Continuous integration side channel testing for ML-KEM

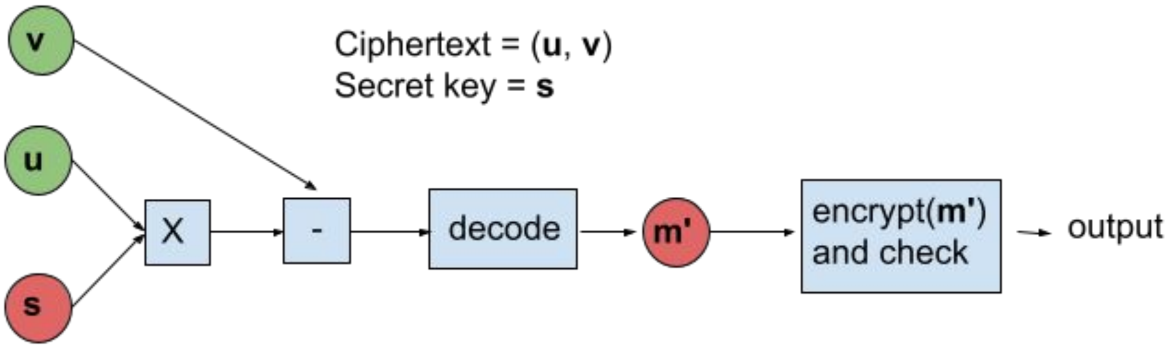
Timo Zijlstra
PQShield

Outline

Main question: how to test SCA security during development of MLKEM implementations?

1. SCA on MLKEM decapsulation
2. Leakage detection using TVLA for continuous integration
3. Limitations of TVLA and alternative methods

Kyber (MLKEM) decapsulation



Input: ciphertext (**u**, **v**) and secret key **s**

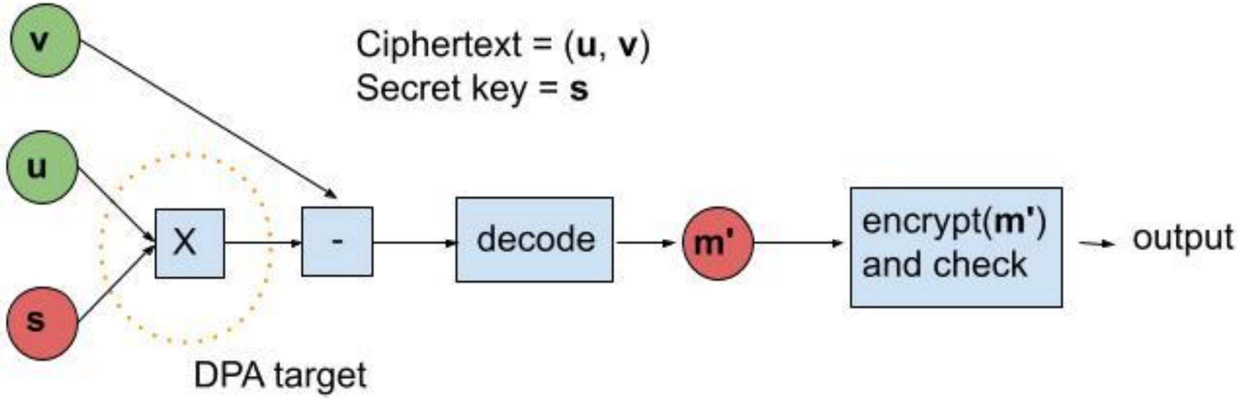
1. Compute **v** - **s***u**
2. Decode/compress result to decoded message bits **m'**
3. Re-encrypt **m'** and check that result equals input ciphertext
4. Output "shared secret"

SCA specific to Kyber decaps

Decaps from draft standard FIPS-203:

