Embedding Belief Propagation within a Multi-Task Learning Model

An example on Kyber's iNTT

Thomas Marquet, Elisabeth Oswald

SPyCoDe

enCRYPTON

Side channel attacks

Power side-channel attacks exploit **physical leakage** during computation:

- **Power consumption** — data-dependent current draw
- **EM emanations** — electromagnetic radiation

Why It Works:

$$\text{Power}(t) \approx \mathrm{L}\big(f(k, p)\big) + \varepsilon(t)$$

The instantaneous power consumption *correlates* with intermediate values that depend on the secret key k.

Masking

Goal: Split a sensitive variable x into d shares so that any $d-1$ shares are statistically independent of x.

Arithmetic Masking

Given random masks $r_1, r_2, \ldots, r_{d-1}$, compute:

$$x_0 = (x - r_1 - r_2 - \cdots - r_{d-1}) \mod q$$

The shares $(x_0, r_1, \ldots, r_{d-1})$ satisfy:

$$x = (x_0 + r_1 + r_2 + \cdots + r_{d-1}) \mod q$$

Shares are assumed unknown in this work/presentation

Profiled attacks

The strongest attacker leverages a clone of the target, to model the function L beforehand

Classical Techniques

- Closed form solution available
- Guaranteed global minima given a set of traces
- Require pre-processing to defeat masking
- Require pre-processing to synchronize the traces

Deep Learning

- **Can** learn arbitrary complex functions
- Stochastic by nature
- **Can** defeat jitter and countermeasures
- Exchange guarantees for potential

The problem with Deep Learning

DL based neural networks can defeat masking or jitter (but in practice with great difficulty)

Problem

- Low SNR makes exploration extremely difficult
- Not a plug and play solution
- Obtaining a successful model requires to iterate over educated guesses (many) many times

d-branch neural networks

One can assume knowledge of the masking scheme! [1]

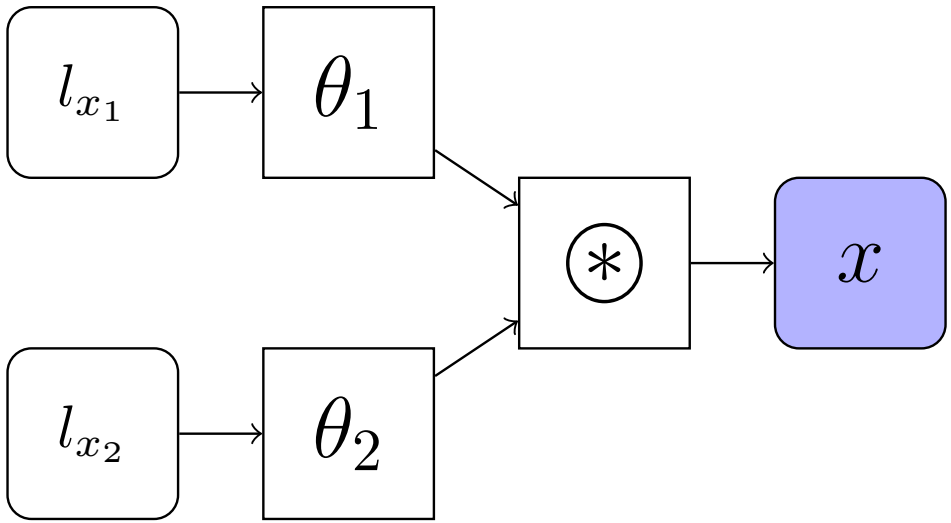

Figure: d-branch neural network

[1] *Dont Learn What You Already Know*, Masure et al. [2]

d-branch neural networks

Cool feature

You can get some idea of the p.m.f. of the shares at the end of the branches.

