On the Feasibility of Single-Trace Attacks on the CDT Gaussian Sampler

Soundes Marzougui[1], Ievgan Kabin[2], Juliane Krämer[3], Thomas Aulbach[3], Jean-Pierre Seifert[1,4]

COSADE 2023, Munich

[1] Technical University of Berlin, Germany
[2] Innovations for High Performance Microelectronics, Germany
[3] Regensburg University, Germany
[4] Fraunhofer Institute for Secure Information Technology, Germany

Motivation

Quantum computers are expected to be widely available in the next decade

Motivation

Quantum computers are expected to be widely available in the next decade

↓

Quantum attacks pose a threat to classical cryptography

Motivation

Quantum computers are expected to be widely available in the next decade

↓

Quantum attacks pose a threat to classical cryptography

↓

Urgent need for quantum-resistant schemes

Motivation

Lattice-based

- Dilithium
- FALCON
- Kyber
- FrodoKEM
- ...

Multivariate

- ~~Rainbow~~
- UOV
- ...

Hash-based

- SPHINCS+
- ...

Code-based

- McEliece
- HQC
- BIKE
- ...

Supersingular

- ~~SIKE~~
- ...

Lattice-based key encapsulation mechanisms

- The security of lattice-based cryptography often relies on the Learning with Errors problem (LWE)
- An LWE instance contains the secret vectors blinded with a noise vector (error)
- Usually, the noise vectors are taken from a Gaussian distribution, typically acquiring many samples for a single run of the scheme

Key encapsulation mechanisms

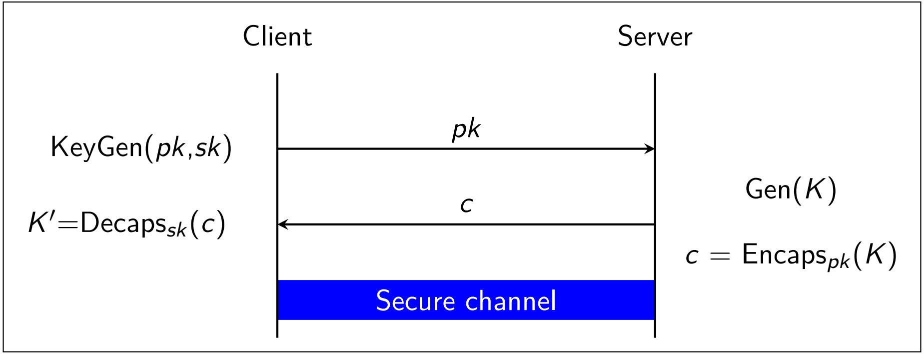


Figure: Simplified Example of TLS Connection Establishment

FrodoKEM: Decapsulation (simplified)

Algorithm 1 FrodoKEM.Decaps($c_1 \| c_2$ and $sk(s \| S)$)

$B, B', C \leftarrow$ Frodo.Unpack(c_1, c_2, b)
Compute $M \leftarrow C - B'S$
Compute $\mu' \leftarrow$ Frodo.Decode(M)
Sample error matrix S', E', and E''
Compute $B'' \leftarrow S'A + E'$ (A is public)
Compute $V \leftarrow S'B + E''$
Compute $C' \leftarrow V +$Frodo.Encode(μ')
if $B' \| C = B'' \| C'$ **then**
    return $(c_1 \| c_2 \| SHAKE(c_1 \| c_2 \| \mu')$
else
    return $(c_1 \| c_2 \| SHAKE(c_1 \| c_2 \| s)$
end if

FrodoKEM: Decapsulation (simplified)

Algorithm 1 FrodoKEM.Decaps($c_1 \| c_2$ and $sk(s \| S)$)

1: $B, B', C \leftarrow \text{Frodo.Unpack}(c_1, c_2, b)$
2: Compute $M \leftarrow C - B'S$
3: Compute $\mu' \leftarrow \text{Frodo.Decode}(M)$
4: Sample error matrix S', E', and E''
5: Compute $B'' \leftarrow S'A + E'$ (A is public)
6: Compute $V \leftarrow S'B + E''$
7: Compute $C' \leftarrow V + \text{Frodo.Encode}(\mu')$
8: **if** $B' \| C = B'' \| C'$ **then**
9: return $(c_1 \| c_2 \| SHAKE(c_1 \| c_2 \| \mu')$
10: **else**
11: return $(c_1 \| c_2 \| SHAKE(c_1 \| c_2 \| s)$
12: **end if**

