Universal Forgery Attack on Dilithium Leveraging Power Side Channels

Vincent Ulitzsch<v.ulitzsch@campus.tu-berlin.de>[1], Soundes Marzougui[1], Mehdi Tibouchi[2], Jean-Pierre Seifert[1]
[1] Technical University of Berlin, Germany
[2] NTT Corporation, Tokyo, Japan

Dilithium, a NIST Post-Quantum Cryptography Candidate

Dilithium is lattice-based fiat-shamir with aborts signature scheme [1]
Private key: $\mathbf{s}_1, \mathbf{s}_2 \in \mathrm{R}_\mathrm{q}^\mathrm{l} \times \mathrm{R}_\mathrm{q}^\mathrm{k}$, where $\mathrm{R}_\mathrm{q} = \mathrm{Z}_\mathrm{q}[\mathrm{X}]/(\mathrm{X}^\mathrm{n}+1)$ and coefficients are small
Public key: $(\mathbf{A}, \mathbf{t}), \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, where **A** is sampled uniformly at random from $\mathrm{R}_q^{\ell \times k}$

Signature: $z = \mathbf{y} + cs_1$

$\mathbf{y} \in \mathrm{R}_q^l$ is a masking vector of polynomials, generated at random. y's coefficients are in $\{-\gamma_1, \dots, \gamma_1\}$.

c is a challenge polynomial derived from the message to be signed. c's coefficients are in $\{0, +1, -1\}$

Research Question: **Assessing the resilience of the Dillithium reference implementation against power side-channel analysis on an ARM Cortex M4.**

Attack Idea

In the reference implementation, **the generation of the masking vector leaks information via power consumption**

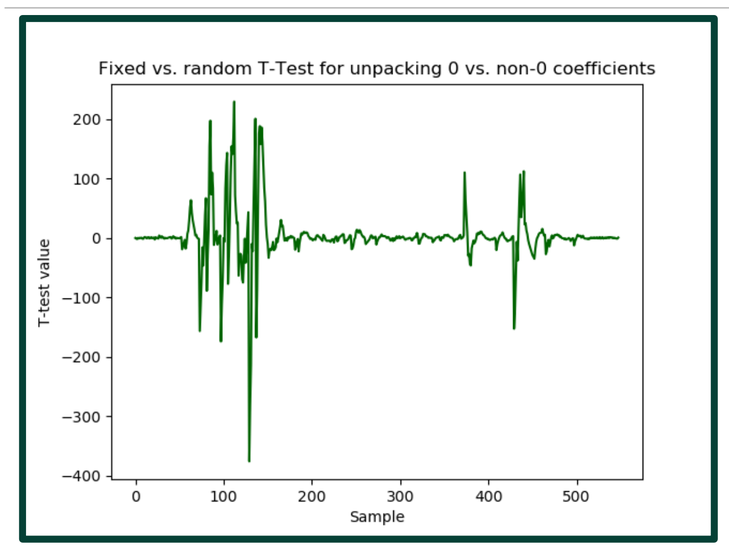
➡ **Using deep neural networks on power traces, we can detect for** coefficient j of polynomial i **whether**

$$\mathbf{y}_{\{i,j\}} = \mathbf{0}$$

Knowing information about $\mathbf{y}$ from the power leak we can **recover the secret key polynomial of s_1** with **integer linear programming**

The knowledge of $\mathbf{s}_1$ is sufficient to perform universal forgery.

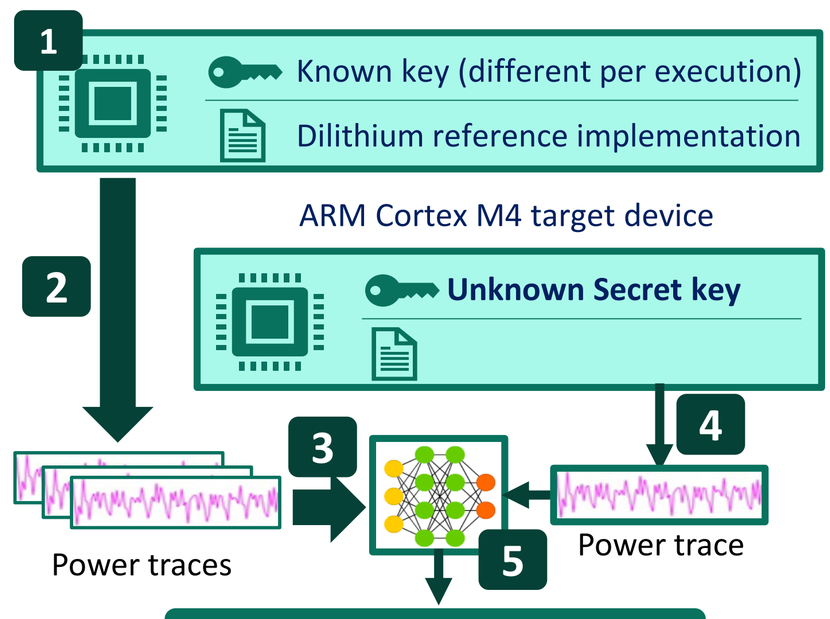
Power traces leak masking vector y



- The masking vector y is generated via unpacking a bit-string expanded from a seed ρ', $\mathbf{y}$ = unpack(expand(ρ'))
- **Power measurements reveal whether the unpacked coefficients are zero or non-zero**