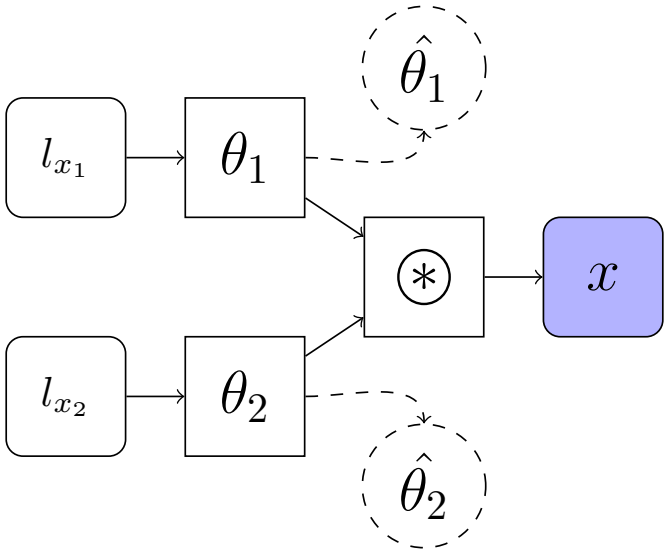

Figure: d-branch neural network

This is exactly what we leverage!

Multi-task learning

Definition

A deep learning model that is trained using multiple training labels.

Potential Benefits for SCA

- Leakage is by definition multivariate
- Data augmentation effect
- Eavesdropping

Tasks can compete as much as they can collaborate!