1. Simple power analysis (SPA) on shared secret **K'**
2. DPA on secret key **s** during Decrypt

$m' \leftarrow \text{K-PKE.Decrypt}(\text{dk}_{\text{PKE}}, c)$
$(K', r') \leftarrow G(m' \| h)$
$\bar{K} \leftarrow J(z \| c, 32)$
$c' \leftarrow \text{K-PKE.Encrypt}(\text{ek}_{\text{PKE}}, m', r')$
if $c \neq c'$ **then**
    $K' \leftarrow \bar{K}$
end if
return K'

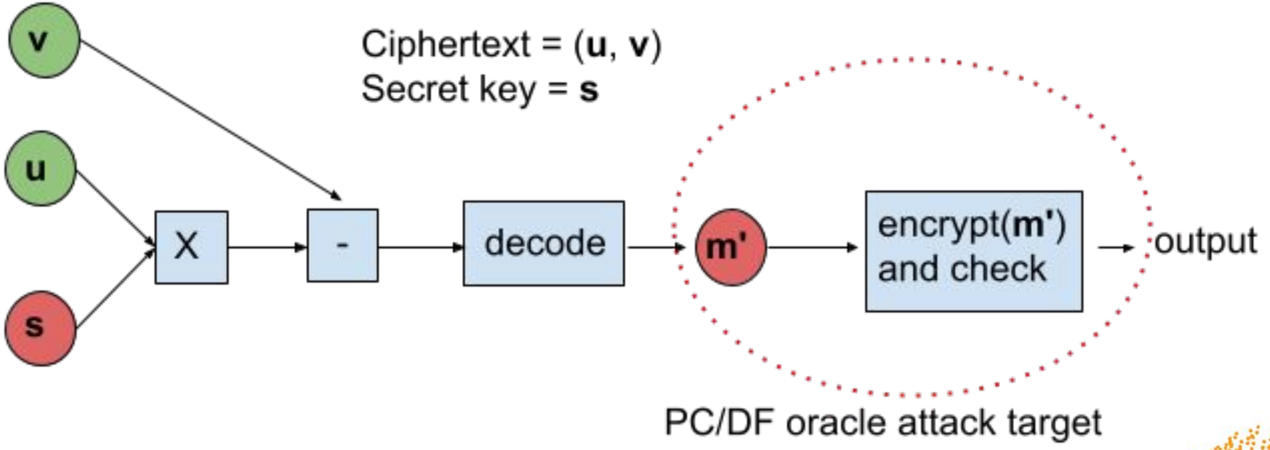


SCA specific to Kyber decaps

1. Simple power analysis (SPA) on shared secret **K'**
2. DPA on secret key **s** during Decrypt
3. Plaintext Checking (PC) oracle

Decaps from draft standard FIPS-203:

$m' \leftarrow$ K-PKE.Decrypt$(\mathrm{dk}_{\mathrm{PKE}}, c)$
$(K', r') \leftarrow G(m' \| h)$
$\bar{K} \leftarrow J(z \| c, 32)$
$c' \leftarrow$ K-PKE.Encrypt$(\mathrm{ek}_{\mathrm{PKE}}, m', r')$
if $c \neq c'$ **then**
    $K' \leftarrow \bar{K}$
end if
return K'



SCA specific to Kyber decaps

1. Simple power analysis (SPA) on shared secret **K'**
2. DPA on secret key **s** during Decrypt
3. Plaintext Checking (PC) oracle
4. Decryption Failure (DF) oracle
   - Exploits the same leakage as PC oracle
   - Additional information during step 9. can be exploited
5. Full Decryption (FD) oracle
   - Similar to PC oracle, but recover 256 bits of **m'** at once
   - Target specific operations that operate sequentially on all bits of **m'**