The CDT Gaussian sampler

Algorithm 2 Constant-time CDT sampling

Require: CDT ψ of length l, σ, τ

Ensure: Sampled value S

1: $S \leftarrow 0$
2: rnd $\leftarrow$ $[0,\tau\sigma) \cup \mathbb{Z}$ unifromly at random
3: sgn $\leftarrow$ $[0,1] \cup \mathbb{Z}$ unifromly at random
4: **for** $(i = 0\ ;\ i < l - 1;\ i + +)$ **do**
5: $\quad S \mathrel{+}= (\psi[i] - rnd) >> 15$
6: **end for**
7: $S \leftarrow ((-sgn) \wedge S) + sgn$
8: **return** S

- The Gaussian sampler is based on a cumulative distribution table CDT
- The CDT length depends on deviation of the Gaussian distribution σ and the Tailcut τ
- The implementation is constant-time
- A sign bit is assigned to the positive output sample

Side-channel analysis of the CDT Gaussian sampler

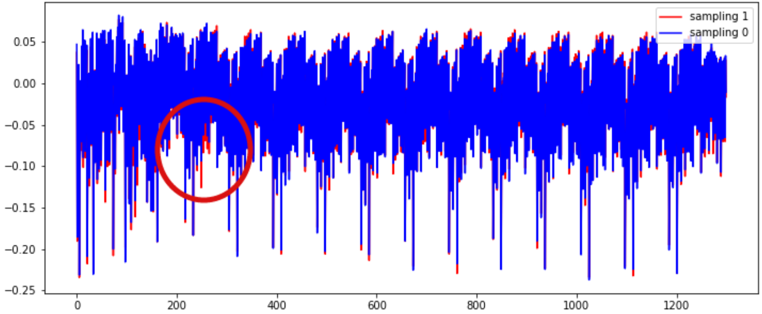


Figure: Overlapped power consumption measurement during the execution of the CDT sampler on an 8-bit Harvard board equipped with an XMega micro-controller; the red color corresponds to the sampling of the value 1, while the blue color corresponds to the sampling of the value 0

Experiments vs. Real-world scenario

- 8-bit Harvard board are used in literature
- An XMega micro-controller which is especially common in educational embedded applications
- In contrast, Cortex-M boards have been embedded in tens of billions of consumer devices
- The frequency and/or the sampling rate have dramatic effect on the accuracy of the power consumption traces
- Noise filtering tools

Measurements on Cortex-M4

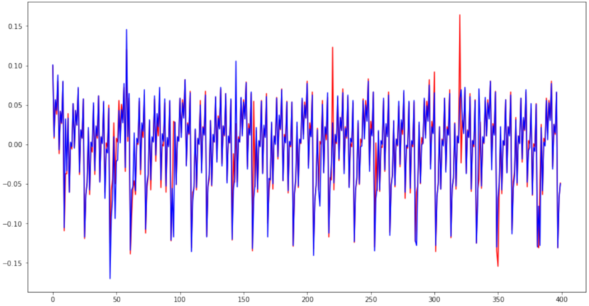


Figure: Overlapped Power consumption measurement during the execution of the CDT sampler on a Cortex-M4 equipped with an STM32F4 microcontroller; the red color corresponds to the sample of the value 0, while the blue color corresponds to the sampling of the value 1

Measurements with different frequencies

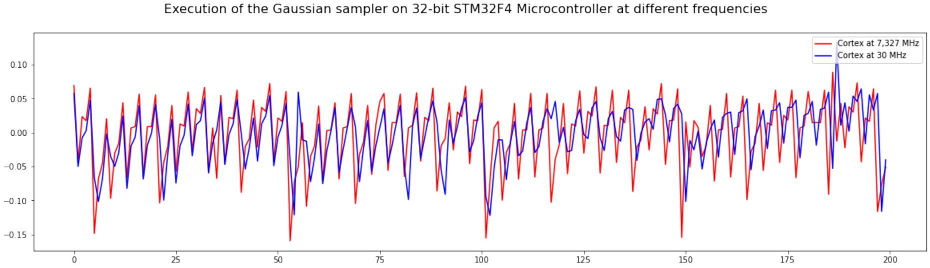


Figure: Overlapped Power consumption measurement during the execution of the CDT sampler on a 32-bit Cortex board equipped with an STM32F4 micro-controller with different frequencies

Power consumption measurement

We write the power consumption at a specific point of time as the following:

$$P = P_{op} + P_{data} + P_{noise} + P_{const}$$

Why machine-learning side channel analysis?