Task-graph multi-task learning

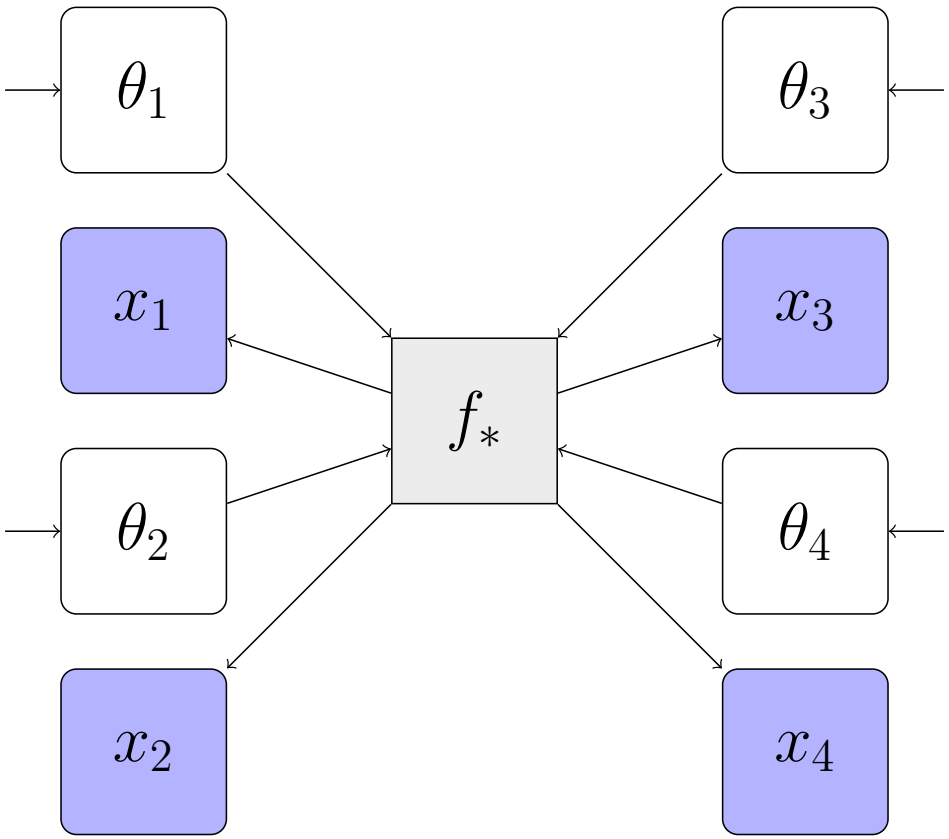

Figure: Example of a task-graph to embed relationships between objectives

Kyber's iNTT

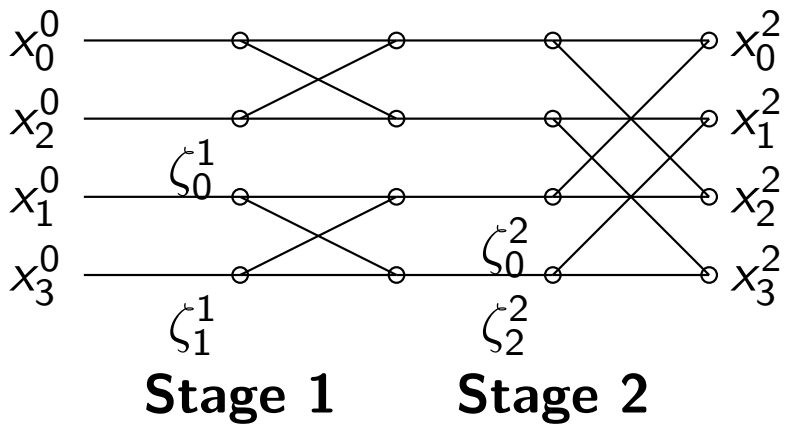

Figure: A 2-stage NTT/iNTT Butterfly

Equations of a single butterfly:

$$\begin{cases} x_k^l = x_k^{l-1} + x_{k+p_l}^{l-1} \pmod q & (1) \\ x_{k+p_l}^l = \zeta_k^l (x_{k+p_l}^{l-1} - x_k^{l-1}) \pmod q & (2) \end{cases}$$

The "perfect" masking scheme

A crypto engineer's dream

Because of linearity of the NTT:

$$NTT(a) = NTT(a_1) + NTT(a_2) \ (mod \ q)$$

Therefore, the flow is:

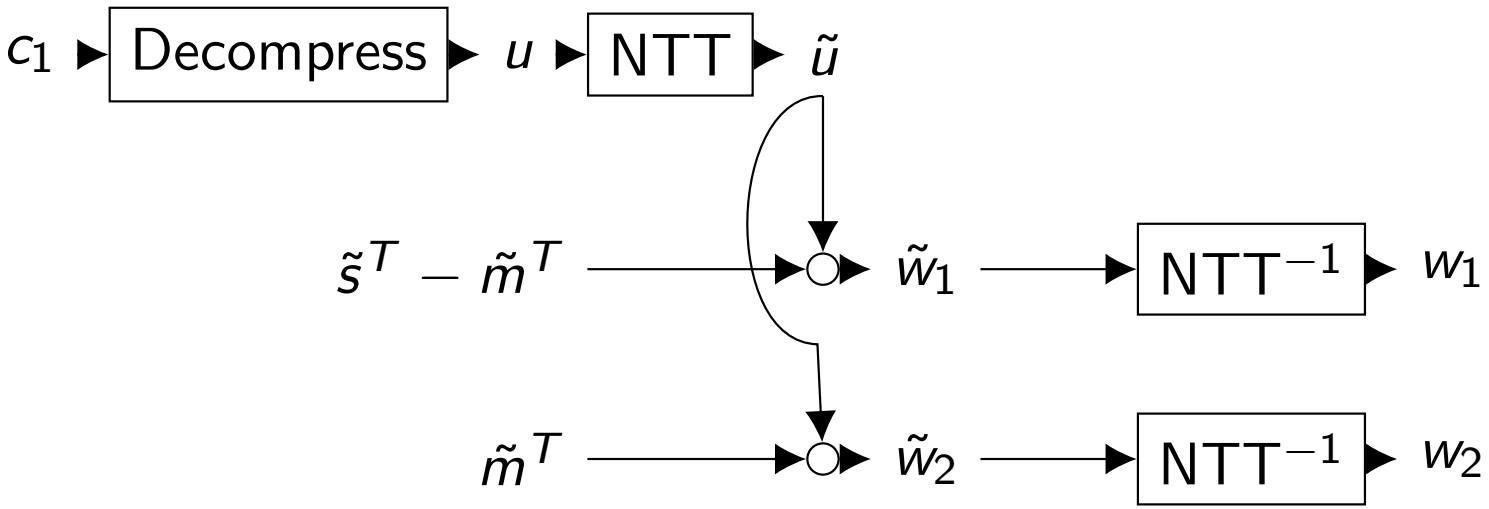

Figure: Arithmetic masking flow around the inverse NTT operation

Belief propagation on Kyber's iNTT

Optimal update rule

The optimal update rule was defined first in Pessel and Primas [2]