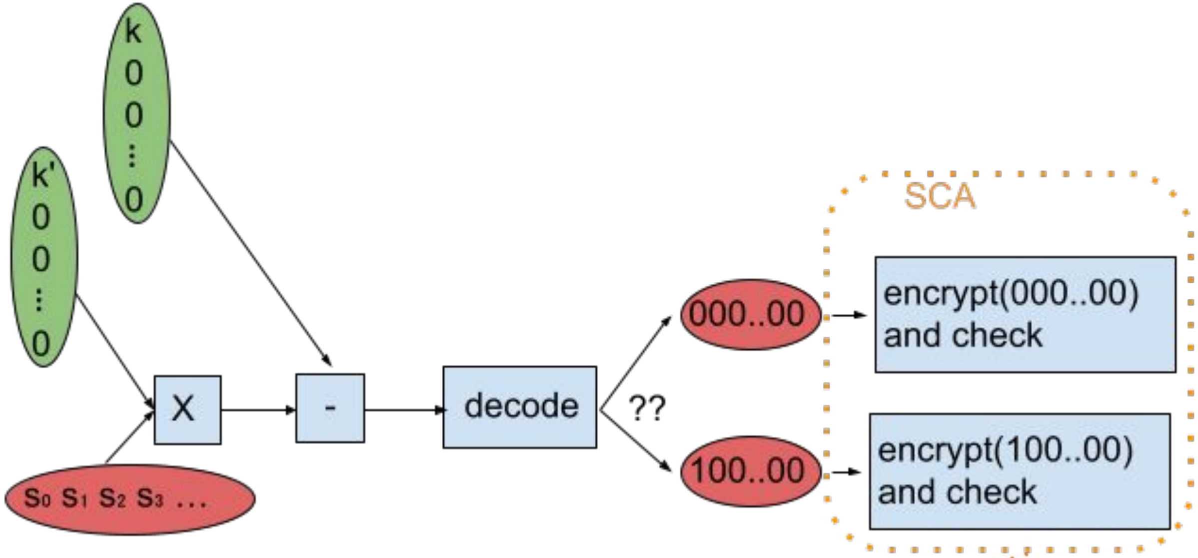
Decaps from draft standard FIPS-203:

5: $m' \leftarrow$ K-PKE.Decrypt$(\mathrm{dk}_{\mathrm{PKE}}, c)$
6: $(K', r') \leftarrow G(m' \| h)$
7: $\bar{K} \leftarrow J(z \| c, 32)$
8: $c' \leftarrow$ K-PKE.Encrypt$(\mathrm{ek}_{\mathrm{PKE}}, m', r')$
9: **if** $c \neq c'$ **then**
10: $\quad K' \leftarrow \bar{K}$
11: **end if**
12: **return** K'

Plaintext checking (PC) oracle attack

SCA attacker wants to recover **s**

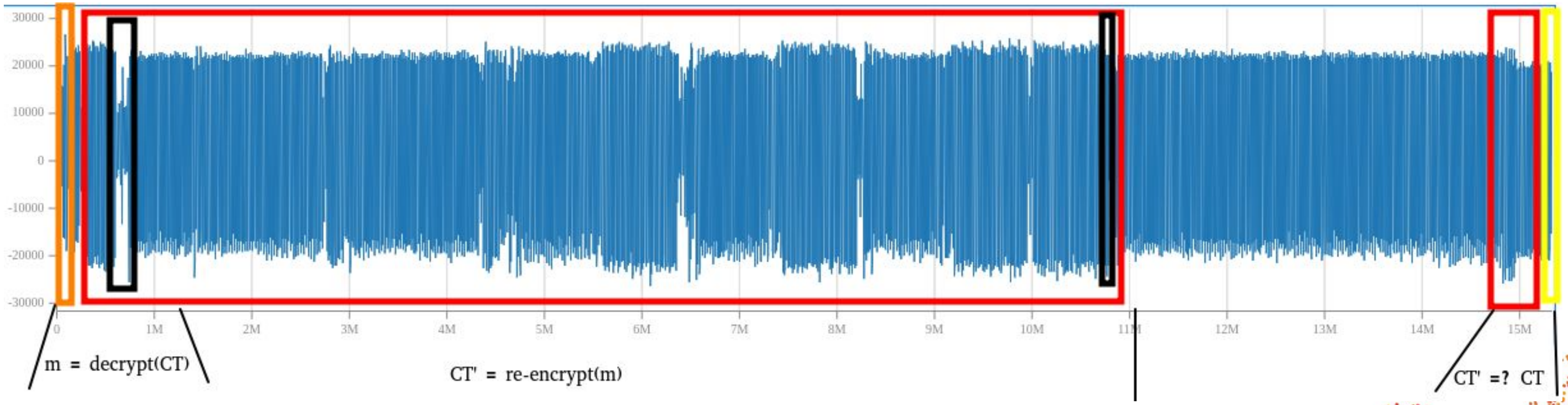
1. Choose **u**, **v** of the form (k, 0, 0, 0, ..., 0)
2. **m'** is either 000...00 or 100..00 depending on $\mathbf{s}_0$
3. Use SCA to distinguish between the two
4. Infer information about $\mathbf{s}_0$, then repeat