- No assumptions on the introduced noise
- Automated selection of Points of Interest (POI)
- Efficient management of large traces/small profiling sets
- Resilience against the addition of useless (i.e. non-informative) leakage samples in the traces

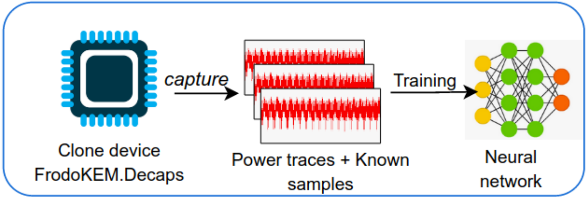
Threat models in profiling attacks

- The classical threat model: A single-device-model
- Portability threat model: A cross-device attack (identical devices, homogeneous devices, heterogeneous devices, etc.)
- Non-profiling supervised threat model: A differential deep learning analysis

Threat models in profiling attacks

- The classical threat model: A single-device-model
- **Portability threat model**: A cross-device attack (**identical devices**, homogeneous devices, heterogeneous devices, etc.)
- Non-profiling supervised threat model: A differential deep learning analysis

Machine-learning profiling attack on FrodoKEM: Profiling phase

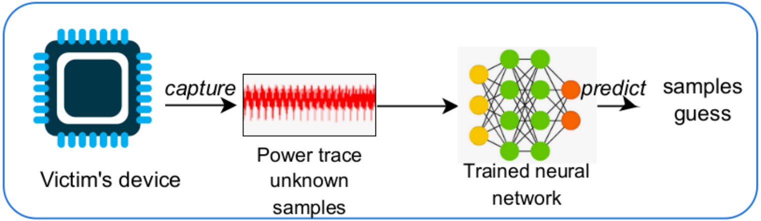


Profiling phase

Machine-learning profiling attack on FrodoKEM: Profiling phase

- The list of these noisy measurements is split into training, validation, and a test set of the Multi-Layer Perceptron (MLP) machine-learning classifier
- The attacker should train a classifier for each board
- Tuning the hyper-parameters of our machine-learning model is of particular importance because it influences the accuracy
- We captured 20,000 power consumption traces. We set 18,000 of them for training and testing and 2,000 for validation.

Machine-learning profiling attack on FrodoKEM: Attack phase



Attack phase

FrodoKEM: Decapsulation (simplified)

Algorithm 1 FrodoKEM.Decaps($c_1 \| c_2$ and $sk(s \| S)$)

$B, B', C \leftarrow$ Frodo.Unpack(c_1, c_2, b)
Compute $M \leftarrow C - B'S$
Compute $\mu' \leftarrow$ Frodo.Decode(M)
Sample error matrix S', E', and E''
Compute $B'' \leftarrow S'A + E'$ (A is public)
Compute $V \leftarrow S'B + E''$
Compute $C' \leftarrow V+$Frodo.Encode(μ')
if $B' \| C = B'' \| C'$ **then**
    return $(c_1 \| c_2 \| SHAKE(c_1 \| c_2 \| \mu')$
else
    return $(c_1 \| c_2 \| SHAKE(c_1 \| c_2 \| s)$
end if

Session key recovery

Having the values of S' and E'', the attacker computes the matrix V:

$$V = S'B + E''$$

Session key recovery

Having the values of S' and E'', the attacker computes the matrix V:

$$V = S'B + E''$$

It is known from the decapsualtion algorithm that:

$$C' = V + Frodo.Encode(\mu')$$

Session key recovery

Having the values of S' and E'', the attacker computes the matrix V:

$$V = S'B + E''$$

It is known from the decapsualtion algorithm that:

$$C' = V + Frodo.Encode(\mu')$$

Then, the attacker obtains the matrix Encode(μ'):

$$C' = S'B + E'' + Frodo.Encode(\mu')$$

Session key recovery

Having the values of S' and E'', the attacker computes the matrix V:

$$V = S'B + E''$$

It is known from the decapsualtion algorithm that:

$$C' = V + Frodo.Encode(\mu')$$

Then, the attacker obtains the matrix Encode(μ'):

$$C' = S'B + E'' + Frodo.Encode(\mu')$$

Hence, μ' can be written as:

$$\mu' = Frodo.Decode(C' - S'B - E'')$$

Conclusion

- We investigated the feasibility of single trace attacks against the CDT Gaussian sampler
- We proved that in real-world circumstances the accuracy of the attack decreases
- We present a machine-learning classifier leveraging the accuracy of the attack to 100%
- We apply our attack on FrodoKEM in real-world circumstances and present a proof of concept of our attack implementation

Thank you for your attention