$$\begin{cases} x_k^{l-1} &= (1-\zeta_k^l)^{-1}(-(1+\zeta_k^l)x_{k+p_l}^{l-1} + x_k^l + x_{k+p_l}^l) \pmod q & (3) \\ x_{k+p_l}^{l-1} &= (1+\zeta_k^l)^{-1}(-(1-\zeta_k^l)x_{k+p_l}^{l-1} + x_k^l + x_{k+p_l}^l) \pmod q & (4) \\ x_k^l &= (1-\zeta_k^l)x_k^{l-1} + (1+\zeta_k^l)x_{k+p_l}^{l-1} - x_{k+p_l}^l \pmod q & (5) \\ x_{k+p_l}^l &= (1-\zeta_k^l)x_k^{l-1} + (1+\zeta_k^l)x_{k+p_l}^{l-1} - x_k^l \pmod q & (6) \end{cases}$$

Previous work assume knowledge of the shares to perform Belief propagation.

[2] *More Practical Single-Trace Attacks on the Number Theoretic Transform* [3]
Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber, Hamburg et al. [1]

Concept

Goal: Improve the chances to obtain a successful model

How?

- Deep learning models can extract the p.m.f. of the shares without labeling of said shares.
- Clear relationships between NTT/iNTT intermediates can be written down.

Can I propagate information inside a deep learning model, without knowing the shares?

Experiments

Setup

We target the iNTT in the decryption

Device under attack

- Chipwhisperer Lite
- STM32F303RCT7 microcontroller (Cortex M4)

Implementation

- pqm4/mkm4 from Kannwisher et al.
- The iNTT are computed sequentially on each share
- Random keys/ciphertexts for training
- Fixed key, chosen ciphertexts for the attack

Experiments

Models

We train three types of masking scheme aware models:

- single-task
- multi-task-ex, with an extra loss before the propagation
- multi-task

Regardless of the type of model, each possess exactly the same amount of hyperparameters!

Masking scheme: f_+

$$x[i] = \sum_{j=0}^{q-1} x_1[j] \times x_2[i-j \pmod{q}] \tag{7}$$

Implemented as a FFT convolution to reduce complexity.

BP Update: f_b^l

With $\mathbf{\Theta_m} = \{\theta_1, \theta_2, \theta_3, \theta_4\}$

$$x_i = \theta_i \; + \; \log\left[\sum_{\mathbf{\Theta}_m \setminus \theta_i} \exp\left(f_b^l(\mathbf{\Theta}_m) \cdot \sum_{\theta_k \in \mathbf{\Theta}_m \setminus \theta_i} \theta_k \right) \right] \tag{8}$$

Single butterfly node only!