Profiling Attack: Machine learning flow

1. **Record traces with known keys and messages on attacker-controlled device**
2. **Collect training data in form of labelled power traces**
3. **Train a DNN to distinguish between zero and non-zero coefficients**
4. **Record traces with unknown keys and random message on device under attack**
5. **Classify traces with trained model and detect 0-coefficients in masking vector**



Recovering the secret key through Integer Linear Programming

1 Identify 0-coefficients of masking vector y

Sign(sk, M)

$\mathbf{y} \leftarrow \mathrm{S}_{\gamma_1 - 1}^{\ell}$

$y_{i,j} = 0$?

$\mathrm{w}_1 = \mathrm{HighBits}(Ay, 2\,\gamma_2)$
$\mathrm{c} \in B_r \coloneqq \mathrm{H}(M \mid w_1)$
$\mathbf{z} \coloneqq y + c\,s_1$
return $\sigma = (\mathbf{z}, \mathrm{c})$

A power consumption side channel gives us information about which coefficients of the masking vector y are 0. After a profiling phase we achieve a true positive rate 98% and true negative rate of around 99.8%, using hyperparameter optimization [2] on the ARM Cortex M4.

2 Add $\mathrm{z} = \mathrm{cs}_1$ to equation system

$$\mathrm{Z} = \mathrm{C}\,\mathrm{s}_1 + \mathrm{e}$$

where for each prediction that in signature $\mathrm{m}, \mathrm{y}_{\{\mathrm{i,j}\}}^{\mathrm{m}} = 0$, we set $\mathrm{Z_l} = \mathrm{z}_{\{\mathrm{i,j}\}}^{\mathrm{m}}$ and derive $\mathrm{C_l}$ from $\mathrm{c_m}$, the current challenge vector. If , $\mathrm{y}_{\{\mathrm{i,j}\}}^{\mathrm{m}}$ is indeed zero, the relationship holds, otherwise the relationship holds up to some error.

For each predicted coefficient, we obtain a linear relationship through observing that $\mathrm{z}_{\{\mathrm{i,j}\}} = (\mathrm{cs}_1)_{\{\mathrm{i,j}\}} + \mathrm{y}_{\{\mathrm{i,j}\}}$ where $\mathrm{y}_{\{\mathrm{i,j}\}}$ is assumed to be zero. This will be true for most of the coefficients (assuming most of our classifications are correct). We insert the row $\mathrm{z}_{\{\mathrm{i,j}\}} = (\mathrm{cs}_1)_{\{\mathrm{i,j}\}}$ in our equation system.

3 Extract secret key by maximizing number of fulfilled equations via integer linear programming

$$\max \sum_{\mathrm{l}} \mathrm{x_l}$$

subject to

$$\mathrm{Z_l} - \mathrm{C_l s} \le \mathrm{M} \cdot (1 - \mathrm{x_l}) \forall\, \mathrm{l} \in \{1, \dots, |\mathrm{L}|\}$$
$$\mathrm{Z_l} - \mathrm{C_l s} \ge -\mathrm{M} \cdot (1 - \mathrm{x_l}) \forall\, \mathrm{l} \in \{1, \dots, |\mathrm{L}|\}$$
$$\mathrm{x_l} \in \{0,1\} \forall\, \mathrm{l} \in \{1, \dots, |\mathrm{L}|\}$$
$$s_i \in \{-2, \dots, 2\} \forall i \in \{1, \dots, n\}$$

We solve ℓ separate integer linear programs, one for each polynomial in $\mathrm{s_1}$. Each ILP finds a secret key polynomial s such that the number of fulfilled equations is maximized. Assuming most of our classifications are correct ($\mathrm{Z_l} = \mathrm{C_l s_1}$), this will exactly match the secret key polynomial s.

Research Result: We were able **to break Dilithium NIST Security Level 2 via profiling power-side channel** using around 750000 signatures on an ARM Cortex M4.

Counter-measure: Apply masking [3], by splitting each coefficient in y into shares $\mathrm{y} = \mathrm{y}_1 + \mathrm{y}_2 + \cdots + \mathrm{y_n} \bmod \mathrm{q}$

References

[1] Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D. (2018). Crystals–dilithium: Digital signatures from module lattices.
[2] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research, 18(1), 6765-6816.
[3] Migliore, V., Gérard, B., Tibouchi, M., & Fouque, P. A. (2019, June). Masking dilithium. In International Conference on Applied Cryptography and Network Security (pp. 344-362). Springer, Cham.