Identifying vulnerable operations

in a power trace of a decapsulation from a masked SW/HW co-design

DPA/CPA PC/DF FD DF SPA



⇒ PC/DF is the most important threat, hardest to protect against

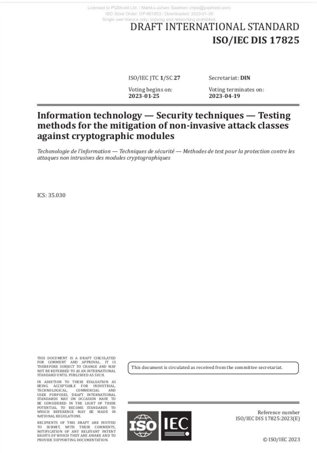
Continuous integration (CI) leakage detection

Detect leakage at an early stage **during development**

- Perform leakage analysis on a regular basis
  - Periodically (daily / weekly)
  - After each change to the code base
- Test must be easy to automate ⟶ "push button"
  - Online test: discard each trace after processing (no trace storage required)

⟶ Test Vector Leakage Assessment (TVLA)

TVLA following ISO 17825



DRAFT INTERNATIONAL STANDARD
ISO/IEC DIS 17825

ISO/IEC JTC **1**/SC **27** — Secretariat: **DIN**

Voting begins on: **2023-01-25** — Voting terminates on: **2023-04-19**

Information technology — Security techniques — Testing methods for the mitigation of non-invasive attack classes against cryptographic modules

Techonologie de l'information — Techniques de sécurité — Methodes de test pour la protection contre les attaques non intrusives des modules cryptographiques

ICS: 35.030

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

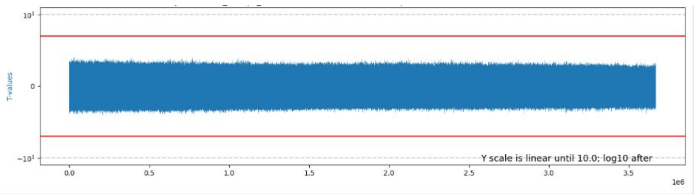
RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

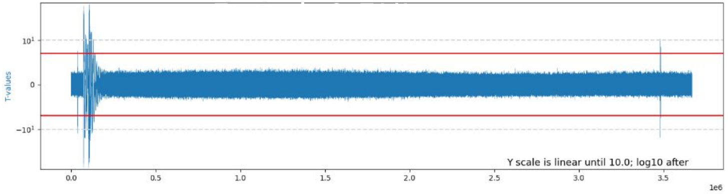
This document is circulated as received from the committee secretariat.