The Models

single-task

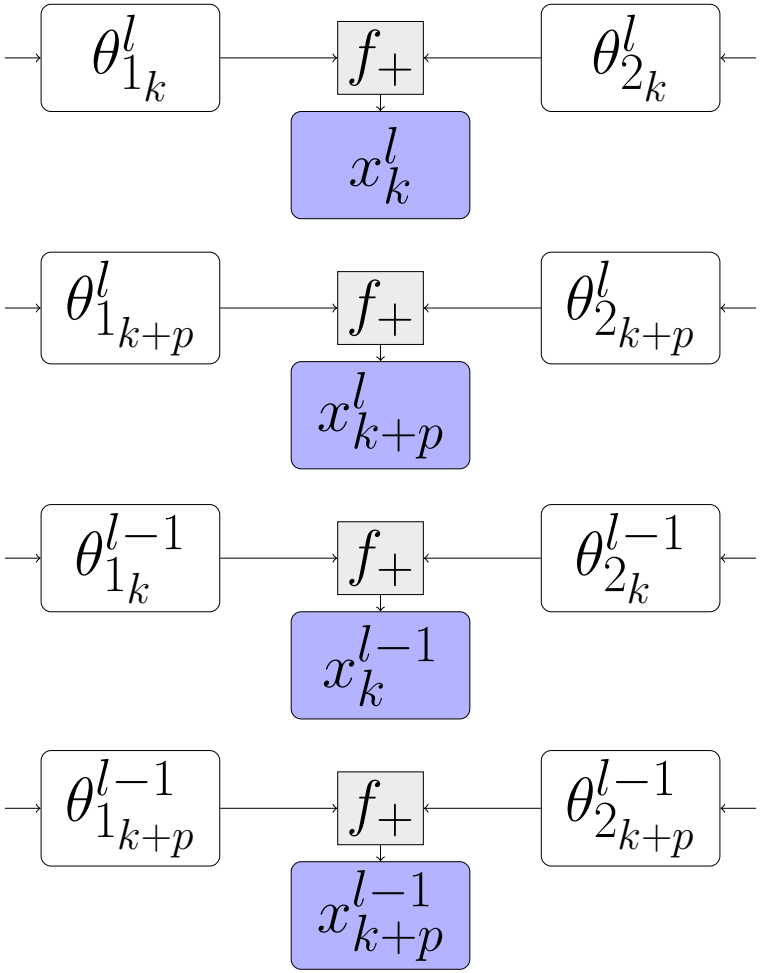

Figure: Model single-task

The Models

multi-task

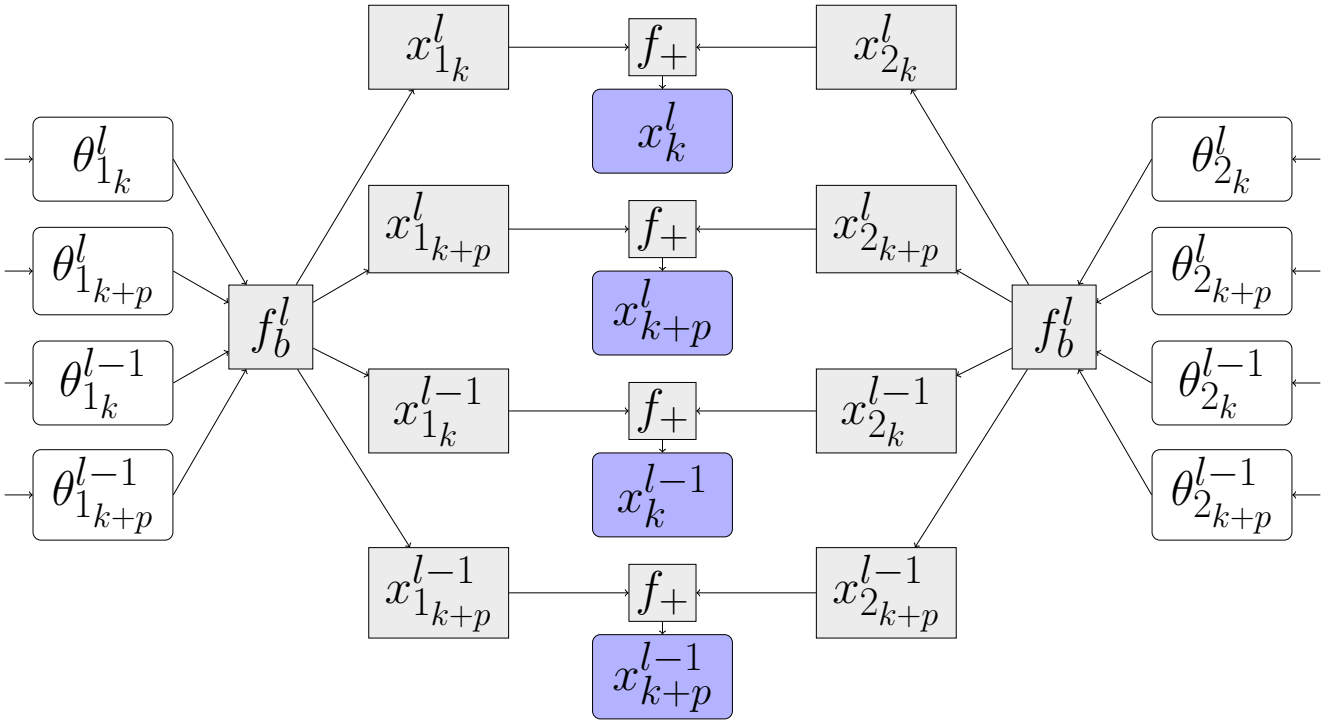

Figure: Model multi-task

The Models

multi-task-ex

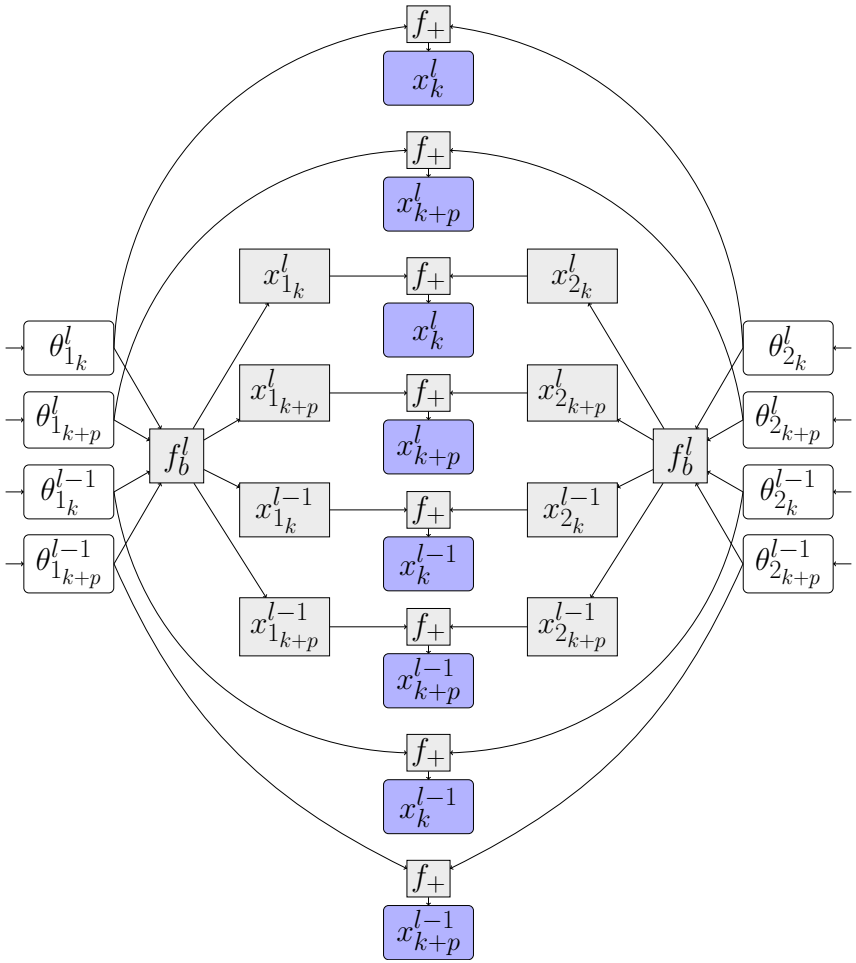

Figure: Model multi-task-ex

To better understand the results

Goal: Observe if the information is propagated by the butterfly layer

Warning:

- Plots of the validation loss
- Dotted black line is the random guess
- Focus on the lines of the same color

