-shuffling
  
  2-shuffling-trigger-sync
  
  2-shuffling-ttestNS
  
  traces-2-coron/
  
  3-coron-without-fakes
  
  3-coron-without-fakes-ttestNS
1. AES-128, 8-bit version, unprotected
   - traces-0-starter/0-unprotected-spa: traces covering the complete execution of AES, for SPA analysis.
   - traces-0-starter/1-unprotected: 20000 first samples of AES encryption
2. AES-128, execution of the AddRoundkey and SubBytes loops in random order
   - traces-1-shuffling/2-shuffling: raw acquisition
   - traces-1-shuffling/2-shuffling-ttestNS: acquisition with a specific set of plaintexts, for the non-specific t-test
   - traces-1-shuffling/2-shuffling-trigger-sync: acquisition with a trigger after the computation of the table of randomized indexes
   - traces-1-shuffling/2-shuffling-no-shuffling: random execution of loops disabled
3. AES-128, fake rounds and temporal desynchronisation, following
   - traces-2-coron/3-coron-without-fakes: 3 first fake rounds disabled (i.e. smaller temporal desynchronisation).
   - traces-2-coron/3-coron-without-fakes-ttestNS: same as above, with a set of plaintexts for the non-specific t-test.
Step #0. instrumentation of the target

Real world: you first start with an instrumentation of the target

- Identification of the crypto cipher used / attack
  - Which crypto cipher
  - Where / when is it executed?
- Repeat encryption or decryption a large number of times
  - Typically: at least $10^6$
  - Best configuration: can control the input plaintext (encryption) or ciphertext (decryption)
  - Also possible: the knowledge of input values is enough if we can’t control them.
- Instrumentation of the trigger acquisition
  - Reduce temporal jitter in acquisition traces
  - Avoid concurrent processing activity (or filter it out)
Step #1. Visual inspection of side-channel traces

Is typically part of step #0, but let’s focus on this step for educational purposes.
Step #2. the Side-Channel Analysis

- First solution: brute force the key with a CPA
  - OK if a small number of traces is enough
  - Expensive computation time in other keys

- Does not work? Use t-tests: get more insights about the nature and the location of the side-channel leakages.