Reference number
ISO/IEC DIS 17825:2023(E)

Test for dependencies between secrets and side channel measurements

1. Create two sets A and B of inputs/keys such that:
   - Secret parameter is fixed in set A
   - Secret parameter is random in set B
2. Execute cryptographic algorithm on target device for A and B and measure power traces
3. Perform static and dynamic trace alignment
4. Perform T-test
   - If the T-value exceeds the threshold output FAIL
   - Else output PASS





TVLA for PC oracle SCA

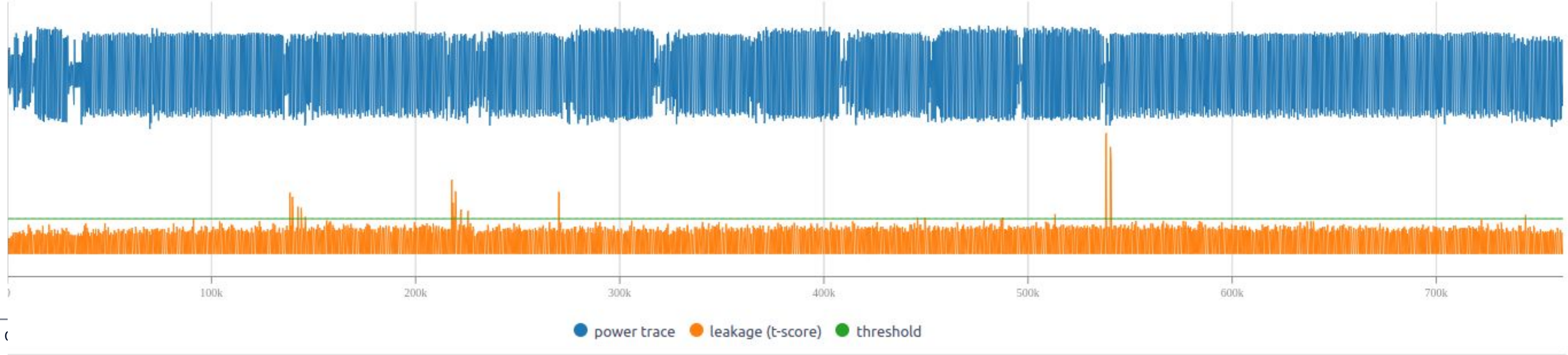
Attacker must distinguish between 2 re-encryptions

1. Craft 2 sets of input ciphertexts CT = (u, v)
   A. Random u, v such that decrypt(u, v) = 000..00
   B. Random u, v such that decrypt(u, v) = 100..00
2. Measure power traces
3. Compute t-test

Problem with crafting CT using Encaps:

$$1: (K, r) \leftarrow \mathsf{G}(m \| \mathsf{H}(\mathsf{ek}))$$
$$2: c \leftarrow \mathsf{K\text{-}PKE.Encrypt}(\mathsf{ek}, m, r)$$

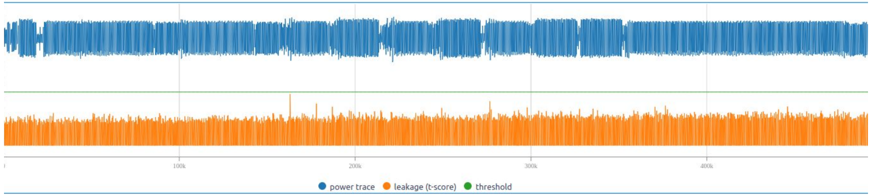
c = (u, v) is **determined by m, ek**
⟶ generate random CT and flip some bits in v such that decrypt(u, v) = 000...00 or 100..00



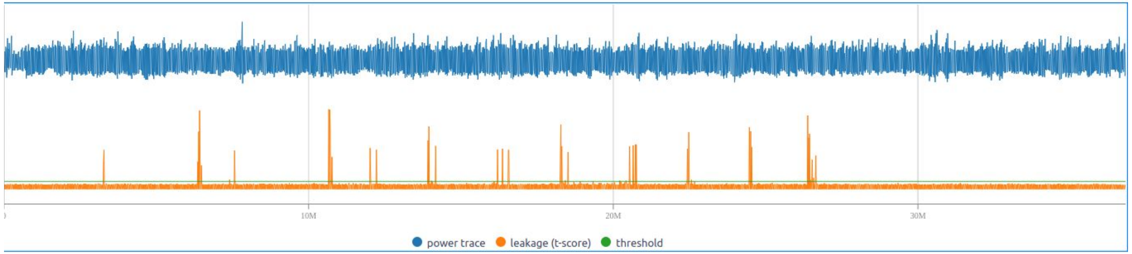
Limitations of TVLA for PC oracle attacks