$$\text{Validation loss} > \text{random guess} = \text{Attack fails}$$

Comparison against the single-task baseline

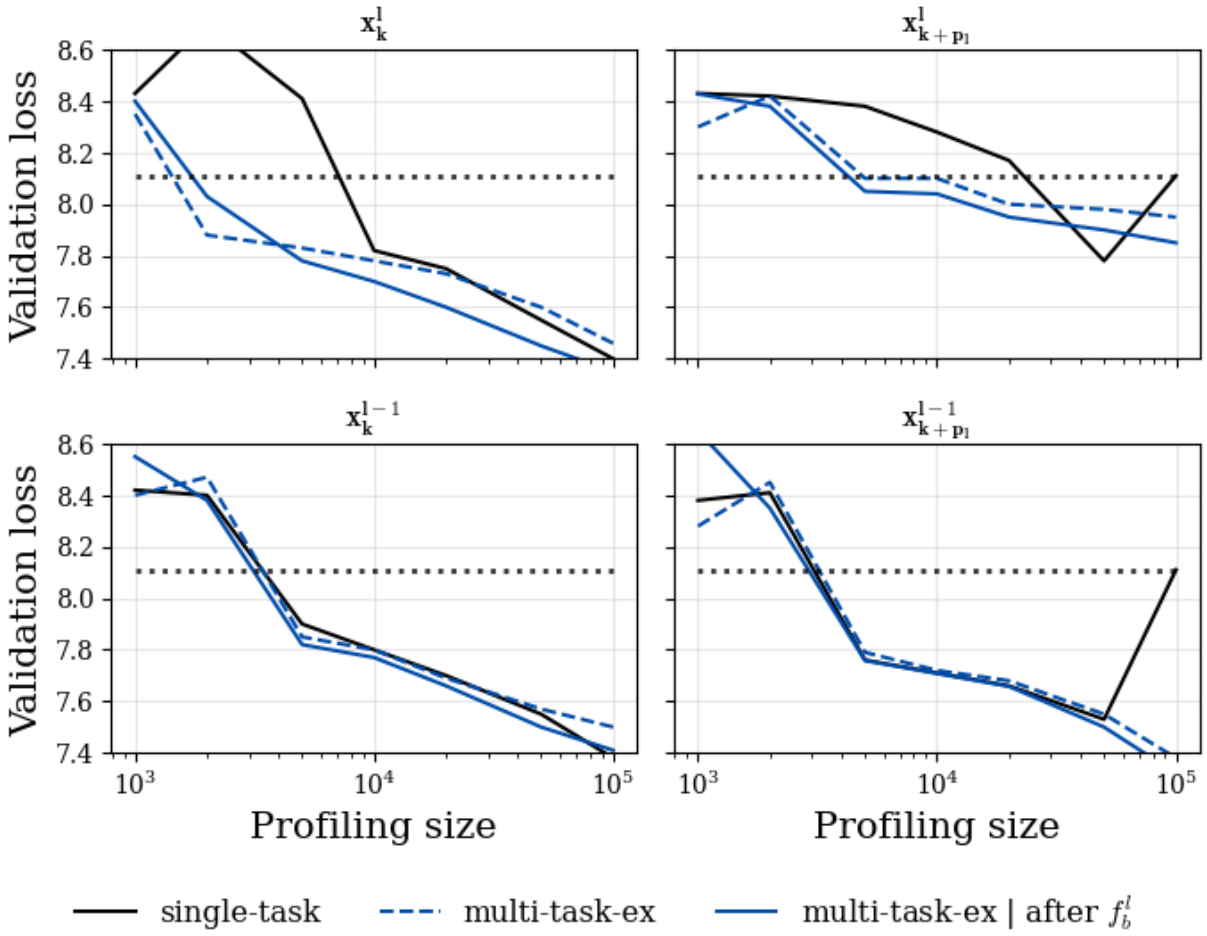

Figure: Validation loss achieved for different sizes of the training set

Comparison between the multi-task models

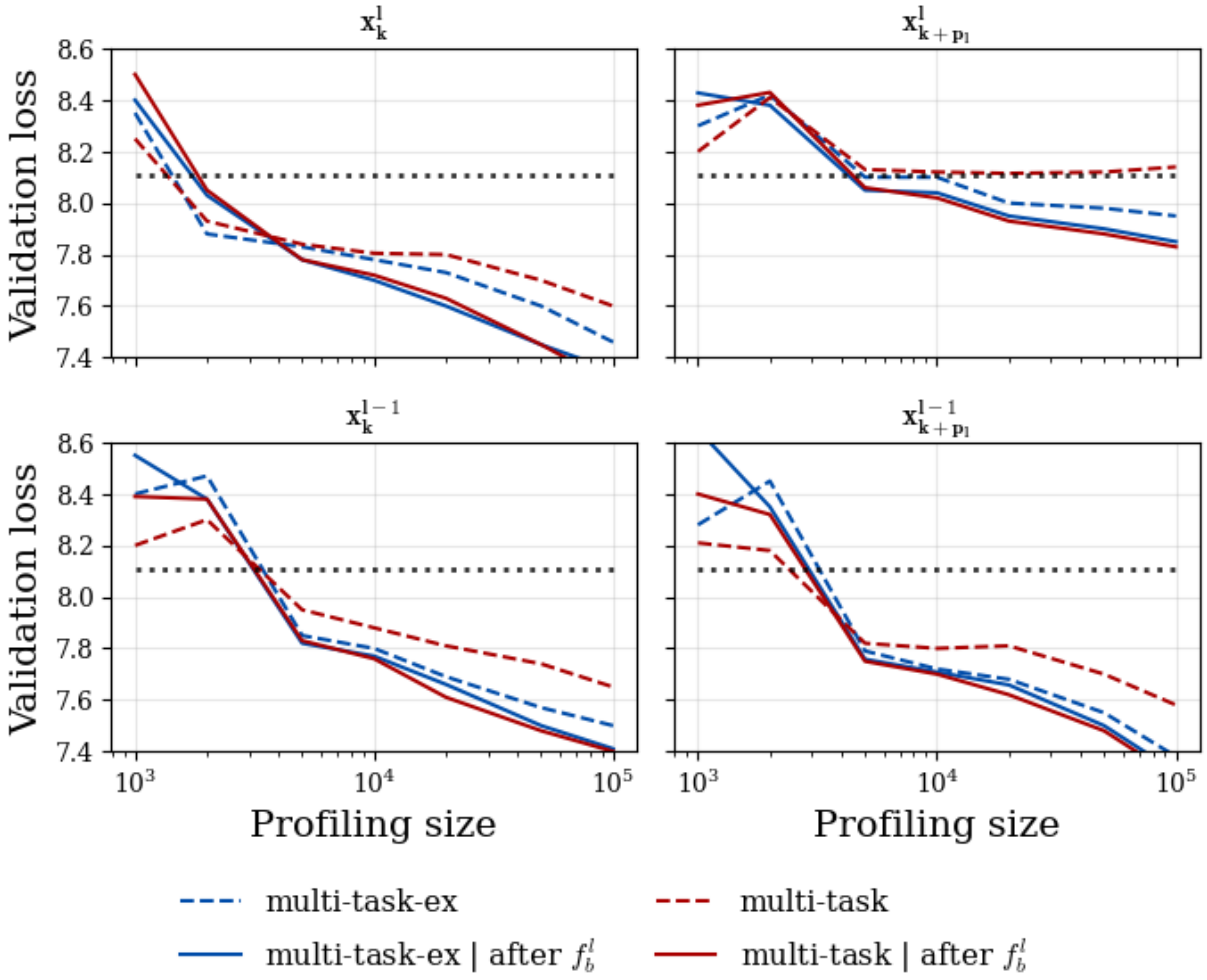

Figure: Validation loss achieved for different sizes of the training set

This propagates!

But

Limitations

- On the expensive side
- Requires significant infrastructure

The main goal of this technique is not to improve the attack, but to obtain a successful model!

Failed attempt to optimize this

Using the FFT domain for convolutions

- Significantly reduces the (memory) complexity
- Did show some sign of propagation with small batch sizes
- Did not show sign of propagation with larger ones

Most likely a mix of issues from my implementation and FFT domain numerical instability

Conclusion

Takeaways

- An attacker can leverage the fact that linear parts are not remasked to improve it's chance of obtain a successful model (And consequently the performances of the attack)
- Only on one butterfly node, but going through two layers with bayesian update rules and the structure is still respected
- Masking linear parts of PQC implementations should not be overlooked

SPyCoDe

enCRYPTON


Hamburg, M., Hermelink, J., Primas, R., Samardjiska, S., Schamberger, T., Streit, S., Strieder, E., van Vredendaal, C.: Chosen ciphertext k-trace attacks on masked cca2 secure kyber. IACR Transactions on Cryptographic Hardware and Embedded Systems **2021**(4), 88–113 (Aug 2021). https://doi.org/10.46586/tches.v2021.i4.88-113, https://tches.iacr.org/index.php/TCHES/article/view/9061


Masure, L., Cristiani, V., Lecomte, M., Standaert, F.X.: Don't learn what you already know: Scheme-aware modeling for profiling side-channel analysis against masking **2023**, 32–59 (Nov 2022). https://doi.org/10.46586/tches.v2023.i1.32-59, https://tches.iacr.org/index.php/TCHES/article/view/9946


Pessl, P., Primas, R.: More practical single-trace attacks on the number theoretic transform. In: Progress in Cryptology – LATINCRYPT 2019: 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2–4, 2019, Proceedings. p. 130–149. Springer-Verlag, Berlin, Heidelberg (2019)