1. No precise estimation of actual security
   - Number of traces required for key recovery
   - Requires profiled attack on selected points of interests (P.o.I.)
2. Only univariate leakage detection
   - What about 2nd order leakage?

1st order TVLA



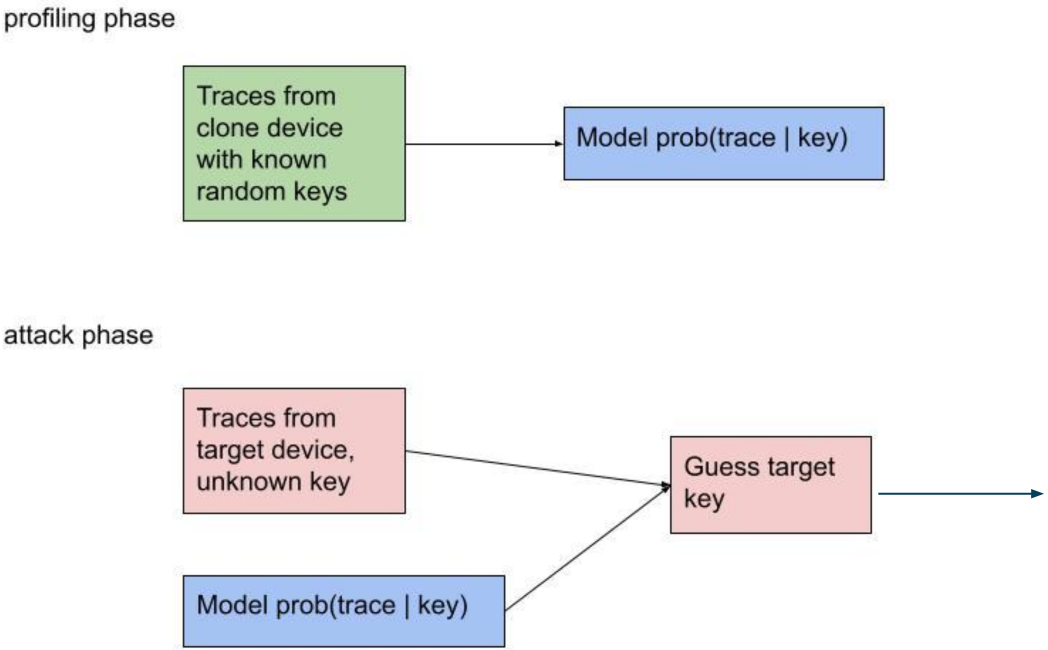
TVLA after pre-processing (combining samples using centred product)



Before collecting power traces, some countermeasures have been disabled in the IP for demonstrational purposes. First order masking is enabled.

3. No combination of multiple P.o.I.
   - Recall that leakage anywhere depends on the same bit of information
   - Minor leakage in many points may be combined to recover the bit

Profiled attack for estimating security



- Model describes the leakage for each possible subkey
- How to create model from traces with known keys:
  - Difference of means
  - Gaussian templates
  - Machine learning

Model accuracy: probability of correct subkey guess

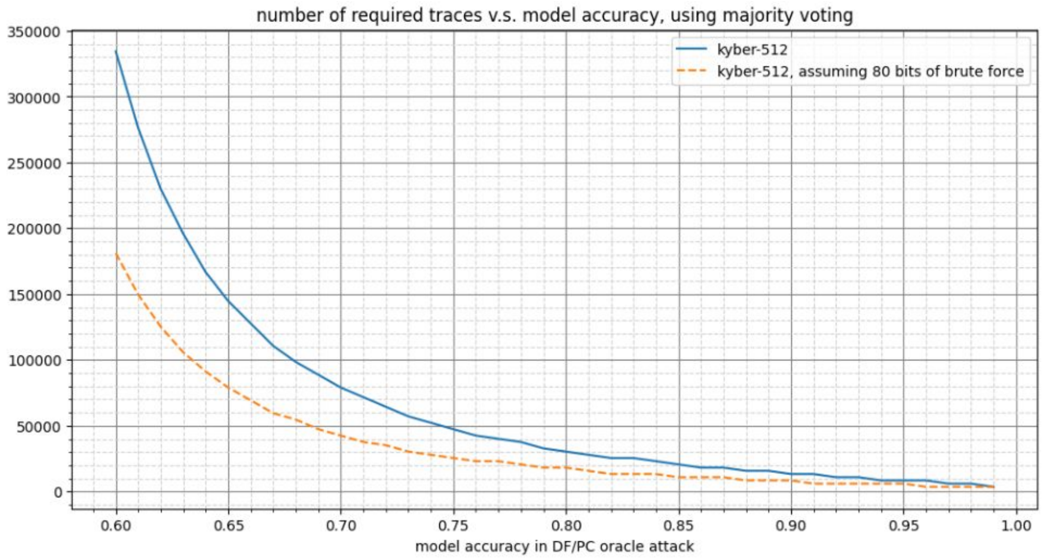
Estimating security against PC oracle SCA

Model accuracy ↔number of traces for key recovery

100% → 1216 queries [QCZ+D21] for MLKEM-512

Accuracy < 100% → use **majority voting** (or [SCZ+22]):

1. Measure N power traces with same input
2. Predict for each trace
3. Return the value that is predicted most



[QCZ+D21]: Qin et al. : "A Systematic Approach and Analysis of Key Mismatch Attacks on Lattice-Based NIST Candidate KEMs"

[SCZ+22]: Shen et al. : "Find the Bad Apples: An efficient method for perfect key recovery under imperfect SCA oracles"

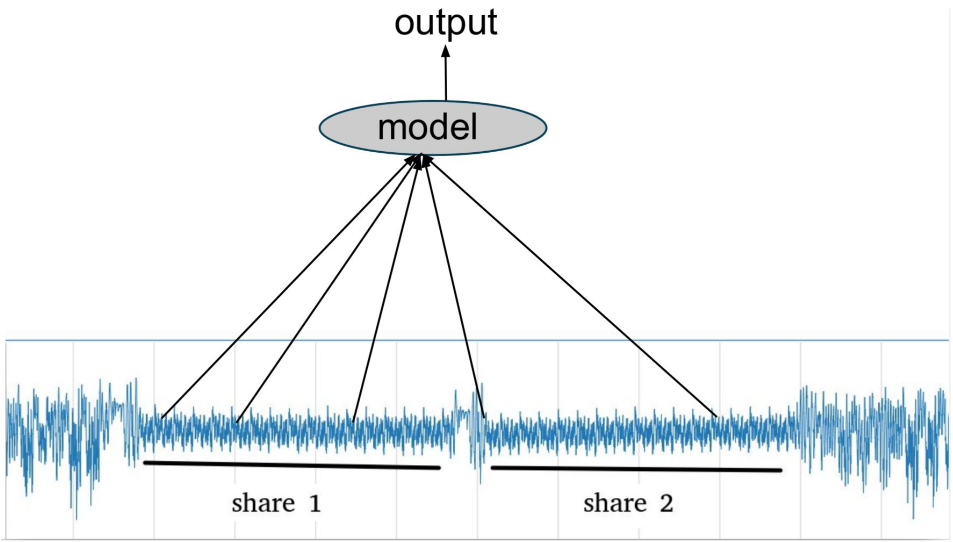
Machine learning for combining trace points

Linear operations masked in software

1. Each operand x is split up in 2 shares
2. Operation is computed on share 1 first, then on share 2

Combining samples for bivariate SCA

- Need to combine trace samples from 2 shares
- Manually: locate trace samples and compute product
- Automatically:
  - feed trace into neural network
  - it will learn which samples to combine



feed whole decaps trace into neural network?
⟶ slow and ineffective training phase

ML-based profiled attack on trace segments

Split up the trace set in segments

For each trace segment set:

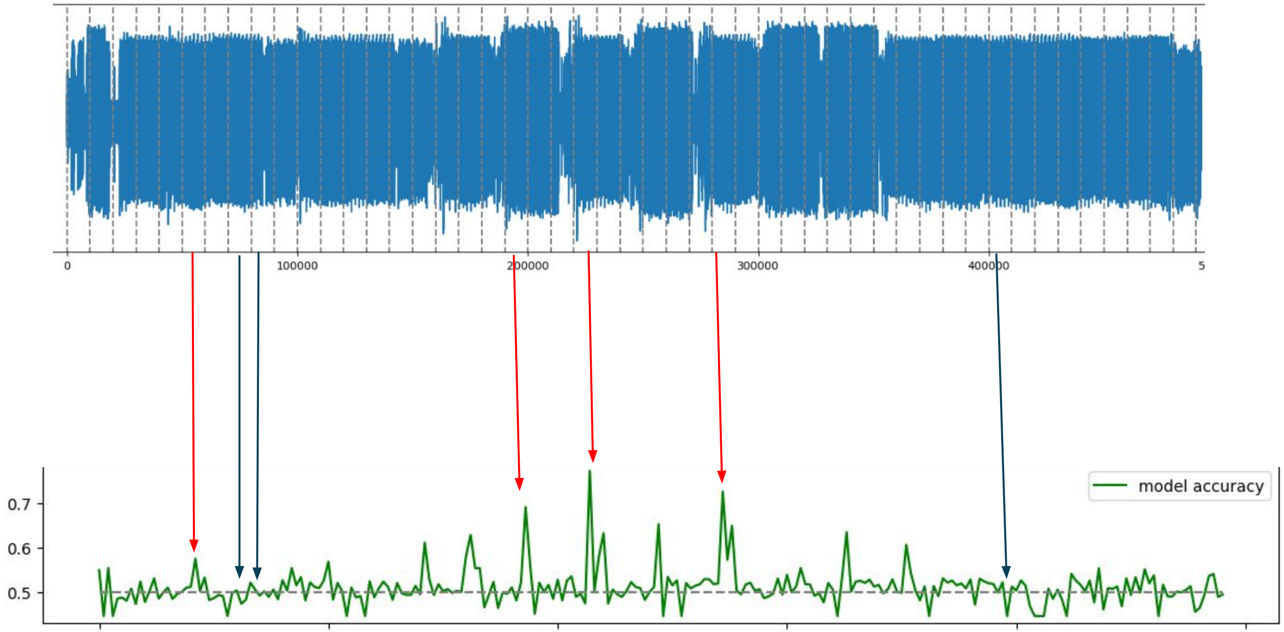
1. Train neural network
2. Save model
3. Plot accuracy

⟶ : close to 50% (no leakage)

⟶ : leakage (1st/2nd order)

Combine leakage peaks:

1. Sum up scores from all models (where ⟶)
2. Re-compute accuracy → 92% → ~11k traces



Conclusion

Method for estimating SCA security against PC-oracle attacks

- Output returns number of traces required for key recovery
- Detects both univariate and (locally) multivariate leakage
- Combines information from the whole re-encaps trace

To be improved

- Method is only semi-automated: captured power traces must be stored
- Model hyperparameters can be tuned
    - Current version uses 1 convolutional layer and 1 dense layer

Any questions?